WTA 2010 – Quarto Workshop de Tecnologia Adaptativa

MEMÓRIAS DO WTA 2010

QUARTO WORKSHOP DE TECNOLOGIA ADAPTATIVA

Laboratório de Linguagens e Técnicas Adaptativas
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

São Paulo
2010

WTA 2010 – Quarto Workshop de Tecnologia Adaptativa

MEMÓRIAS DO WTA 2010

QUARTO WORKSHOP DE TECNOLOGIA ADAPTATIVA

Laboratório de Linguagens e Técnicas Adaptativas
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

São Paulo
2010

FICHA CATALOGRÁFICA

Workshop de Tecnologia Adaptativa (4 : 2010 : São Paulo)
Memórias do WTA´2010. – São Paulo ; EPUSP, 2010.
153 p.

ISBN 978-85-86686-56-6

1.Engenharia de computação(Congressos) 2.Teoria da computação (Congressos) 3.Teoria dos autômatos (Congressos) 4.Semântica de programação (Congressos) 5.Linguagens formais (Congressos) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

CDD 621.39

SUMÁRIO

APRESENTAÇÃO

WTA'2010

A terceira edição do WTA realizou-se em São Paulo, BRASIL, nos dias 21 e 22 de janeiro de 2010, na Escola Politécnica da Universidade de São Paulo, por meio do Departamento de Engenharia de Computação e Sistemas Digitais.

As contribuições encaminhadas na forma de artigos relacionados à Tecnologia Adaptativa, nas seguintes áreas, abrangeram, de forma não exclusiva, os tópicos abaixo:

TÓPICOS

- Teoria da Computação
- Autômatos
- Computação inspirada na biologia
- Compiladores
- Gramáticas
- Processamento de Linguagem Natural
- Jogos eletrônicos
- Tomada de decisões
- Inferência Gramatical

COMISSÃO DE PROGRAMA

Almir Rogério Camolesi (Assis, SP, Brasil)
Amaury Antônio de Castro Junior (Coxim, MS, Brasil)
Aparecido Valdemir de Freitas (São Caetano do Sul, SP, Brasil)
César Alberto Bravo Pariente(São Paulo, SP, Brasil)
Hemerson Pistori (Campo Grande, MS, Brasil)
Marcus Vinicius Midena Ramos (São Paulo, SP, Brasil)

COMISSÃO ORGANIZADORA

André Riyuiti Hirakawa (São Paulo, SP, Brasil)
Cinthia Itiki (São Paulo, SP, Brasil)
Italo Santiago Vega (São Paulo, SP, Brasil)
João José Neto, Chair (São Paulo, SP, Brasil)
Ricardo Luis de Azevedo da Rocha (São Paulo, SP, Brasil)

Memórias do WTA 2010: Quarto Workshop de Tecnologia Adaptativa

R. L. A. Rocha

Resumo — **Esta publicação do Laboratório de Linguagens e Técnicas Adaptativas do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP é uma coleção de textos produzidos para o WTA 2010, o Terceiro Workshop de Tecnologia Adaptativa, realizado em São Paulo nos dias 21 e 22 de Janeiro de 2010. A exemplo da edição de 2009, este evento contou com uma forte presença da comunidade de pesquisadores que se dedicam ao estudo e ao desenvolvimento de trabalhos ligados a esse tema em diversas instituições brasileiras e estrangeiras, sendo o material aqui compilado representativo dos avanços alcançados nas mais recentes pesquisas e desenvolvimentos realizados.**

***Palavras-chave* — Adaptatividade, Autômatos adaptativos, Comportamento Auto-modificável, Sistemas Adaptativos, Tecnologia Adaptativa.**

I. INTRODUÇÃO

Com muita satisfação apresentamos neste documento estas memórias com os artigos apresentados no WTA 2010 – Terceiro Workshop de Tecnologia Adaptativa, realizado na EPUSP nos dias 21 e 22 de janeiro de 2010.

Constatou-se o terceiro sucesso desse evento pela efetiva participação de uma centena de pesquisadores, pelo recebimento de mais de 40 artigos, dos quais 9 foram selecionados completos, 11 selecionados como artigos que configuram pesquisa em andamento e 3 como comunicações. Os artigos mais bem avaliados foram aprimorados por seus autores para publicação na revista IEEE Latin American Transactions.

Na presente edição do WTA observaram-se algumas evoluções em relação à do ano anterior:

- **Chamada aberta de trabalhos**: facilitou-se assim o envolvimento e a aproximação de interessados externos;
- **Diversificação dos participantes e dos trabalhos:** foram recebidas e aceitas excelentes contribuições técnicas, oriundas de fora de S.Paulo e do Brasil;
- **Prestigiosos apoios:** da SBC (Sociedade Brasileira de Computação), SPC (Sociedad Peruana de Computación), IEEE (The Institute of Electrical and Electronics Engineering), EPUSP (Escola Politécnica da USP) e PCS (Dep. de Eng. de Computação e Sistemas Digitais).

R. L. A. Rocha é pesquisador do Laboratório de Linguagens e Técnicas Adaptativas, Escola Politécnica da USP, São Paulo/SP, Brasil; fone: 55 11-3091-5583; e-mail: luis.rocha@poli.usp.br.

II. ESTA PUBLICAÇÃO

Estas memórias espelham o conteúdo apresentado no WTA 2010.

As quatro seguintes áreas dominaram o WTA 2010:

- **Algoritmos**
- **Ambiente de execução adaptativo**
- **Aplicações on-line**
- **Processamento de linguagem natural**

A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estarão acessíveis no portal do WTA 2010, incluindo softwares e os slides das apresentações das palestras. Adicionalmente, o evento foi gravado em vídeo, em sua íntegra, e os filmes serão também disponibilizados aos interessados.

Esperamos que, pela qualidade e diversidade de seu conteúdo, esta publicação se mostre útil a todos aqueles que desejam adquirir ou aprofundar ainda mais os seus conhecimentos nos fascinantes domínios da Tecnologia Adaptativa.

III. CONCLUSÃO

A repetição do sucesso das edições anteriores do evento, e o nível de qualidade dos trabalhos apresentados atestam a seriedade do trabalho que vem sendo realizado, e seu impacto junto à comunidade.

Somos gratos aos que contribuíram de alguma forma para o brilho do evento, e aproveitamos para estender a todos o convite para participarem da próxima edição, em 2011.

AGRADECIMENTO

Às instituições que apoiaram o WTA 2010, à comissão organizadora, ao pessoal de apoio e a tantos colaboradores voluntários, cujo auxílio propiciou o êxito que tivemos a satisfação de observar.

São Paulo, fevereiro de 2010.
Prof. Dr. Ricardo Luis de Azevedo da Rocha
LTA – PCS - EPUSP

Créditos

São muitos os que deram ao nosso evento um auxílio decisivo, que viabilizou sua realização e permitiu que tivesse todo o êxito que apresentou.

Nosso agradecimento inicial a todos os que, de modo formal, institucionalmente apoiaram o WTA 2010:

- SBC – Sociedade Brasileira de Computação;
- SPC – Sociedade Peruana de Computação;
- IEEE – The Institute of Electrical and Electronics Engineering;
- EPUSP – Escola Politécnica da USP;
- PCS – Dep. de Engenharia de Computação e Sistemas Digitais.

Gostaríamos de agradecer ao IEEE, através do Profa. Dra. Mirela Sechi Moretti Annoni Notare, pela parceria com o nosso evento, e pela publicação dos melhores trabalhos na revista IEEE Latin American Transactions.

Agradecemos também, à Microsoft, que, através da pronta ação do Prof. Dr. Jorge Luís Risco Becerra, prestou ao evento uma importante colaboração mediante a cessão dos equipamentos utilizados na mostra de software do evento.

Registramos a seguir nosso reconhecimento público nominal àqueles que, direta ou indiretamente, contribuíram, com suas idéias e seu trabalho dedicado, para a organização e para a viabilização final desta edição do WTA (ordem alfabética):

- Prof. Dr. Almir Rogério Camolesi, a quem devemos: a divulgação do evento; a coleta das inscrições para o evento; a organização dos horários das apresentações; a criação e manutenção do novo portal do WTA;
- Prof. Amaury Antônio de Castro Júnior, a quem devemos: a gestão junto à SBC para apoio ao evento; a atualização do conteúdo do portal do LTA, especialmente das publicações; a criação do novo portal do LTA, com recursos mais automáticos que permitirão uma atualização mais rápida e segura e também pela participação no tutorial deste ano;
- Prof. Dr. João José Neto, pela presteza com que aceitou nosso convite para conduzir os trabalhos com a costumeira competência nos dois dias de duração do evento

Agradecemos finalmente a boa vontade da qual resultou o importante auxílio prestado ao WTA 2009 pelos seguintes profissionais (ordem alfabética):

- Leia Sicília, que secretariou o evento, pela permanente alegria e boa vontade que demonstrou, antes e durante seus trabalhos na recepção do WTA 2009;
- Nilton Araújo do Carmo, pelos trabalhos de apoio técnico prestados durante o evento, inclusive pela gravação de todas as imagens dos trabalhos do WTA 2009;

TUTORIAL

O tutorial apresentado pelo prof. Dr. Amaury Antônio de Castro Júnior apresentou um conjunto de contribuições teóricas e metodológicas para o projeto e a implementação de linguagens de programação, utilizando o autômato adaptativo como dispositivo formal para sua definição.

Como a especificação completa de uma linguagem de programação envolve diversos aspectos distintos em fases também distintas de projeto, é necessário encontrar algum modelo e/ou notação adequada que possa ser utilizada para todos esses aspectos. Esses variados aspectos vão desde a compreensão adequada de princípios e fundamentos comuns entre todas as linguagens de programação, transparentes ao programador, até as suas formas e características externas.

Embora muitos modelos e notações possam ser utilizados na formalização de diferentes aspectos envolvidos no projeto e na implementação das linguagens de programação, o autômato adaptativo demonstra alta aplicabilidade e adequação para uma definição completa da linguagem, sem a necessidade do uso de diferentes notações.

Foi demonstrado como os autômatos adaptativos podem ser utilizados como uma metalinguagem unificada para especificar todas as componentes relevantes da definição formal da linguagem de programação, tais como: análise léxica, reconhecimento da sintaxe livre de contexto e manipulação de alguns aspectos dependentes de contexto da linguagem - declaração e uso de nomes simbólicos, semântica estática, declaração e expansão de macros, entre outros. Foram apresentados os conceitos relacionados, e descritos os aspectos mais importantes da formalização proposta. Para isso, utiliza-se uma linguagem imperativa simplificada, sobre a qual é acoplado um mecanismo de extensão para torná-la extensível.

Foram realizados diversos experimentos usando o ambiente *AdapTools* o que possibilitou uma compreensão mais adequada dos conceitos e a sua aplicacação em cursos de conceitos de linguagens de programação.

COMUNICAÇÕES

Sistema CorPor: uma contribuição para o processamento da fala do português variante brasileira

Z. M. Zapparoli, Professora Associada, USP, Brasil, E. G. Cavalcanti, doutorando, USP, Brasil

Resumo — **Apoiando-se em áreas que partilham a crença nos resultados positivos advindos da interação entre Linguística e Informática, este trabalho insere-se na área da *Linguística Informática* – parte da utilização de *recursos da Informática na Linguística* para a geração do *Sistema CorPor*, que, por sua vez, oferece contribuições às áreas que se servem de *recursos da Linguística na Informática*, a exemplo do processamento automático da língua portuguesa. O *Sistema CorPor* inclui informações ortográficas e fonéticas do português falado no estado de São Paulo (Capital, Campinas, Itu), organizadas, relacionadas e armazenadas em função de anotações linguísticas e extralinguísticas. As informações são de fundamental importância para o desenvolvimento, treinamento e avaliação de sistemas de processamento da fala do português variante brasileira – reconhecimento e síntese –, sobretudo em se tratando de sistemas que utilizam recursos de aprendizagem de máquina através da construção de regras adaptativas.**

Palavras-chave — **Bases de Informações Ortográfico-Fonéticas do Português Falado de São Paulo (*Databanks of Phonetic and Orthographic Information about the Portuguese Language as Spoken in São Paulo*), *Corpora* Eletrônicos do Português Falado de São Paulo (*Electronic Corpora of the Portuguese Language as Spoken in São Paulo*), Linguística Informática (*Linguistic Informatics*), Processamento Automático da Língua Portuguesa (*Automatic Processing of the Portuguese Language*), Sistema CorPor (*CorPor System),* Sistema de Banco de Dados Relacional *(Relational Database System*), Tecnologias Adaptativas nos Estudos Linguísticos (*Adaptive Technologies in Linguistic Studies*).**

I. Introdução

Por envolver o uso de ferramentas informáticas, o trabalho insere-se na interface entre Linguística e Computação e, pois, em área multidisciplinar. Dedica-se à constituição de *Bases de Informações Ortográfico-Fonéticas do Português Falado de São Paulo*, a partir das quais podem ser gerados *corpora* digitalizados de textos orais em português do Brasil, para a sua exploração por recursos computacionais, para diferentes finalidades, como a geração de léxicos, o exame de padrões da língua oral, o processamento de línguas naturais.

Em arquitetura de banco de dados relacional, o Sistema CorPor reúne *Bases de Informações Ortográfico-Fonéticas, Corpora e Léxicos do Português Falado de São Paulo (São Paulo, Campinas, Itu)*, gerados, inicialmente, para a tese de doutorado (1980), em sistemas de computadores de grande porte, conforme em [1].

O armazenamento das bases de textos de língua oral no *Sistema CorPor* facilita a manipulação e o tratamento das informações, contribuindo para suprir a carência de *corpora* eletrônicos com transcrições ortográficas e fonéticas, e de conhecimentos linguísticos necessários ao desenvolvimento de sistemas de processamento da fala.

II. Pressupostos teórico-metodológicos

Numa dimensão mais ampla, o trabalho insere-se na área da *Linguística Informática*. A *Linguística Informática*, como linha de investigação científica, propõe-se, de um lado, à *utilização de recursos da Informática na Linguística* para o armazenamento, processamento e recuperação quantitativa e qualitativa de informações linguísticas; de outro, à *utilização de recursos da Linguística na Informática* para o desenvolvimento de sistemas que exigem equipes multidisciplinares, nas quais se incluem linguistas, como sistemas de tradução automatizada, sistemas de ensino de línguas naturais a distância, sistemas de produção e reconhecimento de línguas naturais.

Ainda, concebendo a *Linguística Informática* como abrangendo as diferentes áreas em que as tecnologias informatizadas estão relacionadas aos estudos da linguagem – *Linguística de Corpus, Linguística Computacional e Processamento de Língua Natural* –, a proposta enquadra-se mais particularmente nos propósitos da *Linguística de Corpus* em uma de suas preocupações, que constitui a condição *sine qua non* para a sua existência – construção de *corpora* eletrônicos a partir de textos e discursos reais. A *Linguística de Corpus* é vista, aqui, mais do que um simples instrumento de trabalho, por acreditarmos que o emprego das tecnologias informatizadas – base da *Linguística de Corpus* – na exploração de grandes quantidades de dados da língua em uso trará informações inéditas sobre as línguas naturais.

III. Procedimentos Metodológicos

A. *Constituição do Corpus de Língua Oral*

Os critérios rigorosos utilizados para a constituição do *corpus* de língua oral para fins do doutorado[1], mediante o controle de variáveis linguísticas – relativas às especificidades da língua falada – e variáveis extralinguísticas – região de origem, sexo, escolaridade, faixa etária, nível socioeconômico, condições extraverbais de interação dialógica – na seleção dos informantes e nos critérios de armazenamento dos dados,

[1] Zapparoli, 1980, v.1, t.1.

permitiram a obtenção de uma amostra representativa do português falado paulista, passível de ser objeto de estudo em diferentes áreas dos estudos da linguagem e de áreas afins. Trata-se de *corpus* compilado, também conhecido como *corpus* de amostragem, porque é fixo, uma vez que foi compilado através de amostras pré-selecionadas.

As amostras das falas dos informantes, recolhidas de 1972 a 1973, totalizam 54 horas de gravações entre documentador e 216 informantes paulistas (São Paulo, Campinas, Itu), de diferentes sexos, escolaridades, faixas etárias e níveis socioeconômicos, num total de 432 diálogos, visto que incluem dois tipos de interação dialógica – entrevistas e conversações.

B. Constituição do Corpus de Fala Transcrito para Tratamento Computacional

O registro[2] dos dados foi planejado para que eles pudessem ser armazenados e recuperados por sistemas computacionais.

O *Diagrama de Registro do Informante* (Fig. 1) mostra a anotação dos dados, a sua estruturação, os seus inter-relacionamentos e as muitas possibilidades de sua recuperação em função do interesse de estudo.

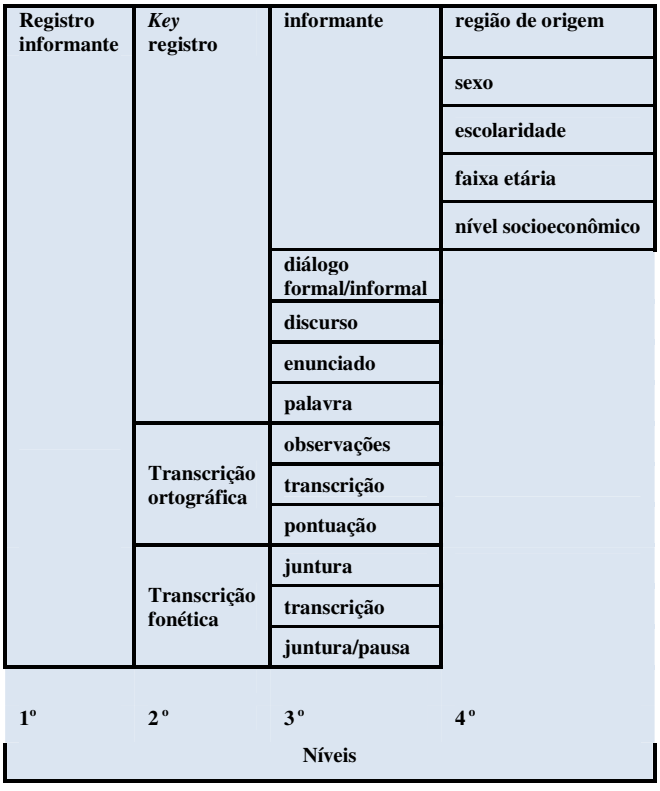


Fig.1. Diagrama de Registro do Informante

Trata-se, então, de *corpus* eletrônico anotado, que traz informações que permitem identificar as variáveis linguísticas (a palavra, a sua posição no enunciado, bem como a do enunciado no discurso, a sua transcrição ortográfica e fonética, o tipo de encontro fônico – juntura – que mantém com a palavra antecedente e com a subsequente) e extralinguísticas (região de origem, sexo, escolaridade, faixa etária, nível socioeconômico, condições de produção do diálogo), do que resulta um código exclusivo para cada item lexical, dentre cerca de 180 mil ocorrências.

A maneira como as informações estão codificadas e estruturadas confere às Bases funcionalidade, com possibilidades de extração de diferentes *corpora* e léxicos por variáveis linguísticas e extralinguísticas.

C. Sistema Gerenciador de Banco de Dados Relacional

As *Bases de Informações* estão armazenadas em Sistema de Banco de Dados e são manipuladas por meio do Sistema Gerenciador de Banco de Dados *Firebird*. A estrutura dos dados segue o modelo relacional, conforme Diagrama de Registro do Informante (Fig. 1), havendo uma correspondência entre os campos da tabela principal do banco de dados e os do diagrama. As Bases constituem, assim, uma coleção de dados ortográficos e fonéticos do português falado de São Paulo, organizados, relacionados e armazenados em função de anotações linguísticas e extralinguísticas, com as diferentes relações existentes entre os dados armazenados.

O ambiente de programação utilizado é o Delphi, produzido pela *Borland Software* Corporation, que utiliza a Linguagem Pascal com extensões orientadas a objetos (*Object Pascal*), associada a recursos da Linguagem Estruturada de Pesquisa (*Structured Query Language* – SQL) [2].

Além de recursos de pesquisa – para o acesso às informações das Bases –, o Sistema abrange recursos de um editor de textos – para os trabalhos de edição dos resultados das pesquisas às Bases de Informações.

IV. Principais Componentes do Sistema

A. Bases de Informações Ortográfico-Fonéticas do Português Falado de São Paulo

As Bases contêm com todas as informações de cada um dos 216 inquéritos pela ordem de registro de gravação e de acordo com os critérios linguísticos e extralinguísticos que foram controlados na seleção dos informantes que forneceram material linguístico para a constituição da amostra.

B. Corpora Eletrônicos do Português Falado Paulista

Também em função das variáveis linguísticas e extralinguísticas, os *corpora* oferecem variadas possibilidades de exploração por programas de análise lingüística, como em [3].

C. Léxico de Frequência Ortográfico-Fonético

O Léxico de Frequência Ortográfico-Fonético traz, para cada palavra em sua transcrição ortográfica, as correspondentes transcrições fonéticas, sem e com separação silábica, com anotação da frequência da unidade fonética e da frequência acumulada da unidade ortográfica.

[2] A palavra registro, aqui, é empregada no sentido de conjunto de informações transcritas.

D. Léxico de Junturas Intervocabulares

O Léxico de Junturas Intervocabulares inclui as categorias de juntura intervocabular – encontros fônicos lexicais que se dão nos limites de duas ou mais fronteiras de palavras –, a transcrição ortográfica das ocorrências de juntura intervocabular com a correspondente transcrição fonética silábico-lexical e a combinatória acentual das sílabas intervocabulares.

Pelas limitações de espaço de um artigo, estendemo-nos, aqui, na apresentação das Bases, visto que elas são o suporte para a geração dos demais componentes.

As *Bases de Informações Ortográfico-Fonéticas do Português Falado de São Paulo* contêm todas as informações de cada um dos 216 informantes, num total de 432 inquéritos, visto que incluem, para cada informante, dois tipos de interação dialógica – entrevistas e conversações. As informações estão organizadas pela ordem de registro de gravação e de acordo com os procedimentos de anotação e de estruturação adotados.

Além da transcrição ortográfica e da transcrição fonética de cerca de 180 mil registros de itens lexicais, as *Bases* incluem anotações relativas a variáveis linguísticas (especificidades da língua oral, categorias de encontros fônicos intervocabulares) e a variáveis extralinguísticas que foram controladas na seleção dos 216 informantes que forneceram material linguístico para a constituição da amostra (região de origem, sexo, escolaridade, faixa etária e nível socioeconômico) e na produção dos diálogos (formal e informal). Ou seja, as Bases trazem a informação lexical organizada em função de relações com dados linguísticos e extralinguísticos, o que permite diferentes possibilidades combinatórias.

A Tabela I traz uma amostra das Bases.

TABELA I. BASES DE INFORMAÇÕES ORTOGRÁFICO-FONÉTICAS DO PORTUGUÊS FALADO DE SÃO PAULO

Chave[1]	*Código Lexical*[2]	*Obs.*[3]	*Transcrição Ortográfica* [4]	*Pont.*[5]	*J / SI*[6]	*Transcrição Fonética* [7]	*J SF/ P*[8]
126	10111100302001		chegamos			$& 'G9 MU	101
127	10111100302002		na		101	NA	101
128	10111100302003	6	França	4	101	'F>@ S	5
129	10111100302004		aquele		5	A 'K& LI	101
130	10111100302005		problema		101	P>O 'BL7 M	5
131	10111100302006		assim	2	5	A 'S1	1
132	10111100302007		a			A	101
133	10111100302008		guerra		101	'GE X	5
134	10111100302009		ainda		5	A '1 DA	101
135	10111100302010		está		101	'T	5
136	10111100302011		ali		5	A 'LI	101
137	10111100302012		presente	4	101	P>& 'Z3) TI	101
138	10111100302013		sabe	9	101	'SA BI	1
139	10111100302014		então	4		1 'T@%	101
140	10111100302015		você		101	'S&	37
141	10111100302016		entra		37	'3) T>A	101

[1] Ordem
[2] Codificação para identificação do item lexical – informante, tipo de diálogo, discurso, enunciado e palavra
[3] Codificação para desvios léxico-morfossintáticos, siglas, nomes próprios, palavras estrangeiras
[4] Transcrição ortográfica
[5] Codificação para pontuação
[6] Codificação para juntura sílaba inicial
[7] Transcrição fonética [4]
[8] Codificação para juntura sílaba final / pausa real

Segue, a título de exemplificação, recorte discursivo extraído das Bases.

Código Lexical: 1011211 – Informante de São Paulo (1), do sexo feminino (0), com curso superior completo (1), 30 a 34 anos (12), classe alta alta (1), registro formal de interação dialógica (1):

Assim, eu... eu acho... eu... o indivíduo, quando escolhe a profissão por... por escolha, independente de influência de qualquer indivi/ qualquer pessoa, tem, ahn..., muito mais possibilidade de realizar se dentro do campo que escolheu. Por exem/ eu acho que dé/ dentro do sta/ do status atual, biblioteconomia é um campo altamente explorável, com boa remuneração econômica e com grande, ah, possibilidade de atividades e especializações; é um campo novo, com poucos especialistas, ih, dentro da o... dentro de São Paulo; quase todos eles são englobados pela Universidade de São Paulo. Acho assim, por exemplo, no momento, nós contamos, aqui na faculdade, com oito bibliotecários, todos de curso superior, dos quais quatro têm especialização em ciências biomédicas — inclusive eu tenho especialização—. Ah o ambiente de trabalho é ideal; não sei, porque não conheço, uhn..., éh..., nenhum; trabalhei um... durante dois anos como bibliotecária da... tsi... do Conselho Regional de Contabilidade do Estado, mas era uma biblioteca independente, com um único profissional; então, você não pode avaliar bem a... o relacionamento; isso eu vim sentir mais aqui na universidade; eu acho um campo... por ser um campo muito novo, todo profissional é muito unido; acho ideal, um trabalho muito bom, e nós trabalhamos aqui, em equipe; embora nós tenhamos todas nós setores bem definidos, pela carga de trabalho ser muito grande, nós todas trabalhamos em comum acôrdo, a ponto de podermos qualquer uma substituir a outra, a qualquer momento. É, eu acho que deu, assim, uma amplitude de trabalho muito grande, o que, muitas vezes, não se verifica em outros campos, né?; nós, graças a Deus, não tivemos esse problema.

V. CONTRIBUIÇÕES

Voltada a aspectos pouco explorados nos estudos linguísticos – se são raros, no Brasil, os *corpora* eletrônicos de transcrições de fala, mais ainda o são os *corpora com* transcrições fonéticas –, os resultados da investigação podem oferecer contribuições e benefícios: no âmbito da Linguística, pelo oferecimento de *corpora* digitalizados de textos autênticos da língua oral paulista para o desenvolvimento de estudos diversos; na interface entre a Linguística e a Informática, pelo oferecimento de conhecimentos linguísticos para o desenvolvimento, treinamento e avaliação de sistemas de processamento da fala do português variante brasileira – reconhecimento e síntese –, uma das áreas de maior complexidade do Processamento de Línguas Naturais.

Estamos certos de que o êxito do processamento de línguas naturais depende tanto do avanço tecnológico como de novos conhecimentos linguísticos. A tarefa que nos cabe, como linguistas e falantes da língua portuguesa como língua

materna, consiste em oferecer contribuições para a aquisição de novos conhecimentos do português. Nesse sentido, o *Sistema CorPor*, que armazena as *Bases* em formato específico de Banco de Dados Relacional, oferece a estudiosos do português facilidade, rapidez e confiabilidade na pesquisa (consulta), na recuperação (acesso) e no tratamento (exploração) automáticos de extensos e variados dados do português falado paulista para o desenvolvimento de estudos de aspectos diversos da língua – fonéticos, fonológicos, lexicais, morfológicos, sintáticos, textuais e discursivos.

No que diz respeito a avanços na área da computação, destacamos que, em se tratando de sistemas com base em tecnologias adaptativas, os ganhos são significativos pela possibilidade de reconhecimento automático de padrões da língua oral paulista e, pois, pelo oferecimento de uma Base de Conhecimentos, indispensável na arquitetura de um sistema de processamento de língua natural.

VI. Considerações Finais

Para concluir, retomamos a referência inicial que fizemos à área da Linguística Informática. Neste trabalho de movimento duplo entre a Linguística e a Informática, de um lado, ressaltamos que as vantagens da utilização das Novas Tecnologias Digitais nas pesquisas linguísticas que desenvolvemos são indiscutíveis; de outro, vislumbramos resultados positivos de uma convergência do *Sistema CorPor* com a área da Inteligência Computacional, através de uma conexão do formalismo já desenvolvido a mecanismos adaptativos, para a geração de uma Base de Conhecimentos da língua oral paulista.

Expressamos o convite a estudiosos interessados no desenvolvimento dessa empreitada.

Agradecimentos

Agradeço a Manoel Vidal Castro Melo a assessoria em análise e programação para o desenvolvimento do Sistema em *Mainframe* e a Edenis Gois Cavalcanti, para a criação do Sistema em PC.

Referências

[1] Z. M. Zapparoli Castro Melo, "Análise do comportamento fonológico da juntura intervocabular no português do Brasil (variante paulista). Uma pesquisa linguística com tratamento computacional", Tese de Doutorado orientada por Francis Henrik Aubert, Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo, 1980.

[2] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Boston: Addison-Wesley, 1998.

[3] Z. M. Zapparoli e A. Camlong, *Do léxico ao discurso pela informática*. São Paulo: EDUSP/FAPESP, 2002.

[4] International Phonetic Association, *Handbook of the International Phonetic Association*. Cambridge: Cambridge University Press, 1999.

Zilda Maria Zapparoli nasceu em Itu, São Paulo, Brasil, em 2 de agosto de 1945. É professora associada aposentada junto ao Departamento de Linguística da Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo (FFLCH-USP), instituição em que obteve os títulos de Mestre, Doutor e Livre-Docente, e onde continua desenvolvendo atividades de ensino, pesquisa e orientação no Curso de Pós-Graduação em Linguística, área de Semiótica e Linguística Geral. É Bolsista de Produtividade em Pesquisa do CNPq e líder do Grupo Interdisciplinar de Pesquisas em Linguística Informática. Tem mais de trinta anos de atuação em Linguística Informática, com tese de doutorado, tese de livre-docência e trabalhos publicados na área. Integrou comissões e colegiados na USP, destacando-se os trabalhos relativos ao processo de informatização da FFLCH-USP, enquanto Membro da Comissão Central de Informática da USP e Presidente da Comissão de Informática da FFLCH-USP por cerca de treze anos.

Edenis Gois Cavalcanti nasceu em São Paulo, Brasil, em 2 de fevereiro de 1962. Possui graduação em Filosofia pela Faculdade de Filosofia Nossa Senhora Medianeira (1986) e mestrado em Semiótica e Linguística Geral pela Universidade de São Paulo (2005). Atualmente, desenvolve projeto de doutorado – Construção de Software para o Estudo da Língua Grega Clássica: o Sistema Nominal do Dialeto Ático na Visão Temática – pelo Departamento de Linguística, USP, área de Linguística Informática. É desenvolvedor de *software* nas linguagens Delphi e C# .NET (Visual Studio), plataformas Desktop e Web, integradas com bando de dados FireBird, MySql e Sql Server. Tem implantado – como metodologia de trabalho em sala de aula, rede municipal de ensino, São Paulo, Capital – projeto de avaliação *on-line*, SIGA WEB .NET (em www.fcavalcanti.com), no qual os alunos, com acesso totalmente gratuito, consultam conteúdos e realizam suas avaliações via *Internet*.

The *Corpor* System: a contribution to electronic speech processing for the Brazilian variety of Portuguese

Zilda Maria Zapparoli (zmz@usp.br)*, Edenis Gois Cavalcanti (edenis@usp.br)#

* Grupo Interdisciplinar de Pesquisas em Linguística Informática, Departamento de Linguística, Universidade de São Paulo – SP – Brazil.

Grupo Interdisciplinar de Pesquisas em Linguística Informática, Departamento de Linguística, Universidade de São Paulo – SP – Brazil.

Abstract

This study belongs in the field of Linguistic Informatics, drawing support from the various areas that share the belief in the positive results of the interaction between Linguistics and Informatics. It makes use of Informatics resources in Linguistics studies in order to build the Corpor System which, in turn, can offer a contribution to the areas that use Linguistics in Computer Sciences, such as the automatic processing of the Portuguese language. The CorPor System includes orthographic and phonetic information about the Portuguese language as spoken in the State of São Paulo (São Paulo City, Campinas, Itu), organized, listed and stored taking into account linguistic and extralinguistic annotations. This information is particularly important for the development, testing and evaluation of speech processing systems for the Brazilian variety of the Portuguese language – recognition and synthesis –, especially in systems that use machine learning resources with the construction of adaptive rules.

Index Terms

databanks of phonetic and orthographic information about the Portuguese language as spoken in São Paulo, electronic corpora of the Portuguese language as spoken in São Paulo, linguistic informatics, automatic processing of the Portuguese language, corpor system, relational database system, adaptive technologies in linguistic studies.

Proposta de aplicação de um algoritmo genético para solução do problema do corte bidimensional em lâminas de vidro

F. Costa, N. C. F. Canto, R. J. Sassi

***Resumo*— O projeto consiste no estudo e desenvolvimento de métodos de solução, baseados em computação evolutiva, para Problemas de Satisfação de Restições (PSR), os quais abrangem uma variedade de problemas de otimização combinatória. Em particular, é proposto o desenvolvimento de um algoritmo genético para solução e otimização do problema de corte de lâminas de vidro, o qual consiste em determinar o arranjo de peças a serem cortadas maximizando a utilização da lâmina e respeitando as restrições impostas pelo fluxo de produção e cronogramas de entregas.**

***Palavras-chave*— Problemas de Satisfação de Restrição, Algoritmo Genético, Problemas de Corte Bidimensional.**

IV. INTRODUÇÃO

Os PSR são caracterizados por uma estrutura comum que consiste em encontrar um conjunto de objetos cujo estado precisa satisfazer um número de restrições ou limitações [1][2]. Os PSR frequentemente apresentam alta complexidade e requerem uma combinação de heurísticas e métodos de busca para que possam ser solucionados em um tempo razoável (complexidade de tempo polinomial) [3]. São vários os sistemas de produção que envolvem os PSR, alguns exemplos são: alocação de tarefas; problemas logísticos de empacotamento e processos de corte de materiais [4][5]. Devido ao potencial das aplicações práticas para otimizar processos industriais e às dificuldades para obtenção de soluções exatas, os PSR têm sido objetos de intensas pesquisas nas áreas Pesquisa Operacional (PO) e Inteligência Artificial (IA), uma vez que o estudo de tais problemas fornece uma base comum para análise e solução de outros problemas que pertencem à mesma categoria. Neste contexto, a computação evolutiva tem ganhado espaço no cenário acadêmico e sendo utilizada cada vez mais na solução de tais problemas. A computação evolutiva compreende um grupo de métodos computacionais que incorporam mecanismos evolutivos presentes na natureza e descritos pela teoria da evolução natural [6][7]. Como exemplo, tem-se os algoritmos genéticos (AG) os quais têm se mostrado uma ferramenta promissora para análise e solução dos PSR [8]. O projeto de Iniciação Científica em desenvolvimento visa o estudo e aperfeiçoamento de métodos de solução, baseados em computação evolutiva, para solução de PSR oriundos de sistemas de produção. Especificamente é proposto o desenvolvimento de um algoritmo genético para solução do problema de corte bidimensional em lâminas de vidro.

V. PROBLEMAS DE CORTE

Problemas de corte são uma importante classe entre os problemas de satisfação de restrição [9][10]. Em particular, o problema do corte bidimensional consiste no arranjo e corte de peças de diferentes tamanhos e formatos com o objetivo de minimizar as perdas de material. As abordagens tradicionais para o corte bidimensional podem ser divididas em duas categorias: programação linear (PL) e métodos heurísticos (MH), sendo os métodos heurísticos os mais populares na solução de problemas reais. Entretanto, a utilização de Algoritmos Genéticos (AG) na solução de problemas de restrição tem crescido e a técnica tem se tornado uma importante ferramenta para modelagem e solução dessa classe de problemas.

III. O CORTE EM LÂMINAS DE VIDRO

O trabalho propõe o desenvolvimento de um AG para a solução de um problema real de uma empresa de corte em lâminas de vidro. O problema consiste em maximinar a utilização das lâminas padrão. As lâminas padrão têm o formato 6000 x 3210 (mm) e são utilizadas para produção de peças retangulares de diferentes dimensões. Atualmente, os leioutes de corte são produzidos por *softwares* que acompanham as máquinas operatrizes, estes *softwares* oferecem um conjunto de algoritmos de otimização baseados em PL e MH e que produzem resultados que variam, dependendo do número de lâminas utilizadas e peças a serem cortadas em determinado pedido, mas que, em média, ficam em torno de 5% do total de material utilizado. A tabela I apresenta um exemplo de pedido encaminhado à produção e a tabela II apresenta valores típicos de desperdício gerado por esses algoritmos.

TABELA I
EXEMPLO DE PEDIDO ENCAMINHADO PARA PRODUÇÃO

Cód. do Produto	Medidas (mm)	Quantidades
10010	720x220	27
10020	515x2150	43
10050	950x1410	15

TABELA II

VALORES TÍPICOS DE DESPERDÍCIO DE MATERIAL PRODUZIDOS POR *SOFTWARES* PROPRIETÁRIOS DE MÁQUINAS OPERATRIZES PARA CORTE DE LÂMINAS DE VIDRO

Lâmina	Área da Lâmina (m^2) – (A)	Desperdício
1	19,26	5,86%
2	19,26	3,08%
3	19,26	3,94%
4	19,26	5,50%

O objetivo é comparar os resultados do AG proposto com os resultados gerados pelos algoritmos do sistema e verificar se existirão ganhos significativos com a utilização de um algoritmo evolutivo. A seção seguinte apresenta os detalhes da arquitetura e codificação do AG proposto.

IV ALGORITMO GENÉTICO PARA OTIMIZAÇÃO DE CORTE EM LÂMINAS DE VIDRO

Inicialmente o AG proposto consistirá de uma versão que selecionará, a partir do pedido solicitado à produção, um número *n* de peças com o objetivo de minimizar as sobras na lâmina de vidro padrão. No problema em questão, trabalha-se com um grande número de peças a serem cortadas e o objetivo é encontrar uma combinação de peças, extraída do pedido, que minimize a sobra de material em cada lâmina. A tabela III mostra o exemplo de um pedido encaminhado para a produção contendo apenas 10 itens e quatro tipos de peças a serem cortadas.

TABELA III
EXEMPLO DE PEDIDO DE PEÇAS PARA CORTE

Peça	Quantidade	Código da Peça	Dimensões
0	3	5010	720x220
1	2	5020	515x2150
2	4	5050	950x1410
3	1	5090	250x1850

O pedido descrito na tabela III será representado por um vetor *p* que será utilizado para geração da população inicial do AG.

$$p = (a_0, a_0, a_0, a_1, a_1, a_2, a_2, a_2, a_2, a_3) \quad (1)$$

A. *População inicial*

Os passos seguintes descrevem o processo de geração da população inicial do AG. A população é formada por candidatos a solução do problema, ou seja, arranjos de peças extraídas do pedido p.

1. Criar uma cópia (p') do pedido p para manipulação;

2. Sortear aleatoriamente um item *i* do pedido p';

3. Remover o item sorteado *i* de p' e calcular
   $S = A - a_i$ (2) sendo:
   A a área da lâmina padrão;
   S a área ainda não utilizada;
   a_i a área da peça *i* sorteada.

4. Verificar se S>0 então colocar a peça a_i no novo indivíduo;

5. Repetir o processo até que o teste acima falhe (S<=0) ou que p' esteja vazio (não é possível encaixar mais nenhuma peça sem ultrapassar a área total da lâmina);

6. repetir os itens de 1 a 5 para gerar os *n* indivíduos da população;

Desta forma, teremos *n* indivíduos contendo um número variado de peças e respeitando a restrição de que a soma das áreas de todas as peças não ultrapasse a área da lâmina.

B. *Codificação dos indivíduos*

Neste exemplo, como o pedido só possui 10 itens, cada peça será codificada com 3 bits, 2 bits para representar o número da peça e 1 bit para representar a posição (em pé ou deitada):

TABELA IV
CODIFICAÇÃO DAS PEÇAS PRESENTES NOS INDIVÍDUOS

Peça	Codificação 1 (em pé)	Codificação 1 (deitada)
0	000	001
1	010	001
2	100	001
3	110	001

Alguns exemplos de indivíduos:
Indivíduo 1 - 010011100 indivíduo formado por 3 peças (peça 1 (em pé), peça 1 (deitada) e peça 2 (em pé).

Indivíduo 2 - 110101011000 indivíduo formado por 4 peças (peça 3 (em pé), peça 2 (deitada), peça 1 (deitada) e peça 0 (em pé).

Portanto, após gerar os *n* indivíduos no item 4, os indivíduos serão codificados para o formato binário e sorteado o último bit representando a posição da peça. Essa representação facilitará a aplicação dos operadores genéticos *crossover* e a mutação.

C. *Fitness*

O cálculo do *fitness* dos indivíduos da população será realizado com o auxílio do algoritmo de posicionamento de peças denominado *bottom-left* [11][12] desenvolvido em Java. O processo consiste em encaixar as peças do indivíduo na lâmina e calcular a área não utilizada da lâmina.

$$f(n) = A - \sum_{1}^{x} a_i \quad (3) \quad \text{Sendo:}$$

f(n) – o *fitness* do indivíduo *n*;

A – área da lâmina padrão;
a_i – área de cada peça que compõe o indivíduo;
x - o número de peças do indivíduo

D. Seleção

O método de seleção de indivíduos utilizado no AG proposto é denominado de Roleta (*roulette wheel*). Neste método, atribui-se a cada indivíduo uma probabilidade proporcional ao *fitness* relativo de cada indivíduo na população.

$$p_{selec} = \frac{f(n)}{\sum_{i=1}^{N} f(n_i)} \quad (4)$$

sendo:

p_{selec} = probabilidade do indivíduo *n* de ser selecionado;

$f(n)$ = o *fitness* do indivíduo *n*, dado pela equação (3);

$\sum_{i=1}^{N} f(n_i)$ = o somatório do *fitness* de todos os indivíduos.

E. Crossover

No algoritmo proposto utiliza-se o operador *crossover* de um ponto com uma taxa fixa de probabilidade de ocorrência entre os indivíduos. Para este operador, são selecionados dois indivíduos (pais) que dão origem após a recombinação a dois novos indivíduos (filhos). A figura 1 ilustra o processo de *crossover* aplicado a um par de indivíduos [ ].

				Ponto de corte								
Pai 1	0	1	0	0	1	1	1	0	0			
Pai 2	1	1	0	1	0	1	0	1	1	0	0	0
Filho 1	0	1	0	1	0	1	0	1	1	0	0	0
Filho 2	1	1	0	0	1	1	1	0	0			

Fig. 1. Exemplo de *crossover* entre dois indivíduos da população.

Uma heurística é implementada ao processo de escolha do ponto de corte para evitar a formação de indivíduos que não atendam as restrições de área da lâmina padrão.

F. Mutação

O objetivo do operador mutação é inserir diversidade na população de indivíduos sem desestabilizá-la. Por isso, a taxa de mutação utilizada é pequena 1% do total de genes da população.

V RESULTADOS ESPERADOS

Espera-se que após um número *n* gerações o algoritmo genético proposto seja capaz de extrair do pedido original um número *k* de peças e alocá-las de forma otimizada na lâmina padrão. A figura 2 apresenta um exemplo de leiaute que deverá ser gerado pelo algoritmo em desenvolvimento.

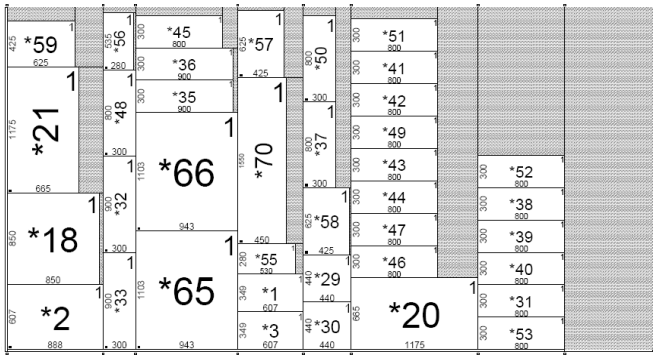


Fig. 2. Exemplo de leioute produzido pelo algoritmo genético.

O modelo apresentado consiste da primeira versão do AG e os resultados obtidos irão orientar os próximos passos da pesquisa com possíveis alterações e melhorias no algoritmo com o objetivo de melhorar a performance do modelo.

REFERÊNCIAS

[1] C. BRODERICK, A. MARY, C. CARLOS e M. ERIC, "Using Constraint Programming to solve Sudoku Puzzles", *Third 2008 International Conference on Convergence and Hybrid Information Technology*, 926-931. 2008.

[2] B. CRAWFORD, M. ARANDA, C. CASTRO e E. MONFROYUSING, "Constraint Programming to solve Sudoku Puzzles", Third International Conference on Convergence and Hybrid Information Technology. pp. 926-931, 2008.

[3] A. N. BERNARD, "Representation Selection for Constraint Satisfaction: A Case Study Using n-Queens", *IEEE Expert*, 885, 16-23, 1990.

[4] H. BAUMGÄRTEL, D. CHRYSLER, "Distributed Constraint Processing for Production Logistics", *IEEE Inteligent System*, 40-48, 2000.

[5] W. SHEN, "Distributed Manufacturing Scheduling Using Intelligent Agents", *IEEE Inteligent System*. pp. 88-94, 2002.

[6] J. H. HOLLAND, "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor, MI, 1975.

[7] J. R. KOZA, "Genetic Programming: On the Programming of Computers by Means of Natural Selection". MIT Press, 1992.

[8] W. BANZHAF, J. R. KOZA, C. RYAN, L. SPECTOR, C. JACOB, "Genetic programming,", *Intelligent Systems and Their Applications* IEEE, 15, 3, 74 – 84, 2000.

[9] E. HOPPER, B. TURTON, "A Genetic Algorithm for a 2D Industrial Packing Problem", Computers and Industrial Engineering, Cardiff, v. .37, n. 1, p. 375-378, 1999.

[10] H. DYCKHOFF, "A typology of cutting and packing problems", "*European Journal of Operational Research*, 44(2), 145–159, 1990.

[11] T. F. GONZALES, "Handbook of Approximation Algorithms And Metaheuristics", Chapman & Hall/CRC, 2007.

[12] C. COTTA, M. SEVAUX, K. SÖRENSEN, Adaptive and Multilevel Metaheuristics, Kenneth, 2008.

Flávio Moreira da Costa: Técnico em informática e redes de comunicação pela Escola SENAI "Suiço-brasileira" (2005), atualmente é estudante do 4º ano do curso de Ciência da Computação pela

Universidade Nove de Julho. Participa de em projetos de iniciação científica relacionado a Inteligência Artificial (Algoritmos Genéticos). Tem experiência nas áreas de Análise e Desenvolvimento de Sistemas Computacionais voltados para Gestão Administrativa e de Produção.

Nilton Canto: graduado em Física pela Universidade de São Paulo (1997), concluiu o mestrado em Engenharia Elétrica pela Universidade Presbiteriana Mackenzie em 2002 e o doutorado em Engenharia Elétrica pela Universidade de São Paulo em 2008. É membro da sociedade Brasileira de Computação (SBC), membro do Institute of Electrical and Electronics Engineers (IEEE) e membro da IEEE Computer Society. Atualmente é professor da Universidade Nove de Julho, do Centro Universitário Unisantanna e do Colégio Etapa. Possui mais de 30 softwares educativos desenvolvidos e em suas atividades profissionais interagiu com cinco colaboradores em co-autorias de trabalhos científicos. Em seu currículo Lattes os termos mais freqüentes na contextualização da produção científica, tecnológica e artístico-cultural são: Córtex Visual, Sincronismo, Osciladores Acoplados, Ferramentas de Autoria, Sistemas Dinâmicos, Softwares Educativos, Tutores Inteligentes, Algoritmos Adaptativos, Algoritmos Genéticos e Sistemas Cognitivos.

Renato José Sassi: bacharel em Ciências Econômicas pela Faculdade de Economia Finanças e Administração de São Paulo (1987), especialista (pós-graduação Lato Sensu) em Administração de Empresas (área de concentração Análise de Sistemas) pela Fundação Escola de Comércio Álvares Penteado (1988), especialista (pós-graduação Lato Sensu) em Didática do Ensino Superior pelo Centro Universitário Sant'anna (1996), mestre em Administração de Empresas (Gestão de Negócios) pelo Centro Universitário Sant'Anna (1999) e doutor em Engenharia Elétrica pela Escola Politécnica (POLI) da Universidade de São Paulo (2006). Atualmente é pesquisador associado do Grupo de Inteligência Computacional, Modelagem e Neurocomputação (ICONE) do Laboratório de Sistemas Integráveis (LSI) da Escola Politécnica da Universidade de São Paulo (USP), docente e pesquisador do Programa de Mestrado em Engenharia de Produção da Universidade Nove de Julho (UNINOVE). Membro do Institute of Electrical and Electronics Engineers (IEEE) e membro da Computational Intelligence Society (IEEE).

Roteirização dinâmica de veículos combinada à Previsão do comportamento do tráfego urbano utilizando uma rede *neuro fuzzy* (22 Janeiro 2010)

R. P. Ferreira, C. Affonso, R. J. Sassi, membro *IEEE*

***Resumo*— O desafio de conquistar e manter clientes impulsiona o desenvolvimento de novas formas de atender aos anseios de consumo cada vez mais tendendo à micro segmentação de produto e mercado. O objetivo deste trabalho é subsidiar o desenvolvimento de um sistema de roteirização dinâmico apoiado pela previsão do comportamento do tráfego na região urbana de São Paulo utilizando uma rede *neuro fuzzy* treinada com ocorrências notáveis verificadas no tráfego urbano na Zona de Máxima Restrição de Circulação (ZMRC) da cidade de São Paulo. A metodologia do trabalho consiste no estudo da cidade de São Paulo, mais precisamente na (ZMRC), região metropolitana de grande concentração de comércio e empresas. Foi realizado um levantamento diário sobre o comportamento do trânsito cotidiano da grande São Paulo. Este trabalho inicial apresentará o resultado parcial da utilização da rede *neuro fuzzy* na previsão do comportamento do tráfego.**

***Palavras chave*— Previsão do Comportamento do Tráfego; Rede *Neuro Fuzzy*; Roteirização de veículos.**

I. INTRODUÇÃO

Os novos hábitos de consumo dos brasileiros trouxeram ao mercado produtos com ciclo de vida mais curto, conseqüentemente volumes crescentes de itens fora de uso e sem destino certo após o uso [1]. As vendas pela *internet* ganharam espaço e as reduções dos estoques aumentaram a freqüência das entregas, e a responsabilidade da entrega em prazos cada vez mais justos, obedecendo a janelas de tempo, geralmente rígidas na distribuição e flexíveis na coleta e exigem soluções inteligentes que com o uso das novas tecnologias disponíveis no mercado apresentem formas alternativas de cumprir a missão de cruzar a região metropolitana e realizar o papel logístico de transportar os mais variados produtos e entregá-los dentro do prazo acordado com os clientes. O cenário implica em maior nível de eficácia na logística do transporte terrestre urbano, nas regiões urbanas de São Paulo a exigência por níveis de serviço regulares e confiáveis tornam a competitividade bastante acirrada entre as *couriers*. A roteirização eficiente de veículos aparece como uma oportunidade de atender adequadamente os anseios dos clientes. A roteirização e programação de veículos tendem a receber um enfoque destacado, observado as exigências do mercado e do cliente final dentro da cadeia de suprimentos e logística de distribuição. Por isso verifica-se a preocupação em criar ferramentas que otimizem o tempo de utilização de veículos, as restrições de circulação e de capacidade dos veículos reduzem a produtividade, dessa forma os veículos quando em operação livre devem ter seu potencial maximizado.

II. FUNDAMENTOS BÁSICOS DA ROTEIRIZAÇÃO E PROGRAMAÇÃO DE VEÍCULOS

Segundo Bodin et al., [2], os problemas de roteirização podem ser classificados inicialmente em três grupos principais:

a) Problemas de roteirização pura de veículos (PRV): neste tipo de problema as condicionantes temporais não são consideradas na geração dos roteiros para coleta e/ou entrega, sendo que em alguns casos pode-se considerar a restrição de comprimento máximo do arco. O problema de roteirização busca definir a seqüência de pontos de parada que cada veículo deve seguir buscando a minimização de custos do transporte;

b) Problemas de programação de veículos e tripulações (PRVT): tipo de problema de roteirização onde as condicionantes temporais devem ser consideradas, ou seja, as restrições adicionais relacionadas aos horários em que as atividades devem ser executadas devem ser consideradas no tratamento do problema;

c) Problemas combinados de roteirização e programação: é uma extensão do problema da roteirização do veículo, onde restrições realistas são incluídas, restrições estas tais como janelas de tempo, precedência de tarefas e alocação da tripulação.

O transporte representa normalmente entre um e dois terços dos custos logísticos totais; por isso mesmo, aumentar a eficiência por meio da máxima utilização dos equipamentos e pessoal de transporte é uma das maiores preocupações do setor. O tempo que as mercadorias passam em trânsito tem reflexos no número de fretes que podem ser feitos por veículo num determinado período de tempo e nos custos integrais do transporte para todos os embarques. Reduzir os custos do transporte e melhorar os serviços ao cliente, descobrir os melhores roteiros para os veículos ao longo de uma rede a fim de minimizar os tempos e as distâncias constituem problemas muito freqüentes de tomada de decisão [3].

Os algoritmos de roteirização representam o nível operacional do sistema proposto, conforme será visto na seção VI.

III. A REDE *NEURO FUZZY* PARA PREVISÃO DO COMPORTAMENTO DE TRÁFEGO URBANO

Atualmente, existe grande interesse nos modelos de redes neurais para resolver problemas não convencionais, nos últimos anos às redes neurais artificiais têm surgido como uma alternativa viável e com inúmeras aplicações. Uma rede *neuro fuzzy* foi desenvolvida utilizando MLP (*Multilayer perceptrons*) arquitetura com *backpropagation* para o aprendizado do algoritmo. Foram colhidos dados de ocorrências notáveis do tráfego na região metropolitana de São Paulo do dia 15 de Dezembro de 2009 com o objetivo de obter o impacto dessas ocorrências na fluidez do tráfego através de ocorrências relevantes sobre o comportamento do tráfego, esses parâmetros foram convertidos através da função de membro específico em conjuntos *Fuzzy*. A tabela 1 apresenta os tipos de ocorrências registradas que foram utilizadas na rede *neuro fuzzy* (50 épocas).

Tabela 1

Ocorrências Notáveis	
1.	Acidente com vítima
2.	Alagamento
3.	Atropelamento
4.	Caminhão quebrado
5.	Defeito na rede de trólebus
6.	Falta de energia elétrica
7.	Incêndio
8.	Incêndio em Veículos
9.	Manifestações
10.	Ocorrência envolvendo carga
11.	Ocorrência envolvendo carga perigosa
12.	Ônibus imobilizado na via
13.	Queda de árvore
14.	Semáforo apagado
15.	Semáforo embandeirado
16.	Veículo com excesso

Fonte: Adaptado de CET, 2009

A figura 1 ilustra os resultados reais do comportamento do tráfego, a linha azul representa o comportamento real em 15 de Dezembro de 2009, a linha verde representa a média inferior e a linha vermelha representa a média superior por horários. O gráfico de lentidão apresenta o índice de lentidão do trânsito registrado a cada 30 minutos, de segunda à sexta, no horário das 7h às 20h, bem como as linhas que indicam o limite inferior e superior [4].

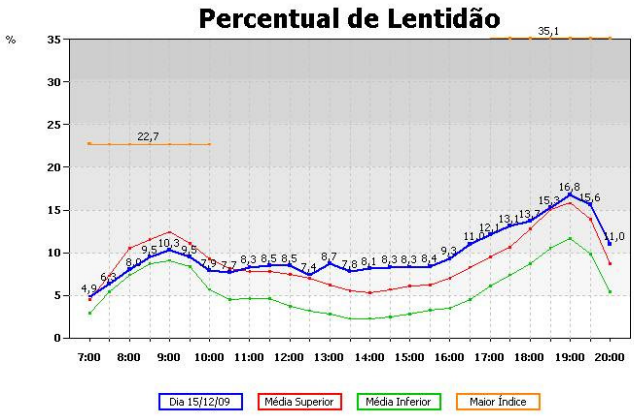


Fig. 1. Comportamento observado em 15.12.2009. Fonte: CET, 2009

A figura 2 ilustra os resultados obtidos pela rede *neuro fuzzy* comparados aos colhidos pela CET em 15 de Dezembro de 2009.

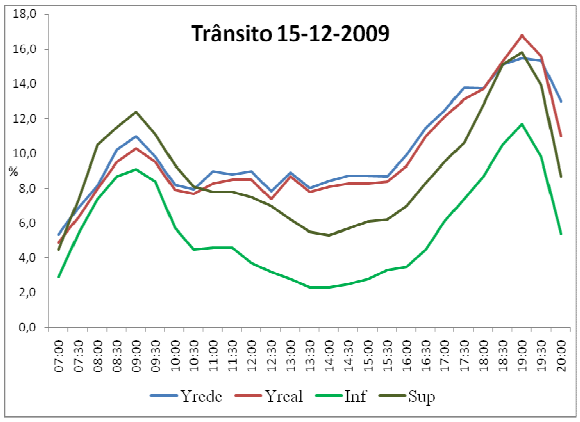


Fig. 2. Comportamento calculado pela rede *neuro fuzzy*. Fonte: Autores

Com os resultados iniciais verifica-se que a rede obteve um resultado razoável do problema proposto, podendo auxiliar na tomada de decisão quanto as janelas de tempo que devem ser evitadas e que apresentam o comportamento acima da normalidade.

A previsão do comportamento de tráfego urbano em São Paulo representa o nível estratégico da roteirização, conforme será visto na seção VI.

IV. A ROTEIRIZAÇÃO DINÂMICA

Na cidade de São Paulo basta um veículo parar em uma via movimentada para produzir imediata diminuição dos veículos dessa via. Nas variações do fluxo de trânsito em ruas perpendiculares ou paralelas, poderemos observar o caos momentâneo. Em cidades sem planejamento urbano, o caos pode ser até permanente [5]. As figuras 3 e 4 a seguir ilustram de forma singela o desvio de vias, congestionadas ou interrompidas, proposto pelo sistema dinâmico de roteirização. A figura 3 ilustra um pequeno bolsão de

distribuição, com a rota inicial oferecida pelo roteirizador, considerando o caminho mínimo do trajeto ou o caminho mais rápido. O ponto verde representa a origem do veículo e o ponto azul o destino do veículo [6].

Fig. 3. Rota inicial. Fonte: Ferreira, 2009

A figura 4 ilustra o veículo realizando o deslocamento conforme rota inicial e uma ocorrência notável gera a interrupção do logradouro onde o veículo trafegaria, assim que detectada a ocorrência é informada à central de processamento, de imediato as informações são avaliadas e repassadas para o GPS que altera o roteiro conforme a ilustração a seguir [6].

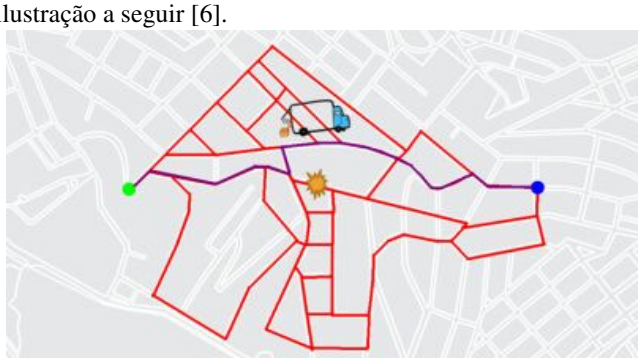


Fig. 4. Rota alterada. Fonte: Ferreira, 2009

A roteirização dinâmica representa o nível tático da roteirização, conforme será visto na seção VI.

V. A Tecnologia da Informação e Comunicação (TIC) na Roteirização

Segundo Santos [7], as ferramentas de Inteligência Artificial (IA) surgem como alternativas de apoio ou substituição dos métodos tradicionais de processamento e tomada de decisão em transportes. A tecnologia de transmissão de informações de trânsito *Traffic Massage Channel* (TMC) possibilita a comunicação de eventos relevantes ao percurso escolhido, são recebidos por ondas de rádio FM-RDS [8] e permitem à comunicação embarcada com os roteirizadores e aparelhos de GPS, essa condição aumenta significativamente o nível de serviço e a rapidez na transmissão de informações de tráfego. Os satélites de comunicação do tipo Geoestacionários contribuem de maneira importante na transmissão de dados em tempo real. O sistema de navegação por GPS nada mais é do que um sistema de rádio-navegação através do uso de satélites, que possui a capacidade de fornecer coordenadas precisas de posicionamento tridimensional e informações de navegação e tempo, para qualquer objeto que possua um transmissor GPS instalado. Verifica-se, dessa forma, que inúmeras são as possibilidades de pesquisa que poderão ser exploradas na roteirização, com o auxílio do GPS [9].

Para Bodin [10], a mais significativa mudança com relação aos sistemas para roteirização e programação de veículos ocorreu no ambiente computacional juntamente com os avanços alcançados nas áreas de Tecnologia da Informação e Comunicação (TIC). A aplicação da tecnologia da geoinformação ou Sistema de Informação Geográfica (SIG) combinados a roteirização também tem contribuído para aumentar o nível de serviço, permitindo a visualização dos pontos de atendimento a serem visitados e a malha viária por onde trafegam os veículos.

VI. A Hierarquia da Roteirização

Através dos três níveis hierárquicos de roteirização é possível considerar não apenas os fatores básicos de roteirização (algoritmos de roteirização) como também os fatores externos, ocorrências relevantes, que influenciam diretamente nos níveis de serviço nas grandes cidades (roteirização dinâmica). A previsão do tráfego representa o nível estratégico da roteirização. A figura 5 ilustra a hierarquia proposta para a roteirização eficiente apoiada pelos três níveis.

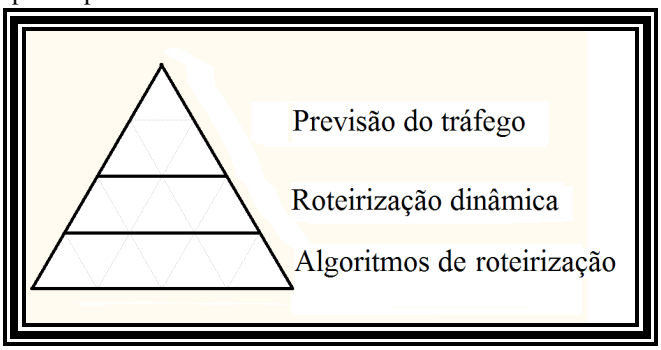


Fig. 5. Hierarquia de roteirização. Fonte: Autores

VII. Conclusão

A previsão do comportamento do tráfego pode ser uma excelente ferramenta para auxiliar a tomada de decisão que antecede a roteirização de forma a viabilizar as etapas de distribuição física com mais efetividade e produtividade. Com a possibilidade de prever as oscilações da fluidez do tráfego é possível escolher as melhores janelas de atendimento de maneira a evitar horários em que a previsão do tráfego aponte para níveis de lentidão que comprometam o atendimento. A roteirização dinâmica combinada a previsão do tráfego pode aumentar significativamente a eficácia da roteirização nas grandes cidades. Os roteiros estáticos não permitem aperfeiçoar todo o roteiro do veículo de maneira que todos os clientes sejam atendidos dentro da janela de tempo estimada. Os desvios inteligentes visam à redução do tempo em trânsito, mesmo quando a distância percorrida for um pouco maior, existindo ainda a economia de tempo e combustível, conclui-

se que as novas técnicas e tecnologias da informação e comunicação são decisivas para a criação de alternativas inovadoras de roteirização. Pretende-se dar continuidade com esse trabalho inicial e parcial com outras amostras de dados coletadas em dias diferentes da semana, em meses diferentes e com dias atípicos da cidade de São Paulo para obter novos resultados utilizando a rede *neuro fuzzy*. A roteirização dinâmica também será estudada em conjunto visando à possível integração dos três níveis de roteirização conforme discutido brevemente no trabalho.

REFERÊNCIAS

[1] *TODAY Logistics & Supply Chain*. São Paulo: Cecilia Borges, Ano III, n. 38, 2009.

[2] BODIN, L.D.; B. GOLDEN; A. ASSAD E M. BALL. *Routing and scheduling of vehicles and crews: The state of the art. Computers and Operations Research,* vol.10, n.2, 1983.

[3] BALLOU, R. H. Gerenciamento da Cadeia de Suprimentos/Logística Empresarial. 5. ed. Porto Alegre: Bookman, 2006. 616 p.

[4] CET - Companhia de Engenharia de Tráfego. Disponível em: http://www.cetsp.com.br Acesso em: 15 Dez. 2009.

[5] PENA, F. Biografias em fractais: múltiplas identidades em redes flexíveis e inesgotáveis. Revista Fronteiras – estudos midiáticos, Vol. VI n. 1, p. 82 - jan/jun. 2004.

[6] FERREIRA, R. P.; SASSI, R. J. Roteirização Inteligente: Uma Ferramenta para o Aprimoramento do Nível de Serviço de Distribuição e Coleta. XVI Simpósio de Engenharia de Produção, Bauru/ SP, 2009.

[7] SANTOS, A. V. N.; FELIX, L. B.; VIEIRA, J. G. V. Estudo da logística de distribuição física de um laticínio utilizando lógica *fuzzy*. V ENCONTRO MINEIRO DE ENGENHARIA DE PRODUÇÃO, 2009.

[8] LOGWEB - Serviço Indica mostra situação do trânsito aos motoristas de São Paulo. n. 89, 2009.

[9] PIMENTA, D. J. Algoritmo de Otimização para o Problema de Roteamento de Veículos no Transporte Conjunto de Cargas e de Passageiros. Belo Horizonte, 2001. 68 p.; Dissertação – Escola de Engenharia da Universidade Federal de Minas Gerais.

[10] BODIN, L.D. *Twenty years of routing and scheduling. Operations Research*, n. 38(4), 571-579, 1990.

Ricardo Pinto Ferreira: Técnico Operacional (2003), bacharel em Administração de Empresas (2005), MBA em Logística Empresarial e *Supply Chain* (2007), MBA em Tecnologia da Informação (2009), Mestrando em Engenharia de Produção pela Universidade Nove de Julho.

Carlos Affonso: Possui graduação em Escola de Engenharia de São Carlos pela Universidade de São Paulo (1995). Atualmente é engenheiro de materiais (Volkswagen do Brasil). Atuando principalmente nos seguintes temas: indústria automotiva, acabamento superficial e polímeros e elastômeros. Mestrando em Engenharia de Produção pela Universidade Nove de Julho.

Renato José Sassi: bacharel em Ciências Econômicas pela Faculdade de Economia Finanças e Administração de São Paulo (1987), especialista (pós-graduação Lato Sensu) em Administração de Empresas (área de concentração Análise de Sistemas) pela Fundação Escola de Comércio Álvares Penteado (1988), especialista (pós-graduação Lato Sensu) em Didática do Ensino Superior pelo Centro Universitário Sant'anna (1996), mestre em Administração de Empresas (Gestão de Negócios) pelo Centro Universitário Sant'Anna (1999) e doutor em Engenharia Elétrica pela Escola Politécnica (POLI) da Universidade de São Paulo (2006). Atualmente é pesquisador associado do Grupo de Inteligência Computacional, Modelagem e Neurocomputação (ICONE) do Laboratório de Sistemas Integráveis (LSI) da Escola Politécnica da Universidade de São Paulo (USP), docente e pesquisador do Programa de Mestrado em Engenharia de Produção da Universidade Nove de Julho (UNINOVE). Membro IEEE.

SHORT PAPERS

Tecnologia Adaptativa Aplicada à Biotecnologia: Estudos de Caso e Oportunidades

H. Pistori, K. P. de Souza

Resumo— Este artigo apresenta diversos exemplos de aplicações de Tecnologia Adaptativa que vêm sendo planejadas e realizadas através do programa de pós-graduação em Biotecnologia da Universidade Católica Dom Bosco e seus parceiros. O principal objetivo deste texto é despertar a comunidade de teoria e tecnologias adaptativas para um importante nicho de pesquisa e desenvolvimento tecnológico envolvendo a integração entre adaptatividade, visão computacional e reconhecimento de padrões.

Palavras-chave— Árvores de decisão adaptativas, biotecnologia, reconhecimento de comportamentos de animais em laboratório.

I. Introdução

Em 2006, foi criado, na Universidade Católica Dom Bosco, o primeiro programa de pós-graduação *stricto sensu* em Biotecnologia do Mato Grosso do Sul e o segundo da região Centro-Oeste do Brasil. Esse programa proporcionou a integração de grupos de pesquisa das áreas biológicas e agrárias, com um grupo que iniciava seus trabalhos de pesquisa nas áreas de visão computacional e tecnologias adaptativas. O perfil da região, bastante voltado para o agronegócio e meio ambiente, ofereceu aos grupos de pesquisa do programa em Biotecnologia uma série de problemas relevantes localmente, que começaram a ser estudados em 2006, com apoio de agências de fomento e outras instituições de pesquisa e de desenvolvimento regionais e nacionais. Vários desses estudos evoluíram para o desenvolvimento de produtos, na forma de software aplicativo, e de uma plataforma de desenvolvimento integrando a implementação de dezenas de algoritmos de processamento de imagens, reconhecimento de padrões, aprendizagem automática e tecnologias adaptativas.

Neste artigo, serão apresentados diversos problemas que vêm sendo explorados através de técnicas de visão computacional, aprendizagem automática e tecnologia adaptativa, além de algumas demandas atuais dos pesquisadores da área de biotecnologia da UCDB. Para cada problema, além de uma contextualização que buscará chamar a atenção para a sua importância, serão descritos os elementos mais diretamente propícios para aplicação de técnicas de reconhecimento de padrões, em particular, daquelas derivadas de tecnologias adaptativas, como as árvores de decisão adaptativas. Todos os problemas apresentados aqui têm em comum a necessidade de se processar imagens e o artigo descreverá também que classe de atributos podem ser extraídos de cada imagem para que se possa aplicar aos problemas as técnicas de árvores de decisão adaptativas.

Na próxima seção serão descritas as árvores de decisão adaptativas e o algoritmo AdapTree, um tipo particular de árvore de decisão adaptativa que foi apresentado em [1]. As seções III e IV apresentarão os exemplos de aplicações em biotecnologia que podem ser tratados através de reconhecimento de padrões. Alguns desses exemplos tratam de projetos já em andamento ou concluídos (seção III), enquanto outros aguardam a formação de novas equipes na área computacional para trabalharem junto com os pesquisadores da biotecnologia (seção IV). As conclusões e expectativas de trabalhos futuros são apresentadas na seção V.

II. Árvores de Decisão Adaptativas

Em [1], foi apresentado um dispositivo adaptativo cujo mecanismo subjacente é uma árvore de decisão. Esse dispositivo, chamado de árvore de decisão adaptativa, permite que a estrutura hierárquica de uma árvore de decisão comum possa ser dinamicamente alterada durante o processo de decisão, quando a árvore é varrida da raiz para as folhas.

A estrutura subjacente das árvores de decisão adaptativas são alteradas através de funções adaptativas capazes de remover ou substituir sub-árvores, indexadas através de um mecanismo de rotulação definido em [1]. Um caso especial de árvore de decisão adaptativa, que acrescenta algumas restrições ao tipo de alterações que a árvore pode sofrer, além de algumas extensões para tratamento de problemas de reconhecimento de padrões com valores contínuos, ausentes (*missing values*) ou inconsistentes, é o algoritmo AdapTree.

Os trabalhos aqui descritos receberam apoio financeiro das seguintes entidades e empresas: Universidade Católica Dom Bosco (UCDB), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Apoio à Ciência e Tecnologia do Mato Grosso do Sul (FUNDECT), Empresa Tecsinapse e Empresa Belt BR Ltda. .

H. Pistori e K. P. de Souza são professores da Universidade Católica Dom Bosco (UCDB), Av. Tamandaré, 6000 79117-900 Campo Grande, MS Brasil (email: pistori@ucdb.br; kriowloo@gmail.com).

K. P. de Souza é mestrando do programa em Ciência da Computação da Universidade Federal do Mato Grosso do SUL (UFMS) .

O AdapTree[4] é um algoritmo de aprendizagem de máquina supervisionado, incremental, que permite que as fases de treinamento e teste sejam executadas intercaladamente. No AdapTree, as amostras ou exemplos de treinamento são armazenados em uma estrutura similar à de uma árvore de prefixos, ou de um autômato adaptativo coletor de nomes, em que cada nível hierárquico corresponde a um dos elementos de um vetor de atributos que representa cada amostra. No caso mais simples, em problemas cuja entrada é uma imagem, o vetor de atributos poderia ser, por exemplo, um vetor com os valores de cada pixel que compõe a imagem. Geralmente, no entanto, algoritmos de extração de atributos são utilizados para gerar informações de mais alto nível a partir dos pixels da imagem. O funcionamento do AdapTree se assemelha ao de algoritmos de aprendizagem baseados em exemplos (*instance based learning*), como o tradicional k-vizinhos-mais-próximos ou k-NN (*k-Nearest Neighbours*) [2].

As funções adaptativas do AdapTree impõem uma ordenação completa, pré-determinada, ao conjunto de atributos. É essa ordenação que determina em que nível da árvore de decisão irão ocorrer os diversos atributos. À medida que exemplos vão sendo lidos da entrada do dispositivo, a estrutura vai crescendo, na forma de uma árvore de prefixos. Como cada nível da árvore corresponde a um único atributo, e vice-versa, a altura máxima da árvore de decisão subjacente da AdapTree é igual ao total de atributos.

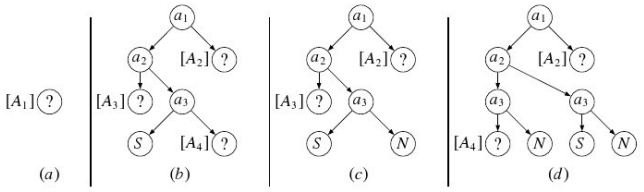


Fig. 1. Evolução do algoritmo AdapTree durante leitura de 4 amostras de treinamento: SNSS, SNNN e SSNN.

A Figura 1 mostra graficamente a evolução de uma árvore de decisão adaptativa para um problema hipotético de aprendizagem com quatro atributos binários $a_1,...,a_4$. O último atributo, a_4 , corresponde à classe. Os valores dos atributos de cada exemplo apresentado ao dispositivo são mostrados através de cadeias de S (Sim) e N (Não), com cada posição da cadeia correspondendo a um dos quatro atributos. Implicitamente, a aresta que parte do lado esquerdo de um nó corresponde sempre ao valor S, enquanto a do lado direito, ao N, por isso estes valores não aparecem nas árvores da Figura 1. Entre colchetes, e apenas nos vértices rotulados com sinal de interrogação, aparecem as ações adaptativas, responsáveis pelo crescimento da árvore conforme novas amostras de treinamento vão sendo apresentadas.

O tratamento de valores contínuos no AdapTree é feito através da discretização. Na implementação atual, o método de discretização de Fayyad e Irani [3] é aplicado, antes do início da operação do AdapTree, para determinar a quantidade e o tamanho dos intervalos que deverão particionar cada um dos atributos contínuos. Esse método é do tipo supervisionado, pois necessita de um conjunto de exemplos de treinamento para decidir sobre os pontos de corte que determinarão os intervalos discretos do atributo.

Valores ausentes são tratados, no AdapTree, através de um pré-processamento do conjunto de treinamento, para substituir valores ausentes por valores válidos. Essa substituição, que segue o método usado no algoritmo CN2 [4], consiste, para o caso de atributos discretos, em usar no lugar do valor ausente o valor mais comum (moda estatística, calculada sobre um conjunto de treinamento). Para atributos contínuos utiliza-se a média aritmética.

Valores inconsistentes[5] são tratados por contadores, atrelados às folhas da árvore de decisão adaptativa, que permitem que o valor da classe associado a cada folha seja sempre aquele de maior frequência entre os exemplos inconsistentes que eventualmente são detectados no conjunto de treinamento. Dessa forma, no caso de valores inconsistentes, a árvore "decidirá heuristicamente" pela classe mais comum, entre os exemplos inconsistentes.

III. Estudos de Caso

Os estudos de caso desta seção referem-se a projetos de pesquisa e desenvolvimento tecnológico concluídos ou em fase avançada de execução.

A. Defeitos em Couro Bovino

O Mato Grosso do Sul é um dos principais produtores de carne bovina do Brasil. No entanto, sua participação no mercado do couro ainda precisa ser fortalecida. Uma etapa bastante importante na cadeia do couro é a avaliação de qualidade, realizada hoje de forma não automatizada e com alto grau de subjetividade, dependendo de técnicos altamente especializados.

Fig. 1. Parte de uma peça de couro no estágio *wet-blue* com algumas regiões defeituosas marcadas a mão.

[4] Originalmente, em [1], o algoritmo de aprendizagem AdapTree foi denominado AdapTree-E, de AdapTree Estendido. Uma vez que o AdapTree "não estendido" possuem sérias limitações para sua utilização em reconhecimento de padrões, pois é incapaz de lidar com variáveis contínuas, valores ausentes ou inconsistentes, propõe-se, no presente artigo, a simplificação do nome AdapTree-E para AdapTree.

[5] Exemplos que, apesar de pertecerem a diferentes classes, possuem atributos idênticos.

O projeto DTCOURO, iniciado através de uma parceria entre UCDB e EMBRAPA, gerou um software capaz de reconhecer regiões defeituosas de uma peça de couro, identificar o tipo de defeito e aplicar uma classificação definida pelo usuário [5,6]. A utilização do AdapTree, e outros algoritmos de aprendizagem automática, neste caso, ocorre na análise de pequenas regiões retangulares, ou janelas, da imagem de uma peça de couro, que é varrida em intervalos espaciais, não necessariamente disjuntos, definidos experimentalmente. Na etapa de treinamento, de cada janela, rotulada por especialistas por um tipo de defeito ou por um indicador de que a região não tem defeito, é extraído um vetor de atributos que alimenta o algoritmo de aprendizagem. Na etapa de teste (ou validação), a árvore de decisão adaptativa ou outro modelo qualquer induzido pelo algoritmo de aprendizagem, recebe o vetor de atributos extraído das janelas de varredura, agora sem rótulos, e retorna o tipo de defeito presente na janela ou um indicativo de que a região não apresenta defeitos.

A Figura 1 apresenta uma imagem de uma peça de couro no estágio *wet-blue*, que é geralmente o primeiro estágio de curtimento pelo qual passa o couro. Nessa peça, foram marcadas à mão algumas regiões defeituosas, que depois foram utilizadas na aprendizagem automática. Quase no centro da imagem, aparece uma região com sarnas (SA), à direita, aparecem veiamentos (VE) e um furo (F). Na parte de baixo, pode-se perceber uma anotação correspondente a uma região com marcas de mosca-dos-chifres (MC).

B. Exploração Horizontal e Vertical de Camundongos

A utilização de camundongos na análise preliminar do efeito de novos medicamentos ou produtos biotecnológicos, de modo geral, é bastante comum. Existem diversos tipos de experimentos para medir mudanças no comportamento desses animais que podem indicar, por exemplo, algum efeito colateral de uma droga. O projeto TOPOLINO resultou em um software que automatiza o processo de medição de diversas variáveis relacionadas com os experimentos de campo aberto, labirinto em cruz e *water-maze*, todos envolvendo a filmagem de camundongos em situações controladas. O reconhecimento de padrões foi utilizado, neste caso, na identificação de dois tipos de comportamentos que o camundongo realiza durante o experimento do campo aberto: (1) exploração horizontal, quando o camundongo está caminhando nas 4 patas, e (2) exploração vertical, quando o camundongo sobe nas paredes laterais, de acrílico, do campo "aberto" ou se apoia nas duas patas traseiras, para realizar algum movimento livre com as patas dianteiras [7,8].

Fig. 3. Camundongos claro e escuro em posições de exploração horizontal e vertical, respectivamente, durante o experimento de campo aberto.

Na Figura 2, pode-se observar, ao lado esquerdo, um camundongo branco em exploração horizontal. Nessa mesma figura, ao lado direito, observa-se um camundongo preto em exploração vertical. Os vetores de atributos, para este problema, são extraídos de partes de imagens obtidas por uma câmera posicionada acima do campo aberto, contendo apenas um camundongo (o isolamento da imagem do camundongo é realizado por técnicas de rastreamento e segmentação, independentes do módulo de reconhecimento de padrões). A partir desses vetores de atributos, devidamente rotulados com o tipo de exploração, o AdapTree pode aprender a distinguir os dois comportamentos.

C. Leveduras Mortas e Vivas

Nos últimos anos, o estado do Mato Grosso do Sul vem se destacando como um grande produtor de cana-de-açúcar, com um crescimento significativo na implantação de usinas sucroalcooleiras. No processo de fermentação alcoólica, o controle microbiológico é uma etapa fundamental para maximização de rendimentos [9]. Esse controle implica na análise microscópica, periódica, de soluções extraídas durante as diversas etapas de produção industrial, para se determinar a necessidade de intervenções para, por exemplo, reduzir o nível de bactérias presentes no caldo ou no mosto.

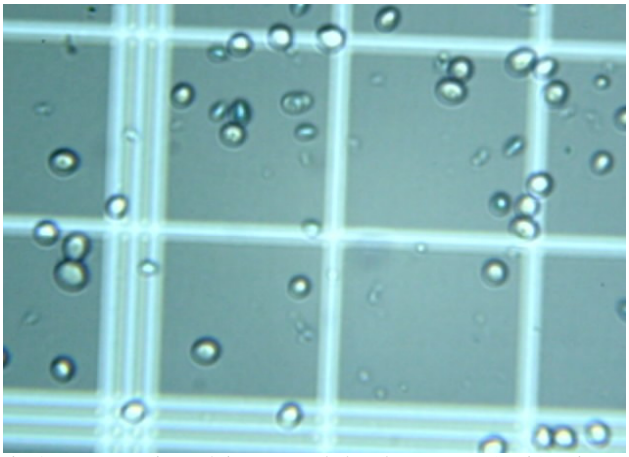


Fig. 2. Imagens microscópicas contendo leveduras mortas (azuis) e vivas.

O projeto BIOVIC tem como objetivo automatizar o processo de contagem de bactérias e leveduras, vivas ou mortas, a partir de imagens microscópicas. No estágio atual, o software realiza a contagem de leveduras mortas e vivas, após aplicação de um corante que faz com que as leveduras mortas se distinguam das leveduras vivas por uma coloração mais azulada. De forma similar ao DTCOURO, o reconhecimento de padrões é realizado através de uma varredura na imagem, com janelas de tamanho compatível com o tamanho das leveduras. A Figura 3 apresenta imagens de dezenas de leveduras mortas (mais azuladas em seu interior) e vivas. A imagem apresenta também algumas impurezas (pequenas manchas, geralmente menores que uma levedura), que precisam ser ignoradas pelo software de contagem de células.

D. Mortalidade de Larvas do Aedes aegypti

As epidemias de Dengue ocorridas nos últimos anos despertaram o país para a necessidade de se buscar formas mais eficazes de combate ao seu mosquito transmissor, o *Aedes aegypti*. O Mato Grosso do Sul se destaca pela busca de novos larvicidas feitos a partir de extratos de sua diversificada flora pantaneira e de cerrado. Experimentos com novos larvicidas envolvem a análise, em várias repetições, da mortalidade de larvas utilizando diversas combinações de concentrações e extratos. Centenas e até milhares de experimentos precisam ser realizados até se encontrar uma substância que produza o efeito desejado. A contagem de mortalidade é feita por análise visual de recipientes contendo larvas expostas durante 24 horas aos produtos em teste.

Fig. 4. Imagens de larvas do *Aedes aegypti* em um experimento, com 6 repetições, do teste de mortalidade.

Através do projeto LARVIC, a contagem de mortalidade está sendo automatizada. As imagens são capturadas através de uma câmera posicionada acima dos recipientes contendo as larvas [7,10]. A Figura 4 mostra 6 recipientes contendo larvas em meio líquido. Pequenos trechos de imagens são analisadas periodicamente para se detectar a quantidade de larvas vivas e mortas. Além de tratar-se de um meio líquido, a introdução de ruídos no processo de captura de imagens impede que a simples detecção de falta de movimento seja utilizada para identificar larvas mortas. O reconhecimento de padrões, neste caso, está sendo utilizado em uma etapa intermediária, onde se busca identificar três formatos possíveis para cada larva: (1) aberta ou esticada; (2) curvada e (3) fechada.

Fig. 6. Larvas em três posições distintas: aberta, curvada e fechada.

A Figura 5 apresenta imagens de larvas com as formas aberta, curvada e fechada, da esquerda para direita. O vetor de atributos que alimenta o algoritmo de aprendizagem busca traduzir informações relacionadas com o contorno e topologia da imagem da larva após um processo de esqueletomização, que resulta em imagens simplificadas, como estas apresentadas na Figura 6.

Fig. 5. Imagem das larvas após processo de esqueletomização

E. Comportamento Alimentar de Ovinos Estabulados

A criação de ovinos é uma interessante alternativa para complementação de renda na agricultura familiar. O desenvolvimento de rações, de baixo custo, que possam também atender de forma adequada às necessidades alimentares desse animais, é uma importante atividade que também exige pesquisa científica. Para o Mato Grosso do Sul, a utilização de mandioca na composição de ração para ovinos pode ser uma boa alternativa, por tratar-se de uma cultura já tradicional nesse estado. O projeto OVIC busca auxiliar a avaliação de novas rações automatizando o processo de análise do comportamento de ovinos estabulados conforme diferentes rações são fornecidas aos animais. Dos projetos apresentados nesta seção, este é o que se encontra em fase menos adiantada. Assim como no caso das larvas, pretende-se utilizar o reconhecimento de padrões como um etapa intermediária para identificação de determinadas posições e posturas dos ovinos que possam facilitar a análise de sequências de quadros através de técnicas baseadas em modelos de Markov ocultos.

IV. Oportunidades

Esta seção apresenta alguns problemas que estão sendo tratados por pesquisadores do programa de mestrado em Biotecnologia da UCDB e que apresentam bom potencial para aplicação das técnicas que já vêm sendo adotadas nos projetos citados na seção anterior.

A. Classificação automática de sementes e grãos para avaliação de desempenho de descascamento automatizado

Devido à grande quantidade de sementes e grãos que precisam ser descascados após suas colheitas, é conveniente fazer uso de máquinas para realizar essa operação. A análise dos resultados desse processo pode ser necessária para algumas áreas de pesquisa, principalmente por se tratar de uma operação automatizada. Contudo, essa análise, geralmente, é realizada por humanos de forma manual sobre uma amostra menor (que pode ser composta por milhares de unidades). Durante a análise, são contados os elementos inteiros (ou seja, que ainda estão com casca), os elementos descascados que estão inteiros e os elementos descascados que não estão inteiros (como exemplo, os elementos quebrados). Por se tratar de uma tarefa minuciosa e que possivelmente leva o analisador à exaustão, a consistência e, consequentemente, a confiabilidade dos resultados podem ser comprometidas. Uma alternativa para esse cenário é a aplicação de técnicas de visão computacional (e outras áreas afins) com o intuito de realizar automaticamente a contagem desses elementos por meio da análise de imagens digitais.

B. Acompanhamento do estado de oxidação de óleos por meio de análise de imagens e correlação com medidas químicas

O nível de oxidação é um dos parâmetros utilizados para se avaliar a qualidade de óleos. Essa oxidação promove alteração na coloração do óleo, a qual poderia ser avaliada como uma medida do estado de oxidação. Devido à existência de tons de cores próximos que se referem a níveis de oxidação diferentes, podem ocorrer inconsistências no processo humano de classificação dessas cores. Realizar essa tarefa de forma automatizada pode trazer consequências positivas na identificação desses níveis, pois elimina a subjetividade e algumas confusões provenientes das limitações humanas durante a análise.

C. Classificação de autenticidade e procedência de méis através de atributos visuais

Nas ciências biológicas, existem métodos para análise de características de méis. Dentre essas características, estão a autenticidade – que indica se o mel em análise é verdadeiro ou falso – e a procedência – que indica características das abelhas produtoras do mel e sua florada (que, basicamente, é tipo de flores que essas abelhas visitam). No entanto, devido à prática, existem pessoas (como produtores rurais, por exemplo) que conseguem identificar tais características analisando apenas a cor do mel. Todavia, essa classificação baseada em cores ainda é subjetiva e sem fundamentação científica.

D. Análise automática de impurezas existentes entre grãos e sementes

Para mensurar a quantidade de impurezas existentes (pedaços de madeira, pedra, etc.) em carregamentos de grãos e sementes, uma amostra menor é analisada separadamente. Essa análise é realizada por humanos através de observação visual dessa amostra, para a contagem de impurezas. Dependendo do tamanho da amostra, essa análise pode se tornar onerosa e, consequentemente, interferir negativamente nos resultados finais.

E. Contagem de eritrócitos policromáticos e normocromáticos micronucleados de camundongos em imagens microscópicas

Algumas substâncias têm a capacidade de produzir alterações na base genética de seres vivos. Tais alterações podem provocar danos irreversíveis ao seu portador, como o câncer. A área da genética que estuda essas alterações é denominada genotoxicidade. Uma das métricas utilizadas por essa área para avaliar essa capacidade dessas substâncias (conhecida como efeito genotóxico) é baseada na contagem de células sanguíneas dos animais: os eritrócitos. Esses eritrócitos podem ser jovens (policromáticos) ou maduros (normocromáticos). Além disso, alguns desses eritrócitos possuem manchas visuais, chamadas de micronúcleos. Geralmente, a contagem dessas células é realizada por especialistas, que classificam cerca de 2000 a 5000 células utilizando um microscópio. Obviamente, essa contagem pode se tornar uma atividade exaustiva e subjetiva, o que pode comprometer a qualidade dos resultados obtidos.

F. Monitoramento de comportamento de pragas em cultura de bambus

O bambu pode ser considerado uma alternativa de baixo custo às necessidades humanas, podendo ser utilizado desde a construção de ambientes até a preservação ambiental. Quando cortado e armazenado, o bambu se torna vulnerável ao ataque de alguns insetos. Dentre eles, o inseto que se caracteriza como principal praga da cultura do bambu é o caruncho-do-bambu. Geralmente, o controle de pragas é realizado por inseticidas químicos. Para a avaliação desses inseticidas, é necessário analisar a ação desses produtos por meio da observação do comportamento e hábitos dos insetos. Tal observação deve ser realizada de forma contínua e em períodos extensos. Além disso, diversos insetos devem ser observados de forma simultânea. Considerando as limitações humanas, a construção de um sistema computacional capaz de classificar e registrar os comportamentos desses insetos torna-se um considerável auxílio aos pesquisadores envolvidos nas análises dos inseticidas acima mencionados.

G. Contagem automática de grãos de pólen em méis para avaliação de procedência e qualidade

Os grãos de pólen presentes no mel podem ser utilizados para analisar a atividade da colmeia que o produziu. Além disso, os milhares de grãos de pólen presentes em cada grama de mel servem como testemunho de sua origem e qualidade. Para avaliar essas características, torna-se interessante a contagem dos diferentes tipos de grãos de pólens contidos em certas porções de mel. Devido à grande quantidade de grãos em pequenas amostras de mel, essa contagem se torna maçante e custosa, podendo ser realizada de forma automática por um sistema baseado em visão computacional. Esse sistema interpretaria as imagens das amostras dos méis e faria a contagem automática dos grãos de pólen existentes nelas.

V. Conclusões

Neste artigo foram apresentados 12 exemplos de projetos relacionados com biotecnologia que podem se beneficiar bastante dos desenvolvimentos nas áreas de visão computacional, aprendizagem automática, reconhecimento de padrões e tecnologias adaptativas. Tratam-se de projetos que já vêm sendo executados por pesquisadores da área da biotecnologia e que podem receber, de forma imediata, pesquisadores e acadêmicos da área da computação, para lidar com os problemas relacionados com a automatização de processos que envolvem análise de imagens. Em todos esses projetos, a adaptatividade pode ser explorada na forma de algoritmos de aprendizagem supervisionada, como o AdapTree, que, a partir de exemplos previamente classificados, convertidos em cadeias de entrada, produz um

modelo de identificação/decisão capaz de distinguir os objetos de interesse nas imagens.

Para o desenvolvimento do AdapTree, foi utilizada a biblioteca Weka, que contém centenas de implementações de algoritmos de aprendizagem e um módulo que permite que o desempenho desses algoritmos seja facilmente avaliado comparativamente em diferentes aplicações. A versão atual do AdapTree exige a utilização de uma versão mais antiga do Weka, o que dificulta a comparação do AdapTree com algoritmos de aprendizagem implementados no Weka mais recentemente. Para o futuro próximo, esperamos ter uma nova implementação do AdapTree que possa ser integrada às versões mais recentes da plataforma Weka.

VI. AGRADECIMENTOS

Os autores reconhecem as contribuições dos mais de 30 acadêmicos e pesquisadores que participaram ou ainda participam dos projetos apresentados neste artigo. A equipe dos projetos em andamento pode ser consultada no website de cada projeto, acessível a partir da página do primeiro autor: www.gpec.ucdb.br/pistori ou do grupo de pesquisa e desenvolvimento em Visão Computacional, INOVISAO: www.gpec.ucdb.br/inovisao.

VII. REFERÊNCIAS

[1] H. Pistori e J. J. Neto, "Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações". Tese de Doutorado. USP. São Paulo, Brasil, Dezembro, 2003.

[2] D. W. Aha, D. Kibler, M. Albert, "Instance-based learning algorithms. Machine Learning", v. 6, p. 37–66, 1991.

[3] U. M. Fayyad, K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning". In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. San Mateo, CA: Morgan Kaufmann, 1993. p. 1022–1027.

[4] P. Clark, T. Niblett, "The CN2 induction algorithm. Machine Learning", v. 3, p. 261–283, 1989.

[5] L. N. B. Quinta and H. Pistori, "Seleção de Atributos para a Segmentação do Couro Bovino". In: WVC - Workshop de Visão Computacional, 2008, Bauru. Anais do IV Workshop de Visão Computacional. Bauru : UNESP, 2008.

[6] R. Viana, R. Rodrigues, M. A. Alvarez and H. Pistori, "SVM With Stochastic Parameter Selection For Bovine Leather Defect Classification". Lecture Notes in Computer Science - Vol. 4872 - Special issue - IEEE Pacific-Rim Symposium on Image and Video Technology PSIVT 2007, Santiago, Chile, December 17-19, 2007

[7] H. Pistori, V. Odakura, J. B. Monteiro, W. Gonçalves, A. R. Roel, J. Silva, B. Machado. "Mice and Larvae Tracking using a Particle Filter with an Auto-Adjustable Observation Model". Pattern Recognition Letters, 2009.

[8] W. N. Goncalves, J. B. Monteiro, J. A. Silva, B. B. Machado, V. Odakura, H. Pistori, "Multiple Mice Tracking using a Combination of Particle Filter and K-Means" . In: SIBGRAPI - Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing, Belo Horizonte, 7-10 Outubro, 2007

[9] L. N. B. Quinta, K. P. de Souza, H. Pistori e M. P. Cereda. "Reconhecimento de Leveduras Vivas e Mortas em Imagens Microscópicas utilizando Atributos de Cores" I Simpósio de Biotecnologia, Campo Grande, Brasil, Setembro, 2009

[10] J. H. de Sá Queiroz, H. Pistori, K. R. Porto e A. A. Roel. "Rastreamento de Múltiplas Larvas utilizando Técnicas de Visão Computacional: Resultados Preliminares", Workshop de Iniciação Científica - XXII SIBGRAPI, Rio de Janeiro, Brasil, Outubro, 2009.

Hemerson Pistori concluiu o doutorado em engenharia de computação e sistemas digitais pela Universidade de São Paulo e o mestrado em ciência da computação pela UNICAMP. Atualmente é professor do programa em Biotecnologia da Universidade Católica Dom Bosco Foi eleito como o representante brasileiro para a International Association for Pattern Recognition (IAPR), em 2007, pela Comissão Especial de Computação Gráfica e Processamento de Imagens (CEGRAPI) da Sociedade Brasileira de Computação (SBC). Participa de conselhos editorias e comitê de avaliadores de revistas como a International Journal for Computer Vision and Biomechanics (IJCV&B) e Electronic Letters on Computer Vision and Image Analysis (ELCVIA), além de ter atuado como membro de comitê de programa e revisor de diversos eventos científicos internacionais como CIARP, IWCIA, PSIVT, COMPIMAGE e VIPIMAGE . É o coordenador mundial do grupo de engenharias das instituições Salesianas de educação superior, IUS EngG. Liderou até 2007 o Grupo de Pesquisa em Engenharia e Computação, GPEC, finalista do premio FINEP, categoria institutos de pesquisa da região Centro-Oeste, em 2005 e 2006. O Prof. Pistori lidera atualmente o grupo de pesquisa INOVISAO.

Kleber Padovani de Souza obteve em 2004 o título de Tecnólogo em Processamento de Dados, pela Universidade para o Desenvolvimento do Estado e da Região do Pantanal. Desenvolveu projeto de iniciação científica como acadêmico bolsista do CNPq, trabalhando na área de visão computacional, focado em conceitos específicos de extração de atributos. Em 2006, obteve o título de Engenheiro de Computação, pela Universidade Católica Dom Bosco (UCDB), em que foi homenageado com o prêmio de aluno destaque da Sociedade Brasileira de Computação. Como membro do Grupo de Pesquisa em Engenharia e Computação (GPEC/UCDB) e bolsista CNPq, realizou o trabalho de estudo e aplicação de conceitos de modelos de Markov ocultos voltados ao reconhecimento de gestos por visão computacional. Está atualmente cursando o programa de mestrado em Ciência da Computação da Universidade Federal de Mato Grosso do Sul e desempenhando, na UCDB, atividades de ensino nos cursos de graduação e de pesquisa e desenvolvimento em visão computacional dentro do GPEC..

SKAN – Skin Scanner: software para o reconhecimento de câncer de pele utilizando técnicas adaptativas

(21 Janeiro 2010)

H. S. Ganzeli, J. G. Bottesini, L. O. Paz e M. F. S. Ribeiro

***Resumo* — Este artigo tem como objetivo apresentar o sistema SKAN: Skin Scanner, um software que procura distinguir imagens de câncer de pele de imagens que contenham outros tipos de lesões cutâneas. A identificação de imagens de melanoma se baseia na regra ABCDE, que é utilizada comumente por dermatologistas para o diagnóstico por inspeção visual de uma mancha. O sistema, a partir da utilização de filtros de imagens e extração de atributos, avalia de forma independente os fatores da regra e decide, com o uso de técnicas adaptativas, o diagnóstico da mancha. A contribuição relaciona-se, portanto, ao sistema de pré-diagnóstico de câncer de pele com o uso de processamento de imagem e técnicas adaptativas na tomada de decisão. Além disso, o artigo apresenta um algoritmo de análise da borda da mancha que pode ser utilizado em outras aplicações, ou adaptado para outros fins. Esse sistema foi proposto pelos autores como projeto de formatura do curso de Engenharia Elétrica com Ênfase em Computação da Escola Politécnica da Universidade de São Paulo.**

***Palavras chave* — Câncer, dermatologia, dispositivos adaptativos, melanoma, oncologia, pele, processamento de imagem.**

I. INTRODUÇÃO

ESTE documento tem como objetivo apresentar um projeto de formatura do curso de Engenharia Elétrica com ênfase em Computação desenvolvido junto ao Laboratório de Tecnologias Adaptativas (LTA). O trabalho propõe, a partir do reconhecimento de imagem, discernir fotografias que contenham câncer de pele do tipo melanoma de outras que apresentem manchas de pele comuns. Esse sistema alia os conceitos de computação com a dermatologia e oncologia.

O artigo apresenta primeiramente um resumo dos conceitos médicos sobre câncer de pele envolvidos no desenvolvimento do projeto. A seguir, detalham-se as características técnicas do sistema desenvolvido, apresentando sua arquitetura e funcionamento, além de mostrar onde e como a utilização de técnicas adaptativas foi importante para a obtenção de resultados. Por fim, são apresentados os resultados atingidos e uma seção de trabalhos futuros, explicitantando algumas possibilidades para a melhoria e continuidade deste trabalho.

II. CÂNCER DE PELE DO TIPO MELANOMA

O melanoma cutâneo é um tipo de câncer que tem origem nos melanócitos e tem predominância em adultos brancos. Embora só represente 4% dos tipos de câncer de pele [4], o melanoma é mais grave devido à sua alta possibilidade de metástase – formação de nova lesão tumoral a partir da primeira.

Existem três tipos principais de câncer de pele: o carcinoma basocelular, o carcinoma de células escamosas e o melanoma, que é o menos comum, porém mais agressivo [3]. O carcinoma basocelular, mais comum dos três, acontece quando as células da camada germinativa se reproduzem descontroladamente. Ele costuma aparecer em áreas da pele que ficam mais expostas ao sol, e é facilmente tratável, causando a morte em pouquíssimas ocasiões. Já o carcinoma de células escamosas é um pouco mais raro, embora ainda seja relativamente comum. O mais raro destes três, e também o mais preocupante, é o melanoma. O melanoma ocorre quando os melanócitos começam a se reproduzir de forma desordenada [3]. A identificação de melanomas utiliza uma regra bastante simples, que é efetiva na maior parte das vezes: a regra ABCDE. Este recurso, baseado na inspeção visual, consiste em analisar algumas características da mancha na pele: a assimetria, as bordas, a coloração, diâmetro e evolução ou elevação [3]. Com a combinação destas características, é possível diferenciar, para a maior parte dos casos, um melanoma de manchas comuns e outros tipos menos agressivos de câncer.

O tratamento de melanoma é eficiente quando diagnosticado e retirado cirurgicamente na fase inicial. Se não for notado a tempo, pode evoluir para o estado invasivo. As células cancerosas podem espalhar-se para outras partes do corpo, invadindo e destruindo órgãos vitais e causando a morte [5].

A. *Assimetria*

Lesões benignas são simétricas, ou seja, suas metades são praticamente iguais. Por outro lado, as lesões malignas podem ser assimétricas. A Fig. 1 mostra uma comparação entre duas lesões quanto ao aspecto de simetria.

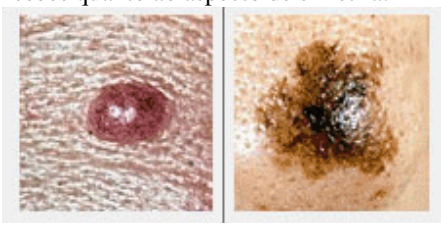


Fig. 1. Lesões simétrica (esquerda) e assimétrica (direita).

B. *Borda*

Lesões benignas apresentam bordas regulares, enquanto as

Este artigo é baseado em um projeto de formatura desenvolvido pelos alunos Heitor de S. Ganzeli, Julia G. Bottesini, Leandro de O. Paz e Matheus F. S. Ribeiro sob a orientação do Prof. Dr. João José Neto, no Laboratório de Linguagens e Técnicas Adaptativas (LTA EP/USP).

malignas podem tê-las irregulares, conforme a Fig. 2.

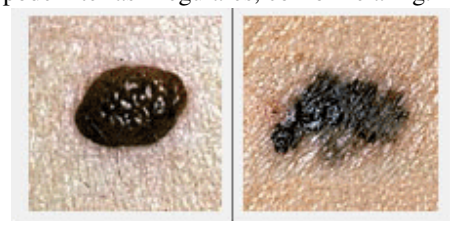


Fig. 2. Lesões com borda regular (esquerda) e irregular (direita).

C. Cor

Lesões benignas normalmente têm coloração marrom uniforme. As malignas podem apresentar variadas tonalidades de marrom ou preto, além de tons avermelhados, azulados e brancos. A Fig. 3 mostra um exemplo de mancha com apenas uma tonalidade (lesão da esquerda) e duas cores (lesão da direita).

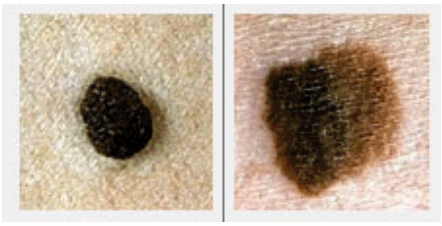


Fig. 3. Lesões monocromática (esquerda) e policromática (direita).

D. Diâmetro

Lesões malignas geralmente apresentam diâmetro maior que 6 milímetros. A Fig. 4 compara o tamanho de duas lesões, ilustrando esse caso.

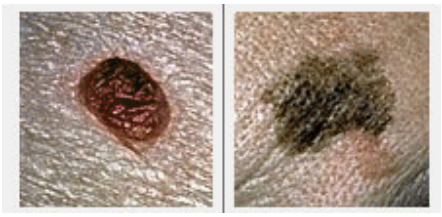


Fig. 4. Lesões com diâmetro menor que 6 milímetros (esquerda) e maior que 6 milímetros (direita).

E. Evolução ou elevação

As lesões malignas apresentam uma alteração de tamanho (aumento ou elevação) ao longo do tempo. Por ora, essa característica da mancha não é levada em consideração na análise feita pelo software.

III. Arquitetura do Sistema

O sistema recebe uma imagem com uma mancha e tem como objetivo identificar características que indiquem a possibilidade de ser um melanoma. Não é responsabilidade do sistema filtrar imagens de entrada, bem como descartar figuras que não sejam manchas de pele, ou que não possuam as características de entrada adequadas. A Fig. 5 ilustra a arquitetura do sistema.

Neste esquema nota-se a presença de um módulo externo ao SKAN, o "Validador da imagem", cuja finalidade é realizar uma triagem das imagens, separando figuras válidas para submissão ao sistema. As imagens válidas devem conter uma mancha no centro da pele em uma área lisa do corpo humano, sem excesso de pelos ou outros inconvenientes que dificultem a localização da mancha. Também são descartadas fotos que não tenham a resolução mínima permitida ou que não sejam suficientemente nítidas.

O sistema, por sua vez, contém três módulos, cada um responsável por uma etapa no diagnóstico da lesão.

A. Tradutor da imagem

Este módulo traduz a imagem para um formato que possa ser facilmente interpretado pelo identificador de irregularidades. Para isso, foram utilizadas diversas técnicas de processamento de imagem, tais como a aplicação de filtros e outras técnicas para extrair valores de interesse. O módulo extrai os atributos da mancha, como contorno da borda, simetria e coloração, e repassa para o próximo módulo na forma de números. Os algoritmos para extração das características da mancha serão detalhados posteriormente.

B. Identificador de irregularidades

Este módulo procura interpretar os dados que recebe do "Tradutor da imagem". Nessa etapa, os atributos extraídos são efetivamente analisados. Aplicam-se diversas técnicas a fim de se obter informações a respeito do formato da borda, simetria e coloração da mancha. Este módulo não associa essas características de forma a conseguir diagnosticar a doença.

C. Tomador de decisão

Módulo que analisa as informações fornecidas pelo identificador de irregularidades a respeito da mancha, e decide se ela é um melanoma ou não. Para isso, ele possui um conjunto de regras, implementadas na forma de árvore de decisão adaptativa. Caso a decisão tomada esteja errada, o paramédico poderá avisar o programa que o diagnóstico era equivocado. Assim, com a teoria adaptativa, esse módulo irá acrescentar ou modificar seu conjunto de regras para tomar a decisão correta, caso volte a encontrar uma entrada similar.. Logo, se uma foto semelhante for submetida ao sistema novamente, as chances de um diagnóstico correto aumentam, tornando o sistema mais robusto.

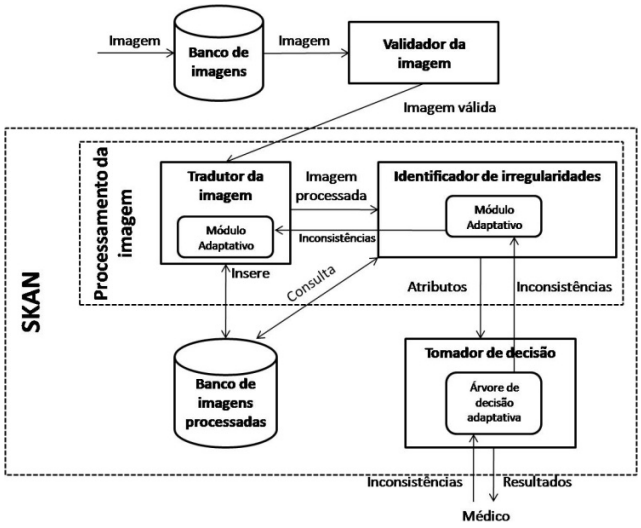


Fig. 5. Arquitetura do software com os três principais módulos.

IV. ALGORITMOS PARA EXTRAÇÃO DE ATRIBUTOS

Para o processamento da imagem, e conseqüente extração de atributos, foram utilizados três filtros: o filtro preto e branco, filtro de contraste e o SRM, *Statistical Region Merging* [7] .

O primeiro filtro aplicado na imagem é o SRM, que delimita regiões com coloração parecida e faz com que todos os pixels de uma mesma região sejam reduzidos à mesma cor [6]. A aplicação desse filtro é essencial para obter-se um resultado razoável na detecção de bordas porque ele localiza e delimita a mancha. A Fig. 6 mostra o resultado desse filtro ao ser submetido a uma imagem com uma lesão.

Fig. 6. Imagem de entrada (esquerda), resultado da aplicação do filtro com o algoritmo de segmentação *Statistical Region Merging* (centro) e resultado da aplicação do algoritmo de detecção de borda Canny (direita).

A detecção de borda diretamente na imagem original pode produzir resultados insatisfatórios, pois o filtro pode detectar bordas internas à mancha, devido a irregularidades na pele do paciente ou variações de iluminação. Assim, torna-se preciso eliminar os elementos da imagem que são irrelevantes para a detecção de borda, sendo essencial a aplicação do SRM.

O próximo passo do "Tradutor da imagem" é a delimitação da borda da lesão. Primeiramente, antes de passar pelo algoritmo de detecção de borda, a imagem do SRM é submetida a um filtro que separa os pixels da imagem em preto ou branco, de acordo com um limiar de cor. A submissão da mancha ao filtro preto e branco ressalta sua forma, separando-a da pele e tornando o cálculo da borda mais preciso.

As bordas em uma imagem são o resultado de mudanças bruscas em alguma propriedade física ou espacial entre duas regiões adjacentes de superfícies iluminadas. A maioria das técnicas de detecção de bordas baseia-se na aplicação de operadores diferenciais de primeira ou de segunda ordem. Esses operadores ressaltam os contornos das bordas mas também amplificam o ruído da imagem. Grande parte dos operadores de borda utiliza algum tipo de suavização da imagem antes da operação diferencial. [8]

O Canny é um operador gaussiano de primeira derivada que suaviza os ruídos e localiza as bordas de uma imagem [9]. O detector tem como uma de suas principais características a dualidade entre a detecção e a precisão de localização das bordas, ou seja, quanto mais se privilegia a precisão de localização, menor é a razão sinal/ruído e, consequentemente, a detecção se torna cada vez mais sensível aos detalhes espúrios da imagem, e vice-versa [9]. Assim sendo, para cada imagem é necessária a escolha de um valor particular para o fator de escala do núcleo de convolução gaussiano. No software, essa variação do parâmetro é desnecessária devido à aplicação do filtro preto e branco anteriormente, que fornece um alto contraste entre a mancha e a pele.

A Fig. 6 também mostra o resultado do algoritmo de Canny aplicado à imagem de entrada. Pode-se notar que a borda é extraída com bastante precisão, embora dependa fortemente da qualidade da imagem de entrada. A aplicação do Canny não garante que a borda da mancha será um contorno fechado. Se a borda for uma linha contínua, o módulo "Identificador de irregularidades" irá realizar uma interpolação entre pontas da borda, a fim de fechá-la.

A partir da borda extraída pelo algoritmo de Canny, utilizou-se outro algoritmo para calcular as coordenadas de cada pixel da borda, primeiro em coordenadas retangulares, e em seguida, transformando-as para coordenadas polares. É feito um processamento de modo a obter um único valor de raio para cada ângulo inteiro. Em seguida, utilizando os pontos em coordenadas retangulares e uma rotina de regressão elíptica, aproxima-se a mancha pela elipse cujo erro quadrático seja o menor possível. Os pontos da elipse encontrada são novamente transformados para coordenadas polares, obtendo-se um raio para cada ângulo. O raio da elipse é subtraído do raio da borda. Assim, elimina-se a variação da borda em função do formato da mancha, ficando-se apenas com as flutuações relativas à irregularidade da borda. Obtém-se um gráfico, como mostrado na Fig. 7, que representa as saliências e reentrâncias da borda. Pode-se imaginar que as manchas são elípticas e essa flutuação é uma textura aplicada a elas. Cabe ressaltar que esse algoritmo proposto não é afetado pela posição da mancha, nem sua direção.

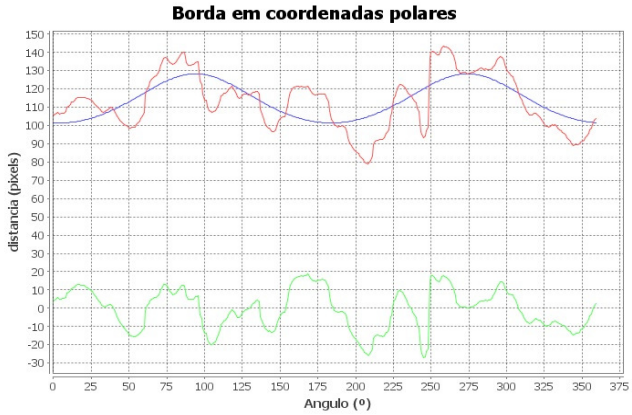


Fig. 7. Borda em coordenadas polares. A linha em vermelho representa a borda da lesão, a linha em azul mostra a melhor elipse que se aproxima do formato da lesão e a linha verde representa a diferença entre a borda da mancha e da elipse.

A área dessa função de flutuação é extraída, e sua grandeza indicará o quão irregular é a borda da mancha. Esse parâmetro não é suficiente para fazer tal afirmação; por isso, calcula-se a ondulação da borda tomando-se o maior e o menor picos, e fazendo-se a diferença entre eles. Em seguida, essa diferença é dividida pelo maior raio da mancha, obtendo-se assim um número representativo da amplitude de variação da borda. Esse número é utilizado como outro parâmetro para determinar se a borda é ou não irregular.

Para extração das cores internas à mancha, decidiu-se por utilizar dois métodos. Primeiro, a partir da imagem original, aplica-se o filtro SRM a fim de detectar na mancha diferentes regiões de cores. Depois, separa-se a mancha do restante da imagem, verificando-se todas as cores presentes na borda da imagem e transformando todos os pixels com essas cores em preto. Assim, a mancha fica isolada, com um fundo preto. Cabe ressaltar que esse método considera que a mancha está centralizada na figura e que não encosta na borda da imagem. A partir da imagem SRM, com a mancha isolada, é extraída a quantidade de cores diferentes presentes na mancha, além das porcentagens das duas maiores regiões em relação à área da mancha. Além disso, percebeu-se que, ao aplicar um filtro de contraste na imagem, diferentes cores ficavam mais exacerbadas. Decidiu-se então repetir o mesmo procedimento após a aplicação do filtro de contraste, e adicionar esse dado como um atributo extra para a análise das cores. A Fig. 8 apresenta a aplicação do filtro de contraste.

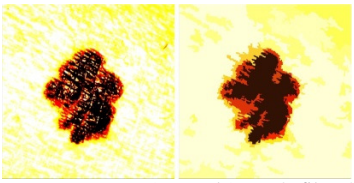


Fig. 8. - Imagem com contraste (esquerda) e após filtro SRM (direita).

Assim, o analisador de cores possui como parâmetros o número de cores presentes nas manchas, tanto com a aplicação do contraste quanto sem, e a área das maiores regiões da mancha comparada à área total da mancha. Ele considera o número de cores para a mancha com e sem contraste separadamente, sendo que quanto mais cores, maiores serão as chances de que a mancha possua mais de um tom. Para as áreas extraídas, o valor da primeira maior área é somado à da segunda maior, e tanto o primeiro valor sozinho quanto a soma dos dois são comparados com a área total. Considera-se que quanto maiores forem essas áreas, mais chances a mancha tem de ser homogênea.

A extração da simetria utiliza um algoritmo que, a partir de uma máscara branco e preto da mancha, identifica o baricentro da mancha e traça círculos concêntricos a esse baricentro para identificar eixos de simetria. Com isso diz-se que quanto mais eixos de simetria a mancha tiver, mais simétrica ela será.

Por fim, o diâmetro da mancha é inserido manualmente no software pelo usuário. No momento não há uma forma de extrair o diâmetro automaticamente pois as fotos não trazem informação de escala.

V. ÁRVORE DE DECISÃO ADAPTATIVA

As árvores de decisão são dispositivos que representam funções discretas sobre um conjunto de variáveis de natureza hierárquica [1]. Elas facilitam a inspeção e uso por seres humanos. Os nós internos de uma árvore de decisão são responsáveis por realizar testes sobre alguma variável V, e de cada nó parte uma aresta para cada possível valor assumido pela variável [1]. O tomador de decisão do SKAN é formado por uma árvore de decisão adaptativa, que pode criar e excluir nós dinamicamente de forma a se adaptar e melhorar seu índice de acertos [10].

O treinamento é a etapa inicial do processo de decisão do SKAN. Um conjunto de imagens com manchas de melanomas e não-melanomas é submetido ao sistema para que ele possa montar suas árvores de decisão. Os atributos extraídos de cada imagem do treinamento são confrontados com sua classificação, previamente feita pelo especialista. Esse processo está divido em duas fases: a primeira consiste na criação de quatro árvores de decisão básicas, uma para cada regra ABCD; e a segunda consiste na modificação e adaptação dessas árvores, submetendo-se a elas imagens de treinamento, de forma que adquiram uma maior capacidade de diferenciação dos resultados encontrados.

Dado um parâmetro P_i, que pode assumir valores no intervalo$]L_0, L_n[$, podemos representar um nó da árvore como mostra a Fig. 9.

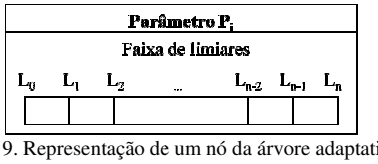


Fig. 9. Representação de um nó da árvore adaptativa.

O nó é formado por um parâmetro e um conjunto de valores possíveis que esse parâmetro pode atingir. Essa faixa é dividida por valores, chamados de limiares, que determinam as arestas da árvore. Cada intervalo $]L_i, L_{i+1}[$ é um ponteiro que indica o próximo nó dá arvore. O sistema pode alterar a faixa de limiares acrescentando e removendo intervalos, a fim de mudar o comportamento da árvore.

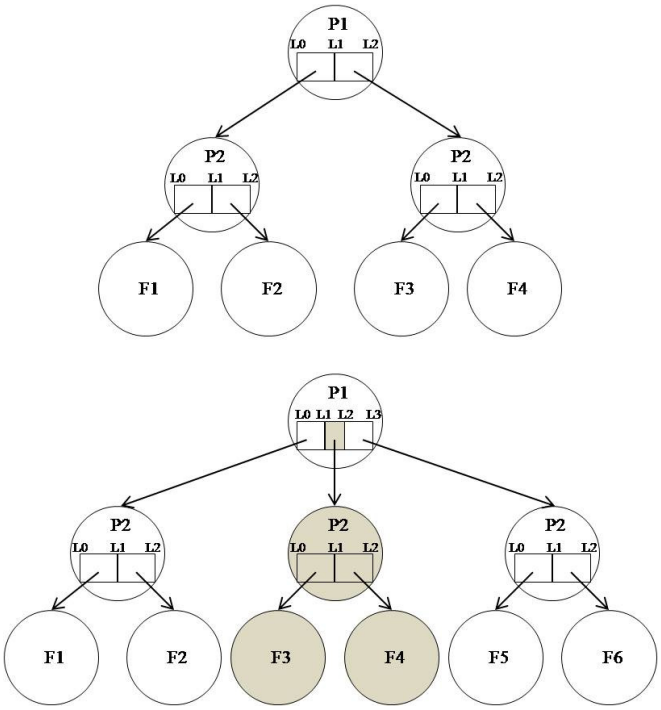


Fig. 10. Adaptatividade aplicada à árvore de decisão

A Fig. 10 mostra uma árvore binária com dois parâmetros de teste. Se o especialista ou o sistema perceber que o seu comportamento não é satisfatório, poderá alterar a estrutura da árvore criando ou excluindo nós. No exemplo, o sistema criou um novo intervalo na faixa de limiares do parâmetro P_1 com uma aresta para um novo ramo da árvore. Esse ramo irá tratar de um caso particular que o sistema incorporou. Analogamente, o mesmo raciocínio pode ser aplicado ao parâmetro P_2. Cabe observar que a faixa de limiares não é única para o mesmo parâmetro, ou seja, a árvore poderá conter nós do parâmetro P_2 com faixas de limiares diferentes, dando mais flexibilidade à decisão dependendo do caminho que a análise seguiu na árvore.

O número de níveis da árvore também pode ser alterado, caso seja necessária a inserção ou remoção de parâmetros extras. Por fim, as folhas contêm informações que indicam o resultado da análise.

A segunda fase consiste em resubmeter as imagens de treinamento para adaptação da árvore.

Quando uma imagem que está percorrendo a árvore chega a uma folha, é feita uma comparação com a classificação original da mancha, dada pelo especialista. O resultado dessa comparação é registrado na folha, rotulando-a como positiva, indicando característica de câncer, ou negativa, caso contrário. Assim, cada folha representa uma combinação de resultados dos testes realizados nos nós pelos quais passou. Mesmo as folhas que antes eram consideradas indecisas passam a ter uma característica de decisão, embora ainda sejam diferenciadas das que já eram positivas ou negativas por apresentarem todos os testes com resultados negativos ou todos com resultados positivos. Uma vez realizada essa segunda fase, outras imagens que incidam nessa mesma folha poderão ser diagnosticadas como sendo também de característica positiva ou negativa.

Além disso, há a possibilidade de existirem folhas sobre as quais não incidiram imagens, e que, portanto, não servem como dado para a conclusão de um diagnóstico, gerando um resultado de indecisão com relação à característica avaliada. Quando duas imagens, uma positiva e uma negativa, incidem sobre a mesma folha há a necessidade de adaptação da árvore, a fim de conseguir diferenciar imagens similares.

A ideia do algoritmo é percorrer em ordem ascendente os nós que levaram até aquela folha, procurando um parâmetro para o qual o valor do atributo da imagem divergente esteja afastado do valor do mesmo atributo para as outras imagens, que incidiram previamente sobre a folha. Achado esse parâmetro, divide-se esse limiar em dois, de forma a separar a imagem divergente das outras, e criar uma nova sub-árvore.
Caso em nenhum dos parâmetros o valor do atributo da imagem divergente esteja isolado dos outros, retorna-se para o nó mais próximo da folha em que o problema foi detectado e divide-se o limiar que aponta para a folha em três, de modo que o valor do atributo para a imagem divergente fique na faixa central. Cria-se, então, a nova sub-árvore a partir dessa faixa intermediária.

Após a montagem da árvore adaptativa na etapa de treinamento, o programa está apto a realizar diagnósticos. Os atributos extraídos das manchas de teste são submetidos às quatro sub-árvores de decisão, uma a uma, de forma a obter um diagnóstico de resultado para cada uma das quatro primeiras regras do método ABCDE. Cada um desses resultados pode ser: positivo, indicando que a mancha contém o problema que a regra se dispõe a identificar, como borda irregular ou heterogeneidade de cores; negativo, indicando o oposto; ou indeciso, indicando que até o momento o programa não possui condições para produzir um resultado com relação a essa imagem. Tendo esses resultados para cada uma das regras, o SKAN decide se a imagem tem chance ou não de ser câncer, com base nos resultados do treinamento, nos diagnósticos realizados previamente, e conceitos teóricos do método ABCDE.

Portanto, o diagnóstico se baseia nos dados históricos utilizando uma estrutura de dados que armazena os resultados de manchas já diagnosticadas. Assim, o software pode procurar nesta estrutura uma imagem com as mesmas características que já tenha sido diagnosticada. Em caso positivo, o diagnóstico será o mesmo. Caso contrário, apresenta-se a interface de correção de diagnóstico para que o especialista indique ao programa as características da mancha.

Essa correção irá alterar a topologia da árvore de decisão, de forma que ela se ajuste ao novo caso. Se o erro tiver acontecido por má interpretação dos atributos extraídos da imagem, as informações fornecidas pelo médico permitirão ao sistema identificar esse erro na árvore, corrigindo-o espontaneamente.

VI. RESULTADOS E CONTRIBUIÇÕES

A Tabela I ilustra os resultados finais com 20 imagens de teste, sendo 10 melanomas e 10 não-melanomas. O treinamento foi realizado considerando a mesma quantidade de imagens e evitando um banco de dados com amostras viciadas.

TABELA I
RESULTADOS DOS TESTES

	Acertos sem adaptação do diagnóstico	Acertos com adaptação do diagnóstico
Assimetria	11/20	9/20
Borda	13/20	15/20
Cor	12/20	13/20
Diâmetro	20/20	20/20
Melanoma	**13/20**	**18/20**

Com relação à assimetria, pode-se concluir que somente a análise do número de eixos de simetria não é suficiente para um bom diagnóstico dessa característica. Além disso, algumas manchas têm o mesmo número de eixos, porém diagnósticos diferentes. Nesse caso, o resultado é impreciso e demanda mais parâmetros de análise para casos dessa natureza. Sugere-se que a assimetria tenha outros parâmetros de decisão, como por exemplo, o nível de precisão usado pelo algoritmo na extração dos eixos.

A análise da borda apresentou um bom resultado devido ao tratamento matemático envolvido no cálculo da potência e da ondulação. Isso prova que o algoritmo desenvolvido nesse projeto foi consistente e apropriado para as análises.

A análise de cor se mostrou um pouco ineficiente, e considera-se que o método para extração das cores deve ser refinado.

O diâmetro permite uma decisão bastante simples quando comparado às demais características e, aliado ao fato de ser inserido manualmente pelo usuário, permitiu que o programa sempre acertasse o seu diagnóstico.

Os testes de adaptatividade foram feitos com as mesmas imagens e na mesma seqüência, corrigindo-se os diagnósticos indecisos. A inserção da adaptatividade aumentou a assertividade do programa diminuindo a quantidade que casos com respostas inconclusivas. Os casos antes indecisos são solucionados para que os próximos que forem similares sejam facilmente resolvidos.

A principal contribuição desse trabalho é a criação de um sistema de pré-diagnóstico de melanoma com a aplicação da tecnologia adaptativa. A adaptatividade interfere positivamente na tomada de decisão conferindo uma base de conhecimento mais consistente para o software. Portanto, o nível de precisão dos diagnósticos pode aumentar com o aprendizado fornecido pela adaptatividade nas tomadas de decisão. Outra importante contribuição, dentro do âmbito interdisciplinar, é o algoritmo de extração de bordas das manchas, que pode ser estendido a diversas aplicações como um reconhecedor de formas fechadas.

VII. TRABALHOS FUTUROS

De um modo geral, o sistema de reconhecimento do SKAN não faz uso de todas as características visuais possíveis para a avaliação da imagem. Sugere-se uma análise mais detalhada das cores internas da mancha; adição de outras características que a regra ABCDE não leve em conta; e a utilização de métodos paralelos para caracterização de cada irregularidade da mancha, buscando um resultado mais refinado. A evolução da mancha não é analisada devido a falta de material para testes, mas pode ajudar na precisão dos diagnósticos.

Quanto à adição de características que a regra ABCDE não considera, tem-se a coceira, sangramento, e outras características visuais, muitas das quais podem ser percebidas com auxílio de um dermatoscópio [2]. A coceira e o sangramento dependeriam de informações clínicas sobre cada uma das fotos, e não se tem acesso a essas informações. Já outras características visíveis por meio de um dermatoscópio exigem que se tenham fotos feitas com o aparelho, que é pouco usado no Brasil, e torna esse tipo de imagem muito difícil de adquirir. No entanto, a adição desses parâmetros poderia ajudar e muito na identificação dos casos de melanoma, fornecendo ainda mais atributos para a análise e diferenciação entre os diversos casos.

Outro parâmetro a ser mais cuidadosamente considerado é a dimensão da mancha, de forma que seja possível sua extração automática. Propõe-se uma solução para esse problema, baseada em tamanhos relativos, na qual, ao fotografar a mancha, o operador insira no campo da foto um objeto de tamanho conhecido e de fácil aquisição, como uma moeda. Desse modo, seria possível obter por meio de software o tamanho da mancha relativo ao objeto, e conhecendo-se as dimensões do objeto, extrair diversas dimensões em valores absolutos, como maior e menor raio, entre outros.

Seria também interessante expandir as implementações de algoritmos para extração dos atributos. Não é preciso que uma característica seja analisada apenas por um algoritmo. Aliás, quanto mais diversificadas forem as informações extraídas para a avaliação de uma característica, e quanto mais processos analisarem essas informações, maior será a confiabilidade do resultado da análise para determinada característica.

Além de implementações que beneficiem diretamente a extração e análise de atributos e características, um conjunto de imagens estatisticamente representativo do que ocorre na população, utilizado para treinamento, poderia melhorar o desempenho do programa. Desse modo, os pesos de cada característica na determinação de um diagnóstico positivo seriam mais realistas.

A ordem de análise dos parâmetros pode influenciar nos resultados na tomada de decisão. Por isso, a adaptatividade [11] também pode ser aplicada na ordem de testes dos parâmetros. Acredita-se que a adaptatividade no módulo de extração de atributos possa também contribuir o aumento do índice de acertos. Na calibração dos parâmetros dos filtros, ela poderia ajudar na extração ótima das características da mancha, facilitando a análise e tomada de decisão.

É importante ressaltar que o pequeno número de amostras limitou as conclusões baseadas nos testes realizados. Os dados precisariam de tratamento estatístico de modo que se possa descobrir a função de distribuição dessas amostras. Com um número elevado de imagens, a melhora com a inserção da adaptatividade se tornaria mais visível.

Finalmente, para que o projeto alcance seu objetivo como quando da sua concepção, implementações equivalentes à feita para melanomas deveriam ser integradas ao SKAN. Desse modo, não apenas seria possível diagnosticar entre melanoma e não melanoma, mas entre diferentes tipos de câncer, ou outras doenças de pele. Quanto mais completo for o repertório

de análises que o programa executa, mais confiável será seu diagnóstico.

VIII. CONCLUSÕES

Este artigo apresenta um software desenvolvido como projeto de formatura e detalha os resultados alcançados pelo mesmo. O sistema extrai os principais atributos da lesão de pele e transforma-os em valores que podem ser submetidos ao dispositivo de decisão para diagnosticar um possível melanoma.

Uma outra aplicação do núcleo desse sistema fora do âmbito médico é o reconhecimento de contornos de formas, como folhas de vegetais, ou países. Há a possibilidade de adaptar o sistema para o reconhecimento de espécies de folhas a partir de seu formato. Também é possível identificar países, continentes ou outros territórios. Nesse caso, o reconhecimento torna-se ainda mais fácil, já que o contorno das fronteiras é constante. Já para folhas, exige-se um pouco mais de flexibilidade, embora essa possa ser provida pela árvore de decisão adaptativa. Todas essas aplicações envolvem uma abstração dos métodos e algoritmos de extração e comparação da borda previamente apresentados. Assim, pode-se perceber que esse projeto possui muito a contribuir nos campos de reconhecimento de imagens e tomada de decisão, além dessa aplicação para a área médica

REFERÊNCIAS

[1] PISTORI, H. e NETO, J.J. *AdapTree - Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas*. Anais Conferência Latino Americana de Informática - CLEI 2002. Montevideo, Uruguai, novembro, 2002.

[2] EHRSAM, E. (2009). *Dermoscopy*. Acesso em 13 de junho de 2009, disponível em http://dermoscopic.blogspot.com.

[3] RIVITTI, E., & SAMPAIO, S. A. (2007). *Dermatologia* (3ª ed.). Editora Artes Médicas.

[4] INCA – Instituto Nacional do Câncer. *Câncer de Pele – melanoma*. Acesso em 01 de novembro de 2009, disponível em http://www.inca.gov.br/conteudo_view.asp?id=335.

[5] ALVES, G. (2008). *Conheça o "ABCD" para diferenciar lesões de pele e câncer*. Acesso em 01 de novembro de 2009, publicado em http://www.belezaestetica.com/content/detail.asp?iArt=539&iType=2&iChannel=1.

[6] CELEBI, M. E. (s.d.). *Fast and Accurate Border Detection in Dermoscopy Images Using Statistical Region Merging*. Acesso em setembro de 2009, disponível em http://www1bpt.bridgeport.edu/~jelee/
[11] pubs/SPIE-MI07.pdf.

[7] LENSONE, E. d., POZ, A. P., & Nogueira, J. R. (setembro de 2009). *Detector de Bordas de Canny*. Acesso em 01 de novembro de 2009, disponível em http://www.sbmac.org.br/eventos/cnmac/cd_xxvii_cnmac/
[12] cd_cnmac/files_pdf/10348a.pdf.

[8] BUENO, M. L. (s.d.). *Detecção de Bordas através de Algoritmo Canny*. Acesso em Setembro de 2009, disponível em http://www2.prudente.unesp.br/area_doc/imai/pdi/ufsc_notas/Canny_ufsc.htm.

[9] SOUZA, J. B. (s.d.). *Apontamentos Sobre Processamento Digital de Sinais*. Acesso em Setembro de 2009, disponível em http://www.deetc.isel.ipl.pt/comunicacoesep/disciplinas/pds/modulo3_20070103_1743.pdf.

[10] TCHEMRA, A. H. (novembro de 2007). *Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão*. Revista IEEE América Latina , Volume 5 (número 7), pp. 552-556.

[11] NETO, J. J. (2001). *Adaptive Rule-Driven Devices - General Formulation and Case Study*. (B. W. Watson, & D. Wood, Eds.) *Implem*entation and Application of Automata 6th International Conference, CIAA 2001 , Vol. 2497, pp. 234-250.

Usando Adaptatividade na Identificação de Padrões

(21 Janeiro 2010)

R. Camargo, Luís Raunheitte

***Resumo*. Este artigo tem como objetivo, mostrar que para aplicações na área de reconhecimento de padrões, a adaptatividade pode em algumas situações, oferecer recursos que a tornam vantajosa em relação a outras técnicas. Entre as principais vantagens do uso da adaptatividade podem ser citadas: não impõe nenhuma notação particular, mas aceita qualquer formulação definida por regras; permite descrever de forma natural coisas dinâmicas complicadas; todos os dados úteis para o reconhecimento de padrões ficam disponíveis e restritos ao conjunto de regras; a evolução do conjunto de regras basta para um acompanhamento direto do aprendizado. Sua aplicabilidade será mostrada em um estudo a ser desenvolvido a partir de uma base de dados organizada, onde utilizando uma técnica de mineração de dados para elaboração de uma matriz de transição da cadeia de Markov e aplicando o conceito de convolução com procedimentos adaptativos, torna-se possível subsidiar processos de tomada de decisão.**

Palavras-chave: **Adaptatividade, padrões, conjunto de regras, base de dados, Markov.**

I. Introdução

A complexidade dos processos de identificação de padrões tem exigido estudos e pesquisas em diversas áreas, tanto na teoria como na prática em desenvolvimento de softwares.

Metodologias vêm sendo utilizadas para identificar padrões e obter conhecimento para posicionar empreendimentos de forma competitiva no mercado.

Este estudo aborda o uso dos formalismos adaptativos como uma alternativa para reconhecimento de padrões apresentado sua aplicabilidade onde haja necessidade, como por exemplo, auxiliar no processo de tomada de decisão.

Os sistemas computacionais que servem de apoio aos processos de tomada de decisão, geralmente, utilizam bancos de dados com informações coletadas previamente. Esses sistemas, além de apresentarem processos de realimentação de dados, permitem aos decisores criar vários cenários sobre um determinado problema de decisão [1]. Das análises dos cenários e da decisão tomada, outras informações são obtidas e armazenadas para servirem de entrada para processos semelhantes.

De acordo com [2], pesquisas em inteligência artificial permitem o desenvolvimento de ferramentas computacionais inteligentes que auxiliam os usuários na execução de tarefas, entre elas, identificando padrões e auxiliando nos processos de tomada de decisão. Estas ferramentas são associadas à inteligência humana, como por exemplo, à capacidade de aprender, raciocinar e solucionar problemas.

Segundo [3], a Tecnologia Adaptativa é uma área da computação cujos estudos e pesquisas sobre técnicas adaptativas possibilitam aos dispositivos adaptativos apresentar como característica principal a capacidade de se automodificarem sem interferência externa, alterando suas estruturas topológicas, adaptando-se às necessidades requeridas de problemas específicos. Essa característica pode, desta maneira, conferir aos métodos adaptativos a classificação de sistemas inteligentes.

O objetivo deste artigo é apresentar o modelo no reconhecimento de padrões, baseado nos fundamentos da tecnologia adaptativa. Como parte dos estudos de pesquisa, é proposto um algoritmo para o modelo (desenvolvido em MatLab), que combina técnicas adaptativas ao processo de convolução.

II. Tecnologia Adaptativa

Na área da Tecnologia Adaptativa existem inúmeros estudos de técnicas com aplicações em diversas áreas [3] [5], cujas contribuições têm estimulado o desenvolvimento de novos recursos computacionais.

Nos dispositivos adaptativos desenvolvidos, encontram-se formalismos conhecidos e tradicionais, tais como autômatos de pilha estruturados, *statecharts*, redes de Markov, gramáticas, árvores de decisão, tabelas de decisão, entre outros [5]. Isso mostra que há certa facilidade de uso das técnicas adaptativas, uma vez que [3] define um dispositivo adaptativo como um dispositivo formado por uma camada subjacente (núcleo do sistema) representada por um formalismo conhecido não-adaptativo e uma camada adaptativa, cujas funções agem sobre o núcleo, o que lhe confere a capacidade de automodificação.

As ações adaptativas são implementadas na camada adaptativa e são responsáveis pelas alterações no conjunto de

[1] R. Camargo – Universidade Presbiteriana Mackenzie (correspondência: R. Oscar Freire, 235 – ap.31- São Paulo, SP, Brasil – cep: 01426-001; e-mail: rubens.camargo@poli.usp.br).
[2] Luís Tadeu Mendes Raunheitte – Universidade Presbiteriana Mackenzie; e-mail: raunheitte@mackenzie.br

regras, gerando uma nova configuração do dispositivo [3]. De maneira geral, as ações adaptativas permitem que regras sejam consultadas, eliminadas ou incluídas no sistema.

Um exemplo de dispositivo dirigido por regras adaptativo é a Árvore de Decisão Não-Determinística Adaptativa definida em [5]. O dispositivo adaptativo simula uma árvore de decisão, que é percorrida a partir da sua raiz (regra inicial) e chega a uma folha (caso determinístico) ou encontra vários ramos a serem percorridos (não-determinístico), após uma seqüência de testes em cada nó (regra). Quando não é possível atingir uma folha, ações adaptativas podem ser executadas na sub-árvore, realizando ações de consultas ou de remoção da sub-árvore ou de inclusão de uma nova sub-árvore à estrutura não-adaptativa.

O dispositivo dirigido por regras adaptativo definido em [3], tem como núcleo um processo que permite descobrir padrões com adaptatividade.

III. CADEIA DE MARKOV

A rede de Markov pode ser vista como um sistema de estados e transições [8], semelhante a um autômato finito. Caracteriza-se de primeira ordem a probabilidade de um estado ser atingido dependendo apenas do estado atual, portanto um sistema estocástico, tratando a probabilidade entre estados, conforme Fig. 1.

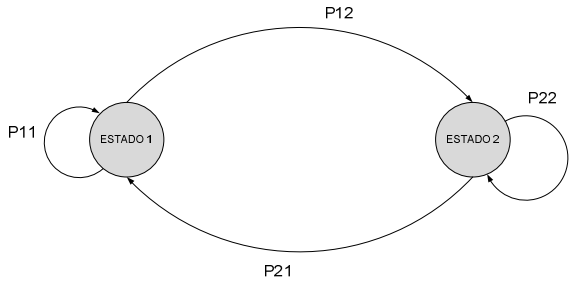


Fig. 1. transições de estados

Em [8], temos outra forma de representação do modelo de Markov a matriz de transição de estados, como sendo (fig. 2):

	Estado 1	Estado 2
Estado 1	0	1
Estado 2	0,49	0,51

Fig. 2. Quadro de transição de estados

Em outras palavras, a probabilidade do estado 1 seguir para o estado 2 é de 1 % , a probabilidade do estado 1 seguir para o estado 1 é de 0 %, a probabilidade do estado 2 seguir par o estado 1 é de 49 % e a probabilidade do estado 2 seguir para o estado 2 é de 51%.

Portanto:

$p_{11} = 0$ $p_{12} = 1$
$p_{21} = 0,49$ $p_{22} = 0,51$

onde a probabilidade do estado 1 seguir para o estado 2 é de 49%, existindo uma probabilidade 51 %, ligeiramente maior do estado 2 seguir para o estado 2.

Conforme [7], a cadeia de Markov é um caso particular de processo estocástico, com tempo discreto, que segue a propriedade de Markov. A definição desta propriedade, também chamada de memória markoviana, é que os estados anteriores são irrelevantes para a predição dos estados seguintes, desde que o estado atual seja conhecido.

Uma cadeia de Markov é uma sequência X_1, X_2, X_3, ... de variáveis aleatórias. O escopo destas variáveis, isto é, o conjunto de valores que elas podem assumir, é chamado de espaço de estados, onde X_n denota o estado do processo no tempo n. Se a distribuição de probabilidade condicional de X_{n+1} nos estados passados é uma função apenas de X_n, então:

$$Pr(X_{n+1} = x \mid X_0, X_1, X_2, \cdots, X_n) = Pr(X_{n+1} = x \mid X_n) \quad (1)$$

onde *x* é algum estado do processo. A identidade acima define a propriedade de Markov.

Seja P_{ij} a probabilidade condicional tal que se o sistema está no estado i em uma observação, então ele estará no estado j na próxima observação, $1 \leq i \leq N$, $1 \leq j \leq N$. Essas probabilidades chamadas de probabilidades de transição. Para cada cadeia de Markov, a matriz P(NxN) cujos elementos são P_{ij} é chamada de matriz de transição da cadeia de Markov (fig. 3), que no contexto da pesquisa será gerada a partir de valores obtidos no processo de mineração de dados.

$$P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{bmatrix}$$

Fig. 3. Matriz de transição

No contexto da pesquisa o sistema em estudo é descrito através de uma matriz de transição da Cadeia de Markov, a partir das probabilidades de ocorrências de seus diferentes estados.

IV. NÚCLEO DE CONVOLUÇÃO

Uma maneira de se entender a convolução é como uma operação que copia uma matriz a partir de cada localização de elemento para outra, considerando o valor de todos os elementos na área onde a cópia acontece, para produzir a alteração no valor original do elemento da matriz, [4].

Pode-se descrever a convolução como um processo de somas ponderadas. Cada elemento da matriz na vizinhança é multiplicado pelo seu similar no núcleo de convolução; a soma de todos os produtos resulta no novo valor do elemento central de interesse. Cada elemento do núcleo de convolução é um fator de ponderação (também chamado de coeficiente de convolução). O arranjo dos fatores de ponderação no núcleo, bem como o tamanho do núcleo, determina o tipo de transformação que será aplicada ao dado da representação matricial. Mudando um fator de ponderação no núcleo de convolução muda-se a magnitude e até o sinal de toda a soma afetando o valor atribuído ao elemento de interesse.

A convolução por soma ponderada apresenta um problema na sua aplicação nas fronteiras da matriz. Com o movimento do núcleo de convolução através da matriz, ao chegar a fronteira da mesma quando o elemento de interesse estiver na fronteira, uma parte dos coeficientes do núcleo não estarão sobre elementos da matriz. Uma maneira de contornar este problema é ignorar as fronteiras da matriz no cálculo, outra é duplicar os dados da fronteira, de forma a adicionar uma fronteira à matriz original, permitindo assim o cálculo sobre a fronteira original.

A operação de convolução substitui o valor do elemento da matriz pela soma de seu valor com a valor dos elementos das vizinhanças, tudo multiplicado por um fator chamado de "núcleo de convolução", supondo que se use uma vizinhança de 3X3 elementos, chamando de p(x,y) os pontos de matriz e os pontos do núcleo de N(x,y) onde x = 0, 1 ou 2, então o elemento central, p(1,1) será substituído pela soma dos pontos, vezes o valor do núcleo.

p(1,1) = p(0,0) * N(0,0) + p(1,0) * N(1,0) + p(2,0) * N(2,0) + p(0,1) * N(0,1) + p(1,1) * N(1,1) + p(2,1) * N(2,1) + p(2,2) * N(2,2).
(2)

ou

$$p(1,1) = \sum_{m,n}^{2} N(m,n) * p(m,n) \quad (3)$$

Esta é uma operação de correlação. Para convolução pode-se inverter a ordem dos valores do núcleo. A correlação é mais fácil de se entender e muitas convoluções de núcleo são simétricas, sendo equivalentes à correlação. Para convoluir uma área da matriz, deve-se repetir esta operação para cada posição de elemento na matriz de dados. Em cada ponto, deve-se multiplicar os valores do núcleo com os valores da matriz sobre ela, somar o resultado, e substituir o elemento do centro do núcleo com o valor. A equação então se torna:

$$p(x,y) = \sum_{m,n=0}^{2} N(m,n) * p(x+m, y+n) \quad (4)$$

Convoluir uma área de tamanho X por Y com um núcleo de tamanho m por n requer X∗Y∗m∗n multiplicações e somas. Então uma matriz de 256 por 256 com um núcleo de 3 por 3 requer 589.824 multiplicações e somas. A Fig. 4 representa o processo de convolução.

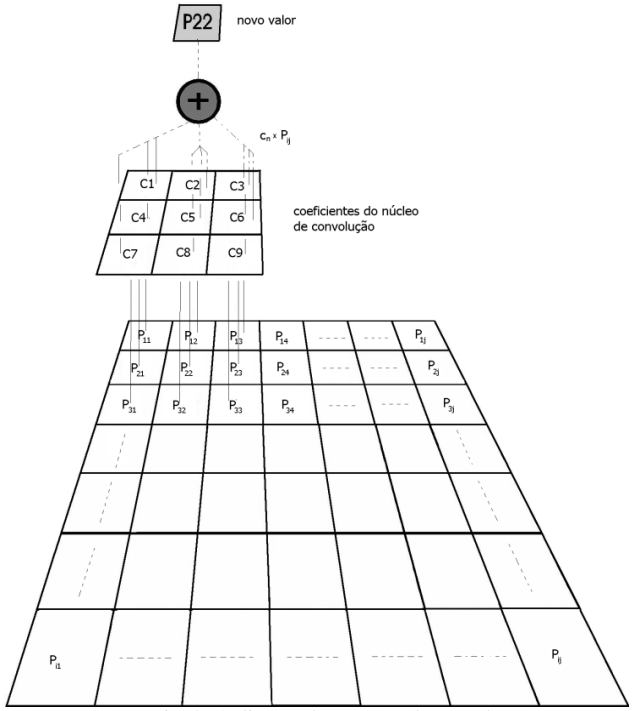


Fig. 4. Aplicação do processo de convolução

Quando se aplica a convolução para um problema de processamento de dados pensa-se na convolução como um filtro espacial. Em um filtro espacial, a convolução de núcleo é essencialmente para análise de uma pequena parte da matriz de dados que se quer amplificar ou detectar. Nesse contexto a escolha eficiente do núcleo pode detectar características na matriz de dados que permitam reconhecer padrões a partir da diferença de frequência dos resultados obtidos. Através da identificação de um determinado padrão o método proposto permitirá alterações no núcleo de convolução o que torna adaptativa a técnica. Essa identificação será feita a partir de um algoritmo que tomará como endereço de uma tabela de consulta podendo ser uma tabela de um banco de dados, este será o índice obtido na convolução, portanto este conteúdo do endereço indicado na tabela conterá os novos valores a serem usados como novo núcleo de convolução adaptativo

V. Conceitos de Mineração de Dados

Abordando pesquisas que tem como objetivo básico a identificação de padrões encontram-se estudos sobre Mineração de Dados ou Data Mining que tiveram origem em análise estatística na década de 60, que evoluíram, posteriormente nos anos 80, para novas técnicas de inteligência artificial, tais como lógica fuzzy, redes neurais, árvores de decisão [9].

Mineração de dados, segundo [10], consiste em um

conjunto de técnicas utilizadas na exploração de conjuntos de dados, normalmente mantidos em tabelas, formando um banco de dados. A mineração de dados tem como objetivo o descobrimento de relacionamentos complexos envolvendo conceitos, tais como: padrões, regras, fatos em dados armazenados.

Conforme [11], o processo de descoberta de conhecimento e mineração de dados (KDD*, Knowledge Discovery and Data Mining*), pode ser tratado em quatro etapas:

1. Seleção de dados: etapa para determinar o agrupamento de dados e atributos de interesse;
2. Limpeza dos dados: consiste na remoção de ruídos, na transformação de alguns campos e na criação de campos combinados;
3. Mineração dos dados: aplicação de algoritmos específicos para extrair padrões de interesse;
4. Avaliação: etapa em que os padrões descobertos são disponibilizados para os usuários em forma inteligível, facilitando a visualização.

Deve-se ressaltar que o conceito de padrão, segundo [11], é uma unidade de informação ou atributo de um registro que se repete, ou então é uma sequência de informações/atributos presentes em uma estrutura que se repete.

Existem cinco tipos de técnicas [10], usadas para a Mineração de Dados:

1. Associações: que identificam afinidades entre um conjunto de dados em um grupo de registros; por exemplo: 72% de todos os registros que contêm itens A, B e C, também contêm itens D e E; dessa maneira, regras associativas procuram estabelecer ligações entre um elemento e outro;
2. Padrões Sequênciais: que identificam sequências de registros que ocorrem em decorrência de outros. Por exemplo: na ocorrência de um evento A, 32% dos clientes com determinadas características realizarão o evento B, em um determinado espaço de tempo;
3. Classificação: que divide as classes predefinidas, permitindo que registros de uma classe permaneçam próximos. Exemplificando: poderia haver classes de registros quanto à frequência do comparecimento de clientes em uma agência bancária: infrequentes (nunca frequentam a agência), frequentes (comparecem de modo frequente) e ocasionais (ocasionalmente frequentam a agência);
4. Agrupamento: a partir da base de dados, descobre classes ocultas, enquanto que a classificação já inicia com classes predefinidas;
5. Previsão: que tem como objetivo o cálculo de previsão do valor futuro de uma variável, como por exemplo, prever uma determinada projeção de vendas, considerando registros devidamente classificados.

VI. RECONHECENDO PADRÕES COM ADAPTATIVIDADE

Adaptatividade conforme [6], refere-se a um conceito onde considerando a "experiência anterior" adquirida por um dispositivo adaptativo, baseado em um histórico de operações, um sistema pode tomar a decisão de modificar seu comportamento sem a inferência de qualquer agente externo. Conforme [5], o formalismo adaptativo se mostra uma opção a ser utilizada na aprendizagem computacional, destacando trabalhos realizados no reconhecimento de imagens, linguagens. Tratando o reconhecimento de padrões, [5] ainda destaca o reconhecimento ótico de caracteres (OCR), baseado em classes de técnicas de aprendizagem tais como: árvores de decisão, redes neurais artificiais, sentenças em lógica de predicados, conjunto de regras "se-então", autômatos, redes *bayesianas* e memorização (*instance-based learning*). Com relação as regras "se-então", pode-se entender como uma opção de fácil entendimento sem o domínio de conceitos complexos, permitindo uma maior transparência do modelo.

Uma proposta de trabalho com a utilização do conceito de adaptatividade no reconhecimento de padrões em uma base de dados, tem como abrangência os seguintes elementos:

a) a utilização da técnica de associação usada na Mineração de dados;
b) a partir de interações que classificam as informações armazenadas, será definida uma camada de aplicação da rede de Markov na identificação das decisões de agrupamento dos dados;
c) definição de uma camada de Convolução que aplicada sobre a rede de Markov gera o fator de ponderação, que determina o tipo de transformação aplicada na representação matricial.

Esta última camada será denominada Convolução Adaptativa, pois apresentará seus fatores de ponderação de forma variável, havendo a possibilidade de inclusão ou exclusão de novos fatores de ponderação, alterando os coeficientes do núcleo de convolução, face aos resultados obtidos (fig. 5).

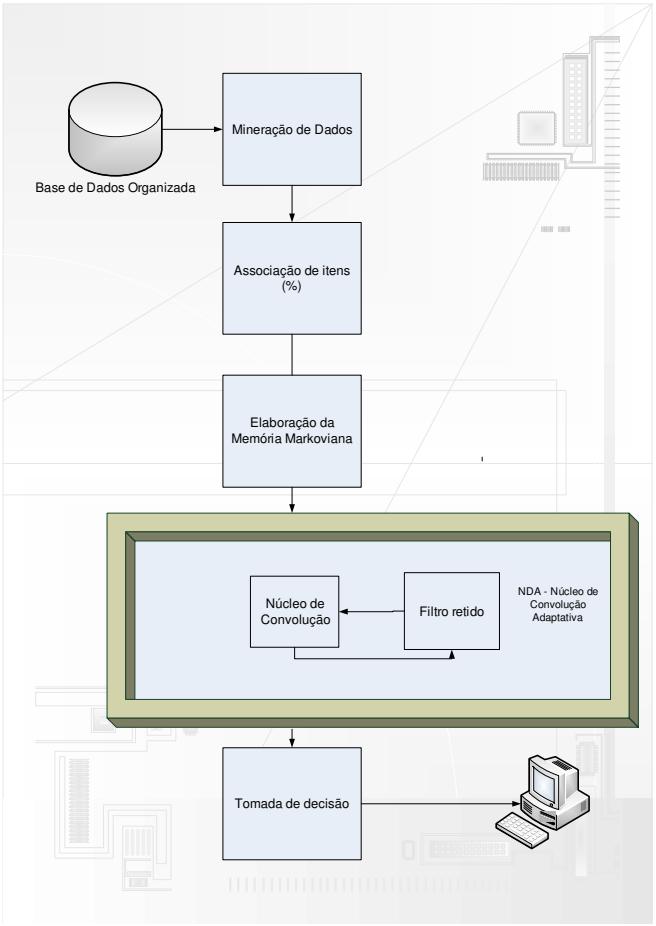


Fig. 5. Modelo do Estudo

VII. Modelo de Aplicação

Como modelo hipotético de aplicação da técnica adaptativa proposta foi utilizada uma Matriz de Transição (5), em duas transições de estados, e calculada a convolução considerando o núcleo (6).

Matriz de Transição da Cadeia de Markov

$$\begin{bmatrix} 0.3750 & 0.5000 & 0.1250 & 0.0000 \\ 0.250 & 0.4375 & 0.2500 & 0.0625 \\ 0.0625 & 0.2500 & 0.4375 & 0.2500 \\ 0.0000 & 0.1250 & 0.5000 & 0.3750 \end{bmatrix} \quad (5)$$

Núcleo hipotético de Convolução:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (6)$$

Como resultado da convolução temos:

$$\begin{matrix} 2.6875 & 3.0000 & 2.3125 & 1.1250 \\ 3.3125 & 4.0000 & 3.3125 & 2.0000 \\ 2.3125 & 3.0000 & 2.6875 & 1.8750 \\ 1.3750 & 2.0000 & 1.9375 & 1.5625 \end{matrix} \quad (7)$$

Calculando o histograma para o resultado obtido:

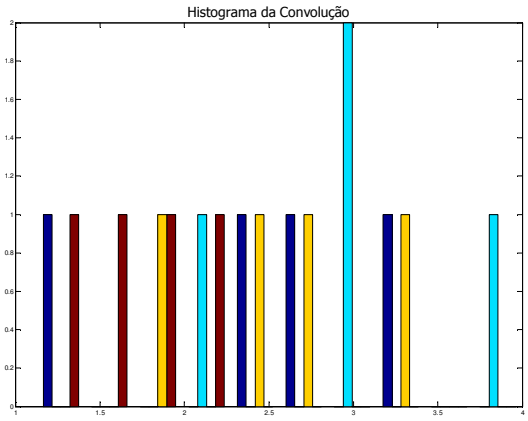


Fig. 6. Histograma da convolação

Pode-se notar que o histograma da convolução (Fig. 6), aplicada à Matriz de transição destaca uma determinada freqüência espacial que pode ser usada como referência em nova aplicação da técnica alterando, por exemplo, o núcleo de convolução.

O resultado da convolução será comparado com níveis pré-estabelecidos que indicarão a necessidade da utilização de novos núcleos de convolução, caso afirmativo um ponteiro será definido usando esse resultado. Assim o ponteiro indicará o endereço onde os coeficientes do novo núcleo estarão armazenados.

VIII. VII. Conclusão

Neste artigo são apresentados mecanismos adaptativos de identificação de padrões, objetivando a criação de um aplicativo de apoio a tomada de decisão, utilizando conceitos de mineração de dados, memória Markoviana, convolução e adaptatividade. É objetivo desta proposta criar um dispositivo adaptativo para aplicação em caráter genérico em processos de tomada de tomada de decisão.

Referências Bibliográficas

[1] PIDD, M., *Modelagem empresarial: ferramentas para tomada de decisão*. Trad. Gustavo Severo de Borba et al. Porto Alegre: Artes Médicas, 1998.

[2] O'BRIEN, J. A. *Sistemas de Informação e as Decisões Gerenciais na Era da Internet*. 2ed. Saraiva, São Paulo, 2004.

[3] NETO, J. J., *Adaptive Rule-Driven Devices - General Formulation and Case Study*. Lecture Notes in Computer Science. Watson, B.W. and

Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol. 2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250.
[4] HU, Osvaldo; RAUNHEITTE, Luís, *Processamento e Compressão Digital de Imagens.* Editora Mackenzie, São Paulo, 2004.
[5] PISTORI, H. *Tecnología Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações*. Tese de Doutorado – Escola Politécnica da USP, 2003.
[6] NETO, J.J., *Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa.* IEEE Latin America Transactions, Vol. 5, No. 7, Nov. 2007.
[7] MAKI, Daniel; THOMPSON, Maynard, *Mathematical Modeling and Computer Simulation*, Thomson Brooks/Cole, USA, 2006.
[8] BASSETO, Bruno Arantes., *Um Sistema de Composição Musical Automatizada, Baseado em Gramáticas Sensíveis ao Contexto, Implementado com Formalismos Adaptativos*. Dissertação de Mestrado - Escola Politécnica da USP, 2000.
[9] Kimbal, R., *The Data Warehouse Toolkit.* 2 ª Ed.Wiley Computer Publishing, USA, 2002.
[10] Moxon, B (1998). *Defining Data Mining-DBMS, Data Warehouse Supplement*, Aug.1996. HTTP//:dbmsmag.com/9608d53.html (30.Out.2008).
[11] RAMAKRISHNAN, R. *Sistema de Gerenciamento de Banco de Dados.* 3ª Ed. McGraw-Hill, São Paulo, 2008.

Rubens de Camargo é bacharel em Administração de Empresas, com especialização em Análise de Sistemas pela Faculdade Associadas de São Paulo e mestre em Administração de Empresas pela Universidade Presbiteriana Mackenzie. Atualmente é professor de cursos de graduação na Faculdade de Computação e Informática da Universidade Presbiteriana Mackenzie. É Doutorando do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

Luís Tadeu Mendes Raunheitte é Engenheiro Eletrônico pela Universidade Presbiteriana Mackenzie, mestre e doutor em Engenharia Elétrica pela mesma Universidade. Atualmente é professor de cursos de graduação na Faculdade de Computação e Informática da Universidade Presbiteriana Mackenzie.

Tecnologia Adaptativa Aplicada ao Processamento da Linguagem Natural

Ana Contier, Djalma Padovani, João José Neto

***Resumo*— Este trabalho faz uma breve revisão dos conceitos de Tecnologia Adaptativa, apresentando seu mecanismo de funcionamento e seus principais campos de aplicação, destacando o forte potencial de sua utilização no processamento de linguagens naturais. Em seguida são apresentados os conceitos de processamento de linguagem natural, ressaltando seu intricado comportamento estrutural. Por fim, é apresentado o Linguístico, uma proposta de reconhecedor gramatical que utiliza autômatos adaptativos como tecnologia subjacente.**

Palavras Chave*— *Autômatos Adaptativos, Processamento de Linguagem Natural, Reconhecedores Gramaticais, Gramáticas Livre de Contexto

I. Autômatos Adaptativos

O autômato adaptativo é uma máquina de estados à qual são impostas sucessivas alterações resultantes da aplicação de ações adaptativas associadas às regras de transições executadas pelo autômato [1]. Dessa maneira, estados e transições podem ser eliminados ou incorporados ao autômato em decorrência de cada um dos passos executados durante a análise da entrada. De maneira geral, pode-se dizer que o autômato adaptativo é formado por um dispositivo convencional, não-adaptativo, e um conjunto de mecanismos adaptativos responsáveis pela auto-modificação do sistema.

O dispositivo convencional pode ser uma gramática, um autômato, ou qualquer outro dispositivo que respeite um conjunto finito de regras estáticas. Este dispositivo possui uma coleção de regras, usualmente na forma de cláusulas if-then, que testam a situação corrente em relação a uma configuração específica e levam o dispositivo à sua próxima situação. Se nenhuma regra é aplicável, uma condição de erro é reportada e a operação do dispositivo, descontinuada. Se houver uma única regra aplicável à situação corrente, a próxima situação do dispositivo é determinada pela regra em questão. Se houver mais de uma regra aderente à situação corrente do dispositivo, as diversas possíveis situações seguintes são tratadas em paralelo e o dispositivo exibirá uma operação não determinística.

Os mecanismos adaptativos são formados por três tipos de ações adaptativas elementares: consulta (inspeção do conjunto de regras que define o dispositivo), exclusão (remoção de alguma regra) e inclusão (adição de uma nova regra). As ações adaptativas de consulta permitem inspecionar o conjunto de regras que definem o dispositivo em busca de regras que sigam um padrão fornecido. As ações elementares de exclusão permitem remover qualquer regra do conjunto de regras. As ações elementares de inclusão permitem especificar a adição de uma nova regra, de acordo com um padrão fornecido.

Autômatos adaptativos apresentam forte potencial de aplicação ao processamento de linguagens naturais, devido à facilidade com que permitem representar fenômenos linguísticos complexos tais como dependências de contexto. Adicionalmente, podem ser implementados como um formalismo de reconhecimento, o que permite seu uso no pré-processamento de textos para diversos usos, tais como: análise sintática, verificação de sintaxe, processamento para traduções automáticas, interpretação de texto, corretores gramaticais e base para construção de sistemas de busca semântica e de aprendizado de línguas auxiliados por computador.

Diversos trabalhos confirmam a viabilidade prática da utilização de autômatos adaptativos para processamento da linguagem natural. É o caso, por exemplo, de [2], que mostra a utilização de autômatos adaptativos na fase de análise sintática; [3] que apresenta um método de construção de um analisador morfológico e [4], que apresenta uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo algoritmo de Mitkov.

II. Processamento da Linguagem Natural: Revisão da Literatura

O processamento da linguagem natural requer o desenvolvimento de programas que sejam capazes de determinar e interpretar a estrutura das sentenças em muitos níveis de detalhe. As linguagens naturais exibem um intricado comportamento estrutural visto que são profusos os casos particulares a serem considerados. Uma vez que as linguagens naturais nunca são formalmente projetadas, suas regras sintáticas não são nem simples nem óbvias e tornam, portanto, complexo o seu processamento computacional. Muitos métodos são empregados em sistemas de processamento de linguagem natural, adotando diferentes paradigmas, tais como métodos exatos, aproximados, pré-definidos ou interativos, inteligentes ou algorítmicos [5]. Independentemente do método utilizado, o processamento da linguagem natural envolve as operações de análise léxico-morfológica, análise sintática, análise semântica e análise pragmática [6].

A análise léxico-morfológica procura atribuir uma classificação morfológica a cada palavra da sentença, a partir das informações armazenadas no léxico [7]. O léxico ou dicionário é a estrutura de dados contendo os itens lexicais e as informações correspondentes a estes itens. Entre as informações associadas aos itens lexicais, encontram-se a categoria gramatical do item, tais como substantivo, verbo e adjetivo, e os valores morfo-sintático-semânticos, tais como gênero, número, grau, pessoa, tempo, modo, regência verbal ou nominal. Um item lexical pode ter uma ou mais representações semânticas associadas a uma entrada. É o caso da palavra "casa", que pode aparecer das seguintes formas:

Casa: substantivo, feminino, singular, normal. Significado: moradia, habitação, sede
Casa: verbo singular, 3a pessoa, presente indicativo, 1ª conjugação. Significado: contrair matrimônio

Dada uma determinada sentença, o analisador léxico-morfológico identifica os itens lexicais que a compõem e obtém, para cada um deles, as diferentes descrições correspondentes às entradas no léxico. A ambiguidade léxico-morfológica ocorre quando uma mesma palavra apresenta diversas categorias gramaticais. Neste caso existem duas formas de análise: a tradicional e a etiquetagem. Pela abordagem tradicional, todas as classificações devem ser apresentadas pelo analisador, deixando a resolução de ambiguidade para outras etapas do processamento. Já pela etiquetagem (POS *Tagging*), o analisador procura resolver as ambiguidades sem necessariamente passar por próximas etapas de processamento. Nesta abordagem, o analisador recebe uma cadeia de itens lexicais e um conjunto específico de etiquetas como entrada e produz um conjunto de itens lexicais com a melhor etiqueta associada a cada item. Os algoritmos para etiquetagem fundamentam-se em dois modelos mais conhecidos: os baseados em regras e os estocásticos. Os algoritmos baseados em regras usam uma base de regras para identificar a categoria de um item lexical, acrescentando novas regras à base à medida que novas situações de uso do item vão sendo encontradas. Os algoritmos baseados em métodos estocásticos costumam resolver as ambiguidades através de um corpus de treino marcado corretamente, calculando a probabilidade que uma palavra terá de receber uma etiqueta em um determinado contexto.

O passo seguinte é a análise sintática. Nesta etapa, o analisador verifica se uma sequência de palavras constitui uma frase válida da língua, reconhecendo-a ou não. O analisador sintático faz uso de um léxico e de uma gramática, que define as regras de combinação dos itens na formação das frases. A gramática adotada pode ser escrita por meio de diversos formalismos. Segundo [7] destacam-se as redes de transição, as gramáticas de constituintes imediatos (PSG ou phrase structure grammar), as gramáticas de constituintes imediatos generalizadas (GPSG) e as gramáticas de unificação funcional (PATR II e HPSG). As gramáticas de constituintes imediatos (PSG), livres de contexto, apresentam a estrutura sintática das frases em termos de seus constituintes. Por exemplo, uma frase (F) é formada pelos sintagmas nominal (SN) e verbal (SV). O sintagma nominal é um agrupamento de palavras que tem como núcleo um substantivo (Subst) e o sintagma verbal é um agrupamento de palavras que tem como núcleo um verbo. Substantivo e verbo representam classes gramaticais. O determinante (Det) compõe, junto com o substantivo, o sintagma nominal. O sintagma verbal é formado pelo verbo, seguido ou não de um sintagma nominal. O exemplo apresentado ilustra uma gramática capaz de reconhecer a frase: O menino usa o chapéu.

F → SN, SV.
SN → Det, Subst.
SV → Verbo, SN.
Det → o
Subst → menino, chapéu
Verbo → usa

No entanto, este formalismo não consegue identificar questões de concordância de gênero e número. Por exemplo, se fossem incluídos no léxico o plural e o feminino da palavra menino, frases como: "O meninos usa o chapéu." e "O menina usa o chapéu." seriam aceitas. Para resolver este tipo de problema existem outros formalismos, tais como o PATR II:

F → SN, SV
  <SN numero> = <SV numero>
  <SN pessoa> = <SV pessoa>
SN → Det, Subst
  <Det numero> = <Subst numero>
  <Det genero > = <Subst genero>
SV → Verbo, SN
o
  <categoria> = determinante
  <genero> = masc
  <numero> = sing
menino
  <categoria> = substantivo
  <genero> = masc
  <numero> = sing
chapéu
  <categoria> = substantivo
  <genero> = masc
  <numero> = sing
usa
  <categoria> = verbo
  <tempo> = pres
  <numero> = sing
  <pessoa> = 3
  <argumento 1> = SN
  <argumento 2> = SN

Neste formalismo, a derivação leva em consideração outras propriedades do léxico, além da categoria gramatical, evitando os erros de reconhecimento apresentados anteriormente. Segundo [7], esse formalismo gramatical oferece poder gerativo e capacidade computacional, e tem sido usado com sucesso em ciência da computação, na especificação de linguagens de programação.

Certas aplicações necessitam lidar com a interpretação das frases bem formadas, não bastando o conhecimento da estrutura, mas sendo necessário o conhecimento do significado dessas construções. Por exemplo, quando é necessário que respostas sejam dadas a sentenças ou orações expressas em língua natural, as quais, por exemplo, provoquem um movimento no braço de um robô. Ou quando é necessário extrair conhecimentos sobre um determinado tema a partir de uma base de dados textuais. Nos casos nos quais há a necessidade de interpretar o significado de um texto, a análise léxico-morfológica e a análise sintática não são suficientes, sendo necessário realizar um novo tipo de operação, denominada análise semântica [7].

Na análise semântica procura-se mapear a estrutura sintática para o domínio da aplicação, fazendo com que a estrutura ganhe um significado [8]. O mapeamento é feito identificando as propriedades semânticas do léxico e o relacionamento semântico entre os itens que o compõe. Para representar as propriedades semânticas do léxico, pode ser usado o

formalismo PATR II, já apresentado anteriormente. Para a representação das relações entre itens do léxico pode ser usado o formalismo baseado em predicados: cada proposição é representada como uma relação predicativa constituída de um predicado, seus argumentos e eventuais modificadores. Um exemplo do uso de predicados é apresentado para ilustrar o processo de interpretação da sentença "O menino viu o homem de binóculo". Trata-se de uma sentença ambígua da língua portuguesa, uma vez que pode ser interpretada como se (a) O menino estivesse com o binóculo, ou (b) O homem estivesse com o binóculo. Uma gramática para a análise do exemplo acima é dada pelas seguintes regras de produção:

F → SN SV
SN → Det Subst
SN → SN SP
SV → V SN
SV → V SN SP
SP → Prep Subst

Uma possível representação semântica para as interpretações da sentença seria:

1. Sentença de interpretação (a):
   agente(ação(ver), menino)
   objeto(ação(ver), homem)
   instrumento(ação(ver), binóculo)
2. Sentença de interpretação (b):
   agente(ação(ver), menino)
   objeto(ação(ver), homem)
   qualificador(objeto(homem), binóculo)

Existem casos em que é necessário obter o conteúdo não literal de uma sentença, ligando as frases entre si, de modo a construir um todo coerente, e interpretar a mensagem transmitida de acordo com a situação e com as condições do enunciado [7]. Por exemplo, para uma compreensão literal da sentença: "O professor disse que duas semanas são o tempo necessário", é possível recorrer aos mecanismos de representação expostos até aqui, porém para uma compreensão aprofundada, seria necessário saber a que problema se refere o professor, já que o problema deve ter sido a própria razão da formulação dessa sentença. Nestes casos, é necessária uma nova operação denominada análise pragmática.

A análise pragmática procura reinterpretar a estrutura que representa o que foi dito para determinar o que realmente se quis dizer [2]. Dois pontos focais da pragmática são: as relações entre frases e o contexto. À medida que vão sendo enunciadas, as sentenças criam um universo de referência, que se une ao já existente. A própria vizinhança das sentenças ou dos itens lexicais também constitui um elemento importante na sua interpretação. Assim, alguns novos fenômenos passam a ser estudados, como fenômenos pragmático-textuais. Inserem-se nessa categoria as relações anafóricas, co-referência, determinação, foco ou tema, dêiticos e elipse [7]. Por exemplo, nem sempre o caráter interrogativo de uma sentença expressa exatamente o caráter de solicitação de uma resposta. A sentença "Você sabe que horas são?" pode ser interpretada como uma solicitação para que as horas sejam informadas ou como uma repreensão por um atraso ocorrido. No primeiro caso, a pergunta informa ao ouvinte que o falante deseja obter uma informação e, portanto, expressa exatamente o caráter interrogativo. Entretanto, no segundo caso, o falante utiliza o artifício interrogativo como forma de impor sua autoridade. Diferenças de interpretação desse tipo claramente implicam interpretações distintas e, portanto, problemáticas, se não for considerado o contexto de ocorrência do discurso [9]. As questões relacionadas à análise pragmática são objetos de estudos de modo a prover mecanismos de representação e de inferência adequados, e raramente aparecem em processadores de linguagem natural [7].

Em [10] são apresentados diversos artigos que mostram pesquisas atuais em processamento de linguagem natural para a Língua Portuguesa. Entre outros, apresenta-se a experiência do Núcleo Interinstitucional de Linguística Aplicada (NILC) no desenvolvimento de ferramentas para processamento de linguagem natural; o projeto VISL – Visual Interactive Syntax Learning, sediado na Universidade do Sul da Dinamarca, que engloba o desenvolvimento de analisadores morfossintáticos para diversas línguas, entre as quais o português; e o trabalho de resolução de anáforas desenvolvido pela Universidade de Santa Catarina. A tecnologia adaptativa também tem contribuído com trabalhos em processamento da linguagem natural. Em [11], são apresentadas algumas das pesquisas desenvolvidas pelo Laboratório de Linguagens e Tecnologia Adaptativa da Escola Politécnica da Universidade de São Paulo: um etiquetador morfológico, um estudo sobre processos de análise sintática, modelos para tratamento de não-determinismos e ambigüidades, e um tradutor texto-voz baseado em autômatos adaptativos.

III. Reconhecedor Adaptativo: Suporte Teórico Linguístico

Para o reconhecedor aqui proposto foi escolhida a Moderna Gramática Brasileira de Celso Luft [12] como suporte teórico linguístico, porque categoriza de forma clara e precisa os diversos tipos de sentenças de língua portuguesa, se diferenciando das demais gramáticas que priorizam a descrição da língua em detrimento da análise estrutural da mesma.

Luft diz que a oração é moldada por padrões denominados frasais ou oracionais. Estes padrões são compostos por elementos denominados sintagmas. Sintagma é qualquer constituinte imediato da oração, podendo exercer papel de sujeito, complemento (objeto direto e indireto), predicativo e adjunto adverbial. É composto por uma ou mais palavras, sendo que uma é classificada como núcleo e as demais como dependentes. As palavras dependentes podem estar localizadas à esquerda ou à direita do núcleo. Luft utiliza os seguintes nomes e abreviaturas:

1. Sintagma substantivo (SS): núcleo é um substantivo;
2. Sintagma verbal (SV): núcleo é um verbo;
3. Sintagma adjetivo (Sadj): núcleo é um adjetivo;
4. Sintagma adverbial (Sadv): núcleo é um advérbio;
5. Sintagma preposicional (SP): é formado por uma preposição (Prep) mais um SS.
6. Vlig: verbo de ligação
7. Vi: verbo intransitivo
8. Vtd: verbo transitivo direto
9. Vti: verbo transitivo indireto
10. Vtdi: verbo transitivo direto e indireto
11. Vt-pred: verbo transitivo predicativo

A Tabela 1 apresenta os elementos formadores dos sintagmas, e a sequência em que aparecem, de acordo com Luft.

TABELA 1.
Elementos formadores de sintagmas [12]

Sintagmas	
Substantivo	Quantitativos+Pronomes Adjetivos+
	Sintagma Adjetivo1+Substantivo+
	Sintagma Adjetivo2+
	Sintagma Preposicional+
	Oração Adjetiva
Verbal	Pré-verbais+
	Verbo Auxiliar+
	Verbo Principal
Adjetivo	Advérbio de Intensidade+
	Adjetivo+
	Sintagma Preposicional
Adverbial	Advérbio de Intensidade+
	Adverbio+
	Sintagma Preposicional
Preposicional	Preposição+
	Sintagma Substantivo

Um padrão oracional é determinado pelos tipos de sintagmas e pela sequência em que aparecem. Por exemplo, o padrão oracional SS Vlig SS, indica que a frase é composta por um sintagma substantivo, seguido de um verbo de ligação e de outro sintagma substantivo. A Tabela 2 apresenta a relação de todos os padrões oracionais propostos por Luft.

Os padrões são classificados em 5 tipos:

1. Padrões pessoais nominais: Neste caso, existe sujeito e o núcleo do predicado é um nome (substantivo, adjetivo, advérbio) ou um pronome (substantivo, adjetivo, advérbio). O verbo, nesses casos, é chamado de verbo de ligação (Vlig).

TABELA 2
Padrões oracionais de Luft [12]

Padrões Pessoais Nominais				
SS	Vlig	SS		
SS	Vlig	Sadj		
SS	Vlig	Sadv		
SS	Vlig	SP		
Padrões Pessoais Verbais				
SS	Vtd	SS		
SS	Vti	SP		
SS	Vti	Sadv		
SS	Vti	SP	SP	
SS	Vtdi	SS	SP	
SS	Vtdi	SS	Sadv	
SS	Vtdi	SS	SP	SP
SS	Vi			

2. Padrões pessoais verbais: São aqueles nos quais existe o sujeito e o núcleo do predicado é um verbo. O verbo pode ser transitivo direto (Vtd), transitivo indireto (Vti), transitivo direto e indireto (Vtdi), e intransitivo (Vi). Se o verbo for transitivo direto (Vtd), o complemento será um objeto direto; se o verbo for transitivo indireto (Vti), o complemento será um objeto indireto; se o verbo for transitivo direto e indireto (Vtdi), o complemento será um objeto direto e um indireto; se o verbo for intransitivo (Vi), não há complemento.

TABELA 2 - CONTINUAÇÃO

Padrões Pessoais Verbo-Nominais			
SS	Vtpred	SS	SS
SS	Vtpred	SS	Sadj
SS	Vtpred	SS	SP
SS	Vtpred	SS	Sadv
SS	Vtpred	SS	
SS	Vtpred	Sadj	
SS	Vtpred	SP	
Padrões Impessoais Nominais			
	Vlig	SS	
	Vlig	Sadj	
	Vlig	Sadv	
	Vlig	SP	
Padrões Impessoais Verbais			
	Vtd	SS	
	Vti	SP	
	Vi		

3. Padrões Pessoais Verbo-Nominais: Neste caso, existe o sujeito e o núcleo do predicado é um verbo transitivo predicativo (Vt-pred), cujo complemento é um objeto direto e um predicativo do objeto.

4. Padrões Impessoais Nominais: Ocorrem quando não existe sujeito e o núcleo do predicado é um nome (substantivo, adjetivo, advérbio) ou um pronome (substantivo, adjetivo, advérbio).

5. Padrões Impessoais Verbais: Neste caso, não existe sujeito e o núcleo do predicado é um verbo.

Luft apresenta uma gramática usada para análise sintática da Língua Portuguesa no modelo moderno, em que as frases são segmentadas o mais binariamente possível: Sujeito+Predicado; Verbo+Complemento; Substantivo+Adjetivo, etc. Neste modelo, a descrição explicita somente as classes analisadas; as funções ficam implícitas. Querendo explicar estas, Luft sugere que sejam escritas à direita das classes: SS:Sj (Sujeito), V:Núc (Núcleo do Predicado), PrA:NA (Adjunto Adnominal), etc.
A gramática proposta por Luft é a seguinte:

F → [Conec] [SS] SV [Conec]
Conec → F
SS → [Sadj] SS [Sadj | SP]
SS → [Quant | PrA] (Sc | Sp | PrPes)
SV→ [Neg] [Aux | PreV] (Vlig | Vtd| Vti | Vtdi| Vi)
[SS | Sadj | Sadv | SP] [SS| Sadj| Sadv| SP] [SP]
SP → Prep (SS | Sadj)
Sadj → Sadj [SP]
Sadj →[Adv] Adj

Sadv → Sadv [SP]
Sadv → [Adv] Adv
PrA → Ind | ArtDef | ArtInd| Dem| Pos
Sendo:
F – Frase
SS – Sintagma substantivo
SV – Sintagma verbal
SP – Sintagma preposicional
SN – Sintagma nominal
Sadv – Sintagma adverbial
Sadj – Sintagma adjetivo
Adv – adverbio
Adj – adjetivo
ArtDef – artigo definido
ArtInd – artigo indefinido
Aux – Partícula auxiliar (apassivadora ou pré-verbal)
Conec – Conector (conjunção ou pronome relativo)
Dem – pronome demonstrativo indefinido
Ind – pronome indefinido
Neg – partícula (negação)
PrA – pronome adjetivo
PrPes – pronome pessoal
Prep – preposição
Quant – numeral
Sc – substantivo comum
Sp – substantivo próprio
V– verbo
Vlig – verbo de ligação
Vi – verbo intransitivo
Vtd – verbo transitivo direto
Vti – verbo transitivo indireto
Vtdi – verbo transitivo direto e indireto

IV. PROPOSTA DE UM RECONHECEDOR GRAMATICAL

O Linguístico é uma proposta de reconhecedor gramatical composto de 5 módulos sequenciais que realizam cada qual um processamento especializado, enviando o resultado obtido para o módulo seguinte, tal como ocorre em uma linha de produção, até que o texto esteja completamente analisado.

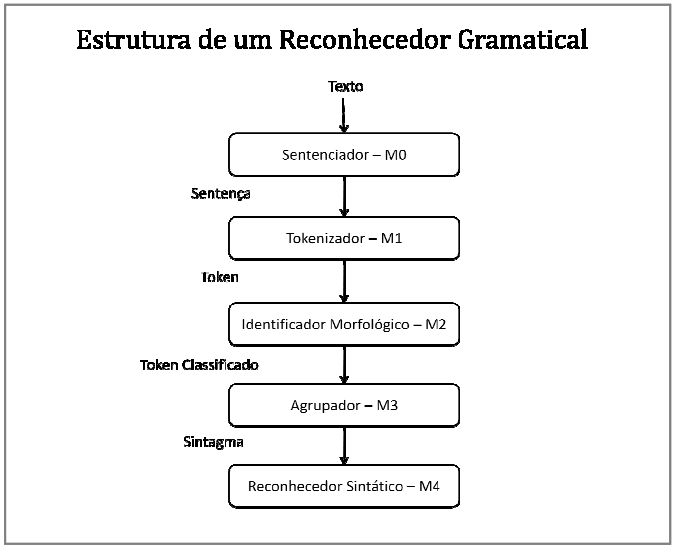


Figura 2. Estrutura para a proposta de um Reconhecedor Gramatical

A Fig.2 ilustra a estrutura do Linguistico. O primeiro módulo, denominado Sentenciador, recebe um texto e realiza um pré-processamento, identificando os caracteres que possam indicar final de sentença, eliminando hífens, apóstrofes e aspas, e substituindo vírgulas e ponto-e-vírgulas por caracteres especiais. Ao final, o Sentenciador divide o texto em supostas sentenças que são analisadas individualmente nas etapas seguintes.

O segundo módulo, denominado Tokenizador, recebe as sentenças identificadas na etapa anterior e as divide em *tokens*, considerando, neste processo, abreviaturas, valores monetários, numerais arábicos e romanos, nomes próprios, caracteres especiais e de pontuação final. Os *tokens* são armazenados em estruturas de dados (*arrays*) e enviados um a um para análise do módulo seguinte.

O terceiro módulo, denominado Identificador Morfológico, recebe os *tokens* da etapa anterior e os identifica morfologicamente, utilizando os textos pré-anotados do corpus Bosque[13] como biblioteca de apoio. O Bosque é um conjunto de frases anotadas morfossintaticamente (conhecido por *treebank),* composto por 9368 frases retiradas dos primeiros 1000 extratos dos corpora CETEMPublico (Corpus de Extractos de Textos Electrónicos MCT/Público) e CETENFolha (Corpus de Extractos de Textos Electrónicos NILC/Folha de S. Paulo). A Fig. 3 apresenta um fragmento do Bosque. Como o Bosque pode apresentar mais de uma classificação morfológica para o mesmo *token*, o Identificador Morfológico usa aquela que é mais frequente, identificando artigos, numerais, substantivos, nomes próprios, verbos, advérbios, adjetivos, preposições e conjunções. Em seguida, o Identificador usa um conjunto de regras heurísticas resultantes da análise prévia de textos de treinamento para resolver situações nas quais não foi encontrada uma classificação ou a classificação encontrada não é a mais adequada no contexto de análise, que é identificado pela classificação do último *token* analisado. Por exemplo, se não for encontrada classificação morfológica para um *token*, ele terminar em 'ado', ou 'ido', e a classificação do último *token* for verbo, então o Identificador Morfológico o classifica como verbo.

```
#9363 CF997-3 Segundo declarações do próprio diretor, ele vive até hoje de forma angustiada:
(STA+fcl (ADVL+pp
                (H+prp Segundo)
                (P<+np
                  (H+n declarações)
                  (N<+pp
                    (H+prp de)
                    (P<+np
                      (>N+art o)
                      (>N+pron-det próprio)
                      (H+n diretor)))))
                (,)
                (SUBJ+pron-pers ele)
                (P+v-fin vive)
                (ADVL+advp
                  (>A+prp até)
                  (H+adv hoje))
                (ADVL+pp
                  (H+prp de)
                  (P<+np
                    (H+n forma)
                    (N<+v-pcp angustiada)))
                (.))
```

Figura 3. Exemplo de frase etiquetada do corpus Bosque [11].

O quarto módulo, denominado Agrupador é composto de um autômato, responsável pela montagem dos sintagmas a partir

de símbolos terminais da gramática e um bigrama, responsável pela montagem dos sintagmas a partir de não-terminais. Inicialmente, o Agrupador recebe do Identificador as classificações morfológicas dos *tokens* e as agrupa em sintagmas de acordo com a gramática proposta por Luft. Neste processo são identificados sintagmas nominais, verbais, preposicionais, adjetivos e adverbiais Para isso, o Agrupador utiliza um autômato adaptativo cuja configuração completa é definida da seguinte forma:

Estados = {1, 2, 3, 4, SS, SP, V, Sadj, Sadv, A},
Onde:
1,2,3 e 4 = Estados Intermediários
SS, SP, V Sadj, Sadv = Estados nos quais houve formação de sintagmas, sendo:
SS= Sintagma substantivo
SP = Sintagma preposicional
V = Verbo ou locução verbal
Sadj = Sintagma adjetivo
Sadv = Sintagma adverbial
A = Estado após o processamento de um ponto final
Tokens = {art, num, n, v, prp, pron, conj, adj, adv, rel, pFinal, sClass}, onde:
art = artigo, num = numeral
n = substantivo, v = verbo
prp = preposição, pron = pronome
conj = conjunção, adj = adjetivo
adv = advérbio, rel = pronome relativo
pFinal = ponto final, sClass = sem classificação
Estados de Aceitação = {SS, SP, V, Sadj, Sadv, A}
Estado Inicial = {1}
Função de Transição = {(Estado, *Token*)→Estado}, sendo:
{(1, art)→2, (2, art)→2, (3, art)→3
(1, num)→Sadv, (2, num)→2, (3, num)→3
(1, n)→SS, (2, n)→SS, (3, n)→SP
(1, v)→SV, (2, v)→ SV, (3, v)→SP
(1, prp)→3, (2, prp)→2, (3, prp)→3
(1, prop)→SS, (2, prop)→SS, (3, prop)→SP
(1, pron)→SS, (2, pron)→SS, (3, pron)→SP
(1, conj)→conj, (2, conj)→Ø, (3, conj)→ Ø
(1, adj)→Sadj, (2, adj)→Sadj, (3, adj)→3
(1, adv)→Sadv, (2, adv)→2, (3, adv)→3
(1, rel)→conj, (2, rel)→ Ø, (3, rel)→conj
(1, pFinal)→A, (2, pFinal)→ Ø, (3, pFinal)→ Ø}

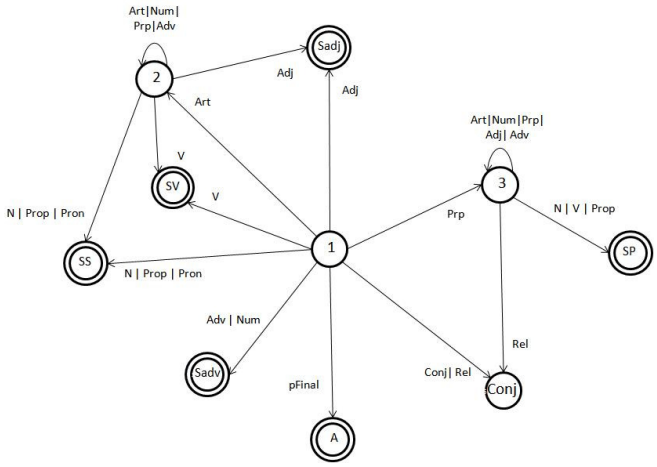


Figura 4. Configuração Completa do Autômato Construtor de Sintagmas.

Por exemplo, segundo a gramática de Luft, os sintagmas substantivos são obtidos através da seguinte regra:

SS → [Quant | PrA] (Sc | Sp | PrPes)

Pela regra acima, o conjunto de *tokens* "A" e "casa" formam um sintagma substantivo, da seguinte forma:

PrA = "A" (artigo definido)
Sc = "casa" (substantivo comum)

Da direita para esquerda, são realizadas as seguintes derivações:

PrA Sc → SS; A Sc → SS; A casa → SS

Já o Agrupador recebe o *token* "A", identificado pelo Tokenizador como artigo definido, e se movimenta do estado 1 para o estado 2. Ao receber o *token* "casa", identificado como substantivo comum, ele se movimenta do estado 2 para o estado SS, que é um estado de aceitação. Neste momento o Agrupador armazena a cadeia "A casa" e o símbolo "SS" em uma pilha e reinicializa o autômato preparando-o para um novo reconhecimento.

Em um passo seguinte, o Agrupador usa o bigrama para comparar um novo sintagma com o último sintagma formado, visando identificar elementos mais altos na hierarquia da gramática de Luft. Para isso ele usa a matriz apresentada na Tabela 3, construída a partir da gramática de Luft. A primeira coluna da matriz indica o último sintagma formado (US) e a primeira linha, o sintagma atual (SA). A célula resultante apresenta o novo nó na hierarquia da gramática.

TABELA 3
Matriz de agrupamento de sintagmas

US \ SA	SS	SP	V	Sadv	Sadj	Conj
SS	SS	SS	-	-	SS	-
SP	SP	-	-	-	-	-
V	-	-	V	-	-	-
Sadv	-	-	-	Sadv	Sadj	-
Sadj	SS	Sadj	-	-	Sadj	-
Conj	-	-	-	-	-	Conj

Esta técnica foi usada para tratar as regras gramaticais nas quais um sintagma é gerado a partir da combinação de outros, como é o caso da regra de formação de sintagmas substantivos: SS → [Sadj] SS [Sadj | SP]. Por esta regra, os sintagmas substantivos são formados por outros sintagmas substantivos precedidos de um sintagma adjetivo e seguidos de um sintagma adjetivo ou um sintagma preposicional. No exemplo anterior, supondo que os próximos 2 *tokens* fossem "de" e "madeira", após a passagem pelo autômato, o Agrupador formaria um sintagma SP. Considerando que na pilha ele tinha armazenado um SS, após a passagem pelo bigrama, e de acordo com a Tabela 3, o sintagma resultante seria um SS e o conteúdo que o compõe seria a combinação dos textos de cada sintagma que o originou. Caso não haja agrupamentos possíveis, o Agrupador envia o último sintagma formado para análise do Reconhecedor Sintático e movimenta

o sintagma atual para a posição de último sintagma no bigrama, repetindo o processo com o próximo sintagma.

O quinto e último módulo, denominado Reconhecedor Sintático, recebe os sintagmas do módulo anterior e verifica se estão sintaticamente corretos de acordo com padrões gramaticais de Luft. O Reconhecedor Sintático utiliza um autômato adaptativo que faz chamadas recursivas sempre que recebe conjunções ou pronomes relativos, armazenando, em uma estrutura de pilha, o estado e a cadeia de sintagmas reconhecidos até o momento da chamada. Caso o Reconhecedor Sintático não consiga se movimentar a partir do sintagma recebido, ele gera um erro e retorna o ponteiro para o último sintagma reconhecido, finalizando a instância do autômato recursivo e retornando o processamento para aquela que a inicializou. Esta, por sua vez, retoma posição em que se encontrava antes da chamada e continua o processamento até o final da sentença ou até encontrar uma nova conjunção, situação na qual o processo se repete.

A configuração completa do autômato é definida da seguinte forma:

Estados = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}

Tokens = {SS, SP, Vli, Vi, Vtd, Vti, Vtdi, Sadj, Sadv, Conj, A}

Estados de Aceitação = { 4, 5, 6, 9, 12, 13,14, 15, 17, 18, 19, 21, 22, 24, 25, 26, 27}

Estado Inicial = {1}

Função de Transição = {(Estado, *Token*)→Estado}, sendo:
{(1, SS)→2, (2, Vti)→3, (3, SP)→4, (4, SP)→4,
(3, Sadv)→5, (2, Vi)→6, (2, Vtdi)→7
(7, SS)→8, (8, SP)→9, (9, SP)→9, (8, Sadv)→10,
(2, Vlig)→11, (11, SP)→12, (11, Sadv)→13,
(11, Sadj)→ 14, (11, SS)→15, (2, Vtd)→16, (16, SS)→17,
(2, Vtpred)→18, (18, SP)→19, (18, Sadj)→20
(18, SS)→ 21, (21, SS)→22, (21, Sadj)→23, (21, Sadv)→24,
(21, SP)→25}

Pilha = {[Texto, Sintagma, Estado]}

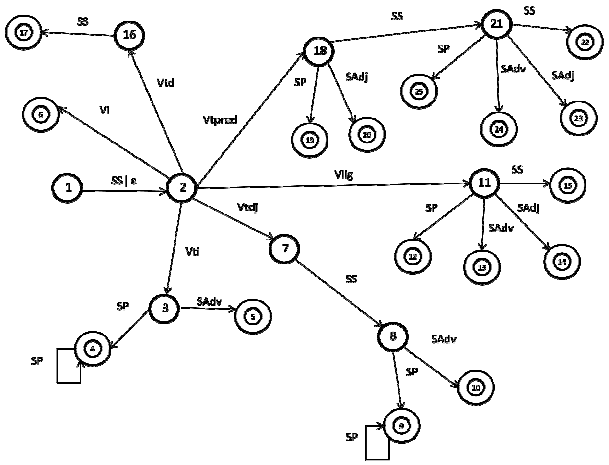


Figura 5.1. Configuração Completa do Reconhecedor Sintático.

No entanto, para que a análise sintática seja feita, não são necessárias todas as ramificações da configuração completa do autômato. Por exemplo, quando se transita um verbo de ligação a partir do estado 2, o autômato vai para o estado 11 e todas as demais ramificações que partem deste estado para os estados 3, 7, 16 e 18, não são usadas. Com a tecnologia adaptativa, é possível criar dinamicamente os estados e transições do autômato em função dos tipos de verbos, evitando manter ramificações que não são usadas.

A Fig. 5.2 apresenta a configuração inicial do autômato adaptativo equivalente ao autômato de pilha apresentado anteriormente. No estado 1, o autômato recebe os *tokens* e transita para o estado 2 quando processa um sintagma substantivo (SS) ou quando transita em vazio. No estado 2, o autômato transita para si mesmo quando recebe qualquer tipo de verbo: Vi, Vtd, Vlig, Vtpred, Vtdi e Vti. Todas as outras ramificações são criadas por meio de funções adaptativas chamadas em função do tipo de verbo processado.

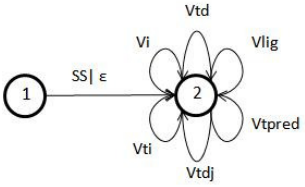


Figura 5.2. Configuração Inicial do Reconhecedor Gramatical.

Por exemplo, se o verbo é de ligação (Vlig), o autômato utiliza as funções adaptativas α (j) e β (o), definidas da seguinte forma:

α (j): { o* :
- [(j, Vlig)]
+[(j, Vlig) :→ o, β (o)]
}

β (o): { t*u*v*x*:
+ [(o, SP) :→ t]
+ [(o, Sadv) :→ u]
+ [(o, Sadj) :→ v]
+ [(o, SS) :→ x]

A função adaptativa α (j) é chamada pelo autômato antes de processar o *token*, criando o estado 11 e a produção que leva o autômato do estado 2 ao novo estado criado. Em seguida, o autômato chama a função β (o), criando os estados 12, 13, 14 e 15 e as produções que interligam o estado 11 aos novos estados. A Fig. 5.3 mostra a configuração do autômato após o processamento do verbo de ligação. Neste exemplo, o autômato criou apenas os estados 11, 12, 13, 14 e 15 e as respectivas transições, evitando alocar recursos que seriam necessários para criar o autômato completo, conforme apresentado na Fig. 5.1.

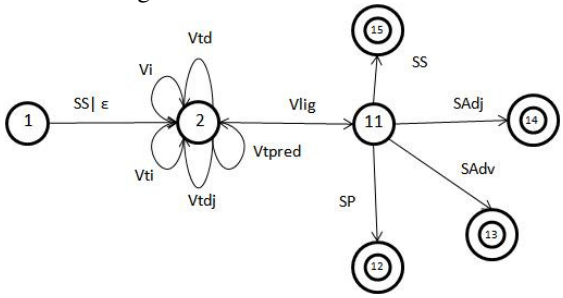


Figura 5.3. Configuração do autômato após o processamento do verbo de ligação.

Toda movimentação do autômato, assim como os sintagmas identificados em cada passagem e a classificação morfológica dos termos das sentenças, são armazenados em arquivos que podem ser acessados por um editor. Se o Linguistico não consegue reconhecer a sentença, ele registra os erros encontrados e grava uma mensagem alertando para o ocorrido.

V. Experimento

Foi realizado um experimento no qual a implementação conceitual descrita na seção anterior foi codificada. A linguagem de programação utilizada foi Python [14] por ela dispor de recursos de processamento de expressões regulares (*Regular Expression* – RE), processamento de linguagem natural (*Natural Language Tool Kit* – NLTK) e autômatos finitos (*Determinist Finite Automon* – DFA) necessários para a realização do trabalho. Segue um exemplo de texto analisado pelo reconhecedor e os resultados obtidos:

Texto:

"A Câmara de Representantes do Congresso americano deu início ao debate de um projeto de lei histórico para limitar as emissões de gases causadores do efeito estufa pela indústria dos Estados Unidos."

Resultados obtidos:

Classificação Morfológica:

```
[['A', 'art'], ['Câmara', 'prop'], ['de', 'prp'],
 ['Representantes', 'prop'], ['do', 'prp'], ['Congresso', 'prop'],
 ['americano', 'adj'], ['deu', 'v'], ['início', 'n'],
 ['ao', 'prp'], ['debate', 'n'], ['de', 'prp'], ['um', 'art'],
 ['projeto', 'n'], ['de', 'prp'], ['lei', 'n'], ['histórico', 'adj'],
 ['para', 'prp'], ['limitar', 'v'], ['as', 'art'], ['emissões', 'n'],
 ['de', 'prp'], ['gases', 'n'], ['causadores', 'adj'], ['do', 'prp'],
 ['efeito', 'n'], ['estufa', 'adj'], ['pela', 'prp'], ['indústria', 'n'],
 ['dos', 'prp'], ['Estados', 'prop'], ['Unidos', 'prop']]
```

Total de Sintagmas= 5

```
[[['A Câmara', 'SS', '2'],
  ['de Representantes do Congresso americano', 'SP', '2'],
  ['deu', 'V', '3'],
  ['início', 'SS', '4'],
  ['ao debate de um projeto de lei histórico para limitar as
    emissões de gases causadores do efeito estufa pela
    indústria dos Estados Unidos', 'SP', '9']]]
```

VI. Considerações Finais

Este artigo apresentou uma revisão dos conceitos de Tecnologia Adaptativa e de Processamento da Linguagem Natural. Em seguida, foi apresentado o Linguístico, uma proposta de reconhecedor gramatical que utiliza autômatos adaptativos como tecnologia subjacente. Ao final, foi apresentado um experimento realizado com o reconhecedor e os resultados obtidos.

IX. Referências

[1] NETO, J.J. Apresentação LTA-Laboratório de Linguagens e Técnicas Adaptativas. Disponível em: http://www.pcs.usp.br/~lta. Acesso 01/11/2009.

[2] TANIWAKI, C. Formalismos adaptativos na análise sintática de linguagem natural . Dissertação de Mestrado, EPUSP, São Paulo, 2001.

[3] MENEZES, C. E. Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos. Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, 2000

[4] PADOVANI, D. Uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo algoritmo de Mitkov. Workshop de Tecnologias Adaptativas – WTA 2009, 2009.

[5] MORAES, M. de Alguns aspectos de tratamento sintático de dependência de contexto em linguagem natural empregando tecnologia adaptativa, Tese de Doutorado, Escola Politécnica da Universidade de São Paulo, 2006.

[6] RICH, E.; KNIGHT, K. Inteligência Artificial**,** 2. Ed. São Paulo: Makron Books, 1993.

[7] VIEIRA, R.; LIMA, V Linguística computacional: princípios e aplicações. IX Escola de Informática da SBC-Sul, 2001.

[8] FUCHS, C., LE GOFFIC, P. Les Linguistiques Contemporaines.

[9] NUNES, M. G. V. et al. Introdução ao Processamento das Línguas Naturais. Notas didáticas do ICMC Nº 38, São Carlos, 88p, 1999.

[13] Paris, Hachette, 1992. 158p.

[10] SARDINHA, T. B. A Língua Portuguesa no Computador. 295p. Mercado de Letras, 2005.

[11] ROCHA, R.L.A. Tecnologia Adaptativa Aplicada ao Processamento Computacional de Língua Natural. Workshop de Tecnologias Adaptativas – WTA 2007, 2007.

[12] LUFT, C. Moderna Gramática Brasileira. 2ª. Edição Revista e Atualizada. 265p. Editora Globo, 2002.

[13] Linguateca : http://www.linguateca.pt/

[14] Python. Python Programming Language – Official Website. Disponível em: <http://www.python.org>.

Ana Teresa Contier: formada em Letras-Português pela Universidade de São Paulo (2001) e em publicidade pela PUC-SP (2002). Em 2007 obteve o título de mestre pela Poli-USP com a dissertação: "Um modelo de extração de propriedades de textos usando pensamento narrativo e paradigmático".

Djalma Padovani nasceu em São Paulo em 1964. Cursou bacharelado em Física pelo Instituto de Física da Universidade de São Paulo, formou-se em administração de empresas pela Faculdade de Economia e Administração da Universidade de São Paulo, em 1987 e obteve o mestrado em engenharia de software pelo Instituto de Pesquisas Tecnológicas de São Paulo - IPT, em 2008. Trabalhou em diversas empresas nas áreas de desenvolvimento de software e tecnologia de informação e atualmente é responsável pela arquitetura tecnológica da Serasa S/A, empresa do grupo Experían.

João José Neto graduado em Engenharia de Eletricidade (1971), mestrado em Engenharia Elétrica (1975) e doutorado em Engenharia Elétrica (1980), e livre-docência (1993) pela Escola Politécnica da Universidade de São Paulo.

Atualmente é professor associado da Escola Politécnica da Universidade de São Paulo, e coordena o LTA - Laboratório de Linguagens e Tecnologia Adaptativa do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

Desenvolvimento de software para preparação semi-automática de atividades de leitura em inglês

J. L. Moreira Filho

Cube – A Knowledge Extraction System (November 30, 2009)

F. S. Komori, F.B. Colombo and M. N. P. Carreño

Abstract— **We describe an expert system that is currently in development and has as a goal automatic knowledge acquisition from documents written in plain English. The envisioned system will be able to answer simple questions based solely on information it has acquired unaided by any human operator. By reading articles in PDF format or web pages, such as Wikipedia, the system will be able to increase its knowledge base, providing the users with better answers. Currently the system works only with texts in English. We make use of the Stanford Parser to help in the natural language interpretation. To store knowledge a representation structure similar to a semantic network was developed. It offers a few advantages which will be described in greater detail.**

Keywords— **Expert systems, knowledge representation, automatic knowledge acquisition.**

I. Introduction

The current revolution promoted by information technology offers anyone with internet access a large, and constantly growing, amount of information. However, in order to gain access to this information, certain tools are needed. Currently search engines like Google, Yahoo! and Bing are essential to find information on the web. These tools return a list of links to pages or documents on the Internet containing data that might be related to the query entered by the user (a set of keywords). The list of links is processed by a non-intelligent algorithm and the user is forced to navigate among the many links returned in order to find the desired information.

This method is important and has been very successful. However, it has expressive limitations. Although such systems can point to sources which have a high probability of containing the answers to a question, it cannot directly answer the question. This has motivated the development of systems capable of directly answering a users question via the web, such as Google Squared (1) and Wolfram Alpha (2). These projects try to automate the information extraction from the existing sources, formatting and filtering the requested information. They are also capable of dynamically generating results for a query based on the acquired knowledge. We consider the next step in this evolution to be the construction of knowledge servers (3); systems that are able of extract, store and provide information in an intelligent manner. The development of such a system is the goal of this project.

II. Objectives

We aim to develop a system, dubbed Cube, which is able to automatically extract knowledge from different sources, like PDF documents and web sites. The system will use natural language processing to extracted knowledge from these sources. It must also be able to store this knowledge in a manner that allows for querying. The system will also be able to generate answer to user submitted queries from the stored knowledge.

III. Methodology

In order to create our knowledge base we must be able to process the source text and extract knowledge from it. An input module that realizes this task was developed. Two important parts of this module are further described below.

A. Natural Language Processing

The input texts are written in plain English. A natural language processor is therefore essential to obtain the semantics of a sentence. To aid in this task, we made use of the Stanford Parser (4). This parser is written in Java and it is statistical, using a probabilistic context free grammar (PCFG).

Our software extracts sentences from the source text and inputs them in the Stanford Parser. The parser returns a tree that represents the grammatical relationships between the words in the sentence. This structure is then parsed by an automaton we developed to generate a graph that represents the knowledge contained in the sentence.

B. Knowledge Representation

There are several well known ways to represent knowledge, such as frames, the entity-attribute-value model and semantic networks. The entity-attribute-value model was discarded because it isn´t efficient at storing knowledge in a generic manner and generating knowledge through inference.

Frames (5) are based on the entity-attribute value model, but are able to represent knowledge in a more structured way. Frames are stereotyped situations that are previously defined, based on frequently experienced situations, which may be changed to define a new situation. A frame is a sort of network where the top levels represent knowledge about what is always true about a given situation and what could be expected from such a situation. There are also terminals which can be filled in to better describe the situation. Kicking a ball can be a situation. Scoring a goal can be an expected result from such an action. The size and type of the ball are terminals that help better describe the situation. However creating the frames to represent any kind of situation that a general knowledge acquisition system might encounter is difficult, so we discarded this structure as well.

The semantic network (6) was the approach implemented here. A semantic network is a graph composed by vertices that represent concepts and edges, that may or may not be directed, which represent relationships between these nodes. An

This project has been supported by FAPESP and CNPq.

example of a semantic network is pictured below.

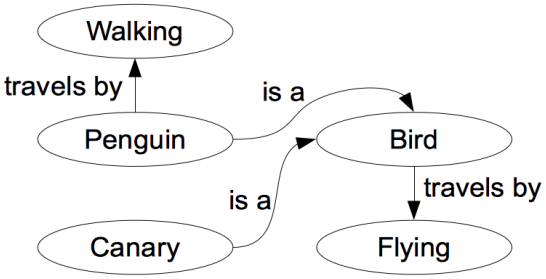


Fig. 1. A semantic network collected from 4 sentences and representing knowledge about two types of birds and how they travel.

Semantic networks provide a simple way to organize knowledge and they also allow inference. Knowledge that may not have been explicitly input into the system may be derived from this representation. If we ask how a canary travels, there is no direct answer. However, we can see from the network that a canary is a bird. If we ask how a bird travels, there is an answer: a bird travels by flying. This mechanism allows us to infer new knowledge from the network: a canary travels by flying.

We should also note that the edge *travels by* connected to the *Penguin* vertex is necessary because otherwise we would incorrectly answer the question "How does a penguin travel?". If we were to infer the answer in the same manner as we did for the canary, the system would answer that the penguin travels by flying, which we know to be false. This is called exception handling.

We found that a semantic network was the best suited structure to our system. However even this structure had a few shortcomings. We therefore modified the structure to better fit our needs.

C. *Extended Semantic Network*

The extended semantic network (ESN) is created from a common semantic network (SN). This modified structure allows us to represent any sentence, and therefore any knowledge that may be expressed by these sentences. An ESN has five types of vertices: entity, relationship, complement, entity modifier and relationship modifier.

An entity vertex represents the same concept that a vertex in a SN represents. The relationships represented by the edges of a SN are equivalent to the relationship vertices of the ESN.

The complement vertex is used to represent any type of complement that may be expressed in English. The complement vertex allows a simple and more elegant way to represent certain rules or limitations. For example, sometimes we have different relationships to model similar sentences, as in "The color of the sky is blue" and "The color of the sky is blue only on sunny days". A SN does not have a simple way to represent this type of exception. An ESN can represent both these sentences in a similar manner because the complement vertex is capable of adding more information about the main phrase.

The entity modifier vertex groups different entity vertices into a new vertex that represents a single more complex entity. Take the sentence "Robert's house is very pretty". Both "Robert" and "house" are entities for which more information could be available. It is therefore desirable to store them as such. However "Robert's house" is also an entity. An entity modifier vertex is able to represent this complex entity formed by two simpler entities. In a similar way, a relationship modifier will be linked to relationship vertices. Considering this, the diagram in Fig.2 represents an ESN created by our software for the same SN in Fig.1.

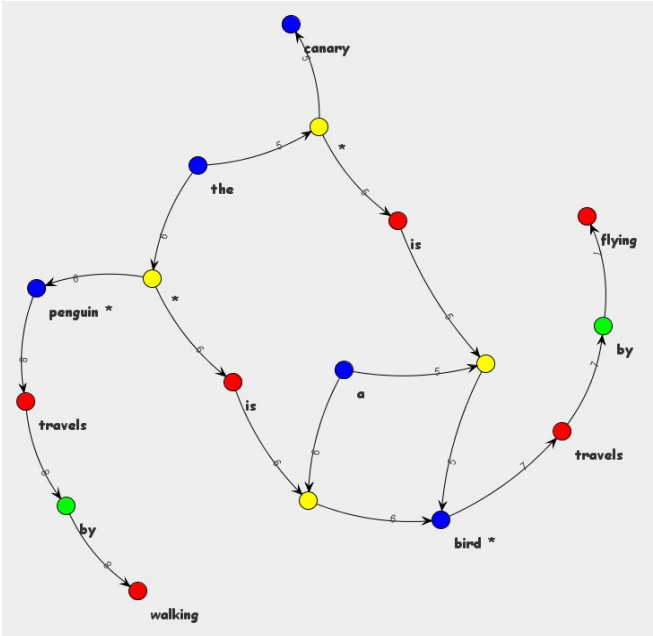


Fig. 2. An ESN representation of the same SN shown in Fig. 1. The vertex colors are: blue for entity, red for relationship, green for complement and yellow for modifiers (both entity and relationship)

Knowledge may also be inferred from an ESN in a manner similar to inference in a SN. The "is a" relationship edge in a SN is replaced by a "is" relationship vertex in an ESN. It is possible to infer how a canary travels by noting that a directed edge from the entity modifier vertex joining "the" and "canary" to an "is" relationship vertex exists. Once again, following the directed edge leaving the "is" vertex one arrives at a entity modifier vertex joining "a" and "bird". Therefore a canary is a bird. Applying a similar logic one determines how a canary travels.

An ESN can be defined through an extended semantic network grammar (ESNG), which is described by the following Wirth notation (7):

esn = "{" extent extrel "}" | "{" extent extrel extent "}" .
exter = extrel | extent .
extrel = genrel { "[" "compl" exter "]" } .
extent = genent { "[" "compl" exter "]" } .
genrel = "rel" | "(" relset ")" "rel" .
relset = "rel" { "rel" } .
genent = "ent" | "(" entset ")" "ent" .
entset = "ent" { "ent" } .
ent = "acorn" | "bird" | "cat" | "dog" | …
rel = "is" | "travels" | "lives" | …
compl = "as" | "by" | "and" | "of" | "in" | …

D. In this notation, "ent" represents an entity; "rel", a relationship and "compl", a complement. The "entset" and "relset" productions correspond to sets of entities and relationships, respectively. The "genent" and "genrel" productions represent an entity or a modified one, with the modified entity and the set of modifiers entities. The "extrel" and "extent" productions allow the insertion of complement phrases. Finally, the esn represents an English sentence, given the previous productions.

E. Based on the ESGN, it is possible to represent English sentences using a new representation that can be obtained from an ESN. For instance, the sentences in Fig. 2 can be defined using the ESNG as: {(the) penguin is (a) bird}, {(the) canary is (a) bird}, {bird travels [by flying]} and {penguin travels [by walking]}. As we can see, the ESNG allows a simple representation of English sentences, adding useful information about the semantics, like the presence of noun modifiers (with the round brackets), complementary phrases (with the square brackets) and sentence boundary (with the curly brackets).

Taking a longer sentence, for instance, "The electron was identified as a particle in 1897 by J. J. Thomson and his team of British physicists", we have the ESN shown in Fig.3:

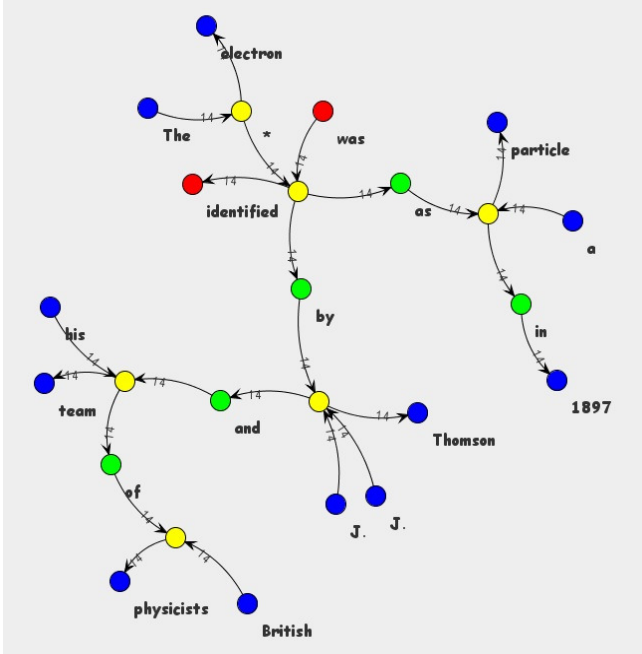


Fig. 3. An ESN representation of a longer sentence.

The corresponding ESNG representation of the ESN in Fig.3 is: {(The) electron (was) identified [as (a) particle [in 1897]] [by (J. J. Thomson) [and (his) team [of (British) physicists]]]}.

F. Architecture

The system is divided in two main modules: an input module (the *Writer module*) which is responsible for populating the database with knowledge gained by examining input sources; and an output module (the *Reader module*) which accesses the database in order to answer queries made by the user.

The general flow of the Writer module is outlined in the flowchart below.

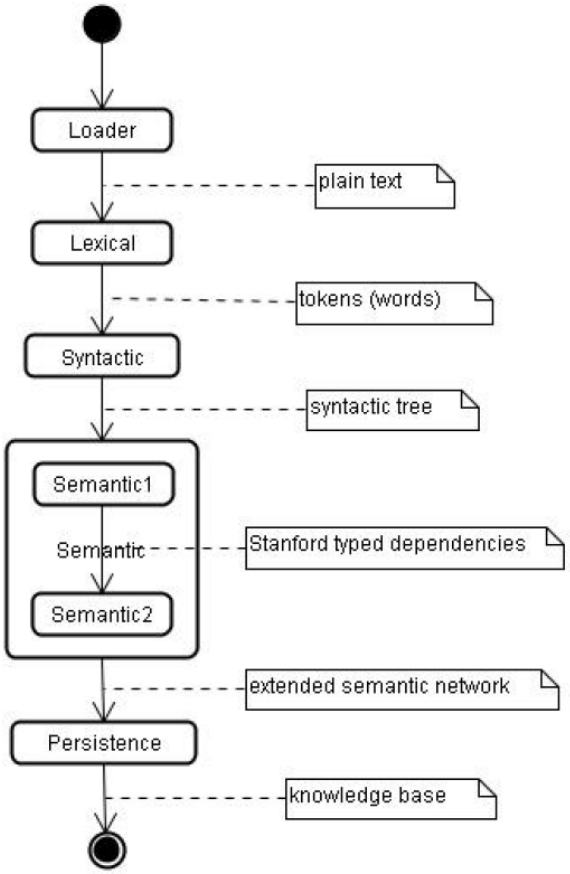


Fig. 4. Flowchart representing the Writer module of the Cube system.

The first component of the Writer module is the Loader. It loads the input texts, independently of the source types. This component isolates the source type from the system which, in turn, allows for different text sources to be processed in a similar way. This modular approach also simplifies the task of creating further functionalities to load data from other sources. Currently we are able to load text from PDF files and from HTML documents, so any web site may be processed. A special processor exclusively for Wikipedia was also developed. It includes the capability of processing links, treating the corresponding sites in a recursive manner. The depth to which this recursion is carried out can be controlled by the user.

The next three components (lexical, syntactic and semantic analyzers) have the usual functions. The lexical analyzer takes the text and splits it into tokens, which in our case represent words. This part is relatively simple and can be implemented using regular expressions. The syntactic analyzer transforms the tokens received from the lexical analyzer into a syntactic tree. A syntactic tree is a tree that represents the text in a sentence in a grammatically structured form. The words are characterized as being nouns, adjectives, etc. and are arranged on the tree based on their order in the sentence. Currently the Stanford Parser is being used to generate the syntactic tree.

The semantic analyzer converts the syntactic tree into the extended semantic network. To accomplish this goal, it uses the Stanford typed dependencies (8) generated by the parser. This functionality is implemented partly by the Stanford Parser and partly by a newly developed component in our software.

The last part of the Writer component is the Persistence module. It basically stores the ESN generated by the previous component in the knowledge base. Currently we use Hibernate (9) to store the ESN in a relational data base. Care must be taken when saving data to the database since the ESN generated for a given sentence must be joined with the existing database. It is important that when this is done, the vertices are not replicated in the database. In other words, there should only be one entity node for "canary", "penguin" and so on.

The Reader module allows the user to query the database. Currently the querying algorithms are relatively simple. We have not yet implemented algorithms for inferring knowledge from the ESN. However the system is already able to answer some simple queries. We have developed two user interfaces. One is in the form of a Java application that can be used to input information into the system as well as to query the database and obtain the answers and check the ESN in the database. The other is a web interface that allows querying and returns formatted answers. Fig.5 shows the Reader module components.

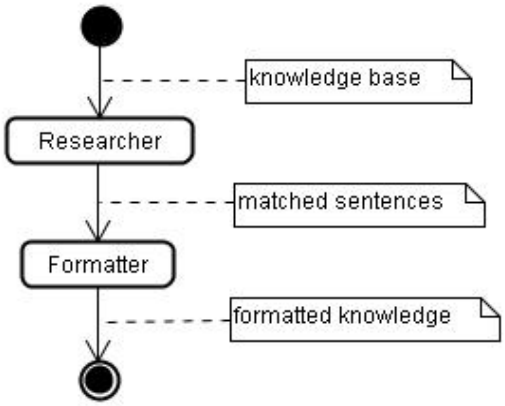


Fig. 5. A flowchart representing the Reader module.

The first Reader module component is the Researcher. This component receives the query provided by the user and then searches for sentences that contain the specified words. Since the ESNs contain data about the complete sentence, which consists of a unique identification for each sentence, we can return pieces of the stored ESN that contain only the desired sentences. Having found the sentences, this component reassembles the ESN in memory, creating the corresponding edge-vertices structure of the ESN.

The next component in the Reader module is the Formatter. This component recognizes knowledge in the sentences returned by the Researcher. For instance, imagine a user queries the word "electron". If the database contains an entity "electron" that is related to a composed entity "subatomic particle" through a relationship node that contains "is", as in Fig. 6, then the Formatter creates a Definition format, retrieving the term ("electron") and the definition ("subatomic particle") from the sentence. The resulting text, created to be displayed as a web page, is shown in Fig. 7.

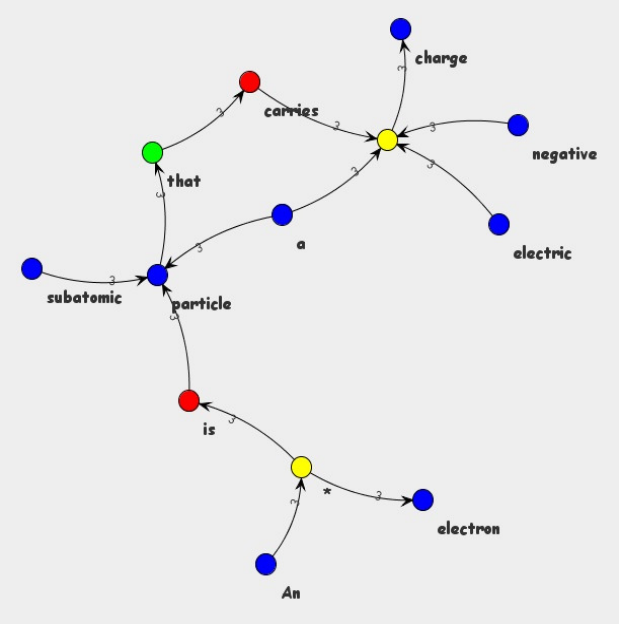


Fig. 6. ESN retrieved from the database for the query "electron".

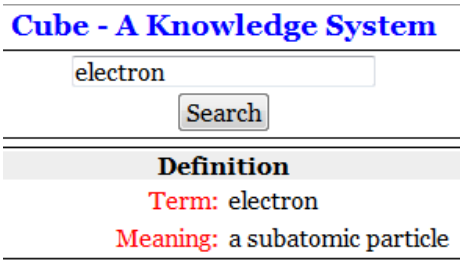


Fig. 7. The web result of a query for "electron".

A Property format was also developed. Take an ESN that contains an entity "color" connected to another entity "sky" by the complement "of" and to another entity "blue" by the relationship "is". This type of relationship describes a property: the main entity is "sky", which has a property "color" with value "blue". The Property format is returned as shown in Fig.8.

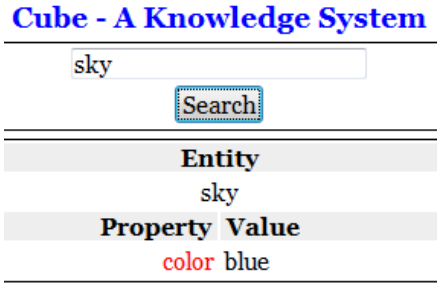


Fig. 8. The web result of a query for "sky".

The Formatter component is extensible and, therefore, other processing components can be incorporated in the system, amplifying its capabilities of finding knowledge from the ESN.

G. Cube as an application of adaptivity theory

The Cube project has some aspects related with adaptivity theory (10). Its main component, the knowledge base, is a rule set. This rule set has a definition expressed by the Extended Semantic Network Grammar (ESNG). The system updates the ESN according to the sentences that it acquires from different possible sources, like PDF documents, web sites and so on. The updating process is not trivial and it has some details that require attention. First, entities can't be duplicated in the database. There must be a verification to merge a newly created ESN with the database ESN. Second, the algorithm must substitute pronouns and other indeterminations, because their presence in the knowledge base has no meaning. Finally, the system must take into account possible conflicts try to resolve them through modifications to the ESN. These last two requisites are not implemented yet on the current version of the program, but they are planned for future work.

The search and formatting process are based on the knowledge base. It is essentially a common application of adaptivity: when the system tries to retrieve some knowledge from the base, this process is done considering a structure that is modified dynamically, through a loading process from text documents. Beyond this, for the syntactic analyzer, we built a dynamic parser generator, that creates a parser (SLR or LR(1)) in execution time, given a grammar. This component permits updates on the grammar in execution time, without the need of recompiling the code in order to generate a new parser. This implementation is another application of adaptativity in this project.

However, developing this piece of software is not part of our current project, which is why we are using the Stanford Parser. In order to create our own complete solution with our own syntactic parser, however, a dynamic parser generator will be useful.

IV. RESULTS

In order to test our system we loaded the Wikipedia web page for the entry "electron", using the previously described writer module. After the site has been processed, the database was updated with the sentences from that page. Searching for the term "electron" using our web interface, we obtain the results shown in Fig. 9.

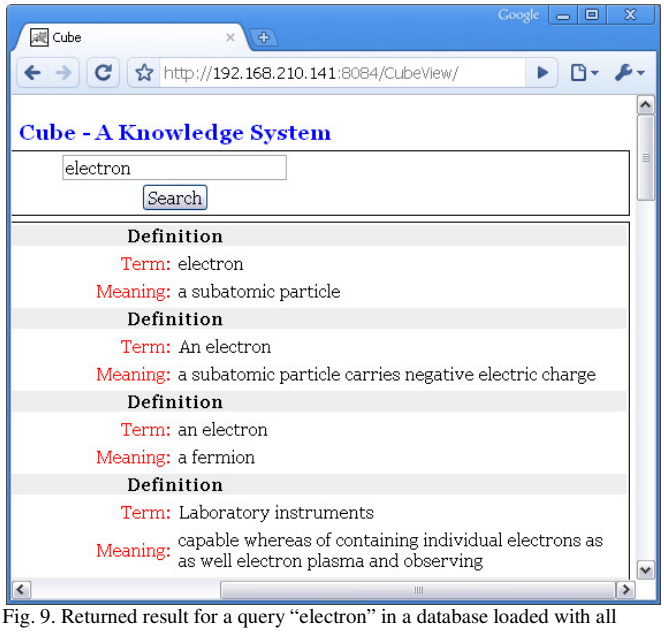


Fig. 9. Returned result for a query "electron" in a database loaded with all the information in Wikipedia web page for the entry "electron".

As can be seen, the system is capable of finding more than one definition for the word "electron". This occurs because the Formatter component analyzes all the sentences returned by the Researcher and verifies, for each of them, if it matches the expected format for a definition.

The process of reassembling the sentence is not yet completed and, therefore, some errors are present (the last definition in Fig.9 is an example). The conversion from an ESN to plain English is not trivial and it must follow English grammar rules. It is important to note that several of the observed mistakes are not caused by our system but are due to incorrect parsing of some sentences by the parser.

Although the Reader module is simple, it provides good results. The mechanisms to extract knowledge are very simple and currently lead to limited results. The describe Definition format is an example. Since not all occurrences of the verb "is" are used to define something, the system sometimes yields incorrect definitions. We believe however, that with little more logic this concept could yield to correct results in a more consistent manner.

The processing time to parse input data is still a bit high. For example, it takes a few minutes to process a Wikipedia page. This is mostly spent parsing the input data via the Stanford Parser.

V. Conclusions

As was shown, the Cube system is still simple and in development. However it is already able to automatically extract knowledge from HTML and PDF documents. In order to store this knowledge, we developed a new semantic representation, the Extended Semantic Network (ESN). The ESN exhibits great flexibility and is capable of representing complex sentences written in English. The ESN has a small set of node types (entity, relationship, modifier, complement) but it is applicable to almost all the sentences in English, because of its generic concepts.

The entire system was intended to be as modular as possible, since this is very important for further development and adding new functionality. For example, adopting a new input source or a different parser, like the Link Grammar Parser (11), is relatively easy. We hope that by implementing the various features mentioned throughout this article and few others the Cube system will become even more useful. This means to generating better and faster results and incorporating new capabilities to provide more information and analyses. As part of our future work, we are studying the introduction of a timeline processing component (based on historical information and object hierarchy) and the addition of capabilities to process and interpret other languages, in special Portuguese.

References

1. **Google.** Google Squared. [Online] [Cited: August 10, 2009.] http://www.google.com/squared.
2. **Wolfram Alpha**. [Online] [Cited: August 10, 2009.] http://www.wolframalpha.com.
3. **Feigenbaum, Edward.** The Age of Intelligent Machines: Knowledge Processing - From File Servers to Knowledge Servers. [Online] [Cited: November 30, 2009.] http://www.kurzweilai.net/meme/frame.html?main=/articles/art0098.html
4. **Stanford.** Stanford Parser. [Online] [Cited: September 21, 2009.] http://nlp.stanford.edu/software/lex-parser.shtml.
5. **Minsky, Marvin.** FRAMES. [Online] MIT, June 1974. [Cited: November 27, 2009.] http://web.media.mit.edu/~minsky/papers/Frames/frames.html.
6. **Sowa, John F.** *Semantic Networks.* s.l. : John Wiley and Sons, Inc., 1987.
7. **Wikipedia**. Wirth Syntax Notation. [Online] [Cited: November 30, 2009] http://en.wikipedia.org/wiki/Wirth_syntax_notation
8. **Manning, Marie-Catherine de Marneffe and Christopher D.** The Stanford typed dependencies representation. *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation.* 2008.
9. **Hibernate.** Hibernate. [Online] [Cited: September 28, 2009.] https://www.hibernate.org/.
10. **Neto, J. J.** Adaptatividade e tecnologia adaptativa. *Revista IEEE América Latina.* 7, 2007, Vol. 5. (Special Edition – of Worshop de Tecnologia Adaptativa, WTA'2007)
11. **University, Carnegie Mellon.** *Link Grammar Parser.* [Online] [Cited: Novermber 30, 2009.] http://www.link.cs.cmu.edu/link/.

AOCR – Adaptive Optical Character Recognition (08 Janeiro 2010)

T. M. D. Bruno, F. S. Douglas, G. J. Rafael

***Resumen*— O trabalho desenvolvido consistiu na elaboração e implementação de um software OCR (Optical Character Recognition) utilizando tecnologia adaptativa. O objetivo do trabalho foi a melhoria do desempenho geral do sistema através da utilização desta tecnologia, elevando sua taxa de acerto e possibilitando a inserção gradual de novas regras adaptativas que possibilitassem a inclusão de diferentes alfabetos/fontes. A primeira parte do projeto consistiu em uma análise dos OCR's e dos processadores de imagem disponíveis no mercado e o modo de funcionamento dos mesmos. Na segunda etapa, realizou-se a elaboração da especificação do software, em que todos os requisitos funcionais e não-funcionais necessários para a elaboração do programa foram listados. Na terceira etapa, houve o estudo da teoria adaptativa e as possíveis formas de inserção desta tecnologia no software, visando à melhoria do desempenho e o aumento do número de funcionalidades. A última fase foi composta pelo desenvolvimento do software e dos algoritmos adaptativos, realização dos testes de cada funcionalidade, assim como a elaboração de toda documentação.**

I. Introdução

O reconhecimento óptico de caracteres é uma dos ramos mais antigos e pesquisados na área de reconhecimento de padrões. Os primeiros OCR's foram desenvolvidos na década de 50 por empresas e centros de pesquisa no exterior.

Atualmente, existe um potencial muito grande a ser explorado no mercado dos OCR's, tendo em vista que apenas algumas grandes e médias empresas de desenvolvimento de softwares desenvolvem tais aplicativos. Esses aplicativos são na maioria das vezes restritos a um alfabeto e possuem um preço muito elevado.

Inserido neste contexto, este trabalho teve como objetivo o desenvolvimento o primeiro OCR que utiliza tecnologia adaptativa. O AOCR é um software que tem como objetivo realizar a tradução de imagens de diversos formatos (PNG, JPEG, TIFF e BMP) em arquivos de texto editáveis (TXT), utilizando técnicas de inteligência artificial e tecnologia adaptativa. Por usar essas tecnologias é possível incluir novas funcionalidades além da tradicional conversão, o que torna este programa único. Essas principais funcionalidades são: aprendizado de novos alfabetos/fontes, aprendizado a partir dos erros cometidos, dentre outras que serão descritas neste artigo.

II. Desenvolvimento

Este projeto teve início em Março de 2009 e foi concluído em Dezembro de 2009. Todas as etapas que fazem parte do projeto (Concepção, Documentação, Desenvolvimento e Testes) foram realizadas durante o último ano do curso de engenharia de computação (ênfase semestral) da Escola Politécnica da USP dos formandos de 2009.

Durante a elaboração e a construção do programa, tentou-se criar um sistema de fácil utilização e de desempenho maior ou próximo aos encontrados nos melhores programas. Nos melhores softwares disponíveis no mercado, a taxa de acerto fica em torno de 97% (utilização de métodos estatísticos e de corretores ortográficos) e o tempo de processamento de uma página é de 1-3 minutos. Vale ressaltar que todos os OCR's realizam tais processamentos com apenas um alfabeto e uma fonte, o que os torna muito eficientes quando utilizados textos que contenham tal alfabeto e fonte.

A. Funcionamento geral do programa

Antes da etapa de desenvolvimento do programa AOCR, houve uma etapa de pesquisa e teste dos programas OCR's de código aberto disponíveis no mercado, como OCRAD [7], OCRopus [8], Tesseract [9] e GOCR [6]. Após diversos testes realizados, verificou-se que o melhor OCR disponível gratuitamente e com o código aberto era o aplicativo GOCR.

Este programa apresenta seu código fonte na linguagem C e está dividido em módulos que facilitam o entendimento de seu funcionamento. Portanto, este sistema foi escolhido como o software base, em que foram realizadas todas as alterações necessárias para a inclusão do algoritmo adaptativo, assim como o tratamento das novas fontes e novos alfabetos. A partir do GOCR foram extraídas somente as rotinas de tratamento de imagem. Os outros módulos do programa não foram utilizados, uma vez que os mesmos eram incompatíveis com os objetivos inicialmente planejados neste trabalho.

O OCR é um sistema responsável pelo processamento de textos no formato de imagem em texto em formatos editáveis. Primeiramente, a imagem é tratada pelo processador de imagem. Em seguida, os parâmetros de cada um dos símbolos são extraídos da figura. Por último, esses parâmetros são utilizados pelos algoritmos de decisão do OCR. Um diagrama do funcionamento básico do programa pode ser visto na Figura 1, em que os principais módulos do software podem ser visualizados.

Fig. 1. Entradas e saídas do algoritmo de decisão.

A funcionalidade básica do software permite o acesso às imagens e a visualização do texto processado através da tela principal do programa. Além da funcionalidade principal, o AOCR possui uma interface gráfica que permite o usuário utilizar todas as principais funcionalidades.

Inclusão/remoção de um novo alfabeto: diferentemente dos OCR's convencionais presentes no mercado, o AOCR permite a inserção e remoção de alfabetos/fontes através de sua interface, ou seja, o usuário é capaz de treinar o programa com os alfabetos/fontes que desejar. Diversos testes foram realizados com sucesso utilizando os alfabetos/fontes latino, grego, cirílico, texto escrito e símbolos matemáticos.

Correção de parâmetros: esta funcionalidade permite a correção dos parâmetros de um determinado caractere. Caso tenha ocorrido uma tradução errada devido ao fato de um parâmetro da letra estar errado, o usuário pode corrigir este parâmetro através da interface.

Escolha dos alfabetos: o usuário pode escolher um alfabeto/fonte ou um conjunto de alfabetos que serão utilizados como fonte de dados para a realização das traduções. Dessa forma, há uma otimização no tempo de processamento, não significando necessariamente uma melhora na taxa de acerto.

Testes automatizados: o usuário pode realizar a tradução de diversos arquivos através desta funcionalidade, diminuindo o tempo de processamento quando são submetidos diversos arquivos.

Foram utilizadas tecnologias de diversas áreas da computação durante o desenvolvimento do projeto. A seguir, as principais tecnologias serão descritas em mais detalhes, assim como a utilização das mesmas no programa.

B. Reconhecimento de padrões

Todos os 37 parâmetros utilizados para determinar um caractere foram criados durante a elaboração do programa. Inicialmente, cria-se um retângulo ao redor do símbolo que será processado. Então o todo o processamento necessário para a obtenção dos parâmetros é realizado. Alguns dos parâmetros resultantes de tal processamento são:

Número de buracos: quantidade de buracos da letra, sendo que um buraco é caracterizado como pixels brancos delimitados por pixels pretos.

Número de linhas horizontais: quantidades de linhas horizontais presentes na letra, sendo que linhas horizontais são caracterizadas como pixels pretos sem interrupção na direção horizontal.

Número de linhas verticais: quantidades de linhas verticais presentes na letra, sendo que linhas verticais são caracterizadas como pixels pretos sem interrupção na direção vertical.

Relação altura/largura: esta é a relação da largura pela altura do retângulo que está ao redor da letra.

Centro de massa proporcional: o centro de massa é calculado de acordo com a seguinte fórmula:

$$CM = \frac{\sum (PixelPreto) * (PosiçãodoPixel)}{TotaldePixelsPretos}$$

Densidade: a densidade é calculada pela quantidade de pixels pretos em relação ao total de pixels da letra, conforme a fórmula a seguir:

$$CM = \frac{\sum (PixelsPreto)}{TotaldePixels}$$

Número de entradas: este parâmetro verifica quantas entradas a letra possui em cada aresta do quadrado que a circunscreve, isto é, verificam-se quantos pixels brancos contínuos existem entre os pixels pretos no limiar da caixa.

Número de linhas cruzadas: este parâmetro verifica o número de linhas cruzadas por um feixe passando pelo centro (horizontal e vertical) da caixa ao redor da letra.

A partir desses parâmetros, a maioria dos caracteres pode ser classificada sem a necessidade de intervenção das técnicas mais avançadas, como adaptativas. Na Figura 2, pode-se observar uma representação dos caracteres latinos de mesma classe que podem na maioria das vezes causam problemas no momento do processamento.

Fig. 2. Classes de objetos semelhantes.

C. *Processamento de imagens*

Antes que os parâmetros de cada símbolo sejam adquiridos, faz-se necessário um tratamento da imagem. Primeiramente, realiza-se a delimitação da região que contém o símbolo. Em seguida, algumas imperfeições presentes no arquivo são removidas, como buracos não fechados, pontos extras ao redor do caractere e ruídos presentes nos contornos. Por último, os caracteres são processados e os atributos enviados para os algoritmos de decisão;

D. *Tecnologia adaptativa*

A utilização desta tecnologia visa melhorar o desempenho do programa, tendo em vista que diversas ações corretivas podem ser aplicadas para eliminar incoerências já conhecidas e comumente encontradas em OCR's atuais. Foram desenvolvidos dois algoritmos que utilizavam árvores de decisão adaptativas para realizarem as escolhas do caractere que mais se aproxima com as características extraídas da figura. Além disso, o usuário pode realizar correções dos parâmetros de cada caractere através da tela de correção de parâmetros. Assim, pode-se interagir com os dados presentes no banco de dados, aumentando a taxa de acerto dos algoritmos do programa.

E. *Algoritmos de decisão*

No que diz respeito ao processo de decisão dos símbolos, realizaram-se algumas estratégias utilizando técnicas adaptativas para superar problemas conhecidos de árvore de decisões, como overfitting[1]. Além disso, visou-se dar maior flexibilidade em relação aos parâmetros, ou seja, os parâmetros da letra sempre vão variar. Caso os valores variem mais que um determinado limite, o algoritmo resultará em um ramo errado da árvore, fornecendo um caractere errado na saída. Com a utilização da adaptatividade é possível tratar este problema, uma vez que mais de um resultado pode ser obtido, como poderá ser visto nos próximos itens.

Todas as técnicas de decisão utilizam o método de ordenação dos parâmetros. Este método utiliza conhecimentos prévios dos parâmetros de desvio padrão e a média, além de uma nota dada pelos conhecimentos prévios dos desenvolvedores, para decidir qual o melhor parâmetro. Define-se o melhor parâmetro como aquele que possuir melhores notas nas três classes a seguir:

1. Confiabilidade: quanto menos este parâmetro variar dentre o que já foi observado para o alfabeto/fonte que está sendo utilizado, melhor será sua confiabilidade. Exemplo: o número de buracos de uma letra é um parâmetro que possui uma alta confiabilidade, porque na maioria dos casos os caracteres possuem uma quantidade de buracos em uma faixa pequena de possíveis valores.
2. Dispersão: quanto mais distantes os valores das letras estão, melhor será para realizar a decisão de qual é a letra, ou seja, o quão bem o parâmetro em questão consegue distinguir qual é a letra correta. Exemplo: o número de buracos não possui uma boa dispersão, visto que a maioria das letras possui zero ou um buraco(s).
3. Classe: uma nota prévia fornecida, que pode ser alterada pelo usuário, para cada parâmetro.

$$PesoParâmetro = Classe + (Pontos_{dispersão} + Pontos_{confiabilidade})$$

As técnicas de decisão disponíveis no programa são quatro. Cada uma delas serão descritas detalhadamente a seguir:

1. Método Estatístico;
2. Árvore de decisão simples;
3. Árvore de decisão adaptativa não-determinística;
4. Árvore de decisão adaptativa "pura".

No primeiro método, são comparados todos os parâmetros de todas as letras dos alfabetos/fontes escolhidos, com os parâmetros encontrados para a letra em questão. Verifica-se a distância entre tais letras, diminuindo a probabilidade de ser a letra conforme esta distância aumenta. Cada parâmetro possui um peso diferente, que influencia na diminuição de probabilidade, baseado na qualidade deste parâmetro (utilizando o método demonstrado anteriormente). Desta forma, a perda da probabilidade segue a seguinte fórmula (para cada parâmetro):

$$Perda = \|Distância\| * CONSTANTE * Peso\_Parâmetro$$

Todos os métodos que utilizam uma árvore de decisão preparam a árvore de forma idêntica, visando deixar os parâmetros com maior qualidade perto da raiz da árvore, e procurando dividir o alfabeto/fonte de forma inteligente, visando dar a maior robustez possível a esta árvore. Desta forma, a montagem da árvore funciona da seguinte forma:

1. Escolhe-se o melhor parâmetro dentre os ainda disponíveis, a partir do alfabeto/fonte disponível no nó atual;
2. Descobre-se o valor mínimo e máximo. Divide-se esse conjunto em 10 partes iguais;
3. Verifica-se a densidade das letras do alfabeto/fonte em cada um dos conjuntos;
4. Verifica-se qual o grupo com a menor densidade;
5. Unem-se esses grupos de menor densidade caso sejam adjacentes;
6. Dividem-se os ramos no centro dessas áreas de menor densidade;
7. Caso o novo nó possua apenas uma letra, o processo acaba. Caso contrário, o processo volta ao passo 1 para cada novo ramo, removendo o parâmetro utilizado nesse passo.

Considerando que no primeiro passo foi escolhido o parâmetro número de buracos, para o alfabeto/fonte latino-arial, a Figura 3 demonstra os passos 2 ao 6:

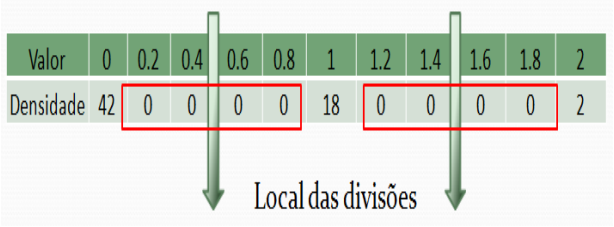


Fig. 3. Distribuição do parâmetro número de buracos das letras da alfabeto/fonte latino-arial.

A árvore de decisão simples percorre a árvore. Ao encontrar uma folha, verifica se existe apenas uma letra, que será a resposta. Caso possua mais que uma letra, utiliza o método estatístico para decidir qual é a mais adequada.

Por fim, ambas as árvores adaptativas utilizam métodos de consulta idênticos, variando-se apenas na ação de adição e remoção adaptativas utilizada.

A ação de consulta consiste em verificar se o valor do parâmetro sendo utilizado está em uma das áreas de incerteza.

Define-se área de incerteza os 10% iniciais e finais de cada ramo. No primeiro ramo a área de incerteza é ao final do ramo e anterior ao início do ramo. Isto também ocorre no último ramo, em que a área de incerteza é o início e após o fim, ou seja, valores nunca vistos anteriormente para aquele valor.

A Figura 4 demonstra as áreas de incerteza para o parâmetro número de buracos, com alfabeto/fonte latino-arial, em que o verde é a divisão dos ramos e em vermelho estão definidas as áreas de incerteza:

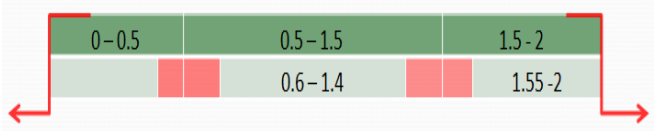


Fig. 4. Incerteza do parâmetro número de buracos das letras da alfabeto/fonte latino-arial.

Logo, quando algum parâmetro está na área de incerteza, aciona-se a ação adaptativa correspondente, que varia de acordo com o tipo de árvore que foi escolhido:

1. Não-determinística: segue todos os ramos possíveis, recolhendo o resultado de cada ramo. Escolhe-se a partir do método estatístico o melhor no final do processo;
2. "Puro": desconsidera o parâmetro atual, podando a árvore. Remonte-se a mesma sem este parâmetro (a partir do nó atual).

Para exemplificarmos a diferença existente entre cada um dos métodos, utilizou-se a Figura 5 como entrada no programa. Foram habilitados alguns alfabetos/fontes previamente existentes no banco de dados do programa (Arial, Símbolos e Grego).

Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text. It is used to convert paper books and documents into electronic files, for instance, to computerize an old record-keeping system in an office, or to serve on a website.

Fig. 5. Arquivo utilizado para realizar o teste.

Optical character recognition, usually abbreviated to OCR, is
the mechan ical o r electron ic translation of images of handwritten , typewritten o r
printed text (usually captured by a scanner] into machine-editable text- It is used
to convert paper books and documents into electronic files, for instance, to
computerize an old record-keeping system in an office, or to serve on a website-

Fig. 6. Arquivo de saída do teste.

O resultado pode ser visto na Figura 6 e os parâmetros relativos a esses testes podem ser observados na Tabela 1, em que estão presentes os valores de tempo de execução e a taxa de acerto para cada um dos algoritmos. Esses são os valores finais da primeira versão software, concluído em dezembro de 2009. Pode-se observar a diferença existente nos resultados obtidos para cada um dos métodos.

TABELA I
COMPARAÇÃO ENTRE OS 4 MÉTODOS DISPONÍVEIS NO PROGRAMA

Método	Tempo (s)	Taxa de acerto (%)
Estatístico	98	84,7
Simples	9	19,5
Adaptativo 1	34	94,5
Adaptativo 2	162	93,6

III. CONCLUSÕES

Podemos observar através dos resultados obtidos pela execução do arquivo da Figura 5 que cada um dos métodos obteve um tempo de resposta e uma taxa de acerto diferente. Levando-se em consideração a taxa de acerto, temos que os métodos adaptativos proporcionaram uma melhor taxa de acerto que o método de árvore de decisão simples e que o método estatístico. Entretanto, pode-se observar que o algoritmo de árvore de decisão simples apresenta um menor tempo de processamento que todos os outros métodos. Um fato importante de ser ressaltado é o fato de que apenas as comparações entre os métodos que utilizam a mesma tecnologia são válidas. Deste modo, as comparações devem ser feitas entre os métodos Adaptativo 1, Adaptativo 2 e Simples. O método estatístico foi inserido devido ao fato de ser o mais utilizado pelos OCR's.

VI. Este trabalho proporcionou aos alunos e à comunidade científica uma maior aproximação com os verdadeiros problemas concernentes ao reconhecimento óptico de caracteres. Este é um dos temas mais antigos e mais estudados dentro da área de reconhecimento de padrões. Acreditamos que a utilização dos conceitos de adaptatividade adquiridos ao

longo do desenvolvimento do projeto ajudou de forma significativa na criação de um software único no mercado, uma vez que o mesmo permite ao usuário o controle da maioria das funcionalidades mais importantes do programa, como a inserção de novos alfabetos/fontes de modo dinâmico, a correção dos parâmetros dos caracteres e o treinamento do OCR.

Este foi o primeiro trabalho desenvolvido no Laboratório de Técnicas Adaptativas na área de reconhecimento óptico de caracteres com a utilização de técnicas adaptativas. Portanto, no que diz respeito ao futuro deste projeto, pode-se afirmar que diversas funcionalidades adicionais poderiam se desenvolvidas, assim como melhorias em processos específicos do programa. Podemos citar as seguintes áreas que poderiam receber contribuições e avanços futuramente:

1. Acredita-se que poderiam ser desenvolvidos novos parâmetros para realizar a escolha dos caracteres. Alguns desses parâmetros haviam sido planejados, mas não foram implementados, como o ângulo de inclinação e o número de pontas presentes nos símbolos;

2. Novos métodos para melhorar o processamento de imagem poderiam ser implementados, tendo em vista que o maior problema encontrado nas imagens durante o desenvolvimento deste trabalho foi a presença de imperfeições e falhas nos caracteres, principalmente quando as imagens eram geradas por scanners;

3. Modificações nos algoritmos adaptativos podem ser realizadas, melhorando a montagem das árvores de decisão, os tempos de processamento e as taxas de acerto.

IV. AGRADECIMENTOS

Os autores reconhecem as contribuições de J. N. João para a o desenvolvimento deste trabalho.

V. REFERENCIAS

[1] N. Morton, P. S. Eric, Pattern Recognition Engineering, New York.

[2] P, Hemerson, N. J. João, "Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.", Tese de doutorado, Universidade de São Paulo, Dec. 2003.

[3] P, Hemerson, N. J. João, "AdapTree - Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas", In: Conferência Latino Americana de Informática, 2002, Montevideo.

[4] T.F. William, "Computer Vision for Interactive Computer Graphics", , IEEE Computer Graphics and Applications, Vol. 18, Issue 3, pp. 42-53, May-June 1998.

[5] J. N. João, "Adaptatividade e Tecnologia Adaptativa", Tutorial baseado no WTA 2007. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 495).

[6] J. SCHULENBURG. Gocr. Disponível em: <http://jocr.sourceforge.net/>. Acesso em: 10 de abril de 2009, 2009.

[7] OCRAD. Ocrad. Disponível em: <http://www.gnu.org/software/ocrad/ocrad.html/>. Acesso em: 10 abril 2009, 2009.

[8] OCROPUS. Ocropus. Disponível em: <http://code.google.com/p/ocropus/>. Acesso em: 10 abril 2009, 2009.

[9] T.; MEZHIROV I. SMITH, R.; BREUEL. Tesseract. Disponível em: <http://code.google.com/p/tesseract-ocr/>. Acesso em: 10 de abril de 2009, 2009.

Bruno Tadayoshi Mendes Doy nasceu no Brasil, na cidade de São Paulo (SP) no dia 03 de setembro de 1985. Cursa atualmente o curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica da Universidade de São Paulo situada na cidade de São Paulo (SP), Brasil.

Realizou durante 3 anos pesquisas na área de eficiência energética e otimização de consume residenciais e empresariais. Trabalha atualmente em uma empresa de desenvolvimento de softwares.

Douglas Fernandes de Souza nasceu no Brasil, na cidade de São Paulo (SP) no dia 13 de outubro de 1986. Cursa atualmente o curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica da Universidade de São Paulo.

Artigos Publicados:

Software de simulação e visualização 3D da corrosão anisotrópica do Silício - D.F. de Souza e M.N.P. Carreño; XVI Simpósio Internacional de Iniciação Científica da USP; Engenharias e Exatas (2008) .

3D Simulation Software for Visualization of MEMS Microfabrication Processes - F.B. Colombo, P.M. Hokama, F. Tsuda, R.G. Jankauskas, D.F. de Souza, P. Kayatt and M.N.P. Carreño; ECS Transactions : Microelectonics Technology and Devices - SBMicro 2008; 14 No 1 (2008) 99-108.

Software CAD em C++ para Projeto de MEMS e MOEMS - D.F. de Souza, R.G. Jankauskas e M.N.P. Carreño; XV Simpósio Internacional de Iniciação Científica da USP - Engenharias e Exatas; Engenharias e Exatas (2007).

Rafael Garib Jankauskas nasceu no Brasil, na cidade de São Paulo (SP) no dia 28 de julho de 1987. Cursa atualmente o curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica da Universidade de São Paulo situada na cidade de São Paulo (SP), Brasil.

Artigos Publicados:

Ferramentas de visualização dinâmica para software de visualização e simulação de processos de microfabricação - R.G. Jankauskas e M.N.P. Carreño; XVI Simpósio Internacional de Iniciação Científica da USP; Engenharias e Exatas (2008).

3D Simulation Software for Visualization of MEMS Microfabrication Processes - F.B. Colombo, P.M. Hokama, F. Tsuda, R.G. Jankauskas, D.F. de Souza, P. Kayatt and M.N.P. Carreño; ECS Transactions : Microelectonics Technology and Devices - SBMicro 2008; 14 No 1 (2008) 99-108.

Software CAD em C++ para Projeto de MEMS e MOEMS - D.F. de Souza, R.G. Jankauskas e M.N.P. Carreño; XV Simpósio Internacional de Iniciação Científica da USP - Engenharias e Exatas; Engenharias e Exatas (2007).

Proposta de uma linguagem de alto nível básica para a codificação de softwares automodificáveis

(21 Janeiro 2010)

S. R. B. da Silva, J. J. Neto

***Resumo* — Este trabalho propõe uma linguagem simples de alto nível, dotada de recursos que facilitam a codificação de programas automodificáveis. Para isso, estendeu-se uma linguagem de programação convencional com comandos e declarações especiais voltados à especificação de alterações dinâmicas do código. A linguagem resultante permite, dessa forma, que o programador indique as alterações desejadas para o seu código, modificações essas que se efetuam no momento da execução do programa. Como resultado, torna-se disponível uma linguagem de programação apropriada para o desenvolvimento de aplicações adaptativas. Neste texto procura-se descrever os principais aspectos do projeto e da implementação de uma linguagem dessa natureza, e um pequeno exemplo de aplicação é apresentado para ilustrar sua utilização.**

***Palavras chave* — Adaptatividade, linguagem de programação, código automodificável, tecnologia adaptativa.**

I. INTRODUÇÃO

EM meados do século passado, quando os computadores careciam de recursos de armazenamento, sempre foi prática corrente entre os programadores procurar reaproveitar a escassa memória física então disponível, à medida que seus programas cresciam além da capacidade máxima dessa memória. Esta prática se manifestava pela automodificação de código – neste artigo, entende-se como automodificação a propriedade de um programa que tem a capacidade de se automodificar enquanto em execução, sem a intervenção de um agente externo ao processo – e era facilitada pelo uso extensivo de programação em linguagens de baixo nível e, também, pelo uso limitado de metodologias de programação, então incipientes [1].

Com o advento da Engenharia de Software e o surgimento de metodologias de programação e novos paradigmas de desenvolvimento de programas, a prática da automodificação de código teve seu uso reduzido drasticamente, uma vez que um código de programa armazenado em memória é considerado imutável. [2]

Recentemente, algumas aplicações especiais voltaram fazer uso dessa técnica conforme pode ser visto em [3], [4], [5], entre várias outras.

De acordo com [6], a adaptatividade, por sua vez, fundamenta seus métodos na evolução dinâmica do conjunto das regras que definem o comportamento do fenômeno adaptativo, podendo-se considerar natural o uso de código automodificável na implementação de programas que realizam procedimentos adaptativos.

O presente texto propõe-se a proporcionar aos programadores de softwares adaptativos, uma linguagem de programação simples e de alto nível, com recursos para especificação de operações de automodificação nos seus programas.

Tratando-se de uma linguagem experimental, através da qual se possam explorar as possibilidades de linguagens dessa natureza, optou-se por implementar uma extensão sintática, que possa ser aplicada a uma linguagem imperativa existente. Com essa extensão, o programador pode especificar as automodificações que deseja imprimir ao programa, cuja realização se efetiva em tempo de execução.

A linguagem resultante deve incorporar, dessa maneira, recursos que permitam ao programador de aplicações adaptativas exprimir seus programas de uma forma natural, apesar da presença de automodificações no mesmo.

II. FUNDAMENTAÇÃO CONCEITUAL

Um programa adaptativo pode ser entendido como sendo um código adaptativo, ou seja, uma sequência automodificável de instruções. À luz da formulação geral apresentada em [7], programas dessa classe podem ser formalmente considerados como sendo dispositivos adaptativos especiais, cujo dispositivo subjacente não-adaptativo seria um programa convencional (este, interpretado como um conjunto de regras).

Analisando-se programas escritos em diversos estilos e paradigmas, constata-se que é viável converter programas escritos em um dado estilo para qualquer outro estilo, preservando-se seu significado. Assim, por exemplo, durante muito tempo foi usual o emprego de práticas baseadas na conversão de programas escritos em estilo recursivo para a forma de programas iterativos e vice versa. [9], [9]. Igualmente, programas desenvolvidos em paradigma funcional e lógico, fortemente baseados no estilo recursivo, costumam fazer uso de recursos de macros, conforme se pode ver em [10].

Considerando que, no estilo adaptativo, a aplicação de automodificações muitas vezes reflete a substituição de grupos de regras por outros grupos de regras, pode-se considerar que

S. R. B. da Silva é professor da Universidade do Estado do Amazonas e cursa mestrado em Engenharia Elétrica pela Universidade de São Paulo (sramosbs@usp.br).

J. J. Neto é o responsável pelo LTA – Laboratório de Linguagens e Técnicas Adaptativas, vinculado ao Departamento de Engenharia e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo – Av. Prof. Luciano Gualberto, trav. 3, No. 158 – Cidade Universitária – São Paulo – SP – Brasil (joao.jose@poli.usp.br) fone 55(11)3591 5402

os formalismos baseados em cálculo lambda [11], que dão apoio à utilização de funções e de macros, possam ser analogamente empregados na formalização do estilo adaptativo.

Macro [10], é um recurso de programação que permite ao programador especificar um procedimento apenas uma vez e usá-lo quantas vezes sejam necessárias dentro do programa, através de uma chamada a essa macro, como se fosse um subprograma (função/procedimento). A principal diferença entre o uso de macro e subprograma reside no fato de que, em tempo de compilação, a macro é acrescentada ao código. Uma vantagem do uso de macro é um aumento de performance do programa, uma vez que esta aumenta a velocidade de execução do código, pois evita o gerenciamento de chamada de função. Todavia, o uso extensivo de macros tende a causar aumentos não desprezíveis no tamanho do código, especialmente no caso em que a macro for grande.

Subprogramas, conforme [11], por sua vez, são procedimentos ditos fechados, cujo código é único no programa, de forma que não acontecem ocorrências múltiplas do mesmo causadas por seu uso em diversos pontos do programa.

No entanto, do ponto de vista formal, pode-se reconhecer que programas em que há algum tipo de repetição podem ser expressos indiferentemente, quer usando macros, quer subprogramas, ou, ainda, técnicas adaptativas, nas quais as chamadas dessas abstrações são expandidas somente quando o comando correspondente for, de fato, acionado em tempo de execução.

Os primeiros computadores foram projetados para processamento de aplicações científicas, as quais utilizavam estruturas de dados simples e elevada quantidade de cálculos em ponto flutuante, de acordo com [12].

A arquitetura básica desses computadores era constituída de uma memória e uma unidade central de processamento – UCP. Os dados e instruções de programa ficam na mesma memória. Os dados são canalizados para a UCP, processados e o resultado retorna para a memória. É a arquitetura de von Neuman. A maioria das linguagens de programação foi projetada em função dessa arquitetura. São as chamadas linguagens imperativas. Essa arquitetura leva os programas a terem as variáveis como recurso central. As instruções são armazenadas em posições contíguas de memória, o que torna o processamento eficiente, diz [12].

Com o passar do tempo, os computadores passaram a ser empregados para outras finalidades e começaram a surgir linguagens de programação com características distintas.

De acordo com as suas características, as linguagens de programação podem ser classificadas em genéricas e de propósito específico, de acordo com 0.

Linguagens genéricas, ou de propósito geral, são as que podem ser utilizadas para o desenvolvimento de aplicações em geral. São dotadas de recursos que permitem empregos na solução de diversos problemas.

Bibliotecas dedicadas podem ser acrescentadas a uma linguagem genérica, o que a torna bem mais poderosa e flexível. Essas bibliotecas são chamadas de extensões funcionais e permitem o desenvolvimento de aplicações de maneira mais eficiente. [14]

A necessidade de implementar determinados tipos de modelos com frequência levou ao desenvolvimento de linguagens com propósitos específicos. São linguagens que possuem interfaces que facilitam o trabalho do programador.

O desenvolvimento de aplicações usando uma linguagem de propósito geral, como foi o caso do FORTRAN, no passado, ou C, C++ e Java, entre outras, nos dias atuais, requer do programador um maior esforço de programação e domínio da linguagem, para o desenvolvimento de solução para problemas específicos, a não ser que se utilize algum bom repositório de bibliotecas disponível. Naturalamente, quando estão disponíveis tais bibliotecas, sua utilização pode até dispensar a de linguagens de propósito específico para essa finalidade.

Um exemplo de linguagem para propósito específico são as linguagens de simulação, projetadas com o fim determinado de permitir o desenvolvimento de simulações, como é o caso das linguagens SIMAN, SIMULA e SIMSCRIPT, como em 0.

Outro tipo de linguagem para fins específicos são as linguagens de marcação. São assim classificadas por utilizarem marcadores, que são palavras chave com função de delimitar blocos de dados e podem conter parâmetros. Como exemplo dessas linguagens temos XML, que foi desenvolvida pela W3C (Word Wide Web Consortium) e é definida como um formato universal de dados estruturados na WEB. [15]

Existem várias outras linguagens de desenvolvimento para Web, entre as quais podemos citar, ainda, OWL, para descrever semântica, RuleML, uma linguagem canônica para regras na Web, SWRL, que propõe combinar RuleML e OWL, como se pode ver em 0.

Linguagens de programação também podem ser classificadas de acordo com o paradigma de programação segundo o qual foram projetadas.

A. *A. Paradigmas*

Problemas podem ser resolvidos de acordo com um padrão de resolução associado a um gênero de linguagem de programação. A esse padrão de resolução de problemas, dá-se o nome de paradigma de programação. Existem diversos paradigmas de programação. De acordo com [11], a partir dos anos 70 quatro desses paradigmas de programação evoluíram: programação imperativa, orientada a objetos, funcional e lógica. Serão descritas, a seguir, sempre com fundamento em [11], as características principais desses quatro paradigmas de programação:

1) Programação Imperativa

O paradigma imperativo tem por base teórica a Máquina de Turing, e como lastro tecnológico, a arquitetura de von Neumann. Nesse paradigma, a idéia central é o uso de efeitos colaterais para a alteração do estado de um programa, e essa função é desempenhada principalmente pelos comandos de atribuição – que consistem em alterar o estado de um programa através da substituição de um valor, contido em uma posição de memória, por algum outro valor. Nesse paradigma,

a atribuição é uma idéia central.

Quando uma linguagem é capaz de fornecer uma base que permita a implementação de qualquer algoritmo possível de ser projetado, essa linguagem é Turing-Completa. Dessa forma, uma linguagem de programação imperativa que disponibilize variáveis e valores inteiros, as operações aritméticas básicas, comandos de atribuição, comandos condicionais e iterativos é considerada Turing-completa.

Além de atribuição, as linguagens de programação imperativas costumam disponibilizar ao programador: declaração de variáveis, expressões, comandos condicionais, comandos iterativos e abstração procedural.

Segundo [12], abstrair é empregar apenas as informações relevantes sobre um problema para representá-lo dentro de uma determinada perspectiva. Destina-se a simplificar o raciocínio e o processo de programação.

A abstração procedimental dá ao programador a possibilidade de atentar para as relações existentes entre um procedimento e a operação que ele realiza (em particular, entre uma função e o cálculo que ela executa) sem preocupações acerca da maneira como essas operações são realizadas.

O refinamento gradual é uma maneira sistemática de desenvolver programas. Usando abstração procedimental, um programador pode particionar uma função, idealizada de forma macroscópica, em um grupo de funções mais simples e/ou mais específicas.

Com o fim de simplificar o desenvolvimento de algoritmos complexos, as linguagens imperativas modernas possuem suporte a matrizes e estruturas de registro, além de bibliotecas extensíveis de funções, que evitam que operações comuns necessitem ser reprogramadas, como é o caso de operações de entrada e saída de dados, gerenciamento de memória, manipulação de cadeias e de estruturas de dados clássicas, e tantas outras.

São exemplos de linguagens imperativas FORTRAN e C.

Como parte do processo de evolução da programação imperativa, buscou-se estender a abstração procedimental para incluir tipos de dados abstratos. Isto se deu através da criação e incorporação do conceito de encapsulamento de tipos, ainda no final da década de 60. Este foi um passo importante para o desenvolvimento do paradigma de programação orientada a objetos, que será comentado a seguir.

2) Programação Orientada a Objetos

Encapsular é agrupar constantes logicamente relacionadas, tipos, variáveis, métodos e outros em uma nova entidade.

A abstração de dados encapsula os tipos de dados e as respectivas funções em um único bloco ou pacote. Mas o bloco (ou pacote), não tem mecanismo de inicialização e finalização de um valor, nem uma maneira simples de acrescentar novas operações. Essas duas situações foram solucionadas pela idéia de classe, um conceito fundamental para a orientação a objetos.

A orientação a objetos envolve a utilização de conceitos, tais como o de classes, herança e polimorfismo, devendo-se notar, no entanto, que nem todas as características teóricas que a definem, estão obrigatoriamente presentes em todas as linguagens consideradas aderentes a esse paradigma.

As classes existem dentro de uma hierarquia. Uma classe pode ser subclasse de outra e esta será, neste caso, considerada sua superclasse. Assim, uma subclasse poderá herdar de sua superclasse variáveis e métodos. A herança é uma forma de reutilização de código.

Quando uma classe herda, de mais de uma superclasse, variáveis e métodos, identifica-se aí o conceito de herança múltipla.

Quando, da chamada de um método, resultarem chamadas a mais de uma forma de implementação de tal método, observa-se uma instância do conceito de polimorfismo.

As linguagens Smalltalk e Java são, basicamente, orientadas a objetos.

3) Programação funcional

O paradigma funcional tem como fundamentação teórica o cálculo Lambda. Devido ao uso extensivo da recursão, apóia-se, tecnologicamente, em arquiteturas nas quais o programador possa, de forma relativamente natural, fazer uso da recursão para exprimir sua lógica e executar seus programas.

O paradigma funcional baseia-se, fundamentalmente, no conceito de função, e tem como uma de suas principais características o conceito de transparência referencial, que se traduz, na prática, na ausência de efeitos colaterais.

Para que isso seja possível, diferentemente do que ocorre em linguagens imperativas, uma linguagem concebida estritamente de acordo com o conceito do paradigma funcional não utiliza variáveis nem comandos de atribuição.

As estruturas típicas encontradas em uma linguagem funcional são:

- Um conjunto de dados, que são representados por estruturas de alto nível, baseadas em listas.
- Um conjunto de definições de funções primitivas préconstruídas, destinadas, principalmente, à manipulação dos dados.
- Especificações de aplicações das funções, que podem ser formas funcionais para construção de novas funções.

A ausência de variáveis impossibilita a construção de programas baseados em lógica iterativa, uma vez que estes exigem variáveis de controle.

Tarefas repetitivas devem, portanto, ser expressas por algum outro tipo equivalente de estrutura, e nas linguagens funcionais costumam ser realizadas por meio de procedimentos e de funções recursivas.

A linguagem LISP foi a primeira linguagem de programação funcional. No entanto, nos dias atuais, incorpora características de outros paradigmas. Outro exemplo de linguagem funcional é a linguagem ML.

4) Programação Lógica

O paradigma lógico apóia-se na fundamentação teórica proporcionada pela Lógica de Primeira Ordem. A recursão também é extensivamente empregada na programação em lógica, a qual se apóia, tecnologicamente, nas arquiteturas que reduzam para o programador as dificuldades impostas pelo uso da recursão.

A programação em lógica tem por base a programação declarativa, na qual se procura privilegiar a especificação da tarefa específica que se deseja que o computador execute, evitando que o programador especifique de que forma o computador deverá proceder para realizá-la.

Dessa forma, enquanto nos demais paradigmas a meta do programador é de informar à máquina a trajetória exata a ser percorrida pelo programa, na programação lógica o programador deve limitar-se a especificar as premissas e os alvos a serem atingidos, deixando para a máquina a determinação do caminho a ser percorrido para solucionar o problema.

Para tanto, um programa lógico costuma efetuar processamento simbólico, não numérico. Programas lógicos costumam usar como linguagem de programação alguma implementação da lógica simbólica.

A exemplo do que acontece com programas escritos no paradigma funcional, os programas lógicos costumam utilizar a mesma notação formal para representar programas e dados, não fazendo distinção entre argumentos de entrada e saída.

Assim como os programas escritos em paradigma funcional, as linguagens voltadas à programação em lógica não costumam dispor de comandos de controle para especificar repetições, os quais são extensivamente substituídos por formas recursivas equivalentes para esta finalidade.

Um programa em linguagem lógica é tipicamente constituído de cláusulas, que podem ser de dois tipos: fatos, que são considerados verdadeiros por definição, e regras, a serem aplicadas sobre os fatos e outras regras, e cuja avaliação pode resultar verdadeira ou falsa.

Prolog, Planner e Oz são linguagens para programação lógica.

Embora algumas linguagens de programação originalmente tenham sido projetadas estritamente de acordo com um dos paradigmas acima, a elas posteriormente foram incorporadas características de outros paradigmas, de modo que raramente se encontra uma linguagem que possa, atualmente, ser classificada como sendo autenticamente pertencente a um ou outro desses paradigmas.

Existem linguagens, denominadas extensíveis, que permitem ao programador definir extensões a uma linguagem básica inicial, facilitando, assim, a criação de características específicas para a solução de classes particulares de problemas de seu interesse.

B. B. Extensibilidade

As primeiras publicações científicas sobre a extensibilidade de linguagens de programação ocorreram ainda na década de 60 e prosseguiram nos anos 70, inclusive com a realização de simpósios sobre linguagens extensíveis nos anos de 1969 e 1971, diz [17].

De acordo com [18], a extensibilidade possibilita ao programador modificar os recursos de uma linguagem de programação, tornando-a adequada a propósitos específicos.

A extensibilidade de linguagens de programação pode ocorrer de três formas:

Extensibilidade léxica, segundo a qual criam-se abreviaturas, paramétricas ou não, para trechos específicos de texto, de tal modo que, toda vez que tal abreviatura for encontrada, o trecho de texto a elas associado é usado em substituição à abreviatura. O mesmo é feito em relação aos argumentos em substituição aos correspondentes parâmetros.

A Linguagem C costuma ser implementada como linguagem lexicamente extensível, uma vez que permite a criação de abreviaturas textuais (DEFINE).

A extensibilidade funcional consiste em encapsular novas funcionalidades a funções definidas e tradicionalmente usadas de uma linguagem de programação. [19].

LISP é um exemplo de linguagem funcionalmente extensível.

A extensibilidade sintática diz respeito à capacidade de uma linguagem de programação permitir a adição de novos constructos sintáticos, sendo que estes podem ser combinados com os já existentes, ainda conforme [18].

Para a solução de problemas de naturezas diversas, vários estudos vêm sendo desenvolvidos empregando programação adaptativa.

C. C. Programação adaptativa

Um fenômeno adaptativo é caracterizado por um comportamento dinâmico, que altera seu funcionamento em função de ter atingido alguma situação particular e, nesta situação, ter recebido um determinado estímulo. Uma maneira possível de representar um comportamento adaptativo através de uma linguagem de programação é utilizar uma linguagem, cujo código executável tenha a capacidade de modificar-se ao acrescentar ou remover porções de seu código a si próprio durante a execução. E muitas das mais importantes linguagens atuais não apenas não foram projetadas para apresentarem essa característica como também incluem propriedades que dificultam ao usuário a execução de tais operações em seus programas.

Linguagens de programação devem, por sua natureza e propriedades, ser classificadas, de acordo com a hierarquia de Chomsky, como linguagens dependentes de contexto (ou sensíveis ao contexto), e, portanto, geradas por gramáticas sensíveis ao contexto. Considerações de ordem prática, entretanto, fazem ser preferível representá-las não em sua forma autêntica, mas aproximada, por variantes livres de contexto, com a vantagem de disponibilizar para elas todo um ferramental disponível para este tipo de linguagens menos complexas. As lacunas introduzidas por tal simplificação, todavia, devem ser compensadas por meio do acréscimo de procedimentos auxiliares que efetuem as operações necessárias ao tratamento das dependências de contexto, e que não são supridas pelo formalismo livre de contexto que define a linguagem simplificada.

Uma linguagem para programação adaptativa constitui um caso particular de linguagens de programação, diferenciando-se das não-adaptativas pela capacidade que possui de permitir a geração de código automodificável.

A próxima seção analisa os requisitos de uma linguagem para programação adaptativa, comenta sobre as exigências do

ambiente de execução e apresenta uma proposta de uma linguagem dessa natureza.

III. PROPOSTA DA LINGUAGEM

A pesquisa acerca do desenvolvimento de uma linguagem para programação adaptativa vem se desenvolvendo há alguns anos, conforme se pode ver em [1], [20][21].

Buscando contribuir com mais um passo nessa direção, propõe-se uma linguagem básica de alto nível, em estilo imperativo, que permite escrever códigos fonte que originarão programas adaptativos, de acordo com o que foi sugerido anteriormente.

Faz-se mister definir, neste ponto, os termos código automodificável e linguagem para programação adaptativa, de acordo com [22].

Neste trabalho, adota-se o termo "código adaptativo" para significar um código executável cujas propriedades permitem que o mesmo se automodifique autonomamente, de forma dinâmica, por acréscimo e/ou supressão de comandos do programa.

Um programa escrito em alguma linguagem de alto nível convencional pode ser considerado estático, na medida em que não pode ser executado diretamente por uma máquina, necessitando, para tal, ser antes interpretado, ou então transformado em código executável por um compilador.

Mesmo assim, se essa linguagem de alto nível permite a produção de código fonte que apresenta comportamento dinâmico, pode-se considerar que se trata de um código adaptativo, pela idéia que traduz, estendendo, dessa forma, o significado dessa expressão ao código-fonte.

Trabalhos anteriores relativos às bases para o estabelecimento de uma linguagem que permita escrever programas com as características acima citadas, como em [20], adotavam o termo "linguagem adaptativa" para significar uma linguagem que permita escrever códigos adaptativos.

Vista como um dispositivo guiado por regras [7], a linguagem não pode se alterar, ainda que as regras gramaticais se modifiquem durante a utilização da gramática. No entanto, o conjunto de sentenças compreendido na linguagem não pode se alterar, ou a linguagem não mais seria aquela.

Assim, é fácil intuir que o termo "linguagem adaptativa" não se aplica e, portanto, passa-se a adotar a denominação "linguagem para programação adaptativa", para especificar uma linguagem de alto nível que venha a permitir o desenvolvimento de códigos adaptativos.

Para [20], é desejável que uma linguagem para programação adaptativa deva ter, como requisitos, principalmente:

- maneiras de endereçar a porção de código que poderá ser modificada, como parte de sua sintaxe, bem como que tipo de modificação será efetuada.

- Estar associada a um ambiente de execução, que possibilite ao compilador referenciar, no código objeto gerado, as operações adaptativas definidas para esse ambiente de execução.

- Um conjunto de operadores adaptativos, que expressem de forma clara e intuitiva as operações adaptativas disponíveis.

Considerando que a lógica de um programa com código automodificável pode ser desenvolvida na forma de um programa convencional, de código estático, pode-se argumentar que a adaptatividade seja supérflua, e dado que o código necessário para redigir programas automodificáveis tende a ser mais extenso e mais complexo que um programa convencional, a programação adaptativa pode exigir práticas nem sempre alinhadas com as usualmente recomendadas para a construção de programas, atualmente considerados de boa qualidade.

Em atenção a posicionamentos como esses, os sistemas operacionais atuais dispõem de mecanismos de segurança que procuram impedir que os códigos executáveis em memória sofram alterações (ou mesmo se imponham automodificações). Esse aspecto dificulta muito, até mesmo, eventualmente, impedindo que um código adaptativo possa ser executado em memória, diretamente sob controle do sistema operacional.

Dentro das limitações de espaço disponível neste trabalho, procuramos apresentar as características mais relevantes dessa linguagem, que passamos a descrever.

a) A implementação da adaptatividade nos programas desenvolvidos nesta linguagem é feita mediante o acréscimo, por extensão sintática, de uma camada adaptativa a um código subjacente não-adaptativo, desenvolvido em alguma linguagem hospedeira disponível e compatível.

b) A linguagem resultante deverá ter sua própria característica, ditada essencialmente pelas extensões acrescentadas à linguagem hospedeira, e que conferem a adaptatividade ao código resultante, mas os programas automodificáveis nela desenvolvidos conterão, como elementos de composição, um conjunto de trechos estáticos, escritos puramente na linguagem hospedeira.

c) Cada trecho estático deve ser escrito de tal forma que seu código esteja em conformidade ao que está especificado em [1]: uma só entrada, uma só saída. Neste artigo, por simplicidade, estão omitidos todos os detalhes da linguagem hospedeira, pois o foco são apenas os recursos sintáticos que se referem à definição dos mecanismos adaptativos no código.

d) A interligação entre os trechos estáticos deve estar, também, em conformidade com [1] para garantir a coerência dos grafos que representam os programas assim construídos. Para isso, convencionou-se, na linguagem proposta, que a declaração dessas interligações se faça sempre a partir da saída de trechos estáticos do programa, e que seja indicada a condição.

e) Convencionou-se, também, que cada trecho estático seja responsável pelo cálculo de um código numérico, que represente univocamente a condição segundo a qual, após a execução do trecho estático corrente, deva ocorrer a ativação de um outro trecho estático do programa.

f) A associação de ações adaptativas às conexões existentes entre os trechos estáticos é que proporciona a adaptatividade aos programas assim desenvolvidos, e isso deve ser feito também de acordo com o especificado em [1].

g) Logo, na camada adaptativa de uma linguagem que deva proporcionar esses recursos ao programador, devem estar

presentes, no mínimo, os elementos sintáticos que permitam:

- Delimitar trechos de código, e identificar tais trechos, estática e dinamicamente, de modo que possam ser referenciados em outra parte do programa quando necessário. Exemplos: declaração de nome, início e final de um trecho de programa.
- Estabelecer conexões entre trechos iniciais do programa, previamente delimitados e identificados, com a finalidade de montar, a partir desses trechos, um grafo inicial subjacente de fluxo para o programa.
- Associar a cada conexão existente, se necessário, algum tipo de atividade de automodificação para o programa através de chamadas de funções adaptativas. Na linguagem aqui proposta, tratam-se de chamadas de funções adaptativas anteriores apenas, não estando disponíveis funções adaptativas posteriores. Os principais casos de atividades de automodificação contemplados nesta proposta são: indicação do trecho a ser processado a seguir, acompanhado da condição de saída do trecho identificado anterior que causa a ativação da presente conexão; indicação da aplicação de funções adaptativas sobre o programa.
- Mecanismos sintáticos para a declaração de funções adaptativas e suas chamadas.
- Mecanismos sintáticos de acesso à ativação de operações adaptativas elementares, tais como consultas, inclusões e exclusões: de trechos, de conexões entre trechos, e da associação entre conexões e chamadas de funções adaptativas.
- Mecanismos sintáticos para a criação dinâmica de novos trechos delimitados e identificados de código, conexões e associações de chamadas de funções adaptativas.
- Mecanismos sintáticos que permitam a especificação da geração de novos identificadores inéditos para trechos recém-criados, sempre que necessário.
- Mecanismos sintáticos de consulta e de memorização de resultados de buscas.

h) A maior independência possível da linguagem proposta em relação à linguagem hospedeira.

Será apresentada, a seguir, no Quadro I, a gramática simplificada da linguagem proposta neste trabalho, a qual foi denominada de Basic Adaptive Language – BADAL. No entanto, trata-se de uma gramática ainda incompleta, cuja linguagem encontra-se em evolução. Dessa forma, nem todas as características acima enunciadas estão presentes na linguagem, como por exemplo, a criação dinâmica de trechos de programas delimitados e identificados.

QUADRO I
GRAMÁTICA SIMPLIFICADA DA LINGUAGEM BADAL

```
programa-adaptativo = "{{" "Start" "at" nome "where"
                          declarações-adaptativas "}}" .

declarações-adaptativas =
          decl-trechos-identificados ";"
          decl-conexões-adaptativas ";"
          decl-funções-adaptativas.

decl-trechos-identificados =
          { decl-trecho-identificado ";" } .

decl-trecho-identificado =
          "Code" nome "is" ":"
          "{" chamada-de-procedimento-hospedeira "}" .

chamada-de-procedimento-hospedeira =
     << chamada de subrotina ou similar na
         linguagem hospedeira >> .

decl-conexões-adaptativas = { conexão-adaptativa ";" }
.

conexão-adaptativa =
     "Conect" "output" "(" nome ")" ":"
                              "//" {   número ":" Link nome
"(" "to" nome ")"
                                                    ["," "Call"
nome
                    "(" [ nome { "," nome } ] ")" ] ";"
                }
                                                  "else" ":"
"Link" nome "(" "to" nome ")"

          ["," "Call" nome
                    "(" [ nome { "," nome } ] ")" ] ";"
                              "//" .
decl-funções-adaptativas =
        { decl-função-adaptativa ";" } .

decl-função-adaptativa =
     "Adapt" nome "(" [ nome { "," nome } ] ")" ";"
                                   "Variable" nome { ","
nome } ";" }
                                   "Generator" nome {
"," nome } ";" }
     "is" ":"
                                   [ "Call" nome
                    "(" [ nome { "," nome } ] ")" ";" ]
                                   "//" {pesquisas}
{remoções} {inserções} "//"

pesquisas =
   { "Search" "Link" nome "from" nome "to" nome ";" } .

remoções  =
   { "Remove" "Link" nome ";" } .

inserções =
   { "Add" (
        "[" "Code" nome "is" ":"
```

Com o fim de demonstrar o uso da linguagem ora proposta, considerou-se um simulador de pilha, apresentado por [24], o qual foi descrito como se segue:

Produções iniciais:

$(1 , "\beta") : \rightarrow 4$

$(2 , "\beta") : \rightarrow 3$

$(3 , ")") : \rightarrow 4$

$(1 , "(") : \rightarrow 2 , \mathcal{A} (2 , 3 , 1)$

Função adaptativa

$\mathcal{A} (i , j , n) = \{ k^* , m^* :$

$+ [(k , "\beta") : \rightarrow m]$

$+ [(m ,$

")") : → j]

+ [(i , "(") : → k , $\mathcal{A}$ (k , m , i)]

- [(n , "(") : → i , $\mathcal{A}$ (i , j , n)]

+ [(n , "(") : → i] }

A representação gráfica desse simulador de pilha consta da Fig. 1.

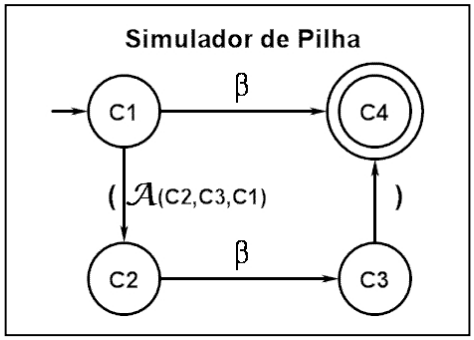


Fig. 1 Simulador de Pilha

A função adaptativa $\mathcal{A}$, realiza as seguintes ações adaptativas:

- Ao ser lido "(", na cadeia de entrada, a transição associada a $\mathcal{A}$ é removida e outra transição sem a função adaptativa associada é adicionada;
- Mais dois estados são criados;
- Três transições são adicionadas: uma associada à função adaptativa $\mathcal{A}$, outra relativa ao símbolo "β" e mais outra correspondente ao símbolo ")".

Consta da Fig. 2. uma ilustração da configuração assumida pelo autômato após ter sido lido o símbolo "(" na cadeia de entrada e a função adaptativa ter sido executada.

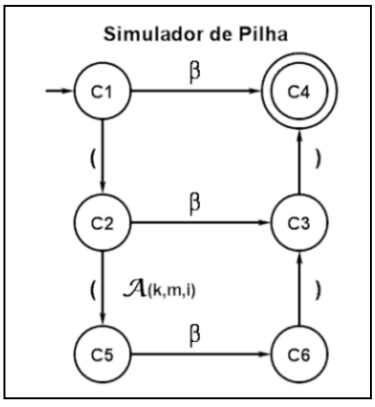


Fig. 2. Simulador de pilha após a execução da função $\mathcal{A}$

Um exemplo de um programa adaptativo, escrito na linguagem BADAL, que representa o simulador de pilha acima é apresentado no Quadro II. C1 é o nome do programa.

QUADRO II
EXEMPLO DE PROGRAMA ADAPTATIVO

```
% programa-adaptativo
% C1 é o nome do trecho do código em que o programa
% tem início

{{ Start at C1 where
```

QUADRO II (CONTINUAÇÃO)

```
% trechos identificados de código
                Code C0 is : { erro ( ) } ;
                Code C1 is : { } ;
                Code C2 is : { } ;
                Code C3 is : { } ;
                Code C4 is : { stop ( ) } ;
% conexões entre os trechos identificados
% (incluindo as adaptativas)

% códigos de saída convencionados:
%               1 = "(" ,    2 = beta ,    3 = ")"

                Conect output (C1) :
                        // 1     : Link L12 ( to C2 ) ,
                Call FA ( C2, C3, C1 ) ;
                         2       : Link L14 ( to C4 ) ;
                           Else : Link L10 ( to C0 ) ;
//
                Conect output (C2) :
                        // 2     : Link L23 ( to C3 ) ;
                           Else : Link L20 ( to C0 ) ;
//
                Conect output (C3) :
                        // 3     : Link L34 ( to C4 ) ;
                           Else : Link L30 ( to C0 ) ;
//
% funções-adaptativas

                  Adapt FA ( PI, PJ, PN ) ;
              Variable lv;
              Generator lk, lm, lkm, k, m;

         is :     //       % remove o Link que usa "(" e
chama FA
                                  Search Link lv from PN
to PI ;
                                  Remove Link lv ;

                % restaura o Link removido,
     % eliminando a chamada de FA
                                  Add [ 1 : Link lv ( to
PI ) From PN ] ;

                                  % cria dois novos
códigos - k e m
                                  Add [ Code k is : { }
] ;
                                  Add [ Code m is : { }
] ;

                                  % cria novo Link lk,
de PI para k,
          % com chamada de FA
                                  Add [ Case 1 : Link lk
```

IV. CONSIDERAÇÕES FINAIS

Tem sido observado nas publicações científicas um interesse pela retomada do uso da automodificação de código para a solução de problemas de diversas naturezas [4], [23].

Esses trabalhos apresentam código automodificável desenvolvido em linguagem de baixo nível [4], [5], ou por extensões a uma linguagem de alto nível [20].

Ao mesmo tempo, despontam, também, publicações sobre os fundamentos de uma linguagem de alto nível que permita a programação de código capaz de se automodificar [1], [20]. Buscando contribuir com essa perspectiva, este artigo propõe uma linguagem básica, que permite desenvolver programas em alto nível com características adaptativas.

A linguagem ainda está em desenvolvimento. Entretanto, após a conclusão do trabalho, espera-se disponibilizar uma

linguagem através da qual seja possível programar códigos adaptativos.

Espera-se, ainda, que o presente trabalho seja motivador de novos estudos, nos quais se possa avaliar os custos computacionais de tempo e de espaço ocupado por repetidas adições de código, durante a execução, que não foram contemplados neste estudo.

V. REFERÊNCIAS

[1] É. J. Pelegrini, "Códigos Adaptativos e linguagem para programação adaptativa: conceitos e tecnologia." Dissertação de Mestrado, apresentada à Escola Politécnica da USP. São Paulo. 2009

[2] C. Hongxu; S. Zhong; V. Alexander. "Certified self-modifying code". SIGPLAN Notices. Vol. 42(6), 2007. pp. 66-77.

[3] B. Anckert; M. Madou, K. D. Bosschere. "A Model for Self-Modifying Code". Lecture Notes in Computer Science, Springer Berlin / Heidelberg, ISSN 0302-9743, Volume 4437/2007, Information Hiding, ISBN 978-3-540-74123-7, Pages 232-248, September 14, 2007.

[4] J. T. Giffin, M. Christodorescu, L. Kruger. "Strenghtening software self-checksumming via self-modifying code". Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005).

[5] Y. Kanzaki, A. Monden, M. Nakamura, K. Matsumoto. "Exploiting selfmodication mechanism for program protection". In Proc. of the 27th Annual International Computer Software and Applications Conference, pages 170-181, 2003.

[6] J. J. Neto. "Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa." Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 496-505).

[7] J. J. Neto. "Adaptive Rule-Driven Devices - General Formulation and Case Study." Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol.2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250.

[8] R. S. Bier. "Recursion elimination with variable parameters". The Computer Journal 1979 22(2):151-154; doi:10.1093/comjnl/22.2.151 by British Computer Society. 1979.

[9] Y. A. Lui, S. C. Stoller. "From Recursion to iteration. What are the optimization?" In Proceedings of the ACM SIGPLAN Workshpo on Partial Evaluation and Semantics-Based Programming Manipulation (PEPM '00), pages 73-82. ACM Press, Jan. 2000.

[10] M. Flatt. Composable and Compilable Macros. Proceedings of the seventh ACM SIGPLAN international conference on Functional programming. Pag. 72-83. 2002.

[11] A. B. Tucker e R. E. Noonan. *Linguagens de programação: princípios e paradignmas*. 2ª. ed. São Paulo: Mc Graw Hill. 2009. p. 362-366.

[12] R. W. Sebesta. *Conceitos de Linguagens de Programação*. 5ª. Ed. São Paulo: Bookman, 2002.

[13] P. J. da Rocha, MARQUES, Silva S. "Simulação de um sistema automático de logística interna para a indústria de calçados." Dissertação de mestrado em Automação, Instrumentação e Controle, apresentada à Universidade do Porto, Portugal. 2007.

[14] M. Bravenboer e E. Visser. "Designing Syntax Embeddings and Assimilations for Language Libraries". In *Models in Software Engineering: Workshops and Symposia at MoDELS 2007*, volume 5002 of LNCS, 2008.

[15] W3C. "Extensible Markup Language. (XML)", disponível em http://www.w3.org/XML/, acessado em 16/01/1010.

[16] A. Kamada. "Execução de serviços baseada em regras de negócios." Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação, da Universidade Estadual de Campinas. São Paulo. 2006.

[17] T. Standish. "Extensibility in programming language design". ACM SIGPLAN Notices, Volume 10, Issue 7, Pages: 18–21, ISSN: 0362-1340, 1975.

[18] J. Alghamdi, J. Ilrhaa**. "**Comparing and Assessing Programming Languages: Basis for A Qualitative Methodology". Proceedings of ACM/SIGAPP, pag. 222-229. 1993.

[19] R. Rüdiger. "Extensible programming in Oberon. A tutorial". Technical Report. FH Braunchweig /Wolfenbüttel – University of Applied Science. 1999.

[20] A. V. de Freitas. "Considerações sobre o desenvolvimento de linguagens adaptativas de programação." Tese de Doutorado, apresentada à Escola Politécnica da Universidade de São Paulo. 2008.

[21] A. A. de Castro Júnior. "Aspectos de projeto e implementação de linguagens para codificação de programas adaptativos." Tese de Doutorado apresentada à Escola Politécnica da Universidade de São Paulo. 2009.

[22] J. J. Neto. "Um glossário sobre adaptatividade". III Workshop de Tecnologias adaptativas. Universidade de São Paulo. 2009.

[23] C. Tschudin, L.Yamamoto. "Harnessing Self-Modifying Code for Resilient Software". Proc 2nd IEEE Workshop on Radical Agent Concepts (WRAC), NASA Goddard Space Flight Center Visitor's Center, September 2005. Springer LNCS/LNAI 3825, 2006.

[24] J. J. Neto. "Contribuições à metodologia de construção de compiladores". Tese de Livre Docência, apresentada à Escola Politécnica da Universidade de São Paulo. 1993.

Salvador Ramos Bernardino da Silva é graduado em Estatística (1986). Atualmente, é professor do Curso de Engenharia de Computação da Universidade do Estado do Amazonas – UEA e aluno do Curso de Mestrado em Engenharia Elétrica, do Departamento de Engenharia de Computação e Sistemas Digitais – PCS, da EPUSP. Suas áreas de interesse são, principalmente: programação de computadores, teoria da computação, compiladores. Está desenvolvendo pesquisa sobre dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e suas aplicações à Engenharia de Computação, particularmente sobre programação adaptativa.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Tecnologia Adaptativa do PCS – Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

Reconhecimento de Padrões em Classificadores – Comparação de Técnicas e Aplicações

R. L. Stange, J. J. Neto

***Resumo* — O reconhecimento de padrão utiliza diferentes abordagens em classificadores, a estatística e a sintática são as abordagens de maior interesse neste trabalho. A escolha de um método de classificação depende da natureza do problema. Este trabalho tem o objetivo de comparar técnicas utilizadas para a construção de classificadores e de suas aplicações. No final, é apresentada uma proposta para construção de classificadores, baseada em métodos híbridos incluindo métodos adaptativos.**

***Palavras chaves* — Adaptatividade, Classificadores, Métodos de Reconhecimento de Padrão, Técnicas de Aprendizagem.**

I. INTRODUÇÃO

O PROBLEMA de reconhecimento de padrão é um problema de classificação. O objetivo é discriminar amostras de objetos e classificar corretamente amostras futuras. Objetos ou indivíduos são caracterizados por um conjunto de atributos. Cada atributo possui um valor. Os atributos podem ser categóricos, binários, ordinais ou contínuos. Em alguns casos, é necessária a conversão dos valores de atributos (ex: discretização, binarização).

O estudo em reconhecimento de padrão está em constante desenvolvimento e tem alcançado diversas áreas de aplicação, tais como, biologia, medicina, agricultura, entre outras.

Atualmente, diferentes técnicas provenientes de diversas áreas de pesquisa são utilizadas para o desenvolvimento de algoritmos computacionais para reconhecimento de padrõesAs principais abordagens segundo Jain et al. [3] são: e*statística:* baseada nos modelos probabilísticos para geração de padrão e na teoria da decisão (ex: aprendizagem bayesiana); *sintática e estrutural:* fundamentada principalmente na teoria das linguagens formais (ex: aprendizagem de árvores de decisão, inferência gramatical); *neural:* o conjunto de pesos é a base para geração de padrão na abordagem neural. Essa abordagem incorpora as propriedades das redes neurais para classificação (ex: redes neurais artificiais). Alguns autores incluem a abordagem baseada na lógica fuzzy para classificação não-determinística, chamada de abordagem *difusa*. Além dessas, a principal abordagem para aprendizagem não-supervisionada é conhecida como *Clustering* [1].

R. L. Stange é estudante de Mestrado em Engenharia Elétrica da Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Computação e Sistemas Digitais, Av. Prof. Luciano Gualberto, travessa 3, número 158 - Cidade Universitária - São Paulo - SP - CEP: 05508-900 (rlstange@usp.br).

J. J. Neto atua no Departamento de Engenharia de Computação e Sistemas Digitais (PCS) da Escola Politécnica (POLI) da Universidade de São Paulo (USP), Av. Prof. Luciano Gualberto, Trav. 3, N. 158 - CEP: 05508-900 - São Paulo/SP, Brasil. (joao.jose@poli.usp.br).

A finalidade deste trabalho é comparar técnicas para reconhecimento de padrão provenientes de diferentes áreas de aplicação.

II. ABORDAGEM GERAL SOBRE RECONHECIMENTO DE PADRÃO

A definição de aprendizagem de máquina para Mitchell [5] é "um programa de computador é dito aprender a partir de uma experiência *E* com respeito a uma classe de tarefas *T* e medida de desempenho *P*, se seu desempenho nas tarefas em *T*, segundo a medida *P*, melhora com a experiência *E*". Para Witten et al. [13] a aprendizagem é a capacidade que os programas de computador têm de aprender coisas que mudam seu comportamento e melhoram seu desempenho futuramente. Na aprendizagem de máquina, de acordo com Russel et al. [9], dispositivos computacionais se adaptam a novas circunstâncias para detectar padrões.

O campo da aprendizagem de máquina incorpora o reconhecimento de padrões. O reconhecimento de padrões considera a capacidade de programas de computador encontrar regularidades através de experiências e melhorar automaticamente [5].

O reconhecimento de padrão de acordo com Theodoridis et al. [10] é a descoberta automática de regularidades em dados através de algoritmos computacionais e uso dessas regularidades para classificar objetos em categorias ou classes.O termo genérico padrão é utilizado para referir-se a esses objetos. A figura 1 apresenta um diagrama geral para sistemas de reconhecimento de padrão [1].

O processo de reconhecimento de padrão é divido basicamente em duas fases: treinamento (aprendizagem) e reconhecimento (classificação). A fase de reconhecimento ou classificação tem início com a aquisição dos dados e sensoriamento, nesta tarefa ocorrem as medições das variáveis físicas. Em seguida um pré-processamento é realizado para remoção de ruídos nos dados. Além disso, qualquer outro tratamento realizado sobre os dados capturados ocorre nesta fase. Na extração de características é necessário encontrar uma nova representação para o padrão em termos de funcionalidades. No modelo de aprendizagem é necessário mapear as características entre grupos de padrões e categorias. A classificação usa as características e o modelo de aprendizagem para atribuir uma categoria a um padrão. Em seguida é realizada a avaliação de confiança nas decisões.

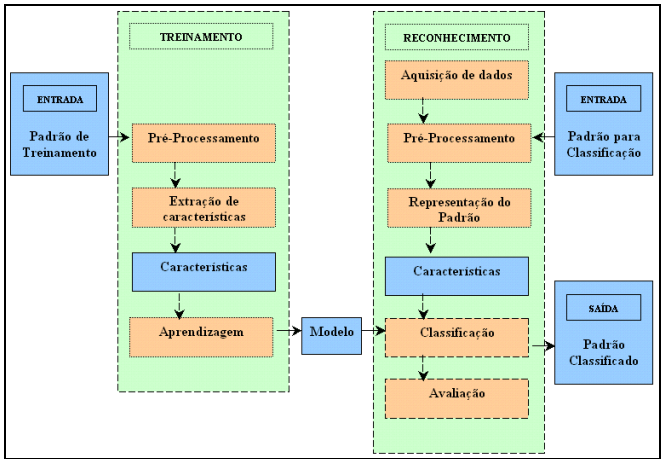


Figura 1 – Sistema de reconhecimento de padrão

Na componente de treinamento a finalidade é inferir, a partir de um conjunto de exemplos de treinamento, um conjunto de regras que define um padrão. O resultado final do treinamento é um modelo de aprendizagem que define o classificador. Na componente de reconhecimento, o classificador é utilizado com a finalidade de classificar cadeias de entrada de acordo com critérios definidos durante sua construção. A cadeia de entrada é uma sequência de estímulos que alimenta o dispositivo. O modelo apresenta duas entradas. Na entrada para treinamento, exemplos de um padrão, chamados exemplos positivos, são inseridos para aprendizagem. Em alguns modelos são inseridos exemplos que não pertencem ao padrão, esses exemplos são conhecidos como exemplos negativos. Na entrada para reconhecimento, um padrão não classificado é inserido para ser classificado de acordo com o modelo de aprendizagem definido durante o treinamento. O critério de classificação é definido a partir das diferentes técnicas de reconhecimento de padrão.

A seguir, são apresentadas técnicas de reconhecimento de padrão utilizando métodos estatísticos e métodos determinísticos.

A. Aprendizagem baseada em métodos estatísticos

Os métodos estatísticos tendem a obter resultados rapidamente. Os métodos estatísticos são apropriados quando o foco é evidenciar tendências no espaço amostral e quando as particularidades dos indivíduos são irrelevantes.

Na abordagem estatística, cada padrão é representado em termos de *d* características ou medições, onde *d* é o número de características. O padrão é visto como um ponto em um espaço *d*-dimensional. A finalidade é escolher características que separem vetores de padrão, pertencentes a diferentes categorias, em regiões compactas. A eficácia da representação do espaço (conjunto de características) é determinada por quão bem os padrões de classes distintas podem ser separados. Dado um conjunto de padrões de treinamento, o objetivo é estabelecer fronteiras de decisão no espaço de característica que separa padrões pertencentes a diferentes categorias [1] [3].

Na abordagem baseada em teoria de decisão estatística o Teorema de Bayes é aplicado e as fronteiras de decisão são determinadas pelas distribuições de probabilidade de padrões pertencentes a cada classe[1][5]. Além da teoria da decisão, uma abordagem baseada na análise discriminante para a classificação pode ser adotada, neste caso podemos utilizar o Discriminante Linear de Fisher [1].

Considere os padrões $X = (x_1, x_2,..., x_d) \in \mathbb{R}^d$ e as classes $c = \{\omega_1, \omega_2,..., \omega_c\}$, em reconhecimento de padrões. Estamos interessados em associar um padrão a uma classe. A abordagem Bayesiana supõe que as probabilidades de cada classe $P(\omega_i)$ e as densidades de probabilidade condicionais $\rho(x|\ \omega_i)$ de x com respeito a cada um das classes ω_i i = 1,2,...,c, são conhecidas. Na ausência de qualquer outra informação, poderíamos classificar um padrão x como sendo da classe ω_i de maior probabilidade. Porém, dado que x foi observado, isto parece uma decisão muito ingênua, pois o classificador acertaria a classificação com probabilidade $P(\omega_i)$, porém erraria sempre com probabilidade $\sum_{i\neq j} P(\omega_j)$. Como temos as condicionais, podemos utilizar o teorema de Bayes e calcular a probabilidade $P(\omega_i|x)$, ou seja [1][10]:

$$P(\omega_i|\mathbf{x}) = \frac{P(\omega_i)p(\mathbf{x}|\omega_i)}{p(\mathbf{x})}$$

Na qual $P(\omega_i)$ é a *priori* , $\rho(x|\ \omega_i)$ é a densidade condicional ou verossimilhança, $\rho\ (x) = \bullet\ ^{c}_{i\neq j} P(\omega_j)\ \rho(x|\ \omega_j)$ é a evidência, e $P(\omega_i|x)$ é a *posteriori*. Dentre as variações do classificador de Bayes podemos aplicar classificação com mínima taxa de erro e mínimo risco, classificação de Bayes para distribuição gaussiana, classificação Naïve Bayes e outras [1].

B. Aprendizagem baseada em métodos determinísticos

Alguns problemas de classificação envolvem atributos nominais ou categóricos, sem qualquer noção natural de semelhança ou mesmo de ordem. Nestes casos, a distribuição de probabilidade não é adequada. Para classificar objetos utilizando atributos nominais, métodos apropriados devem ser adotados. Dentre os principais métodos podemos citar a indução árvores de decisão, o reconhecimento com String, os métodos gramaticais, a inferência gramatical e a aprendizagem baseada em regras.

A seguir é destacado o método de indução de árvores de decisão e na sequência a aprendizagem baseada em regras.

Segundo Duda et al. [1], é natural e intuitivo classificar um padrão através de uma sequência de questões, onde a próxima questão depende da resposta da questão corrente. Uma árvore de decisão é uma sequência de questões. Por convenção, o primeiro nó da árvore fica no topo e é chamado nó *raiz*. Em seguida, ramificações sucessivas são conectadas a outros nós, essas conexões são chamadas *links* ou *ramos*. O número de links em cada nó é chamado *fator* de ramificação. Em árvores binárias o fator de ramificação de cada nó é 2 (dois), em árvores n-árias o fator de ramificação é *n*. Essas conexões ocorrem até alcançar um nó *terminal*, ou *folha*. Um nó folha é atingido quando termina a ramificação. Para classificar um item, dada uma árvore de decisão percorre-se a árvore do nó raiz até um nó folha seguindo o link apropriado para o nó subsequente ou descendente. Cada link deve ser mutuamente distinto e exaustivo, ou seja, um e somente um link deve ser seguido [1]. Cada folha contém uma categoria rotulada, chamada de *classe*. Cada nó intermediário especifica um teste de um atributo ou instancia pertencente à classe. Cada link contém um dos valores possíveis de cada atributo. Dentre os

principais algoritmos para indução de árvores estão: ID3 [8], C4.5 [7], ID5R [14].

No algoritmo ID3, conforme figura 2, *Examples* são os exemplos de treinamento. *Target_attribute* é o atributo cujo valor será predito pela árvore. *Attributes* é uma lista de atributos que podem ser testados pela árvore de decisão. O algoritmo retorna uma árvore de decisão que classifica corretamente os exemplos dados. Na sequência do algoritmo, o primeiro passo é criar um nó *raiz* para a árvore. Se os exemplos de treinamento são corretos, retorna uma árvore de nó único (raiz), com rótulo "+". Se os exemplos são incorretos, retorna uma árvore de nó único (raiz), com rótulo "-". Se a lista de atributos testada pela árvore estiver vazia, retorna uma árvore de nó único com o rótulo contendo o valor mais comum do atributo predito pela árvore. Caso contrário, escolhe o atributo da lista de atributos que melhor classifica os exemplos de treinamento, e o nó raiz da sub-árvore recebe o atributo escolhido. Para cada valor possível do atributo é adicionado um novo ramo da árvore abaixo da raiz, correspondendo ao teste "atributo = valor do atributo". Na seqüência, são criados subconjuntos de exemplos com o valor do atributo. Se o subconjunto for vazio, adicionar uma folha embaixo deste novo ramo com rótulo = "valor mais comum do atributo, cujo valor foi predito pela árvore nos exemplos". Se o subconjunto for diferente de vazio adicionar a sub-árvore e repetir o algoritmo considerando a lista de atributos, retirando o subconjunto testado. No final, retornar raiz.

```
01. ID3(Examples, Target_attribute, Attributes)
02. Create a Root node for the tree
03. if all Examples are positive, Return the single-node tree Root, with label = +
04. if all Examples are negative, Return the single-node tree Root, with label = -
05. if Attributes is empty, Return the single-node tree Root,
06.    with label = most common value of Target_attribute in Examples
07.
08. Otherwise Begin
09. A <- the attribute from Attributes that best* classifies Examples
10. The decision attribute to Root <- A
11. For each possible value, vi, of A,
12.      Add a new tree branch below Root, corresponding to the test A = vi
13.      Let Examples_vi be the subset of Examples that have value vi to A
14.      if Examples_vi is empty
15.         then below this new branch add a leaf node with label = most common
16.          value of Target attribute in Examples
17.          else below this new branch add the subtree
18.          ID3(Examples_vi, Target_attribute, Attributes - {A})
19. end
20. return Root
21.
22. "The best attribute is the one with highest information gain"
```

Figura 2 – Algoritmo *ID3* (Figura adaptada de Michell, 1997)

Métodos baseados em regras são partes integrantes de sistemas em inteligência artificial, mais sua adoção em reconhecimento de padrões ainda é modesta [1]. Uma visão geral desse método para representação e aprendizagem, concentra-se em classes de regras do tipo **"se...então"**. Mitchell [5] considera o conjunto de regras do tipo se-então uma das formas mais expressiva e legível para representar hipóteses em aprendizagem de máquina. Uma regra pode ser exemplificada em [1]: "se *Swims(x)* e *HasScales(x)* então *Fish(x)*", (lê-se: se um objeto *x* tem a propriedade de que nada e a propriedade de ter escamas então esse objeto é um peixe). Um *predicado*, como *Swims(x)*, *HasScales(x)* é um teste que retorna um valor lógico, que pode ser verdadeiro ou falso. Os predicados podem ser aplicados a problemas onde os dados são numéricos, não-numérico, sequências linguísticas, entre outros. A tarefa de escolha de predicados apropriados e a avaliação desses predicados dependem fortemente do problema, em geral, é mais difícil do que aprender a regra. Mitchell [5] considera dois tipos principais de regras se-então, as regras proposicionais e as regras de primeira-ordem. A regra proposicional descreve um caso particular, a desvantagem da lógica proposicional é não fornecer uma maneira para representar as relações gerais entre um grande número de casos. As regras de primeira ordem usam variáveis, o que permite expressar propriedades gerais, pertencentes a todos os indivíduos e propriedades existenciais, pertencentes a alguns indivíduos. Para Russell et al. [9] a lógica de primeira ordem é considerada satisfatória para representar o conhecimento comum. Além disso, de acordo com Mitchell [5], ao contrário da árvore de decisão, que aprende apenas regras proposicionais, as regras de primeira ordem são mais expressivas e permitem representar variáveis e relacionamentos entre elas. Neste caso, árvores de decisão ou técnicas estatísticas, não são capazes de aprender a regra perfeitamente, mesmo tendo em conta um número extremamente grande de exemplos [1].

Na classificação de padrão, dado um conjunto de regras escritas em lógica de primeira ordem, avaliar as proposições e regras para classificar um item desconhecido. Regras proposicionais podem ser aprendidas por algoritmos conhecidos como algoritmos de Cobertura Sequencial, já as regras de primeira ordem utilizam o algoritmo FOIL [5] para aprendizagem.

No algoritmo de cobertura sequencial, a entrada é um conjunto de exemplos de treinamento positivos e negativos e a saída é uma regra que cobre vários dos exemplos positivos e nenhum, ou poucos, negativos. A regra deve ter uma boa taxa de acerto, porém, não necessariamente precisa englobar muitos exemplos para evitar *overffiting*.

A figura 3 mostra o pseudocódigo do algoritmo de cobertura.

Os métodos para aprender um conjunto de regras, [5] dado à função *learn-one-rule* é: construir uma árvore de decisão e fazer a conversão para um conjunto de regras ou utilizar o algoritmo *Sequential Covering*. Neste último: invocar Learn-one-rule sobre todos os exemplos; remover os exemplos positivos cobertos pela regra aprendida; repetir o processo até atingir a fração desejada de exemplos positivos cobertos pelas regras.

```
   view plain  copy to clipboard  print  ?
01. Sequential Covering (Classe, Atributos, Exemplos, Limiar)
02. regras_aprendidas <- {};
03. regra <-  LEARN-ONE-RULE(classe, atributos, exemplos)
04.           while PERFORMENCE(regra, exemplos) > limiar do
05.                   regras_aprendidas <- regras_aprendidas + regra;
06.                   exemplos <-  exemplos - {exemplos cobertos pela Regra}
07.                   regra <- LEARN-ONE-RULE(classe, atributos, exemplos);
08. regras_aprendidas <- ordene regras_aprendidas por PERFORMENCE;
09. return regras_aprendidas;
```

Figura 3 – Algoritmo *Sequential Covering* para aprendizagem de regras (Figura adaptada de Michell, 1997)

Uma das diferenças entre aprendizagem de árvores e aprendizagem de regras é que o algoritmo *Sequential Covering* aprende as regras uma de cada vez enquanto as árvores de decisão aprendem o conjunto de regras de uma só vez [5].

C. *Aprendizagem baseada em métodos adaptativos*

A adaptatividade em sistemas de software permite um comportamento dinâmico através da automodificação da funcionalidade desses sistemas. O estudo da tecnologia adaptativa tem como objetivo propor modelos baseados em dispositivos adaptativos para solução de problemas práticos. A tecnologia adaptativa é o conjunto de aplicações práticas baseadas no conceito de adaptatividade, com a finalidade de solucionar problemas em diversas áreas.

Um dispositivo é adaptativo quando seu comportamento pode mudar dinamicamente em resposta a estímulos de entrada, sem a interferência de agentes externos. Dentre os principais dispositivos adaptativos, consideramos apropriados para solução de problemas de classificação as árvores de decisão adaptativas [6], as tabelas de decisão adaptativas [11] e as gramáticas adaptativas [2].

A indução de árvores de decisão proposta em [6] baseada na teoria dos dispositivos adaptativos pode ser aplicada em problemas de aprendizagem computacional. O resultado é uma árvore de decisão adaptativa com capacidade de automodificação através da inserção e/ou remoção de sub-árvores. As árvores de decisão adaptativas são aplicadas na solução de problemas relacionados a processamento digital de imagens e aprendizagem de máquina [6].

A tabela de decisão adaptativa, cujo formalismo pode ser encontrado em [11], pode apoiar o processo de tomada de decisão e ser aplicada na busca de padrões [12]. Uma tabela de decisão adaptativa é um dispositivo guiado por regras que permite mudanças no conjunto de regras da tabela de forma dinâmica, através de ações adaptativas. As tabelas são formadas por um conjunto de condições, ações, regras e funções adaptativas. Na execução de uma condição em uma tabela de decisão adaptativa, são verificadas as regras não adaptativas e caso uma única delas se aplique, as ações correspondentes são executadas. Caso mais de uma regra não adaptativa satisfaça a condição, as ações correspondentes às condições devem ser aplicadas em paralelo. Porém, se nenhuma regra satisfazer a condição, essa é uma condição excepcional e no caso de uma tabela de decisão convencional não seria possível prosseguir. Porém, na solução adaptativa é verificado se uma regra adaptativa se aplica. Se existir, a ação é executada e o conjunto de regras não adaptativas é alterado. Neste último caso, o comportamento do sistema é modificado e uma vez aplicada à regra, utiliza-se novamente a tabela de decisão adaptativa em sua nova configuração.

As gramáticas adaptativas [2], são capazes de representar linguagens sensíveis ao contexto. O que diferencia estas gramáticas das convencionais é a sua capacidade de automodificação, característica dos dispositivos adaptativos. As modificações acontecem durante a geração da sentença da linguagem, as ações adaptativas são associadas às regras de produção que possibilitam alterar tanto o conjunto de símbolos não-terminais, como as regras de produção da gramática.

III. PROPOSTA DE UM MÉTODO HÍBRIDO PARA CONSTRUÇÃO DE CLASSIFICADORES

Na literatura a abordagem estatística para reconhecimento de padrão é predominante, porém, cada abordagem tem seu domínio de aplicação onde uma é mais apropriada que outra. Existem muitas aplicações potenciais para as diferentes abordagens de reconhecimento de padrão [1].

Métodos estatísticos pressupõe uma distribuição de probabilidade associada ao espaço de características. A noção de erro de classificação é formulada em termos estatísticos e probabilísticos. Os métodos estatísticos utilizam variáveis aleatórias sobre um espaço amostral. Quanto menor a variância das variáveis aleatórias, maior a precisão. Assim, se considerarmos problemas de alta de precisão a variância deve ser nula.

Métodos estatísticos são apropriados em casos que o indivíduo é irrelevante, pois esses métodos massificam as amostras e eliminam a importância dos indivíduos. Em problemas em que as características de cada indivíduo são fundamentais, é mais apropriado usar métodos determinísticos auxiliares ou utilizar métodos híbridos.

A segunda opção é promissora e uma proposta para construção de classificadores utilizando métodos híbridos é considerada.

Um método híbrido é apropriado na busca de uma solução considerando um problema de natureza estruturada e dinâmica. Também é adequado onde às relações entre os indivíduos, bem como seu comportamento individual e coletivo, são de grande importância. Suponha uma amostra grande de indivíduos. Os métodos estatísticos têm a função de localizar a região provável da solução procurada, convergindo rapidamente para o espaço de possíveis soluções. Na sequência, métodos determinísticos têm a função de refinar a solução do processo, considerando os aspectos estruturais relacionados aos indivíduos. Assim, métodos estáticos e dinâmicos, dependendo intimamente da natureza do problema, são aplicados para tomada de decisão do processo. Esses métodos incluem métodos não adaptativos (estáticos) e métodos adaptativos (dinâmicos). Para exemplificar, em problemas de processamento da língua natural, um método híbrido pode ser uma alternativa de busca de solução no processo de reconhecimento. Aplicando um método estatístico, a busca por uma possível solução em amostras grandes é reduzida, o seja, o espaço amostral é reduzido, porém a decisão final não é baseada na probabilidade. Em seguida, mecanismos de inferência gramatical e gramática adaptativa são utilizados para tratamento de características estruturais e dinâmicas do problema. A decisão de classificação é baseada nesses métodos. Em geral, métodos híbridos tendem a produzir melhores resultados, quando os indivíduos são relevantes.

É importante destacar que se trata de uma proposta inicial, sem resultados práticos. Porém, a possibilidade de combinar técnicas é muito utilizada e pode agregar as vantagens e capacidades de diferentes métodos tradicionalmente conhecidos.

IV. CONCLUSÃO

O estudo comparativo entre métodos estatísticos e métodos determinísticos permitiram identificar as características

relevantes em cada uma das abordagens. Métodos estatísticos estão preocupados em buscar regularidades em um espaço amostral sobre uma massa de dados e classificar novos indivíduos com uma taxa de erro aceitável. Esse tipo de abordagem desconsidera a precisão. Métodos determinísticos buscam regularidades em amostras mapeando a estrutura relativa aos indivíduos. Considera cada indivíduo relevante e particular, proporcionando alta precisão na classificação de novos indivíduos. Além disso, métodos estáticos são apropriados quando o fenômeno que descreve o comportamento dos indivíduos é fixo. Em casos onde o fenômeno que descreve o comportamento dos indivíduos é variável, métodos dinâmicos são mais apropriados. Neste trabalho, os métodos dinâmicos correspondem aos métodos adaptativos.

A proposta de uso de adaptatividade em métodos para reconhecimento de padrão considera as seguintes características do problema: problemas com amostras grandes de dados; comportamento coletivo entre os indivíduos, comportamento isolado de cada indivíduo e as relações entre os indivíduos; problemas com comportamento dinâmico.

Em problemas de classificação em que é necessária a obtenção de resultados de alta precisão, devemos considerar: 1) uso de métodos estatísticos para localizar a região provável da solução procurada. 2) uso de métodos determinísticos para refinamento da resolução do processo. 3) uso de métodos estáticos para tratamento de fenômenos de comportamento fixo. 4) uso de métodos dinâmicos (adaptativos) para tratamento de comportamentos variantes no tempo.

Os resultados mostram que a combinação de diferentes métodos e técnicas na construção de classificadores permite: 1) obtenção rápida de resultados utilizando métodos estatísticos com uma estimativa aceitável para a solução, ainda que com baixa porcentagem de acerto. 2) eliminação de imprecisões e flutuações com a introdução de técnicas determinísticas mais onerosas. 3) obtenção eventualmente lenta de refinamentos proporcionados por métodos não estatísticos, resultando em alta porcentagem de acerto.

Em trabalhos futuros, o método híbrido deve ser detalhado, sendo aplicado os conceitos apresentados neste trabalho. Isso inclui delimitar a atuação de cada abordagem e método no processo de reconhecimento.

A combinação de métodos e técnicas em reconhecimento de padrão é comum, porém a capacidade de tratar aspectos dinâmicos de forma expressiva, na busca de uma solução mais precisa, é uma das principais contribuições da aplicação de métodos adaptativos na construção de classificadores.

VI. Referências

[1] DUDA, R. O.; HART, P. E. ; STORK, D. G.. *Pattern Classification*. 2ª Edição. Wiley, 2001. 680 p. ISBN: 0471056693.

[2] IWAI, M. K. Um formalismo gramatical adaptativo para linguagens dependentes de contexto. Tese (Doutorado), Escola Politécnica da USP. São Paulo, 2000.

[3] JAIN, A.K.; DUIN, R. P.W.; MAO, J.. Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22 (no.1):4–37, 2000.

[4] NETO, J. J. Adaptive Rule-Driven Devices – General Formulation and Case Study. Revista de Engenharia de Computação e Sistemas Digitais,São Paulo, v.1, n.1, p. 45-57, nov. 2003.

[5] MITCHELL, T. M.. Machine Learning. 1ª Edição. McGraw-Hill, 1997. ISBN: 0070428077.

[6] PISTORI, H. Tecnologia Adaptativa em Engenharia de Computação: estado da arte e aplicações. Tese (Doutorado), Escola Politécnica da USP, 2003.

[7] QUINLAN, J. R. C4.5: Programs for Machine Learning. Morgan-Kaufmann, San Francisco, 1993.

[8] QUINLAN, J. R. (1986). Induction of decision trees. Machine Learning, 1, 81-106.

[9] RUSSEL, S. J.; NORVIG, P.. Artificial Intelligence: A Modern Approach. 2º Edição. Prentice-Hall, 2002. ISBN - 10: 0137903952

[10] THEODORIDIS, S.; KOUTROUMBAS, K. Pattern Recognition. 3º Edição. Academic Press, 2006.

[11] TCHEMRA, A. H. Aplicação da Tecnologia Adaptativa em Sistemas de Tomada de Decisão. I WTA – Workshop sobre Tecnologia Adaptativa, Jan 2007 - www.pcs.usp.br/~lta.

[12] TCHEMRA, A. H., CAMARGO, R. Descoberta de padrões em bases de dados utilizando Técnicas Adaptativas. III WTA – Workshop sobre Tecnologia Adaptativa, Jan 2009 - www.pcs.usp.br/~lta.

[13] WINTEN, I. H. e EIBE, F.; Data Mining: Practical Machine Learning Tools and Techniques.

[14] UTGOFF, P. E., Incremental Induction of Decision Trees. Machine Learning, Machine Learning, 4, 161-186,1989.

Autores

Renata Luiza Stange é mestranda em Engenharia Elétrica (Área de concentração: Computação e Sistemas Digitais) pela Escola Politécnica da Universidade de São Paulo. Especialista em Administração de Sistemas de Informação pela Universidade Federal de Lavras e Bacharel em Análise de Sistemas pela Universidade Estadual do Centro-Oeste do Paraná.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo (EPUSP) e coordena o LTA – Laboratório de Linguagens e Tecnologia Adaptativa do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

Uso de Tabelas de Decisão Adaptativas em Redes de Sensores Sem Fio (15 dezembro 2009)

L. Gonda, C. E. Cugnasca, J.J. Neto

***Abstract*— Devido ao avanço de diversas tecnologias as Redes de Sensores Sem Fio têm se tornado uma importante tecnologia no monitoramento de diversas aplicações, tais como militar, agricultura, monitoramento de pacientes e outras. Devido à limitação de recursos, essas redes, ao contrário das redes tradicionais, trabalham de maneira colaborativa e os protocolos de comunicação devem se preocupar não só com o transporte de dados, mas também com o consumo de energia na transmissão. Estas redes possuem um comportamento dinâmico, além de serem capazes de se auto-organizarem, pois muitas vezes não é possível a troca de suas baterias ou estão localizadas em locais de difícil acesso. Este trabalho apresenta uma proposta de utilização de grafos dinâmicos e Tabelas de Decisão Adaptativas na modelagem e controle de topologia em RSSF.**

Palavras chave*— *Controle de Topologia, Grafos Dinâmicos, Redes de Sensores Sem Fio, Tabelas de Decisão Adaptativa.

I. Introdução

UMA Rede de Sensores Sem Fio (RSSF) é composta por diversos módulos distribuídos, denominados de nós sensores, que possuem capacidade de sensoriamento, processamento e comunicação [1]. Esta tecnologia foi viabilizada pela evolução de diversas áreas, em especial comunicação sem fio, microprocessadores e dispositivos eletromecânicos, possibilitando que pequenos dispositivos, dispostos em uma área a ser monitorada, pudessem reproduzir, p de sensoriamento, processamento e comunicação com outros nós, fenômenos físicos com mais precisão [2]. Diversas aplicações para RSSF estão surgindo, incluindo medicina, agricultura, controle de ambientes, militar, entre outras [3].

Em geral, os recursos dos nós sensores apresentam limitação em relação à capacidade de processamento, armazenamento, de energia e de alcance do seu rádio de comunicação. Conseqüentemente, os nós sensores devem ser agrupados para executarem tarefas de maneira colaborativa, pois individualmente não são muito eficazes.

Uma das características essenciais em uma RSSF é a capacidade de auto-organização.

Dessa forma, ao serem espalhados em grandes quantidades em uma área de monitoramento, os nós sensores devem se organizar automaticamente, para que seja estabelecida a comunicação entre os diversos dispositivos da rede [3]. É importante ressaltar que a auto-organização é necessária, não somente no instante em que a rede está sendo criada, mas também na manutenção da rede, devido a possíveis mudanças em decorrência de falhas e inserção de novos nós sensores.

Com a finalidade de manter as funções da rede e economizar recursos, especialmente energia, a utilização de mecanismos para manutenção e prevenção a falhas são extremamente importantes em protocolos de comunicação para RSSF. Neste sentido, o objetivo deste trabalho é apresentar uma proposta inicial de utilização de Tabelas de Decisão Adaptativas (TDA) no desenvolvimento de mecanismos de comunicação para RSSF.

O restante deste trabalho está organizando conforme descrito a seguir. A Seção II mostra alguns trabalhos relacionados à comunicação e manutenção em RSSF. Na Seção III são apresentadas as definições de grafos dinâmicos e Tabelas de Decisão Adaptativas, bem como características de RSSF que demonstram a necessidade de um dinamismo em suas organizações e topologias. Por fim, na Seção IV são apresentadas as considerações finais do trabalho.

II. Trabalhos Relacionados

Na transmissão de dados em uma RSSF, ao contrário do que acontece nas redes tradicionais, como a Internet, por exemplo, além de se preocupar com o envio de informações, os protocolos de comunicação devem levar em consideração também o consumo de energia, já que na maioria das vezes os nós sensores são alimentados por baterias, que possuem carga limitada [2].

Devido à limitação de recursos nos nós sensores, o estudo de mecanismos e protocolos para gerenciamento em RSSF, tais como, manutenção de topologia em RSSF, cobertura da área a ser monitorada e redução de falhas são importantes para prolongar o funcionamento da rede, sem que haja nenhuma intervenção externa, principalmente, em casos em que os nós sensores estão dispostos em locais de difícil acesso.

Neste sentido, diversos protocolos e mecanismos para controle de topologia foram propostos na literatura. Por

C.E. Cugnasca e J. J. Neto são docentes da Escola Politécnica da Universidade de São Paulo (carlos.cugnasca@poli.usp.br, joao.jose@poli.usp.br)
L. Gonda é Doutorando na Escola Politécnica da Universidade de São Paulo e docente da Universidade Católica Dom Bosco(gonda.ucdb@gmail.com).

exemplo, Sun *et. al.*[4] apresentam um mecanismo denominado de REMUDA, que tem por objetivo a construção e manutenção de topologias em RSSF. Neste mecanismo cada nó da rede possui um nó denominado de pai, para quem o mesmo deve enviar as informações coletadas. Para garantir o funcionamento do mecanismo, os nós devem enviar periodicamente informações de controle para os nós pais. Caso algum envio detecte alguma falha, o mesmo deve iniciar o processo de escolha de um novo nó pai.

Outro mecanismo de disseminação de dados é apresentado por Figueiredo, Nakamura e Loureiro [5]. Este mecanismo é denominado de Multi e corresponde à utilização alternada de dois outros protocolos: SID (*Source-initiated Dissemination*) e EF-Tree (*Earliest First Tree*).

Em [5], Gonda *et al.* apresentam uma proposta de uso de Tecnologia Adaptativa (TA) em RSSF utilizando grafos dinâmicos. Entretanto, além da utilização de grafos dinâmicos são necessários mecanismos mais eficientes para melhorar o processo de Controle de Topologia. Neste trabalho, os autores descrevem formalmente um grafo dinâmico como um dispositivo adaptativo e um algoritmo dividido em duas etapas: construção da topologia e manutenção da topologia.

III. Dispositivos Adaptativos Utilizados

Um dispositivo formal é dito adaptativo sempre que o seu comportamento muda dinamicamente, em resposta a estímulos de entrada, sem a interferência de agentes externos, incluindo seus usuários.

Em sua formulação geral, os dispositivos adaptativos são caracterizados por meio de suas camadas conceituais: o formalismo subjacente (ou dispositivo subjacente) e o mecanismo adaptativo (ou camada adaptativa) [6].

O formalismo subjacente é representado por um dispositivo guiado por regras (formalismo convencional, não-adaptativo) e opera, movendo-se de uma configuração para outra, sucessivamente, em resposta a estímulos recebidos em sua entrada.

Tais movimentos são definidos por um conjunto de regras que mapeiam cada possível configuração em uma configuração seguinte correspondente. O mecanismo adaptativo responde pelas mudanças incrementais no comportamento do dispositivo subjacente. As propriedades de auto-modificação são representadas pelas funções e ações adaptativas, capazes de alterar o conjunto de regras do dispositivo subjacente, determinando assim, conseqüentemente, seu comportamento independente de decisões externas a partir de então. Na literatura existem diversos dispositivos adaptativos, porém, somente os grafos dinâmicos e as Tabelas de Decisão Adaptativas serão descritas neste trabalho.

A. *Grafos Dinâmicos*

Uma RSSF pode ser representada como um grafo ou como uma árvore. Neste trabalho, optou-se por utilizar um grafo direcionado para representar a estrutura da RSSF.

Entretanto, na maioria das aplicações de algoritmos em grafos, os grafos são dispositivos estáveis que sofrem pouca ou nenhuma mudança. Em um grafo que representa uma RSSF, este tipo de dispositivo estável não pode ser utilizado devido à necessidade de mudanças que podem ocorrer durante o tempo de vida da rede. Neste sentido, são necessárias alterações na estrutura de um grafo, como inserção e remoção de arestas, mudança no estado dos vértices (que pode ser representado por mudanças na coloração dos mesmos).

Baseado na definição de Neto [6], pode-se definir um dispositivo adaptativo que usa um grafo como formalismo subjacente (não-adaptativo). Embora o dispositivo formal apresentado em [6] seja uma 5-upla, um grafo dinâmico para modelar o comportamento de uma RSSF não precisa de todos os conjuntos e símbolos. O conjunto A (que define aceitação) não é utilizado, pois os estímulos e eventos que podem ocorrer na RSSF são infinitos e não há final de entrada, ou seja, a cadeia de estímulos w é infinita. Dessa forma, um dispositivo adaptativo que utiliza um grafo como formalismo subjacente pode ser definido como uma 4-upla ND = (C; NR; S; c_0) na qual:

- ND: corresponde ao grafo que modela a RSSF em questão;
- C: conjunto de possíveis configurações do grafo que modela a RSSF. c_0 é sua configuração inicial, ou seja, a configuração que é montada inicialmente com base nas regras de conexão da RSSF;
- NR: conjunto de regras que definem o grafo ND pela relação C X S X C. Neste caso, as regras $r \in NR$ têm a forma $r = (c_i; s; c_j)$, indicando que, em resposta a um estímulo de entrada $s \in S$, a regra r altera a configuração corrente c_i do grafo para configuração c_j. Neste caso, a mudança de configuração representa uma alteração na topologia da RSSF;
- S: conjunto (finito) de todos os possíveis eventos válidos e considerados como estímulos de entrada para ND. Neste caso, ε também é desconsiderado, pois o estímulo "vazio" não provoca alterações na RSSF.

É importante observar que o grafo dinâmico é utilizado para representar graficamente a RSSF, porém, as mudanças na rede são provocadas por estímulos que são analisados e configurados através de uma Tabela de Decisão Adaptativa, descrita na próxima seção.

B. *Tabela de Decisão Adaptativa*

As Tabelas de Decisão Adaptativas (TDA) podem ser consideradas uma versão adaptativa das Tabelas de Descisão Convencionais, que são utilizadas nas mais diversas áreas com o objetivo de auxiliar na descrição de problemas complexos, além de se mostrarem mais eficientes do que outras formas de expressão, tais como narrativas, escritas, diagramas e outras[7].

Uma Tabela de Decisão Convencional ou Não-Adaptativa (TDN) é composta por condições, ações e regras e pode ser dividida em quatro partes [7]:

- Linha das codições: corresponde a uma variável

do problema que será avaliada no processo de decisão e cujo valor está presente na lista de valores das condições;

- Possíveis valores das condições: neste item são apresentados todos os possíveis valores que as condições podem assumir;
- Linha das ações: conjunto de procedimentos ou operações a serem executadas de acordo com as condições;
- Ações a serem tomadas: as ações a serem tomadas após a avaliação das regras.

É importante observar que as condições assumem valores binários do tipo falso (F) ou verdadeiro (V) para uma determinada regra R_k. Quando isto ocorre indica que a condição é verificada para esta regra e caso alguma ação deva ser tomada, é necessário fazer a sinalização. A Tabela I mostra um exemplo de uma Tabela de Decisão na qual o *firewall* de uma rede deve tomar decisões baseadas nas informações de pacotes.

TABELA I
EXEMPLO DE UMA TABELA DE DECISÃO

Endereço IP de Origem Confiável	V	F	V	V	F
Endereço IP de Destino Confiável	V	V	F	V	F
Porta de Destino Confiável	V	V	F	V	V
Liberar Pacote	X			X	
Encaminhar Pacote			X		

Na Tabela I, as regras são apresentadas na parte superior e as ações são representadas na parte inferior da tabela. Quando uma ação a ser tomada é representada por um X, significa que a mesma será executada caso as condições possuírem o valor correspondente de sua coluna. Quando a célula de ações é deixada em branco significa que nenhuma ação será tomada para aquela regra específica.

Após visualizar brevemente uma Tabela de Decisão Convencional, é possível definir uma TDA. A TDA proposta em [6] utiliza como dispositivo subjacente uma Tabela de Decisão Convencional e uma camada adaptativa composta por linhas que correspondem às funções adaptativas. Dessa forma, as colunas da TDA correspondem às regras da mesma. Sempre que uma função adaptativa é executada, o número de regras é alterado e, conseqüentemente, o número de colunas da TDA é alterado.

Portanto, uma TDA que representa uma RSSF pode ser definida como uma dupla TDA = (TDN, CA), na qual TDN é a Tabela de Decisão Não Adaptaiva e CA é a o mecanismo adaptativo[7]. A TDN que modela uma RSSF pode ser definida como 5-upla: TDN = (CT, NRT, CV, t_0, CRA), na qual:

- CT: é o conjunto de todas as possíveis configurações da Tabela de Decisão, que corresponde ao estado da RSSF;
- NRT $\subseteq$ CT X CV X CT X CRA é o conjunto de regras de decisão da tabela para verificar possíveis mudanças na RSSF;
- CV é o conjunto finito de valores válidos das condições para a RSSF;
- t_0 é a configuração inicial da tabela, que correspondem a todas as ações, regras e condições existentes para a RSSF;
- CRA é o conjunto finito dos possíveis resultados obtidos pela execução das ações quando da aplicação das regras NRT;
- cada regra rt $\in$ NRT é da forma rt = (t_i, cv, t_j, r) , na qual com a entrada de valores das condições cv $\in$ CV, a regra rt gera a resposta r $\in$ CRA e muda a configuração da tabela corrente t_i para a tabela t_j.

É importante observar que esta definição é baseada na definição apresentada por [7], porém, o conjunto AT não é necessário para as RSSFs.

Para um melhor entendimento de uma TDA, a Tabela II apresenta a Estrutura Geral de uma TDA, baseada na proposta por Neto [6].

TABELA II
ESTRUTURA GERAL DE UMA TABELA DE DECISÃO ADAPTATIVA[7]

Linha de cabeçalho (tag)	Coluna das funções adaptativas	Coluna das regras
Linha das condições		Valores das condições
Linha das ações	Declaração das funções adaptativas	Ações a serem aplicadas
Linha das funções adaptativas		Ações adaptativas a serem executadas

Na Tabela II, as Linhas das Condições e das Ações correspondem à estrutura da Tabela de Decisão não Adaptativa, enquanto a linha das funções adaptativas corresponde à Camada Adaptativa do mecanismo.

C. Tabela de Decisão Adaptativa Estendida

Na seção anterior, foi apresnetada a estrutura geral de uma Tabela de Decisão Adaptativa. Entretanto, apesar de o objetivo do Controle de Topologia em RSSF ser único (aumentar o tempo de vida da rede), existem diversos critérios que podem ser levados em consideração. Dessa forma, é necessário realizar uma extensão da Tabela Adaptativa apresentada na Seção 2.B para que múltiplos critérios possam ser levados em consideração.

Neste sentido, esta seção apresenta incialmente alguns conceitos envolvendo decisões multicritério e, em seguida, apresenta a estrutura de uma Tabela de Decisão Adaptativa Estendida (TDAE).

Segundo Tchemura [7] APUD Fülöp [8], os métodos multicritério possibilitam uma maior compreensão dos problemas relacionados com decisão que apresentam um número finito de critérios e alternativas. Além disso, estes métodos apóiam os processos de análise de decisão, especialmente quando estes são inseridos em conceitos multidisciplinares.

Os métodos multicritério podem ser classificados como de escolha, classificação ou ordenação das alternativas existentes nos problemas. Os métodos de escolha permitem que apenas uma alternativa seja escolhida de um subconjunto de opções, enquanto os métodos de classificação permitem que mais de alternativa seja escolhida. A diferença entre eles é que os métodos de classificação em geral distribuem cada alternativa

em uma categoria, enquanto os métodos de ordenação estabelecem uma ordem de preferência entre as diversas alternativas[7].

No caso da aplicação em RSSF, como só existe uma topologia possível por vez, é necessário que o método multicritério a ser utilizado seja o de escolha. Dessa forma, sempre que houver alguma mudança na topologia da rede, dentre as possíveis alternativas, apenas uma delas será escolhida para tentar melhorar o desempenho da mesma.

Após verificar conceitos básicos de multicritérios, será apresentada agora uma estensão da TDA, denominada de Tabela de Decisão Adaptativa Estedida (TDAE). A TDAE tem por objetivo apoiar processos decisórios semi-estruturados, com aplicação de multicritérios, possibilitando ao decisor obter soluções viáveis para o problema, por meio da análise e interação com o sistema[7].

Para um mellhor entendimento, a Figura 1 apresenta a estrutura geral de um TDAE.

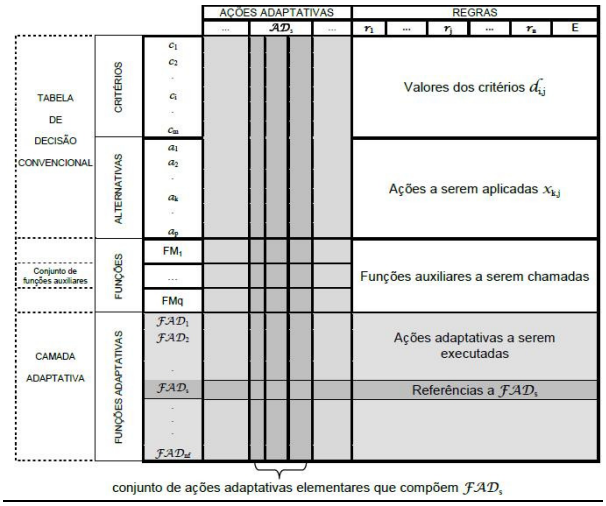


Figura 1. Estrutura Geral de uma Tabela de Decisão Estendida[7].

Formalmente, uma TDAE pode ser definida como uma tripla TDAE = (TDA, FM, M), na qual TDA é a Tabela Adaptativa apresentada na Seção 2.B, FM é o conjunto de Funções Auxiliares e M é o método multicritério adotado para um particular problema [7].

No caso das RSSF, a TDAE funcionará como um decisor, que deverá apresentar uma solução para otimizar o controle de topologia da mesma.

IV. Algoritmo Proposto

O algoritmo proposto é baseado na proposta apresentada em [5]. O algoritmo é dividido em duas fases: construção da topologia inicial e manutenção da topologia através de ações adaptativas. É importante observar que a escolha de uma topologia consiste na mudança de configuração tanto do grafo que representa a RSSF, como também, da TDAE.

Na fase de construção da topologia do algoritmo é responsável por determinar a topologia inicial da RSSF, de maneira que as informações coletadas pelos nós possam ser encaminhadas ao nó *gateway*. Inicialmente existe um conjunto de vértices que estão isolados. Como o objetivo é fazer com que as informações possam ser enviadas para o *gateway*, este nó é responsável por enviar uma mensagem do tipo *inicio* na RSSF. Esta mensagem é responsável por detectar os nós existentes, bem como, determinar quais nós participarão da RSSF inicialmente. Ao receber uma mensagem do tipo *inicio*, cada nó escolhe o nó remetente como pai (cria-se uma aresta entre os dois nós) e encaminha esta mensagem através de *broadcast*. É importante observar que um nó pode receber a mensagem de *inicio* de mais de um nó. Neste caso, o nó apenas cria uma aresta entre o nó emissor e ele mesmo, porém coloca-a como inativa (neste caso a mensagem *inicio* não é enviada para os demais nós).

Após esta fase, são montadas as configurações c_0 e t_0, do Grafo Dinâmico e da TDAE, respectivamente. Nesta etapa, são definidos também os critérios, regras e alternativas iniciais a serem adotados.

No caso de uma RSSF, possíveis critérios seriam, nível de energia, potência do nó, número de vizinhos, função do nó na rede roteador ou não), dentre outros. Estes seriam os critérios iniciais e de acordo com seus valores, poderia haver ou não mudanças na topologia da rede. Caso alguma regra não seja encontrada, a TDAE cria uma nova regra, por meio das Funções Adaptativas para que a topologia da rede seja mantida.

V. Considerações Finais

A tecnologia de RSSF tem se tornado cada vez mais presente e importante em aplicações de monitoramento nas mais diversas áreas. Entretanto, a limitação de recursos, em especial, a energia é um fator que vem se destacando, principalmente, no desenvolvimento de algoritmos de comunicação que permitam uma melhor utilização dos recursos de bateria, sem que haja perdas na comunicação.

Além da restrição de energia, um dos fatores que pode influenciar na comunicação em RSSF é o dinamismo da topologia, devido a mecanismos de prevenção e manutenção que são inerentes ao comportamento da rede.

Dessa forma, a modelagem do comportamento dinâmico de uma RSSF em dispositivo adaptativo é um desafio importante do pronto de vista da aplicação de TA.

Para a modelagem do comportamento da rede, podem ser utilizados os Grafos Dinâmicos, enquanto a TDAE permite que, por meio de um método decisório de escolha, uma nova topologia seja escolhida baseando-se em multicritérios. Além disso, a TDAE permite que caso algum comportamento não previsto pelo algoritmo seja encontrado, funções adaptativas possam ser executadas para garantir que novas regras sejam incorporadas e a topologia da rede seja reconfigurada da melhor maneira possível.

Portanto, a TA pode ser aplicada em RSSF, garantindo que o comportamento dinâmico da mesma seja modelado adequadamente. Uma das coisas a serem levadas em consideração ainda é o tempo de processamento e a memória a ser utilizada para que estes dispositivos funcionem

adequadamente .

REFERÊNCIAS

1. I. F. Akyildz et al. "Wireless Sensor Networks: a survey", *Computer Networks*, 2002, v. 38, n. 4, pp. 393-422.
2. D.Cueller, D. Estrin, M. Srivastava. "Overview of Sensor Networks". *IEEE Computer Society*, 2004, pp. 41-49.
3. P. Baronti et al. "Wireless Sensor Networks: a survey on the state of the art and the IEEE 802.15.4 and Zigbee Standards", *Computer Communications*, 2007, v.30, pp. 1655-1695.
4. L. Sun, T. Yan, Y. Bi. "REMUDA: A Practical Topology Control and Data Forwarding Mechanism for Wireless Sensor Networks", ACTA AUTOMATICA SINICA, 2006, v. 36, n. 6, pp. 867-874.
5. L. Gonda et al. "Aplicação de Tecnologia Adaptativa em Redes de Sensores Sem Fio", in *Proc. Workshop de Tecnologia Adaptativa, 2009.*
6. J. J. Neto, "Adaptative rule-driven devices - general formulation and a case study," in *CIAA'2001 Sixth International Conference on Implementation and Application of Automata*, Pretoria, South Africa, July 2001, pp. 234–250.
7. A. H. Tchemura, "Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério", Tese de Doutorado, orientada por João Jose Neto, Escola Politécnica da Universidade de São Paulo, São Paulo – SP, Junho, 2009.
8. J. Fülöp. *Introduction to Decision Making Methods. Laboratory of Operations Research and Decision Systems*, Computer and Automation Institute. Hungarian: Academy of Sciences, 2005.

[10]

[11]

Luciano Gonda é graduado em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (1999) e mestre em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (2004). Atualmente, é aluno de doutorado da Escola Politécnica da USP, sob a orientação do Prof. Carlos Eduardo Cugnasca. É membro do Laboratório de Automação Agrícola da Escola Politécnica da USP e do Grupo de Pesquisa em Engenharia e Computação da UCDB e professor da Universidade Católica Dom Bosco (UCDB). Tem experiência na área de Ciência da Computação, e seus interesses em pesquisa concentram-se em Redes de Sensores Sem Fio, Algoritmos Paralelos e Distribuídos e Redes de Computadores.

Carlos Eduardo Cugnasca é graduado em Engenharia de Eletricidade (1980), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Elétrica (1993). É livre-docente (2002) pela Escola Politécnica da Universidade de São Paulo (EPUSP). Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo, e pesquisador do LAA - Laboratório de Automação Agrícola do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Supervisão e Controle de Processos e Instrumentação, aplicadas a processos agrícolas e Agricultura de Precisão, atuando principalmente nos seguintes temas: intrumentação inteligente, sistemas embarcados em máquinas agrícolas, monitoração e controle de ambientes protegidos, redes de controle baseados nos padrões CAN, ISO11783 e LonWorks, redes de sensores sem fio e computação pervasiva. É editor da Revista Brasileira de Agroinformática (RBIAgro).

[12]

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975) e doutor em Engenharia Elétrica (1980). É livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo, e coordena o LTA - Laboratório de Linguagens e Tecnologia Adaptativa do PCS - Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseados em tecnologia adaptativa.

[13]

[14]

Framework para o desenvolvimento de aplicações baseadas em modelos de regras adaptativas

F. C. N. Campos, R. F. Feldberg, E. M. M. Jorge, B. M. Trigo

***Resumo*— O objetivo deste trabalho é o desenvolvimento de um framework para o desenvolvimento de aplicações utilizando sistemas de regras adaptativas, visando facilitar a utilização de adaptatividade em aplicações computacionais. Para isto, esta ferramenta se baseia na utilização de tabelas de decisão adaptativas, em conjunto com a simplicidade de descrição do modelo através de uma linguagem baseada em XML. Espera-se, com este trabalho, difundir a utilização de sistemas adaptativos tanto por usuários com pouca experiência no campo da adaptatividade e poucos conhecimentos de programação como por profissionais experientes desejando agilizar o processo de desenvolvimento.**

***Palavras Chave*— Sistemas baseados em regras, Adaptatividade, Tabelas de Decisão, Desenvolvimento, Framework.**

I. Introdução

MECANISMOS adaptativos possuem um largo campo de aplicações em sistemas computacionais, muitos deles ainda inexplorados sob o ponto de vista da adaptatividade. Um dos problemas no qual a plena difusão da tecnologia adaptativa esbarra é a baixa disponibilidade de ferramentas auxiliares, tanto para aprendizado quanto para o desenvolvimento.

Com este trabalho, pretende-se desenvolver um framework para desenvolvimento mais rápido de aplicações utilizando sistemas baseados em regras adaptativas. Para isto, apóia-se na premissa de que o aprendizado e utilização de adaptatividade em aplicações computacionais muitas vezes é desestimulado pela dificuldade do aprendizado de todo o formalismo necessário para o desenvolvimento do motor adaptativo da aplicação, dificultado ainda mais pela ausência de ferramentas que suavizem a curva de aprendizado.

Além de prover apoio a pesquisadores, estudantes e desenvolvedores interessados na tecnologia adaptativa, o framework proposto deve ao mesmo tempo permitir uma grande flexibilidade no que diz respeito aos campos de atuação onde poderia ser aplicado. Com isto, o framework teria não só apelo didático, como poderia ser utilizado para o desenvolvimento de aplicações em escala comercial.

F.C.N. Campos, R.F. Feldberg, E.M.M. Jorge e B.M. Trigo são estudantes de Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo, e podem ser contactados pelo e-mail felipe.pqi@gmail.com, rafael.feldberg@gmail.com, edummjorge@gmail.com e brunomtrigo@gmail.com, respectivamente.

II. Sistemas de Regras e Tabelas de Decisão Adaptativas

Sistemas baseados em regras são, de forma simples, um conjunto de fatos e regras SE-ENTÃO que coordenam todo o funcionamento de um dado sistema. Apesar do conceito simples, sistemas baseados em regras são de suma importância no campo dos sistemas especialistas, uma vez que podem representar de forma simples o raciocínio de um especialista. [4] Além disso, a utilização de bancos de regras e fatos permite uma separação do desenvolvimento da aplicação e das regras que regem seu controle, de forma que modificações no funcionamento do programa possam ser feitas simplesmente alterando o conjunto de regras.

A arquitetura de um sistema simples baseado em regras pode ser vista na figura 1.

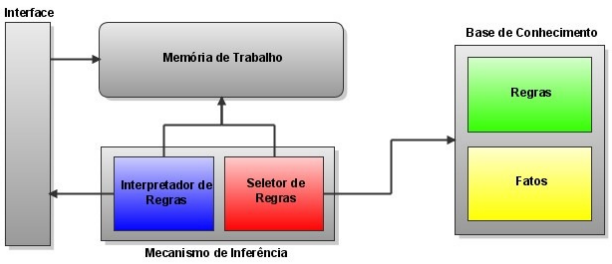


Fig. 1. Arquitetura de um sistema simples baseado em regras.

Através da arquitetura mostrada na figura 1, pode-se ver os principais componentes de um sistema baseado em regras. A memoria de trabalho armazena o estado atual da aplicação, tal qual, valores de variáveis, e quaisquer valores temporários do processamento de regras. A base de conhecimento é um banco onde são armazenadas as regras e fatos que regem o sistema. Este banco é acessado pelo seletor de regras, que seleciona a regra a ser aplicada e a passa ao interpretador de regras, que aplica a regra recebida. O interpretador de regras pode tanto modificar valores da memória de trabalho, como enviar comandos à aplicação através da interface. A aplicação, por sua vez, pode modificar a memória de trabalho como entrada do sistema baseado em regras.

O componente seletor de regras pode seguir uma variedade de estratégias de seleção de regras. No caso de apenas uma regra ser aplicável ao sistema, não há muito sobre o que

decidir. Já no caso de mais de uma regra poder ser aplicada ao estado atual do sistema, estas estratégias, denominadas estratégias de resolução de conflitos, são utilizadas. A estratégia mais simples conhecida é a estratégia da "Primeira Aplicável". Esta estratégia consiste simplesmente em aplicar a primeira regra encontrada. Outras regras comuns são a "Aleatória" (uma das regras aplicáveis é selecionada aleatoriamente), "Mais específica" (a regra que possui maior número de condições cumpridas, dentre as aplicáveis, é selecionada), "Melhor Aplicável" (através da utilização de pesos nas regras, de forma que a regra com maior peso é aplicada), entre diversas outras. Cada uma é melhor aplicada em determinadas situações. Em jogos, por exemplo, a estratégia aleatória é muito útil na inteligência artificial de adversários, devido à imprevisibilidade.

Além disso, existem dois tipos mais conhecidos de algoritmos para sistemas baseados em regras. O primeiro é o algoritmo de Encadeamento à Frente (*Forward-Chaining*) ou Dirigido a Estímulos (*Stimulus-Driven*). Este algoritmo parte do estado do sistema e, buscando as regras que se aplicam a este estado, tentam chegar a uma conclusão ou ação a ser tomada. Já o algoritmo de Encadeamento Reverso (*Backward-Chaining*) ou Dirigido a Objetivos (*Goal-Directed*) busca, a partir de um determinado objetivo, quais regras chegam a ele. Ao encontrar a regra, transforma todas as suas condições em novos objetivos, e assim prossegue até encontrar um conjunto de fatos e determinar a seqüência de regras para se atingir o objetivo inicialmente proposto.

Já a base de conhecimento pode ser definida através de uma diversidade de formalismos. Um deles são as tabelas de decisão, que são estruturas tabulares para a representação da lógica de tomada de decisão de uma aplicação. Tabelas de decisão são uma forma de mostrar toda a lógica de funcionamento de uma aplicação de forma simples, compacta e fácil leitura. Além disso, são meios eficientes de representação tanto para pessoas dentro da área de computação como fora, facilitando a comunicação entre os desenvolvedores e especialistas e não limitando a sua utilização à área da computação. Outra vantagem trazida pela utilização de tabelas de decisão é que estas podem ser traduzidas diretamente para lógica computacional, sem necessidade de tradução. [6]

Tabelas de decisão são representadas basicamente através de quatro quadrantes, conforme mostra a tabela 1.

TABELA I
ESTRUTURA BÁSICA DE UMA TABELA DE DECISÃO

Linhas de Condição	Entradas de Condição
Linhas de Ação	Entradas de Ação

Note que a tabela de decisão exemplificada possui as linhas de condição separadas das linhas de ação através de uma linha dupla: este padrão é adotado para facilitar a leitura da tabela.

Na tabela, cada coluna representa uma regra. Assim, as linhas de condição expressam qual a condição sendo testada, e as entradas de condição especificam os valores de cada condição para o qual aquela regra é aplicada. Já as linhas de ação especificam quais as ações a serem tomadas, e as entradas de ação especificam parâmetros ou se aquela ação deve ser executada quando a regra é aplicada. Um exemplo de tabela de decisão pode ser vista na tabela 2.

TABELA 2
EXEMPLO DE TABELA DE DECISÃO

	Regra 1	Regra 2	Regra 3
Crédito	SATISFATÓRIO	INSUFICIENTE	INSUFICIENTE
Histórico		BOM	RUIM
Compra	ACEITAR	REJEITAR	REJEITAR

Poderíamos adicionar às tabelas de decisão previamente mostradas um módulo adaptativo subjacente. Desta forma, poderíamos ter sistemas baseados em regras adaptativas, com regras mutáveis durante a execução do sistema sem interferência externa. Assim, a estrutura básica de uma tabela de decisão adaptativa poderia ser representada conforme a tabela 3.

TABELA 3
ESTRUTURA BÁSICA DE UMA TABELA DE DECISÃO ADAPTATIVA

Linhas de Condição	Entradas de Condição
Linhas de Ação	Entradas de Ação
Funções Adaptativas "before-"	Entradas de Função
Funções Adaptativas "after-"	Entradas de Função

Desta forma, as linhas de funções adaptativas "before-" definem as funções adaptativas que são executadas antes da aplicação da regra, e as linhas de funções adaptativas "after-" definem as funções adaptativas que são executadas após a aplicação da regra. As entradas de função definem quais funções adaptativas são chamadas e com quais valores de parâmetros.

Uma função adaptativa é definida através de um conjunto de ações adaptativas, que podem adicionar regras, remover regras, entre outros. Assim, um exemplo de tabela de decisão adaptativa pode ser vista na tabela 4.

TABELA 4
EXEMPLO DE TABELA DE DECISÃO ADAPTATIVA

	H	-	-	+	+	+	Regra 1	Regra 2
Estado =		p_1	p_1	p_1	g_1	p_1	1	1
Entrada =		*a*	*b*	*b*	*b*	*a*	*a*	*b*
Estado :=		p_1	p_2	g_1	p_2	p_1	1	2
Função A	A	✓				✓	✓	
p_1	P	p_1				p_1	1	
p_2	P	p_2				p_2	2	
g_1	G							

Note no exemplo dado na tabela 4 que antes das colunas de regra vêm as colunas que definem as funções adaptativas. A coluna H define o cabeçalho de uma função adaptativa: o valor identificado após o nome da função indica se é uma função "after-" (A) ou "before-" (B). Abaixo da função vêm seus parâmetros (P) e geradores (G). As colunas subseqüentes identificam as ações adaptativas que fazem parte daquela função, sendo um (-) para remoção de regra e um (+) para adição de regras. Os valores destas colunas nas entradas de condição e ação definem os valores da regra sendo adicionada ou removida, e os valores nas linhas de funções adaptativas indicam as chamadas e valores de parâmetros das regras sendo adicionadas ou removidas. Os valores das linhas de funções nas colunas de regras identificam se a função adaptativa é chamada, e dão os valores para cada parâmetro para esta chamada.

III. DESENVOLVIMENTO DO FRAMEWORK

Com base nos objetivos e nos conceitos explorados anteriormente, foi definida a arquitetura básica para o framework demonstrada na figura 2.

Fig. 2. Arquitetura do framework adaptativo.

O framework seria executado internamente à aplicação, sendo responsável por todo o controle de execução do modelo. Como pode ser visto pelo diagrama da arquitetura, todo o framework utiliza uma tabela de decisão como fundação para seu funcionamento. Tal escolha se deu, inicialmente, pelo fato que tabelas de decisão incorporam as principais idéias de um sistema básico baseado em regras. [7] Além disso, a estrutura simples e tabular das tabelas de decisão facilitaria a modelagem do sistema por usuários menos experientes. Outra grande vantagem na utilização de tabelas de decisão adaptativas é a relativa simplicidade para a modelagem de outros dispositivos adaptativos utilizando a mesma, quando comparado a outros formalismos. Uma tabela de decisão permite que novos dispositivos adaptativos sejam modelados através de manipulações simples e diretas de variáveis, ou seja, sem necessidade de conversões trabalhosas e dispendiosas do ponto de vista de recursos computacionais.

A tabela de decisão utilizada pelo framework foi modelada seguindo o conceito apresentado, porém algumas modificações foram feitas para que houvesse uma maior flexibilidade e facilidade de modelagem. Uma destas modificações é a presença da variável de "Estado" já embutida no sistema, como pode ser visto pela presença de um módulo dedicado aos estados da aplicação. Outra modificação diz respeito ao módulo de funções adaptativas. Na teoria formal, as funções adaptativas são definidas previamente como sendo do tipo "after-" ou "before-". Neste projeto, as funções não mais são definidas previamente, mas podem ser chamadas a qualquer momento da aplicação da regra, seja antes, depois ou até mesmo ambos. Com isso, espera-se ganhar em flexibilidade e facilidade de modelagem.

A Tabela de Regras é o módulo responsável por armazenar todas as regras atuais do modelo. Este módulo é alterado pelas ações adaptativas de cada função presente no módulo de Funções Adaptativas quando são chamadas. Este módulo foi desenvolvido visando um sistema baseado por regras com Encadeamento à Frente, ou seja, buscando uma ação de acordo com o estado atual do sistema. A busca de regras utiliza, como estratégia de resolução de conflitos, o método de "Primeira Aplicável". Tal decisão foi tomada visando facilitar a modelagem dos sistemas, por ser mais simples, e também por possuir melhor desempenho em comparação às demais. A forma modular como foi desenvolvido o framework permite que, mais tarde, outras estratégias de resolução de conflito sejam adicionadas.

Pode-se notar que existem dois módulos de variáveis: as variáveis da tabela e as variáveis do modelo. Isto foi feito pois uma das funcionalidades do framework é prover, além das tabelas de decisão adaptativas mencionadas, outros dispositivos adaptativos prontos para uso mais simples. Desta forma, caso um dispositivo adaptativo diferente de uma tabela de decisão esteja em uso, as variáveis do modelo são utilizadas pelo usuário para definir o estado do sistema e então são convertidas pelo framework nas variáveis da tabela, utilizadas para modelar o dispositivo adaptativo escolhido pelo usuário. Além disso, os métodos do modelo trazem métodos de nível mais alto para facilitar a interação do usuário com o framework. Por exemplo, a tabela de decisão adaptativa só permite que o usuário execute um passo da tabela por vez através do método *runStep()*, uma vez que não se pode saber como é o funcionamento do modelo previamente. Caso o usuário esteja utilizando um Autômato Adaptativo Finito, no caso um dispositivo já disponibilizado pelo framework, este dispõe de um método adicional que permite que seu modelo seja executado inteiramente para uma dada cadeia de entrada, verificando se a cadeia é ou não aceita

pelo sistema. Sendo assim, estes módulos visam facilitar a utilização do framework de forma mais direta e prática por aqueles que não possuem tanto conhecimento ou desejam desenvolver algo mais rapidamente.

Outro módulo bastante útil adicionado ao framework é a Interface. Este módulo dá ao usuário uma interface que pode ser implementada por sua aplicação, permitindo uma comunicação mais próxima entre a aplicação e o modelo. A interface permite que o modelo, durante a execução de uma regra, chame um método da aplicação onde reside, seja para avisar de um evento ou para requisitar algum tratamento adicional. Esta chamada de método é feita como uma ação adaptativa, e pode enviar para a aplicação dois parâmetros. Desta forma, o modelo possui um tipo de chamada que funciona quase como um tipo de interrupção à aplicação, passando parâmetros do modelo e permitindo um controle maior sobre o fluxo do mesmo.

IV. Linguagem de Descrição de Modelos

Um dos problemas encontrados durante testes e desenvolvimento de provas de conceito utilizando o framework foi a dificuldade em ler e entender o modelo através do código-fonte da aplicação, onde este era construído inteiramente através de métodos do framework. Apesar de possuir métodos de alto nível voltados para uma construção mais simples do modelo, a leitura do modelo através dos parâmetros dos métodos não era eficaz quando este modelo crescia em complexidade, e assim surgiu uma necessidade de criar uma segunda forma de criar o modelo na aplicação.

Para atender a esta necessidade, foi criada uma linguagem baseada em XML para descrever o modelo em utilização. O arquivo contendo esta descrição pode então ser importado pela aplicação durante a execução para carregar o modelo. A escolha de uma linguagem baseada em XML se deu, inicialmente, pela estrutura em árvore da mesma, o que permite uma melhor visualização da hierarquia do modelo, além de ser de fácil aprendizagem e de simples leitura.

A utilização de um arquivo de descrição do modelo trouxe outras vantagens, como o fato de que o modelo agora pode ter toda a sua lógica de execução modificada sem necessidade de modificar a aplicação em que reside. O arquivo de descrição, por ser em uma linguagem diferente daquela em que a aplicação é escrita, também permite que outras pessoas envolvidas com o projeto, mas não necessariamente programadores, leiam, entendam e modifiquem o modelo, facilitando a comunicação e interação.

Uma vez que o framework já era capaz de importar os modelos através da linguagem de descrição, decidiu-se também adicionar um módulo para exportação de modelos em execução para um arquivo de descrição. Desta forma, se torna mais fácil realizar uma varredura por problemas de modelagem, verificar estados intermediários do modelo e até mesmo realizar a persistência de modelos em execução. Esta possibilidade permite que o framework deste trabalho seja utilizado também em aplicações onde é desejável persistir o estado atual do modelo, o que é especialmente útil em aplicações que utilizam aprendizado e treinamento. Desta forma, aumentou-se a abrangência e campos de aplicação do framework.

V. Implementação e Testes Executados

O framework foi desenvolvido, ao fim, seguindo o diagrama de classes simplificado mostrado na figura 3.

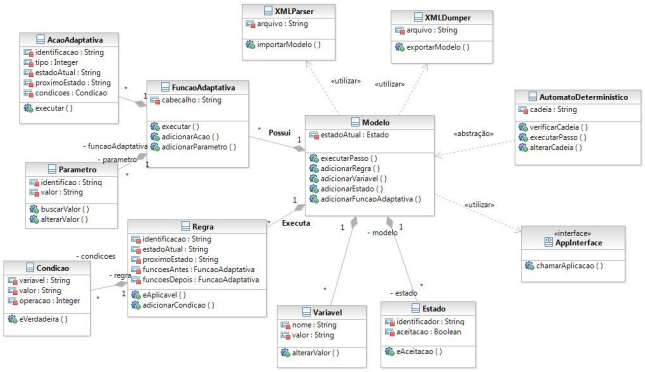


Fig. 3. Diagrama de classes simplificado do framework adaptativo.

A linguagem escolhida para desenvolvimento do framework foi Java. Esta decisão tomou por base não só o fato de ser uma linguagem popular, mas também por, devido a ser executada sobre uma máquina virtual, permitiria que o framework fosse utilizado sobre diversas plataformas diferentes. Assim, podemos citar, entre os diversos campos de aplicação, a possibilidade de utilização do framework para desenvolver aplicações adaptativas para celulares e outros dispositivos móveis, aplicações Web, conteúdo interativo adaptativo para discos Blu-ray (BD-J), além das aplicações tradicionais para computadores pessoais.

Os testes executados tomaram como base o autômato adaptativo que resolve a linguagem $L = \{a^{2n}b^{2n}e \mid n \geq 0\} \cup \{a^{2n+1}b^{2n+1}o \mid n \geq 0\}$, mostrado graficamente na figura 4. Este autômato foi selecionado pois utiliza, para sua execução, todas as funções do framework, tais quais gerador de novos estados, criação e remoção de regras, entre outros.

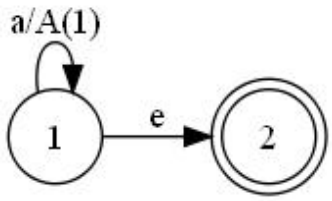


Fig. 4. Autômato utilizado para testes do framework adaptativo.

Tal autômato, ao receber o primeiro caracter "*a*" da cadeia de entrada, executa a função adaptativa A que remove a ligação entre o estado de aceitação 2 e o estado dado como parâmetro p_1 para o símbolo "*e*", remove a ligação entre o estado 1 e ele mesmo, cria um novo estado entre os estados 2 e o estado dado como parâmetro e cria ligações entre o estado do parâmetro p_1 e o novo estado g_1 para o símbolo "*b*", entre o estado g_1 e o estado de aceitação 2 para o símbolo "*o*" e entre o estado 1 e ele mesmo para o símbolo "*a*", chamando a função adaptativa B com o estado criado por g_1 como parâmetro. Dessa forma, o autômato da figura 4, ao receber o símbolo "*a*", se modificaria para o autômato visto na figura 5.

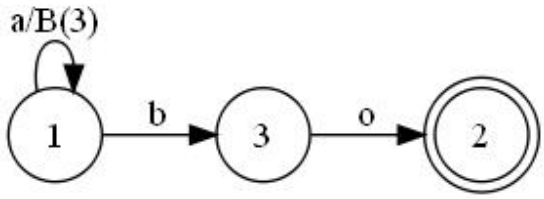


Fig. 5. Autômato utilizado para testes do framework adaptativo após consumir um símbolo "a".

A função adaptativa B, por sua vez, realiza as mesmas ações que a função adaptativa A, porém invertendo o símbolo "*e*" por "*o*" e vice-versa. Além disso, foi utilizada uma ação adaptativa adicional em cada função adaptativa para chamar um método da aplicação de testes que realizava a exportação do modelo para cada passo, para que fosse verificado seu funcionamento.

VI. Conclusões

O desenvolvimento deste framework trouxe uma importante contribuição no preenchimento da lacuna existente hoje na área de tecnologia adaptativa no que diz respeito a ferramentas de auxílio de desenvolvimento e aprendizagem. A definição de uma nova ferramenta que permita o desenvolvimento de aplicações adaptativas sem necessidade de conhecer previamente todo o formalismo de dispositivos regidos a regras adaptativas permitiria a difusão deste novo campo de conhecimento através do estímulo a pesquisadores, estudantes e desenvolvedores, tornando a curva de aprendizado mais suave para aqueles que ainda não estão familiarizados.

Além disso, a utilização de uma linguagem de descrição simples permite que seja utilizado até mesmo por programadores iniciantes ou inexperientes, e ao mesmo tempo provendo uma ferramenta poderosa para programadores experientes que podem desenvolver aplicações de forma simples e rápida utilizando um framework de fácil utilização e ao mesmo tempo bastante flexível.

Assim, esperamos que o projeto aqui desenvolvido incentive e colabore com o desenvolvimento de novas aplicações no campo da tecnologia adaptativa.

Referências

[1] F. C. N. Campos, E. M. M. Jorge, R. F. Feldberg e B. M. Trigo, "*Framework Adaptativo*", Website do projeto. Disponível em: http://www.edu.poli.usp.br/felipe/adapdk. Acesso em: 19 dez. 2009.

[2] M. V. M. Ramos, J. J. Neto, I. S. Vega, "*Linguagens formais: Teoria, Modelagem e Implementação*", Porto Alegre, Bookman, 2009.

[3] J. Freeman-Hargis, "*Rule-based Systems and Identification Trees*". Disponível: http://ai-depot.com/Tutorial/RuleBased.html. Acesso em: 14 Jun. 2009.

[4] F. Hayes-Roth, "Rule-based systems," *Communications of the ACM*, vol. 28, no. 9, pp. 921-932, Sept. 1985.

[5] H. W. Kirk, "Use of decisión tables in computer programming," *Communications of the ACM*, vol. 8, no. 1, pp. 41-43, Jan. 1965.

[6] U. W. Pooch, "Translation of Decision Tables," *Computing Surveys*, vol. 6, no. 2, pp. 125-151, June 1974.

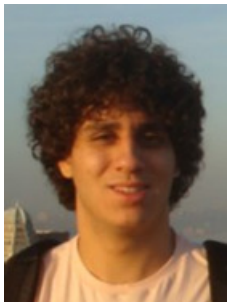
[7] A. Ligeza, "*Logical foundations for rule-based systems*," 2nd ed., Springer, 2006.

[8] J. J. Neto, "Adaptive rule-driven devices – general formulation and case study," *CIAA'01: Revised Papers from the 6th International Conference on Implementation and Application of Automata*, London, UK: Springer-Verlag, 2002, pp. 234-250, ISBN 3-540-00400-9.

[9] H. Pistori, "*Tecnologia adaptativa em engenharia de computação: estado da arte e aplicações*," Dissertação de doutorado, orientada por J. J. Neto, Escola Politécnica, Universidade de São Paulo, 2003.

Felipe C. N. Campos nasceu em Juiz de Fora/MG, Brasil, em 07 de outubro de 1986. É formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo.

Realizou iniciação científica na área de sistemas em Lógica Fuzzy pelo KNOMA (Laboratório de Engenharia do Conhecimento) e trabalha na IBM na área de consultoria em Business Intelligence. Já realizou projetos relacionados à utilização de recursos auxiliares e Robótica no ensino de fundamentos de computação para Engenharia na Escola Politécnica da Universidade de São Paulo, resultando no laboratório de Robótica utilizado hoje como atividade complementar para alunos dos primeiros anos, apresentados em conferência internacional de ensino em engenharia.



Rafael F. Feldberg nasceu em São Paulo/SP, Brasil, em 14 de Dezembro de 1986. É formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo e aluno de graduação da Faculdade de Direito da Universidade de São Paulo.

Trabalhou na Oracle do Brasil na área de Pré-vendas de tecnologia (produtos relacionados a banco de dados e middleware). Na empresa realizou projetos relacionados com segurança da informação, principalmente com gestão de identidade.

Eduardo M. M. Jorge é formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo. Realizou iniciação científica com o desenvolvimento de software para computador de mão voltado à pesquisa agrícola pelo LAA (Laboratório de Automação Agrícola) e trabalha na área de desenvolvimento do Credit Suisse.

Bruno M. Trigo é formado em Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo. Já realizou diversos projetos de pesquisa na área de didática para o ensino de Engenharia, com artigos publicados em congressos no exterior sobre a utilização de Applets no ensino de Química.

ARTIGOS

Uso de adaptatividade na modelagem de cursos para software educacional

W. J. Dizeró e J. J. Neto

***Resumo* — Este artigo apresenta uma proposta de aplicação da tecnologia adaptativa na modelagem e elaboração de cursos para softwares educacionais. A princípio, soluções adaptativas podem ser incorporadas a qualquer tipo de dispositivo guiado por regras. Neste trabalho, o dispositivo subjacente utilizado será Máquina de Moore, que é um autômato finito com saídas associadas aos seus estados. Assim, para cada estado pode-se associar o material didático a ser apresentado pela função de saída. Aplicando-se os conceitos de adaptatividade nesse transdutor, é possível elaborar cursos dinâmicos, que se auto-modifiquem com base em regras definidas pelo professor, e pelas experiências e nível de conhecimento individualizado dos alunos. Para complementar o trabalho, é apresentado um exemplo baseado em Máquina de Moore Adaptativa.**

***Palavras chaves* — tecnologia adaptativa; Máquina de Moore Adaptativa; ensino por computador.**

I. INTRODUÇÃO

Nos últimos anos, inúmeras ferramentas computacionais foram propostas e desenvolvidas para ajudar no ensino nas mais diferentes áreas do conhecimento. É fato que o uso de tecnologias computacionais tem sido importante no auxilio do aprendizado para inúmeras modalidades de ensino, seja quando aplicada à educação infantil ou de nível superior, em programas regulares ou de educação continuada, no ensino presencial ou a distância.

É verdade, também, que diversos grupos de pesquisa vêm explorando o uso do computador no ensino e atingindo um significativo grau de sucesso. Porém, as metodologias para o desenvolvimento de tais sistemas costumam empregar processos de desenvolvimento informais e muitas vezes inflexíveis. Processos informais podem levar a criação de sistemas com falhas de projetado, enquanto sistemas inflexíveis podem desmotivar seu uso. Ambas as situações podem ser cruciais no sucesso de um sistema de ensino auxiliado por computador.

Neste contexto, propõe-se um modelo baseado em máquina de Moore adaptativa, com o objetivo de definir formalmente cursos e, ainda, permitir que os cursos possam ser dinâmicos. Assim, enquanto o mecanismo formal do transdutor permite que se especifique o sistema através de uma notação matemática rigorosa, o potencial da tecnologia adaptativa [22, 23, 32] permite ao aluno uma experiência de aprendizagem individualizada, apresentando o material didático e seu roteiro de estudo adaptado ao nível de conhecimento e preferências.

Assim, a descrição deste artigo envolve, basicamente, a exposição dos conceitos acerca de máquina de Moore adaptativa, além da apresentação de um pequeno exemplo da especificação um curso baseado no modelo proposto. No capítulo 2 é feita uma breve explanação sobre o uso de computadores na educação. O capítulo 3 apresenta alguns conceitos sobre tecnologia adaptativa, a definição de dispositivo adaptativo dirigido por regras, e, apresenta o conceito de Máquina de Moore Adaptativa, cujo dispositivo subjacente guiado por regras é um transdutor com saídas associadas aos estados. No capítulo 4, encontra-se um esquema do funcionamento de um curso projetado com base no modelo proposto. Por fim, a seção 5 traz algumas considerações finais e propostas de trabalhos futuros para enriquecer este artigo. Ao final do texto, encontram-se as referências bibliográficas utilizadas.

II. COMPUTADORES NA EDUCAÇÃO

De forma geral, os computadores têm tido um papel de destaque no processo educativo. Ao contrário de outras mídias utilizadas na educação, o computador, tem a característica de processar e manipular a informação que recebe. Ele pode transformar, traduzir, usar em cálculos, ordenar, arrumar e mesmo fazer inferências a partir de uma informação fornecida.

Na educação, o computador tem sido utilizado tanto para ensinar sobre computação, como para ensinar praticamente qualquer assunto. No ensino de computação, o computador é usado como objeto de estudo. Ou seja, o aluno usa o computador para adquirir conceitos computacionais, como princípios de funcionamento do computador, noções de programação e implicações sociais do computador na sociedade. Já no ensino mediado por computador, ele pode ser usado como máquina de ensinar ou como ferramenta pedagógica.

Sendo a educação um processo de exploração, descoberta, observação e construção do conhecimento, é importante destacar que cada pessoa prefere aprender de maneira

W. J. Dizeró; (e-mail: wagner.dizero@poli.usp.br)
J. J. Neto; (e-mail: joao.jose@poli.usp.br)

diferente. Algumas preferem através de recursos visuais, outras através de métodos verbais, algumas preferem explorar, outras deduzir. Ou seja, qualquer ferramenta pedagógica deve considere as diferenças individuais de cada aprendiz.

Mas, ao considerar a aplicação do computador ou qualquer produto tecnológico na educação é preciso ter claro, e em destaque, que a aprendizagem - a aquisição de um conhecimento novo - só ocorre com o engajamento pessoal do aprendiz. Nenhuma máquina é capaz de colocar conhecimento em uma pessoa. Ela pode ser usada, para ampliar as condições do aprendiz de descobrir e desenvolver suas próprias potencialidades.

Não há dúvida de que atrás de qualquer tecnologia utilizada para sistemas de ensino existem pessoas, que preparam os materiais e os disponibilizam. Assim, as tecnologias não mudam necessariamente a relação pedagógica. As tecnologias tanto servem para reforçar uma visão conservadora, como uma visão progressista. Aqui, não serão discutidos assuntos relativos a questões pedagógicas, sendo apenas apresentados conceitos e aplicações de soluções relacionadas ao uso dos computadores e da informática na educação.

III. TECNOLOGIA ADAPTATIVA

O termo tecnologia é empregado para designar o uso do conhecimento científico na resolução de problemas práticos. Para a palavra adaptatividade e outras similares, adotou-se, como definição, a propriedade que um modelo tem de alterar seu próprio comportamento, de forma espontânea, sem auxílio externo. Logo, Tecnologia Adaptativa (TA) refere-se às técnicas, métodos e disciplinas que estudam as aplicações práticas da adaptatividade.

Dessa forma, TA compreende qualquer modelo de representação formal que tenha capacidade de mudar dinamicamente seu próprio comportamento, em resposta direta a um estímulo de entrada, sem qualquer intervenção externa. Num sistema definido por um conjunto de regras, a incorporação de algum mecanismo, através do qual possa se auto-modificar durante sua execução, o torna adaptativo.

Portanto, a TA pode ser identificada em qualquer modelo que possua a propriedade de alterar seu próprio comportamento, de maneira independente, sem a interferência de qualquer agente externo. Assim, se um sistema for definido por um conjunto de regras, pode-se conferir-lhe adaptatividade pela incorporação de algum mecanismo através do qual ele possa se auto-modificar durante sua operação, alterando o próprio conjunto de regras que define seu comportamento.

De acordo com Neto [23], a principal propriedade apresentada pela Tecnologia Adaptativa consiste no fato dela constituir um formalismo com regras dinamicamente variáveis, em contraste com a maioria dos modelos formais tradicionais, cujas regras, uma vez estabelecidas, permanecem imutáveis durante toda a sua operação. É importante destacar, também, seu poder de expressão computacional, que é equivamente ao das máquinas de Turing [33].

A adaptatividade pode ser incorporada a qualquer tipo de dispositivo guiado por regras. O termo dispositivo é empregado aqui como alguma abstração formal, na qual o comportamento do dispositivo é conduzido por um conjunto finito e explícito de regras. Tais regras especificam, para cada situação em que se encontre o dispositivo, sua nova situação. Quando se cria uma camada adaptativa num dispositivo subjacente qualquer, tem-se, então, um dispositivo adaptativo guiado por regras, ou simplesmente, um dispositivo adaptativo [22]. Um dispositivo adaptativo é, portanto, uma técnica adaptativa, ou seja, uma forma particular de aplicação da TA.

O formalismo geral que caracteriza os dispositivos adaptativos foi apresentado pela primeira vez à comunidade científica em [22], embora em formulações mais restritas venha sendo utilizado a mais tempo. Tal formulação fundamenta-se em um núcleo constituído por um dispositivo não-adaptativo dirigido por regras, acrescido de uma camada adaptativa que provê os recursos de mecanismos adaptativos ao mesmo. Uma característica fundamental dos dispositivos adaptativos é a possibilidade de reaproveitar integralmente os formalismos consolidados, com aumento de seu poder de representação, ao custo de um pequeno acréscimo na sua complexidade formal [32]. A teoria dos dispositivos adaptativos surgiu da busca por um formalismo simples de usar, mas capaz de representar problemas complexos, envolvendo linguagens não-regulares e até mesmo dependentes de contexto. Outras informações sobre linguagens e técnicas adaptativas, artigos relacionados e projetos desenvolvidos podem ser encontrados em [15].

Nas seções a seguir, são apresentados: uma formulação geral para dispositivos adaptativos; e, a formulação para Máquina de Moore Adaptativa.

A. Dispositivos Adaptativos Dirigidos por Regras

Historicamente, dispositivos adaptativos dirigidos por regras, ou simplesmente dispositivos adaptativos, emergem do campo de linguagens formais e autômatos. Os métodos formais permitem que se especifique, desenvolva e verifique um sistema baseado em computadores, através da aplicação de uma notação matemática rigorosa. A utilização de uma linguagem de especificação formal proporciona meios para especificar um sistema de tal modo que sejam obtidas a consistência, a completeza e a correção desejadas [1].

Um dispositivo não-adaptativo dirigido por regras, pode vir a ser qualquer máquina formal cujo comportamento dependa exclusivamente de um conjunto finito de regras, que determinem, para cada possível configuração corrente do dispositivo, a sua próxima configuração. Já, um dispositivo formal é dito adaptativo sempre que seu comportamento puder

alterar-se dinamicamente, como uma resposta espontânea aos estímulos de entrada que o alimentam. Essa alteração deve se dar sem que haja qualquer interferência de agentes externos, inclusive de usuário. Para que isso possa acontecer, os dispositivos adaptativos devem ser, portanto, automodificáveis.

Na construção de um dispositivo adaptativo dirigido por regras [22], conceitualmente é possível separar de forma clara dois componentes importantes: um dispositivo subjacente, tipicamente não-adaptativo, e um mecanismo adaptativo, responsável pela incorporação da adaptatividade.

O mecanismo adaptativo tem dois elementos: as declarações das funções adaptativas e as associações de chamadas de funções adaptativas às regras que definem o dispositivo subjacente (tais chamadas denominam-se ações adaptativas).

A primeira parte consiste de um conjunto de declarações de funções adaptativas paramétricas, as quais especificam, de uma forma semelhante ao que ocorre com as declarações de sub-rotinas e funções em uma linguagem de programação usual, as modificações a serem impostas ao conjunto de regras, na ocasião em que a função adaptativa correspondente for ativada.

Programam-se as modificações a serem impostas ao conjunto de regras através de uma lista de ações adaptativas elementares. Uma ação adaptativa elementar indica três tipos de primitivas de edição: consultas, inclusões e exclusões.

- ✓ As ações adaptativas elementares de consulta permitem inspecionar o conjunto de regras que definem o dispositivo, em busca de regras que sejam aderentes a algum um padrão fornecido.
- ✓ Ações elementares de exclusão permitem remover do conjunto de regras qualquer regra aderente ao padrão fornecido.
- ✓ Ações de inclusão permitem especificar a adição de uma nova regra, de acordo com um padrão fornecido.

Quando uma ação adaptativa é executada, todas as ações elementares de consulta e exclusão são executadas em primeiro lugar, e por último são efetuadas todas as inclusões.

A segunda parte refere-se à associação de, no máximo, duas ações adaptativas a regras selecionadas do dispositivo subjacente. Uma dessas ações adaptativas é especificada para ser executada antes que a regra à qual está associada seja aplicada. A segunda é executada após a aplicação da regra. Ambas são opcionais, portanto regras que não se estejam associadas a ações adaptativas comportam-se como simples regras não-adaptativas.

A seguir, é apresentada uma figura com o esquema geral para Dispositivos Adaptativos. Na figura, é possível ver uma cadeia de entrada e o comportamento inicial do dispositivo. Para se consumir um símbolo de entrada da cadeia, a camada adaptativa do dispositivo é acionada, realizando, opcionalmente, ações adaptativas antes de executar as regras do dispositivo subjacente. Em seguida, são execuções as regras do próprio dispositivo subjacente. Após a execução das regras do dispositivo subjacente, ações adptativas posteriores podem ser executadas. E, finalmente, tem-se o novo comportamento do dispositivo.

Figure 1: Esquema de um dispositivo adaptativo.

Em outros termos, um dispositivo adaptativo pode ser obtido pela incorporação de ações adaptativas às regras da formulação subjacente, de tal modo que, sempre que alguma delas for aplicada, a ação adaptativa associada é acionada, causando as devidas alterações correspondentes no conjunto de regras do dispositivo não-adaptativo subjacente. Porém, quaisquer possíveis alterações no comportamento do dispositivo devem ser plenamente conhecidas a priori, em quaisquer etapas de sua operação em que tais alterações de comportamento devam se efetivar.

A principal característica dessa formulação é que ela apresenta a desejável propriedade de preservar a natureza do formalismo não-adaptativo, de tal forma que o dispositivo adaptativo resultante possa ser facilmente compreendido por todos os que tiverem familiaridade com o formalismo não-adaptativo subjacente original.

Em [15] é possível encontrar diversos tipos de dispositivos adaptativos já propostos, sendo alguns citados a seguir: Autômatos Adaptativos, Statecharts Adaptativos, Redes de Markov Adaptativas, Tabelas de Decisão Adaptativas, Árvores de Decisão Adaptativas, Redes de Petri Adaptativas, ISDL Adaptativos, entre outros.

B. Aplicações de Dispositivos Adaptativos

Inúmeras aplicações potenciais existem para a tecnologia derivada da adaptatividade, incluindo: inferência [21, 18], arte usando computador [20, 2], processamento de linguagem natural [19, 39], síntese de voz [32], reconhecimento de padrões [9, 32], tomada de decisão [32, 26, 40], linguagens de programação adaptativas [34; 11, 28], otimização de código [16], meta-modelagem [5], computação evolutiva [31, 3], engenharia de software [35], robótica [9, 37] segurança (security) [27, 7, 8] e, ensino por computador [10], entre outras.

C. Classificação de Dispositivos Adaptativos

Os formalismos subjacentes dos dispositivos adaptativos podem ser classificados em variadas categorias, conforme sua forma de operação. Entre outras, têm-se as seguintes, definidas em [24]:

- ✓ dispositivos de reconhecimento, da classe dos autômatos, baseados na sucessão de mudanças de estados;
- ✓ dispositivos de processamento, como as linguagens de programação adaptativas, que permitem descrever a lógica de programas com código automodificável;
- ✓ dispositivos de geração, da classe das gramáticas, baseados na aplicação sucessiva de regras de substituição;
- ✓ dispositivos para a representação de sistemas assíncronos, tais como os statecharts, que incorporam mecanismos responsáveis pela representação de fenômenos de sincronização;
- ✓ dispositivos estocásticos, como as redes de Markov, capazes de representar fenômenos de caráter aleatório;
- ✓ dispositivos de auxílio à tomada de decisões, representados principalmente pelas tabelas de decisão e pelas árvores de decisão;

D. Máquina de Moore (não-adaptativa)

Uma Máquina de Moore [12, 13, 14] é um autômato finito determinístico modificado, que possui saídas associadas aos estados. Tal máquina possui uma função que gera uma palavra de saída para cada estado da máquina, podendo essa ser uma palavra vazia.

É representada por uma héptupla:
ND = (C, NR, S, c0, A, NA, RS), onde:

- ✓ ND é uma Máquina de Moore (não-adaptativa), cuja operação é determinada pelo conjunto de regras *NR*.
- ✓ C é o conjunto de todas as suas possíveis configurações, e $c_0 \in C$ é sua configuração inicial.
- ✓ S é o conjunto (finito) de todos os possíveis eventos que são estímulos de entrada válidos para *ND*, com $\varepsilon \in S$.
- ✓ $A \subseteq C$, (respectivamente, $F = C - A$) é o subconjunto de todas as suas configurações de aceitação (respectivamente, configurações de rejeição).
- ✓ ε denota "vazio", e representa o elemento nulo de conjunto ao qual ele pertence.
- ✓ $w = w_1 w_2 ... w_n$ é uma cadeia de estímulos de entrada, onde $w_k \in S - \{\varepsilon\}$, $k = 1, ..., n$, com $n \geq 0$.
- ✓ *NA* é um conjunto (finito), com $\varepsilon \in NA$, de todos os possíveis símbolos a serem emitidos como saídas por *ND*.
- ✓ *RS* é a função de saída por meio da relação $RS \subseteq C \times NA^*$ a qual é uma função total que determina a geração de uma palavra de saída para cada estado.
- ✓ *NR* é o conjunto de regras que definem *ND* por meio da relação $NR \subseteq C \times S \times C$. As regras $r \in NR$ apresentam-se na forma $r = (c_i, s, c_j)$, indicando que, em resposta a um estímulo de entrada qualquer $s \in S$, r modifica a configuração corrente de ci para a nova configuração *cj*, e consome *s*.

Uma regra $r = (c_i, s, c_j)$, com $r \in NR$; $c_i, c_j \in C$; $s \in S$, é dita compatível com a configuração corrente c se e apenas se $c_i = c$, desde que s seja vazio ou igual ao estímulo de entrada corrente do dispositivo. Neste caso, a aplicação de uma regra compatível move o dispositivo para a configuração c_j (denotada por $c_i \Rightarrow^s c_j$). Note-se que *s* pode ser vazio.

Um seqüência opcional de movimentos em vazio, finalizada por um movimento que consuma o símbolo w_k, é denotada por: ci □~ cm, m $\geq c_i \Rightarrow^{\sim} c_m$, $m \geq 0$, e abrevia $c_i \Rightarrow^{\varepsilon} c_1 \Rightarrow^{\varepsilon} c_2 \Rightarrow^{\varepsilon} \ldots \Rightarrow^{\varepsilon} c_m$.

Um seqüência opcional de movimentos em vazio, finalizada por um movimento que consuma o símbolo w_k, é denotada por: ci $c_i \Rightarrow^{\sim wk} c_j$ e representa
$c_i \Rightarrow^{\sim} c_m \Rightarrow^{wk} c_j$.

Diz-se que uma cadeia de entrada $w = w_1 w_2 ... w_n$ é aceita por *ND* quando $c_0 \Rightarrow^{\sim w1} c_1 \Rightarrow^{\sim w2} ... \Rightarrow^{\sim wn} c_n \Rightarrow^{\sim} c$ (abreviadamente denotada como $c_0 \Rightarrow^{w} c$, com $c \in A$). De forma complementar, diz que *w* é rejeitada por *ND* quando $c \in F$.

A linguagem definida por *ND* é o conjunto: $L(ND) = \{ w \in S^* \mid c_0 \Rightarrow^{w} c, c \in A \}$ de todas as cadeias $w \in S^*$ que são aceitas por *ND*.

O processamento de uma Máquina de Moore para uma dada entrada w consiste na sucessiva aplicação da função programa para cada símbolo de w (da esquerda para a direita), até ocorrer uma condição de parada. A palavra vazia como saída da função programa indica que nenhuma gravação é realizada e, portanto, a cabeça da fita de saída não se move. Se todos os estados geram saída vazia, então a Máquina de Moore se comporta como se fosse um autômato finito.

Outro tipo de transdutor é conhecido como máquina de Mealy, que também é um autômato finito modificado, mas que possui as palavras de saída associadas com as transições entre os estados. Neste tipo de máquina, as palavras de saída dependem do estado atual e do valor das entradas. Essa máquina não será abordada aqui.

E. Máquina de Moore Adaptativa

Uma Máquina de Moore Adaptativa (MMA) é um caso particular de aplicação do conceito de dispositivo guiado por regras adaptativo [22, 32], no qual o mecanismo subjacente utilizado é o da Máquina de Moore (descrito na seção anterior).

Complementarmente a um autômato adaptativo, uma Máquina de Moore Adaptativa pode adicionar ou remover as saídas associadas a cada estado, além de ter funções adaptativas para adicionar ou remover estados e transições.

Defina-se um contador T, que é iniciado com o valor *zero*, e automaticamente incrementado de uma unidade sempre que uma ação adaptativa não-vazia for executada. Cada valor k assumido por T pode ser usado para indexar os nomes de conjuntos variantes no tempo. Neste caso, tal indexação seleciona o passo k de operação de AD.

Um dispositivo $AD = (ND_0, AM)$ é dito adaptativo sempre que, para qualquer passo de operação $k \geq 0$, AD seguir o comportamento de ND_k, até que a execução de alguma ação adaptativa não-vazia inicie seu $(k+1)$-ésimo passo ao modificar seu próprio conjunto de regras.

Em qualquer passo $k \geq 0$ de operação de AD, sendo ND_k o correspondente dispositivo subjacente definido por R_k, a execução de qualquer ação adaptativa não-vazia provoca a evolução de R_k para R_{k+1}. Assim, AD começa a executar seu $(k+1)$-ésimo passo de operação já com o conjunto R_{k+1}, resultante das alterações impostas, no passo anterior, ao conjunto R_k. Daí em diante, AD seguirá o comportamento de ND_{k+1} até que alguma outra ação adaptativa não-vazia provoque o início de um novo passo de operação. Esse procedimento se repete até que a cadeia de entrada tenha sido integralmente processada ou que algum erro interrompa a operação de AD.

O dispositivo adaptativo AD inicia sua operação em c_0, em sua forma inicial $AD_0 = (C_0, R_0, S, c_0, A, NA, BA, AA)$. No passo $k \geq 0$, um estímulo de entrada move AD para sua próxima configuração e então AD inicia seu $(k+1)$-ésimo passo de operação se e somente se uma nova ação adaptativa não-vazia for executada. Dessa forma, estando AD em seu passo k, e apresentando-se na forma $AD_k = (C_k, R_k, S, c_k, A, NA, BA, AA)$, a execução de uma ação adaptativa não-vazia deve conduzi-lo a $AD_{k+1} = (C_{k+1}, R_{k+1}, S, c_{k+1}, A, NA, BA, AA)$.

Nessa formulação, tem-se:

- ✓ $AD = (ND_0, AM)$ representa algum dispositivo adaptativo, dado por um dispositivo subjacente iniciado por ND_0 e um mecanismo adaptativo AM.
- ✓ ND_k é o dispositivo não-adaptativo subjacente de AD em algum passo de operação k. ND_0 é o dispositivo não-adaptativo subjacente de AD, definido pelo conjunto inicial de regras não-adaptativas R_0. Por definição, quaisquer regras não-adaptativas, em qualquer R_K, espelham as regras adaptativas correspondentes em R_K.
- ✓ C_K é o conjunto de todas as possíveis configurações que ND pode assumir no seu $(k+1)$-ésimo passo de operação, e $c_k \in C_k$ é a configuração em que inicia tal passo de operação. Para $k = 0$, tem-se, respectivamente, C_0, o conjunto inicial de configurações válidas, e $c_0 \in C_0$, a configuração inicial, tanto para ND_0 como para ND.
- ✓ ε denota "vazio", no contexto em que for empregado, a ausência de qualquer outro elemento válido do conjunto correspondente.
- ✓ S é o conjunto (finito, invariável) de todos os possíveis eventos que são possíveis estímulos de entrada para AD ($\varepsilon \in S$).
- ✓ $A \subseteq C$, (respectivamente, $F = C - A$) é o subconjunto de todas as suas configurações de aceitação (respectivamente, de rejeição).
- ✓ BA e AA são os conjuntos de ações adaptativas, ambos contendo a ação adaptativa vazia ($\varepsilon \in BA \cap AA$).
- ✓ $w = w_1 w_2 \ldots w_n$, $k = 1, \ldots, n$ é uma cadeia de estímulos não-vazios de entrada, ou seja, $w_k \in S - \{\varepsilon\}$.
- ✓ NA, com $\varepsilon \in NA$, é um conjunto (finito, invariável) de todos os possíveis símoblos passíveis de serem gerados por AD como efeitos colaterais da aplicação de regras adaptativas.
- ✓ AR_k é um conjunto de regras adaptativas, dadas por uma relação $AR_k \subseteq BA \times C \times S \times C \times NA \times AA$. Em particular, AR_0 define o comportamento inicial de AD. Ações adaptativas mapeiam o conjunto corrente de regras adaptativas AR_k de AD em um novo conjunto AR_{k+1}, através da aplicação de um conjunto de operações de inclusão e/ou remoção de regras adaptativas AR_k. Regras adaptativas $ar \in AR_k$ apresentam-se na forma $ar = (ba, c_i, s, c_j, z)$, significando que, em resposta a algum estímulo de entrada $ar \in AR_k$, ar executa inicialmente a ação adaptativa $ba \in BA$; a execução de ba é descontinuada caso venha a eliminar ar de AR_k; caso contrário, aplica-se a regra não adaptativa subjacente $nr = (c_i, s, c_j, z)$,, conforme descrito anteriormente para o dispositivo não-adaptativo; finalmente, executa-se a ação adaptativa $aa \in AA$.
- ✓ Defina-se AR como o conjunto de todas as possíveis regras adaptativas para o dispositivo AD.
- ✓ Defina-se R como o conjunto de todas as possíveis regras não-adaptativas para ND, o dispositivo subjacente de AD.
- ✓ $AM \subseteq BA \times R \times AA$, definido para um determinado dispositivo adaptativo AD, é o mecanismo adaptativo de AD que deve ser aplicado, em qualquer passo k de operação, a cada regra $R_k \subseteq R$, e deve ser tal que opere como função, quando aplicado a qualquer subdomínio $R_k \subseteq NR$. Isso determina um único para de ações adaptativas a serem associadas a cada regra não-adaptativa.

Note-se que a definição de AR_k como subconjunto do produto cartesiano $BA \times C \times \ldots \times AA$ pode ser refinada, à luz da definição do mecanismo adaptativo AM. Assim, pode-se reinterpretar AR_k como subconjunto do produto cartesiano $BA \times AR_k \times AA$.

O conjunto $AR_k \subseteq AR$ pode ser construído como a coleção de todas as regras adaptativas que possam ser obtidas pela associação de pares de ações adaptativas com as regras não-adaptativas em AR_k.. Equivalentemente, como as regras de NR_k. espelham as de AR_k. em cada momento, é possível construir AN_k. simplesmente removendo-se, das regras de AR_k., todas as referências a ações adaptativas.

IV. EXEMPLO DE APLICAÇÃO

Primeiramente, é apresentada a estrutura geral do sistema, na qual depois de sua abertura, é apresentado um menu para que o aluno possa escolher o curso que deseja participar. De maneira genérica, o sistema prevê o oferecimento de diferentes cursos. Assim, após a escolha do curso desejado, é realizada uma chamada à sub-máquina correspondente, passando de "A" e retornando em "B" após a realização do curso. Ao término do curso, retorna-se ao menu de escolha de cursos. As figuras 2 e 3, representam os autômatos desse modelo.

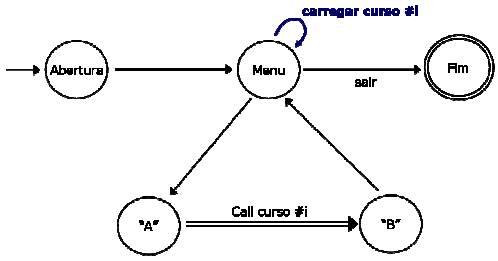


Figure 2: Autômato das chamadas aos cursos disponíveis.

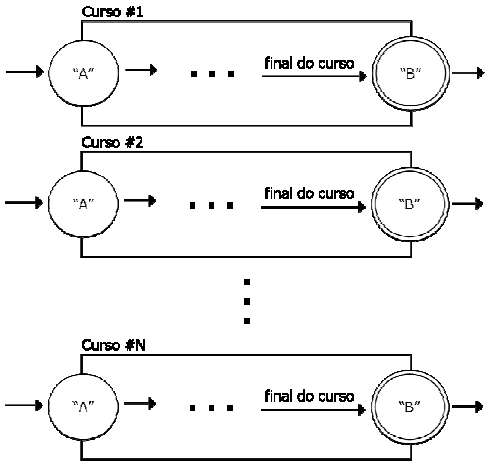


Figure 3: Representação das chamadas dos cursos.

A seguir, será, então, apresentado um exemplo demonstrando como será a representação de cada um dos cursos disponibilizados aos alunos. O exemplo será apresentado através de uma Máquina de Moore (não adaptiva). Posteriormente, serão apontadas as mudanças propostas para tornar o curso mais dinâmico, através do uso da adaptatividade.

A idéia central é elaborar um curso desmembrando-o seu material didático em pequenos blocos chamados de objetos de aprendizagem [41]. Assim, o curso pode ser montado com o mesmo material, de diferentes formas, ou ainda, ser modificado dinamicamente.

Cada curso é definido como um autômato [17] e referencia um conjunto de objetos de aprendizagem independentes, tal que o roteiro de estudos do curso permanece separado do material didático. Cada estado representa um tópico de estudo dentro do curso. As transições compõem o roteiro das aulas. A função de saída funciona como a ligação lógica dos estados com os objetos de aprendizagem.

A. Curso com Máquina de Moore

Em [17] é apresentado uma forma de modelagem de cursos para Web utilizando autômatos finitos determinísticos com saída, cuja principal finalidade é a criação de material hipermídia independente do esquema de navegação. Na proposta, um curso pode ser representado tanto por Máquina de Moore ou quanto por Máquina de Mearly, tendo o material didático vinculado, respectivamente, aos estados ou às transições do transdutor. Em ambos os casos, esse autômato com saída fornece uma máquina abstrata para o controle da navegação em hipertextos.

Segundo esse modelo, cada autômato constitui um curso definido sobre um conjunto de hiperdocumentos (unidades de informação) independentes. As n-uplas dos autômatos de cursos apresentam uma correspondência às estruturas de hiperdocumentos na Web. O alfabeto de símbolos de entrada é um conjunto de nomes que identificam as âncoras de navegação no hipertexto. As saídas estão relacionadas a unidades de informação constituídas por hiperdocumentos e as palavras de saída, presentes na função de saída, são hipertextos apresentados como uma única página Web no navegador do usuário. As transições, definidas na função programa, funcionam como ligações lógicas (e não ligações físicas no documento) entre os conteúdos dos cursos e definem possíveis links a serem selecionados durante a navegação pelo hipertexto.

Uma das vantagens da solução utilizada é que a estrutura dos autômatos com saída permite a criação do material hipermídia de forma independente do autômato em si, possibilitando a programação de seqüências de estudo com objetivos e enfoques específicos, reuso de parte ou íntegra das páginas Web em diversos cursos, eliminando a redundância na criação de páginas, bem como a modularização que facilita a expansibilidade do sistema.

A estrutura do sistema de hipertexto que não utiliza links diretos permite a estruturação e edição do conjunto de materiais hipermídia sem a necessidade do autor dos documentos se preocupar com a inserção de links diretamente nos textos, facilitando a tarefa de edição dos documentos. Um benefício adicional é que, como páginas de um curso podem ser constituídas por outras sub-páginas concatenadas, o autor pode construir funções de saída/programa diferentes que contenham acesso a páginas de mesmo conteúdo, evitando-se redundância na criação dos documentos Html. Por exemplo, pode existir uma função que gere uma página contendo uma "definição" e um "exemplo", enquanto outra página pode ter o mesmo "exemplo" como "dica" para um exercício.

Tomando como exemplo um mini-curso para ensino de

algoritmos, usando Máquina de Moore, tem-se:

Alfabeto de entrada:
{ próxima, anterior, exercício, resumos, saída }
Alfabeto de saída:
{ A, B, C, D, E, F, G, H, I }
A – Introdução a Algoritmos
B – Definição de Desvio Condicional
C – Exemplo de Desvio Condicional
D – Exercícios sobre Desvio Condicional
E – Definição de Laço de Repetição
F – Exemplo de Laço de Repetição
G – Exercícios sobre Laço de Repetição
H – Conclusões
I – Fim

A figura a seguir ilustra uma possível configuração de Máquina de Moore, com suas regras de transição e regras de saída para a representação de um mini-curso sobre algoritmos.

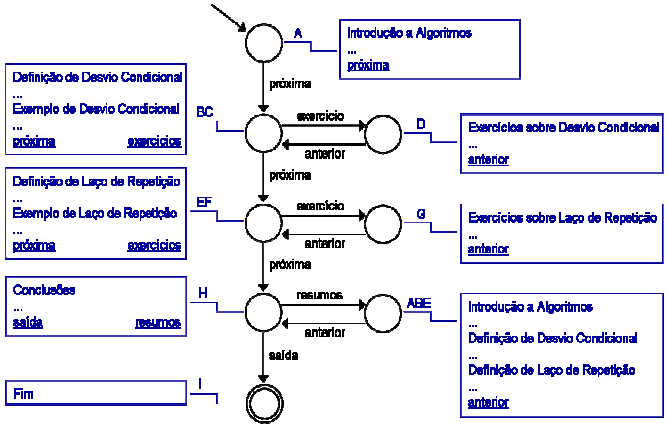


Figure 4: Curso de Algoritmos representado com Máquina de Moore.

É possível notar que num mesmo estado podem existir mais de uma unidade de informação associada, uma vez que a função de saída pode gerar palavras a partir do fecho de kleen do alfabeto de símbolos de saída. É possível, também, notar o reuso dessas unidades de informação, que podem aparecer por mais de uma vez durante o curso. Da mesma maneira, é também verdade que esse material didático possa ser reaproveitado em outros cursos.

B. Curso com Máquina de Moore Adaptativa

Para a modelagem de um curso estático, o funcionamento de uma Máquina de Moore é suficiente para se conseguir especificar formalmente todo o modelo. Contudo, um enriquecimento nesse tipo de máquina, através da camada adaptativa, permite maior poder de representação, podendo tornar o acompanhamento do curso mais flexível e dinâmico, podendo tratar questões dependentes de contexto. Desta forma, um sistema adaptativo tenta antecipar as necessidades e desejos dos alunos a partir de modelos que representam o seu perfil, nível de conhecimento e preferências.

Com base nos modelos de Sistemas Hipermídia Adaptativos [4, 25], é proposta a aplicação de ações adaptativas em três níveis:

- Adaptatividade na navegação
- Adaptatividade na apresentação
- Adaptatividade no conteúdo

A adaptatividade na navegação tem por objetivo demonstrar um caminho ideal para que um aluno em particular alcance seus objetivos, evitando que o mesmo se disperse, diante de um amplo conjunto de opções e sinta-se desorientado diante da grande quantidade de material oferecido. A adaptação na navegação é obtida acrescentando-se novas transições no modelo do curso ou, ainda, removendo-se algumas das transições existentes. Uma vez que as transições definem o roteiro de aulas do curso, pode-se manter o mesmo conteúdo programático, porém oferecer seqüências alternativas para seguir do curso. De acordo com o perfil de cada aluno, pode-se ofertar um número maior de caminhos a serem seguidos ou restringir essas possibilidades, tornando o curso mais linear e assim limitar o espaço de busca As regras adaptativas nesse nível incidem exclusivamente sobre a criação e remoção de transições.

A adaptatividade na apresentação diz respeito à forma como as informações serão exibidas para os alunos, dependendo do perfil de cada aluno. Ou seja, um mesmo conteúdo pode ser apresentado utilizando-se diferentes layouts ou mesmo através de diferentes mídias, conforme as preferências do usuário. Nota-se, também, que o material a ser apresentado ao aluno não está necessariamente pronto e pode ser gerado em tempo real. Essencialmente, a adaptatividade na apresentação implica em se modificar as palavras de saída associadas aos estados.

A adaptatividade no conteúdo refere-se a possibilidade de se substituir um objeto de aprendizagem por outro mais elaborado, ou mesmo mais simplificado, dependendo da situação. Com isso, o curso pode ser atualizado trocando-se as palavras de saída ou até mesmo criando-se novos estados, com tópicos complementares não contidos no curso original.

As regras para o acompanhamento do curso podem ser definidas pelo professor, sendo aplicadas de acordo com a disciplina a ser abordada ou a linha pedagógica do professor.

A seguir é apresentado um simples exemplo no qual o professor define que o aluno precisa obter conhecimentos téoricos para somente depois poder fazer os exercícios sobre o assunto abordado. O aluno poderá retornar ao material teórico, caso necessite, e retomar o desenvolvimento do exercícios. Contudo, somente após realizar os exercícios, o aluno poderá fazer a avaliação sobre o conteúdo abordado. No entanto, a avaliação poderá ser realizada uma única vez. Para tratar essa dependência de contexto, é criada uma ação adaptativa disparada após o aluno terminar sua avaliação. Essa ação adaptativa irá remover a transição entre os exercícios e a avaliação, impedindo que o aluno volte a fazer a avaliação.

Figure 5: Configuração inicial, antes do aluno realizar a avaliação.

Figure 6: Configuração do curso após a avaliação.

Uma questão importante a ser pensada diz respeito a onde e como aplicar regras adaptativas. A princípio, essas regras devem ser criadas pelo professor responsável pela elaboração do curso. Contudo, a critério do próprio professor, essas regras podem ser baseadas na própria inteligência coletiva do grupo de alunos participantes. Ou seja, a participação ativa dos alunos através de sugestões de novos roteiros de aula ou novos objetos de aprendizagem pode contribuir e difinir as novas configurações do curso para os novos aprendizes.

V. CONCLUSÕES

O uso de um modelo baseado em Máquina de Moore Adaptativa para projetar sistemas de ensino assistidos por computador apresenta um grande potencial para se criar ambientes de ensino flexíveis, que se ajustem ao perfil de cada estudante.

Uma das características importantes, que ficou evidenciada na proposta apresentada, é a possibilidade de se criar o material didático de forma independente do autômato. A independência do material didático em relação ao roteiro das aulas permite, por exemplo: reutilização do material instrucional em diversos cursos. Outra característica importante é a possibilidade de se criar, para um mesmo curso, diferentes roteiros de aulas, com enfoques diferenciados, pois um curso pode ter seu autômato modificado dinamicamente.

O projeto encontra-se em fase de desenvolvimento. Já existe disponível um protótipo capaz de executar o modelo proposto de Máquina de Moore Adaptativa, representando cursos sobre quaisquer assuntos, obedecendo ao roteiro de aulas previstas e controlando, por exemplo, questões como pré-requisitos entre módulos. O protótipo está sendo desenvolvido para uso em ambiente Web. Dessa forma, basta apenas um programa navegador da Internet para poder utilizar o ambiente, independente de plataforma ou sistema operacional utilizado.

Como trabalhos futuros, pode-se: aplicar o conceito de máquina de Moore adaptativa para a elaboração de provas adaptativas; monitorar o comportamento de diversos alunos e, através dos conceitos de Inteligência Coletiva, permitir que os materiais didáticos do curso e/ou roteiros de estudos do curso sejam modificados; expandir o conceito usado para curso, para que o mesmo possa ser aplicado, por exemplo, numa grade curricular completa, e, assim, trabalhar com adaptatividade multi-nível; criar uma ferramenta de autoria, que ofereça uma interface gráfica amigável para que o professor possa montar o roteiro do curso, sem a necessidade de ser treinado.

REFERÊNCIAS

[1] ALMEIDA Jr, J. R.; *STAD – Uma Ferramenta para Representação e Simulação de Sistemas Através de Statecharts Adaptativos*; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; 202p;1995.

[2] BASSETO, B. A.; *Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos*; Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, 2000.

[3] BRAVO, C. et al. *Towards an Adaptive Implementation of Genetic Algorithms* INBI 2007, XXXIII CLEI – Conferencia Latinoamericana de Informática, San José, Costa Rica, 2007.

[4] BRUSILOVSKY, P. *Methods and Techniques of Adaptive Hypermedia. User Modeling and User Adapted Interaction. Special issue on adaptive hypertext and hypermedia*. Pittsburgh, 1996.

[5] CAMOLESI, A. R.; *Proposta de um Gerador de Ambientes para a Modelagem de Aplicações usando Tecnologia Adaptativa*; Tese de Doutorado; Escola Politécnica; Universidade de São Paulo; São Paulo; 2007.

[6] CAMOLESI, Almir Rogério; JOSÉ NETO, João. *Modelagem AMBER-Adp de um ambiente para Gerenciamento de Ensino a Distância*. Anais do XIII Simpósio Brasileiro de Informática na Educação. São Leopoldo-RS, 2002. v. 1, p. 401-409.

[7] CEREDA, P.R.M. & ZORZO, S.D.; ***Access Control Model Formalism using Adaptive Automaton.*** IEEE Latin America Transactions. Volume 6, Issue 5, ISSN: 1548-0992, September 2008.

[8] CEREDA, P.R.M.; ***Modelo de Controle de Acesso Adaptativo.*** Dissertação de Mestrado, UFSCAR, São Carlos/SP, 2008.

[9] COSTA, E. R., HIRAKAWA, A. R., NETO, J. J.; *An Adaptive Alternative for Syntactic Pattern Recognition*. Proceeding of 3rd International Symposium on Robotics and Automation, ISRA 2002, pp. 409-413. Toluca, Mexico, 2002.

[10] DIZERÓ, W. J.; *Uso de Máquina de Moore Adaptativa na Modelagem de Cursos*; 3° Workshop de Tecnologia Adaptativa – WTA'2009; São Paulo-SP, Brasil, 2009.

[11] FREITAS, A.V. & NETO, J. J.; *Adaptive Languages and a New Programming Style* 6th WSEAS International Conference on Applied Computer Science (ACS'06) – Tenerife, Canary Islands, Spain, December, 2006.

[12] HARRISON, Michael A.: *Introduction to Formal Language Theory*, Ed. Addison-Wesley; 1a edição; Califórnia – USA (1978)

[13] HOPCROFT, J. E., Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley (1979).

[14] LEWIS, Harry R. & Papadimitriou, Christos H.: *Elementos de Teoria da Computação*. Ed. Bookman, 2ª edição, (2000)

[15] LTA - Laboratório de Linguagens e Tecnologias Adaptativas; Poli-USP; São Paulo; http://www.pcs.usp.br/~lta/; site visitado em Novembro/2009.

[16] LUZ, J.C.; *Tecnologia Adaptativa Aplicada à Otimização de Código em Compiladores;* Dissertação de Mestrado, EPUSP, São Paulo, 2004.

[17] MACHADO, Júlio P. et al.; *Autômatos Finitos: um Formalismo para Cursos na Web*; XIII Simpósio Brasileiro de Engenharia de Software 1999, SBES'99, Florianópolis, Brasil.

[18] MATSUNO, I.P.; *Um Estudo do Processo de Inferência de Gramáticas Regulares e Livres de Contexto Baseados em Modelos Adaptativos*. Dissertação de Mestrado, USP, São Paulo, 2006.

[19] MENEZES, C.E.D.; *Um Método para a Construção de Analisadores Morfológicos, Aplicado à Língua Portuguesa, Baseado em Autômatos Adaptativos*. Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, 2000.

[20] NETO, J. J. & BASSETO, B. A.; *A Stochastic Musical Composer Based on Adaptative Algorithms*; Proceedings of the 6th Brazilian Symposium on Computer Music - SBC&M99, Rio de Janeiro, 1999.

[21] NETO, J. J. & IWAI, M.K.; *Adaptive Automata for Syntax Learning*. CLEI 98 – XXIV Conferencia Latinoamericana de Informática, MEMORIAS. pp. 135-149, Quito, Equador, 1998.

[22] NETO, J. J., *Adaptive Rule-Driven Devices - General Formulation and Case Study*. Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol. 2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250.

[23] NETO, J. J., *Contribuições à Metodologia de Construção de Compiladores*, Thesis (Livre Docência) – Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[24] NETO, J. J.; ***Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa*; IEEE Latin America Transactions; vol. 5 n° 7; novembro** 2007.

[25] PALAZZO, L. A. M.; *Sistemas de Hipermídia Adaptativa*, In Anais do XXII Congresso da Sociedade Brasileira de Computação (SBC), Florianópolis, 2002

[26] PEDRAZZI, T., TCHEMRA, A. H., ROCHA, R. L. A.; ***Adaptive Decision Tables - a Case Study of their Application to Decision-Taking Problems*;** Proceedings of International Conference on Adaptive and Natural Computing Algorithms - ICANNGA 2005, Coimbra, March 21-23, 2005.

[27] PELEGRINI, E. J. & NETO, J. J.; ***Applying Adaptive Technology in Data Security*.** Proceedings of the 6th Peruvian Computer Week - JPC 2007, Trjillo, Peru, Novembro 5-10 (pp. 31-40), 2007.

[28] PELEGRINI, E. J.; ***Códigos Adaptativos e Linguagem para Programação Adaptativa: Conceitos e Tecnologia*.** Dissertação de Mestrado, EPUSP, São Paulo, 2009.

[29] PIMENTA, Pedro e BAPTISTA, Ana Alice. *Das plataformas de E-learning aos objetos de aprendizagem*. In. DIAS, Ana Augusta Silva e GOMES, Maria João. Elearning para e-formadores. Minho, TecMinho, 2004, p. 97-109.

[30] PISTORI, H. et al. *Defect Detection in Raw Hide and Wet Blue Leather CompIMAGE* – Computational Modelling of Objects Represented in Images: Fundamentals, Methods and Applications, Coimbra, 2006.

[31] PISTORI, H., MARTINS, P.S., CASTRO Jr., A.A.; *Adaptive Finite State Automata and Genetic Algorithms – Merging Individual Adaptation and Population Evolution*. Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005, Coimbra, March, 2005.

[32] PISTORI, H.. *Tecnologia Adaptativa em Engenharia de Computacão: Estado da Arte e Aplicações*. Tese (Doutorado) - Universidade de São Paulo, 2003.

[33] ROCHA, R. L. A.& NETO, J. J.; *Autômato adaptativo, limites e complexidade em comparação com máquina de turing*; In: Proceedings of the second Congress of Logic Applied to Technology - LAPTEC'2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia: [s.n.], 2000.

[34] ROCHA, R. L. A.& NETO, J. J.; *Uma proposta de linguagem de programação funcional com características adaptativas*. IX Congreso Argentino de Ciencias de la Computación. La Plata, Argentina, Octubre de 2003.

[35] SILVA, P. S. M. & NETO, J. J. *An Adaptive Framework for the Design of Software Specification Languages*. Proceedings of International Conference on Adaptive and Natural Computing Algorithms - ICANNGA 2005, Coimbra, March, 2005.

[36] SILVA, P. S. M. & NETO, J. J. *An Adaptive Framework for the Design of Software Specification Languages*. Proceedings of International Conference on Adaptive and Natural Computing Algorithms - ICANNGA 2005, Coimbra, March, 2005.

[37] SLONNEGER, K. & KURTZ, B.L. *Formal Syntax and Semantics of Programming Languages – a Laboratory Based Approach* Addison Wesley, 1995.

[38] SOUZA, M. A. A. & HIRAKAWA, A. H.; *Robotic Mapping and Navigation in Unknown Environments Using Adaptive Automata*. Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005, Coimbra, March, 2005.

[39] STANDISH T. A.; *Extensibility in Programming Language Design* ACM SIGPLAN Notices, p. 18-21, vol 10 n. 7, July, 1975.

[40] TANIWAKI, C. Y. O.; *Formalismos Adaptativos na Análise Sintática de Linguagem Natural* Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, 2001.

[41] TCHEMRA, A. H.; ***Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério*.** Tese de Doutorado, Escola Politécnica da USP, São Paulo, 2009.

[42] WILEY, D. A. *Conecting learning objects to instructional theory: A definition, a methaphor and a taxonomy*. The Instructional Use of Learning Objets. Wiley, D. (Ed.) 2001. Disponível na URL: http://www.reusabilility.org/read/chapters/wiley.doc. 2001. Acesso em 20/03/2005.

Wagner José Dizeró nasceu em Piracicaba-SP, Brasil, em 29 de Agosto de 1975. Se graduou na Faculdade de Tecnologia de Americana (1996) e defendeu mestrado em Ciência da Computação pela Universidade Federal de São Carlos (1999). Atualmente, é aluno de doutorado do departamento de Engenharia da Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo. Profissionalmente, é Professor no Centro Universitário de Lins a 9 anos, onde também é coordenador dos cursos de Bacharelado em Sistemas de Informação e Pós-Graduação em Gestão de Sistemas de Informação. Entre seus campos de interesse encontram-se: tecnologias adaptativas, engenharia de software, sistemas para Internet, informática na educação e educação a distância.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Tecnologia Adaptativa do PCS – Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

Servidor Web Adaptativo

P. R. M. Cereda

***Resumo*— A navegação do usuário na Internet gera uma procura por métodos que possam melhorar a experiência do usuário em um determinado website. Este artigo apresenta uma proposta de um servidor web adaptativo que assume um papel mais ativo durante a navegação do usuário, fornecendo técnicas de personalização e recomendação aos websites que estão hospedados em seu domínio, além de poder tratar de modo mais consistente e amigável os erros que eventualmente acontecem. São apresentados os aspectos de implementação e um experimento para verificação e análise dos módulos utilizados no servidor web.**

***Palavras-chave:*— Servidor Web, Autômato Adaptativo, Sistema de Recomendação, Personalização.**

I. Introdução

A navegação do usuário na Internet tornou-se um parâmetro fundamental para analisar a qualidade dos serviços disponibilizados. A grande maioria dos websites procura obter informações acerca dessa navegação e oferecer recursos personalizados e funcionalidades que possam melhorar essa experiência de navegação dos usuários visitantes.

Os servidores web são responsáveis por disponibilizar conteúdo para o usuário na Internet. De modo geral, o servidor web tem apenas uma participação passiva no processo de melhoria da experiência da navegação do usuário. A aplicação web quase sempre assume a tarefa de personalizar recursos e oferecer características inerentes aos interesses dos usuários.

Este artigo apresenta uma proposta de um servidor web adaptativo, chamado *Apache HTTPD Adaptativo*. O servidor em questão possui seis módulos que buscam melhorar a experiência de navegação dos usuários e utilizam autômatos adaptativos para oferecer tais funcionalidades. O autômato adaptativo foi escolhido para representar as características de cada módulo devido à sua simplicidade, capacidade de modificação autônoma e poder computacional.

A organização deste artigo é a seguinte: a Seção I contextualiza a navegação dos usuários na Internet, apresentando técnicas para melhoria da navegação e alguns problemas comuns encontrados. A Seção II apresenta algumas considerações sobre a adaptatividade em servidores web. O servidor web adaptativo é apresentado na Seção III. Os aspectos de implementação e implantação são apresentados na Seção IV. A Seção V apresenta um experimento realizado para analisar as possíveis melhorias na navegação do usuário. As discussões sobre o servidor web proposto neste artigo são apresentadas na Seção VI. As conclusões são apresentadas na Seção VII.

O autor Paulo Roberto Massa Cereda pode ser contactado através do seguinte e-mail: cereda@users.sf.net.

II. Navegação do Usuário na Internet

Com o advento e popularização da Internet, os usuários passaram a interagir de modo mais ativo em websites, buscando cada vez mais um conteúdo inerente aos seus interesses. Assim, a forma de navegação do usuário na Internet mobilizou companhias de comércio eletrônico e grupos de pesquisa acadêmica na procura de métodos que pudessem melhorar a experiência do usuário em um determinado website.

Um dos métodos de melhoria da experiência de navegação é a personalização de serviços. De modo geral, a personalização pode ser descrita como tornar algo pessoal, individual, dependente das características e dos interesses humanos. Ao personalizar um objeto de acordo com um usuário, cria-se uma relação de afinidade e interesse. Grande [1] afirma que "um produto ou serviço pode atender as necessidades fundamentais de uma pessoa por suas funcionalidades e características primárias. Além disso, um serviço, através da personalização, pode possuir determinadas características que o torna mais parecido com um indivíduo. Essas qualidades secundárias são consideradas tão importantes que em muitos casos a escolha do produto ou serviço é regida somente através delas." No âmbito do comércio eletrônico, a personalização é um recurso indispensável para tais websites.

Kimball e Merz [2] listam algumas funcionalidades de personalização, imprescindíveis para a melhoria da navegabilidade e experiência do usuário em um determinado website. Tais funcionalidades incluem: *reconhecimento de re-visitas*, que possibilita a identificação do usuário como um novo visitante ou um visitante recorrente; *interface de usuário e personalização de conteúdo*, que permite ao usuário ter vários tipos de menus de acesso rápido, de acordo com sua preferência; *vendas colaterais*, que são recomendações a partir de conhecimentos prévios obtidos do usuário; *vendas por impulso*, que são sugestões geralmente relacionadas a um determinado grupo de interesse, como por exemplo, se o usuário comprou um determinado livro em um website de comércio eletrônico, o website poderá relatar outros produtos comprados por outros usuários que também compraram o mesmo livro; *filtragem colaborativa ativa*, que são sugestões e

indicações de outros usuários; *eventos de calendário*, no qual o conteúdo de um website pode ser relacionado de acordo com eventos baseados em feriados ou datas importantes, como por exemplo, dia das Mães, Páscoa ou Natal; *eventos de estilo de vida*, no qual o conteúdo pode ser relacionado de acordo com situações comuns do dia-a-dia, por exemplo, nascimento de um filho, casamento ou formatura; e *localização*, que utiliza um conhecimento prévio acerca do usuário para personalizar um conteúdo de acordo com o seu idioma ou localização geográfica.

Para que a personalização exista, é necessário que exista a coleta de algumas informações do usuário. Quanto mais informações são obtidas, maior será a capacidade de oferecer personalização ao usuário. Entretanto, existem vários questionamentos sobre a privacidade do usuário e a exposição do mesmo; assim, quanto mais informações obtidas, menos privacidade esse usuário terá [3], [4]. A melhor opção, na maioria dos casos, é oferecer uma personalização mínima sem, entretanto, macular a privacidade do usuário [5].

Existem diversos mecanismos para coleta de dados na Internet com o intuito de oferecer personalização. Geralmente, são utilizadas análises de dados em formulários e sobre a navegação do usuário [3]. Com os dados obtidos, são aplicadas técnicas de mineração de dados para determinar preferências, tendências e outras informações relevantes.

Um dos mecanismos mais interessantes para a análise da navegação do usuário em um website é o *clickstream*. Também conhecido como seqüência de cliques ou *clickpath*, o *clickstream* representa o caminho que o usuário percorre enquanto está visitando um determinado website. A seqüência de cliques obtida, quando aplicada e analisada de forma correta, pode proporcionar um conjunto de informações muito valioso para empresas e organizações [3].

Os dados de *clickstream* também podem ser coletados por Provedores de Serviço de Internet (ISPs), painéis de acesso ou mesmo manualmente, apesar da coleta através dos arquivos de *log* dos servidores ser a mais comum [6].

O *clickstream* é muito importante, do ponto de vista comercial, pois é possível identificar as preferências e os padrões de comportamento do usuário; isso inclui qual área lhe interessa, a freqüência que a procura e quais as informações úteis para criar estratégias de marketing mais direcionadas e com maior chance de sucesso [7], [5].

Outra forma de melhorar a experiência de navegação é através dos chamados *sistemas de recomendação*. O *sistema de recomendação* é um método tradicional para retornar informações relevantes ao usuário. De um modo geral, um sistema de recomendação tenta identificar quais são os recursos mais importantes para um determinado usuário e recomendá-los [8], [9]. [10].

Apesar da existência de tais melhorias, a navegação na Internet também pode ser prejudicada de modo significativo. Com a disseminação de programas maliciosos através de websites suspeitos, o usuário sente-se ameaçado e, por muitas vezes, interrompe sua navegação [11]. Em uma escala de risco muito menor, porém muito mais freqüente, erros comuns de recursos não encontrados, *hyperlinks* inválidos ou servidores web indisponíveis comprometem a permanência do usuário em um determinado website e sua navegação em geral.
É importante que o usuário possa ter uma navegação eficiente e consistente, minimizando os erros e oferecendo subsídios para melhoria da qualidade da mesma.

III. Adaptatividade em Servidores Web

Um *servidor web* é um programa de computador responsável por atender requisições e retornar conteúdo utilizando o protocolo HTTP[6]. O objetivo principal de um servidor web é retornar páginas web e todo o conteúdo associado a elas (imagens, folhas de estilo, entre outros). Um cliente (um *browser*[7] ou um *crawler*[8]) faz uma requisição atráves do protocolo HTTP para o servidor, que recebe esta requisição, processa-a e retorna o conteúdo.

A quase totalidade das aplicações web busca fornecer para o usuário os subsídios necessários para personalização e recomendação de conteúdo e serviços. O servidor web tem uma participação passiva no processo, apenas servindo a seu propósito mais geral – responder requisições – e lançar erros quando estes acontecem.

A adição de uma camada adaptativa em servidores web justifica-se pelo fato de que a qualidade da navegação do usuário está intimamente relacionada com a qualidade das técnicas de personalização e recomendação presentes na aplicação web. Em outras palavras, uma aplicação web que não implementa tais técnicas – ou as implementa de modo precário – não pode oferecer uma experiência adequada ao usuário. Assim, propôs-se que o servidor web assumisse um papel mais ativo no processo, fornecendo técnicas de personalização e recomendação aos websites que estão hospedados em seu domínio, além de poder tratar de modo mais consistente e amigável todos os erros que eventualmente acontecem. Além disso, tal característica permite que as aplicações web sofram o mínimo de ajustes necessários para poder incorporar as funcionalidades oferecidas pelo servidor web adaptativo.

A. Autômato Adaptativo

O *autômato adaptativo*, proposto por Neto [12], é uma extensão do formalismo do autômato de pilha estruturado que permite o reconhecimento de linguagens do tipo 0, segundo a Hierarquia de Chomsky. O termo *adaptativo*, neste contexto, pode ser definido como a capacidade de um dispositivo em alterar seu próprio comportamento. Logo, um autômato adaptativo tem como característica a possibilidade de provocar alterações em sua própria topologia durante o processo de reconhecimento de uma dada cadeia [13].

Essa capacidade de alteração do autômato faz-se possível através da utilização de ações adaptativas, que podem ser executadas antes e/ou depois de uma transição. A cada

[6] HTTP é o acrônimo de *Hypertext Transfer Protocol*, ou Protocolo de Transferência de Hipertexto.
[7] *Browser* ou navegador é um programa de computador que permite ao usuário navegar na Internet. Os *browsers* mais conhecidos são *Internet Explorer* e *Mozilla Firefox*.
[8] *Crawler* é uma aplicação que navega automaticamente por páginas web, indexando seu conteúdo e seguindo seus *hiperlinks*.

execução de uma ação adaptativa, o autômato tem sua topologia alterada, obtendo-se uma nova configuração. O objetivo de uma ação adaptativa é lidar com situações esperadas, mas ainda não consideradas, detectadas na cadeia submetida para reconhecimento pelo autômato [14]. Uma transição pode ter ações adaptativas associadas, que permitam a inclusão ou eliminação de estados e transições.

Um *autômato adaptativo M* é definido por $M = (Q, S, \sum, \Gamma, P, q_0, Z_0, F)$, tal que Q é o conjunto finito de estados, $Q \subset Q^A$, Q^A é o conjunto de todos os estados possíveis, Q^A é enumerável, S é o conjunto finito de sub-máquinas, $\sum$ é o alfabeto de entrada, $\sum \subset \sum^A$, $\sum^A$ é o conjunto enumerável de todos os símbolos possíveis, Γ é o alfabeto da pilha, $\Gamma \subset \Gamma^A$, $\Gamma = Q \cup \{ Z_0 \}$, Γ^A é o conjunto enumerável de todos os símbolos possíveis da pilha, $\Gamma^A = Q^A \cup \{ Z_0 \}$, P é um mapeamento $P: Q^A \times \sum^A \times \Gamma^A \rightarrow Q^A \times (\sum^A \cup \{ \varepsilon \}) \times (\Gamma^A \cup \{ \varepsilon \}) \times H^0 \times H^0 \}$, H^0 definido a seguir, $q_0 \in Q$ é o estado inicial, $Z_0 \in \Gamma$ é o símbolo inicial da pilha, e $F \subset Q$ é o conjunto de estados finais. Os conjuntos "^{A}" ("*All*" – para todos) são convenientes porque as *funções adaptativas* podem *a)* inserir novos estados q, $q \in Q$ mas $q \in Q^A$, e *b)* usar novos símbolos de pilha $\gamma \in \Gamma$ mas $\gamma \in \Gamma^A$. Em resumo, as *funções adaptativas* podem modificar o autômato, mas os novos símbolos que elas introduzem estão todos nos conjuntos "^{A}" [15].

H^0 é o conjunto de todas as funções adaptativas no autômato adaptativo M. Define-se $H^0 = \{ f \mid f : E \times G_1 \times G_2 \times \ldots \times G_k \rightarrow E \}$, onde f é uma função, $k \in \mathbb{N}$ é o número de argumentos em f, e $G_i = Q^A \cup \sum^A \cup \Gamma^A$ [15].

E é o conjunto de todos os autômatos adaptativos que têm o estado inicial q_0, o símbolo inicial de pilha Z_0 e o conjunto de estados finais F iguais aos do autômato adaptativo M. Define-se $E = \{N \mid N$ é um autômato adaptativo $N = (Q', \sum', \Gamma', P', q_0, Z_0, F)$, onde $Q' \subset Q^A$, $\sum' \subset \sum^A$, $\Gamma' \subset \Gamma^A$, P': $Q^A \times \sum^A \times \Gamma^A \rightarrow Q^A \times (\sum^A \cup \{ \varepsilon \}) \times (\Gamma^A \cup \{ \varepsilon \}) \times H^0 \times H^0 \}$. Observe que q_0, Z_0 e F são os mesmos em qualquer $N \in E$ [15].

O conjunto de todas as sub-máquinas do autômato adaptativo M é representado por S. Cada *sub-máquina* s_i é definida como $s_i = (Q_i, \sum_i, P_i, q_{i0}, F_i)$, onde $Q_i \subseteq Q$ é o conjunto de estados da sub-máquina s_i, $\sum_i \subseteq \sum$ é o alfabeto de entrada da sub-máquina s_i, $P_i \subseteq P$ é o mapeamento da sub-máquina s_i, $q_{i0} \in Q_i$ é o estado inicial da sub-máquina s_i, e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina s_i.

A linguagem aceita por um autômato adaptativo M é dada por $L(M) = \{ w \in \sum^* \mid (q_0, w, Z_0) \vdash^* (q_f, \varepsilon, Z_0)$, onde $q_f \in F \}$. A capacidade de auto-modificação confere ao autômato adaptativo o poder computacional de uma Máquina de Turing [16].

Os autômatos adaptativos [12] [13] constituem um mecanismo formal para a descrição de linguagens recursivamente enumeráveis; sua simplicidade e facilidade de entendimento em relação ao modelo clássico de reconhecimento destas linguagens, a Máquina de Turing, fazem com que sua utilização seja muito ampla [17].

B. *Servidor Web Adaptativo*

Espera-se que um servidor web adaptativo contemple algumas funcionalidades desejadas para que ocorra uma melhoria significativa na experiência de navegação do usuário. Algumas destas funcionalidades incluem:

- a capacidade de oferecer personalização e recomendação de conteúdo ao usuário sem requerer modificações drásticas no código-fonte das aplicações web existentes;
- a verificação de erros de recursos não encontrados no servidor;
- a detecção de possíveis causas de erro e eventual tentativa de correção;
- a possibilidade de uma estrutura de navegação modificável ao longo do tempo; e
- a tentativa de gerenciamento eficiente de processos.

A Figura 1 ilustra uma possível arquitetura de um servidor web adaptativo. Essa arquitetura foi utilizada no desenvolvimento desta proposta.

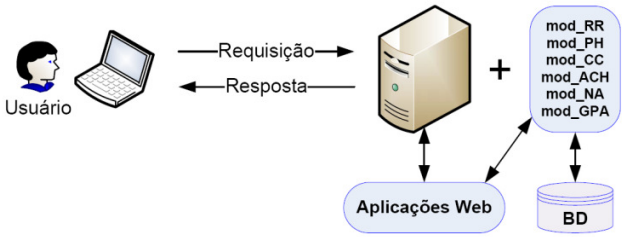


Figura 1 Uma possível arquitetura de um servidor web adaptativo.

De acordo com a Figura 1, espera-se que um servidor web adaptativo possua um conjunto de funções (módulos) incorporadas ao processo principal que proporcione as funcionalidades mencionadas anteriormente. Eventualmente, algumas destas funcionalidades podem requerer a persistência dos dados processados, justificando a utilização de um banco de dados ou algum arquivo de configuração disponibilizado. Além disso, algumas funcionalidades podem requerer também a alteração de alguns arquivos disponíveis no espaço das aplicações web. Deseja-se ainda que esse conjunto de funções esteja disponível de modo transparente ao usuário.

É importante que o servidor web adaptativo possa realizar tais funcionalidades de modo consistente e eficiente, para que a navegação do usuário não seja prejudicada, tanto em tempo de resposta quanto de qualidade de serviço.

Ao utilizar um servidor web com características de adaptatividade para oferecer melhorias na navegação do usuário, alguns questionamentos importantes surgem a partir dessa escolha. A Seção VII trata desse assunto, apresentando tais questionamentos em maiores detalhes.

IV. APACHE HTTPD ADAPTATIVO

O programa de computador *Apache HTTPD* é atualmente o servidor web mais comum existente, abrangendo em torno de 60% do mercado de servidores. Desenvolvido pela *Apache*

Software Foundation[9], possui seu código-fonte disponível livremente para download, na página do projeto[10].

O servidor Apache HTTPD foi projetado para oferecer um nível muito alto de flexibilidade. Uma de suas grandes características é a capacidade de permitir que usuários escrevam seus próprios módulos (também chamados de *Apache Modules*). Internamente, o Apache HTTPD é construído sobre módulos independentes projetados para realizar tarefas específicas. Em outras palavras, diversos módulos podem ser incorporados ao processo principal do servidor para estender suas funcionalidades. No geral, os módulos do Apache HTTPD abrangem suporte à CGI[11], controles de acesso, negociação de conteúdo, depuração, monitoramento, entre outros. Alguns módulos são incorporados por padrão; outros, entretanto, são adicionados à medida que são necessários [18].

A característica modular do Apache HTTPD mostrou-se adequada para um estudo sobre adaptatividade direcionada aos servidores web em diversos níveis de cobertura. Optou-se pelo desenvolvimento de módulos que utilizem elementos da tecnologia adaptativa e que possam contribuir para uma análise acerca da experiência de navegação do usuário. A Seção VII apresenta uma discussão sobre a implementação dessas funcionalidades sob a forma de módulos do Apache HTTPD.

Essa versão modificada do servidor Apache HTTPD recebeu o nome de *Apache HTTPD Adaptativo* e possui seis módulos disponíveis, a saber: recomendação de recursos, promoção de *hyperlinks*, criação de conteúdo, auto-correção de *hyperlinks*, navegação adaptativa e gerenciamento de processos adaptativo. Tais módulos são apresentados em detalhes a seguir.

A. *Módulo de Recomendação de Recursos*

O *Módulo de Recomendação de Recursos* (*mod_RR*) é responsável por verificar a existência de alguma relação entre os recursos disponíveis no servidor. Tais recursos podem ser representados por páginas web ou mesmo conteúdo multimídia, devidamente classificados e rotulados. O *mod_RR* utiliza o sistema *RecomAA* [10], [19] para recomendar os recursos disponíveis no servidor e apresentá-los sob a forma de *layers* durante a visualização de uma determinada página web.

O *RecomAA* atua da seguinte forma sobre os recursos disponíveis: dados dois recursos, o sistema verifica se existe alguma relação entre eles. Se a relação existe, o segundo recurso pode ser recomendado em função do primeiro e vice-versa. Por exemplo, dados os recursos *agenda* e *livro*, deseja-se saber se são relacionados; em outras palavras, o sistema responde *sim* ou *não* à seguinte pergunta: *dado o recurso agenda, é possível recomendar o recurso livro?*, ou *dado o recurso livro, é possível recomendar o recurso agenda?* [10].

Para auxiliar a execução do *mod_RR*, as páginas web recebem anotações especiais, inseridas em comentários no código-fonte, que são interpretadas em tempo de execução:

```
<!-- resource("Página inicial") -->
<!-- resource("Downloads") -->
```

Essas anotações especiais são submetidas ao *mod_RR* para verificação de relação entre possíveis recursos. Caso exista um recurso compatível com a recomendação, o servidor web adicionará um *layer* contendo a recomendação ao código-fonte da página em questão e retornará a requisição normalmente, conforme ilustra a Figura 2.

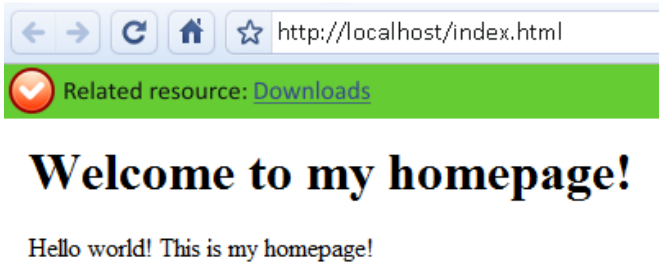


Figura 2 Exibição de uma recomendação.

O *mod_RR* também é responsável pelo monitoramento das escolhas do usuário em relação a um recurso recomendado. De acordo com a escolha do usuário (aceitar ou não uma determinada recomendação), o autômato adaptativo utilizado pelo *RecomAA* sofre alterações em sua topologia. Essas alterações buscam refletir de modo mais consistente as recomendações disponíveis no servidor web.

B. *Módulo de Promoção de Hyperlinks*

O *Módulo de Promoção de Hyperlinks* (*mod_PH*) é responsável pelo gerenciamento de *hyperlinks* em uma página web. O *mod_PH* avalia os *hyperlinks* disponíveis e registra o número de acessos a cada um. Eventualmente, os links mais acessados são promovidos, isto é, ascendem um nível hierárquico em relação aos demais. Para desempenhar tal tarefa, o módulo utiliza um autômato adaptativo, representado na Figura 3.

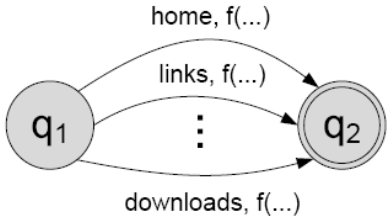


Figura 3 Autômato Adaptativo do Módulo de Promoção de *Hyperlinks*.

O autômato da Figura 3 representa a quantidade de acessos a cada *hyperlink* de uma determinada página web. Cada *hyperlink* α é representado como um símbolo na transição q_1 q_2, e a função adaptativa f é responsável por inserir um novo estado q_i entre q_1 e q_2, tal que (q_1, α) (q_i, ε) e (q_i, ε) (q_2, ε). Assim, a cada acesso ao *hyperlink* α adiciona-se um

[9] Página oficial: http://www.apache.org .
[10] Página oficial: http://httpd.apache.org .
[11] CGI é o acrônimo de *Common Gateway Interface*.

estado ao trecho do autômato que representa o número de acessos de α até o momento.

Após a atualização da configuração corrente do autômato adaptativo, o *mod_PH* realiza a contagem dos estados de cada trecho do autômato que representa um *hyperlink* e gera uma lista ordenada. Após a contagem, o *mod_PH* altera a ordem dos *hyperlinks* no código-fonte da página web de acordo com os números de acesso da lista obtida, do maior até o menor, e retorna a requisição ao usuário.

O *mod_PH* também faz uso de anotações especiais nas páginas web, inseridas em comentários no código-fonte, que são interpretadas em tempo de execução:

```
<!-- process-ranking -->
<a href="links.html">Links</a>

<!-- process-ranking -->
<a href="downloads.html">Downloads</a>

<!-- process-ranking -->
<a href="research.html">Research</a>
```

As anotações assinalam quais *hyperlinks* podem participar de promoções na página web. Esse tipo de marcação faz-se necessário devido ao fato de que alguns *hyperlinks* são imutáveis na página web, ou seja, não podem – e não devem – mudar de posição ao longo do tempo. Assim, optou-se por anotar explicitamente quais *hyperlinks* podem efetivamente sofrer promoções.

A Figura 4 exibe uma página web contendo uma lista de *hyperlinks* disponíveis.

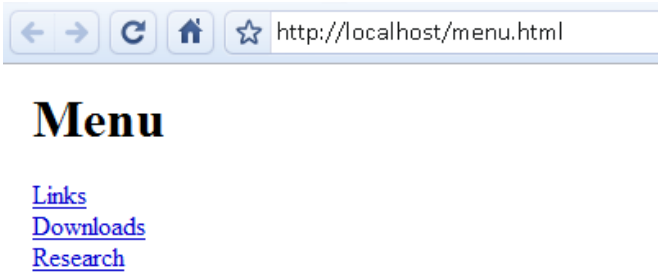


Figura 4 Página web contendo uma lista de *hyperlinks*.

Suponha que o *hyperlink Downloads* seja o mais acessado entre os três. A Figura 5 ilustra a promoção dos *hyperlinks* da página web, realizada pelo *mod_PH*.

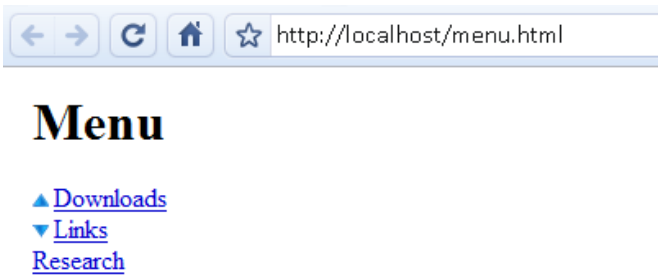


Figura 5 Página web contendo os *hyperlinks* já promovidos.

É importante observar que essa personalização de conteúdo baseada na promoção de *hyperlinks* não atua no código-fonte original da página web armazenada no servidor web, mas na instância a ser exibida em função de uma determinada requisição. O *mod_PH* encarrega-se de construir o autômato, analisá-lo e alterar o código-fonte da página web antes de enviá-la.

C. *Módulo de Criação de Conteúdo*

O *Módulo de Criação de Conteúdo* (*mod_CC*) é responsável pela criação de conteúdo no espaço de aplicações do servidor web. Em algumas situações, os websites apresentam *hyperlinks* inválidos (*broken hyperlinks*) para documentos, conteúdo multimídia ou mesmo para outras páginas web. Ao seguir um *hyperlink* que não pode ser resolvido, o servidor web lança o já conhecido *erro 404 – Não encontrado* (em inglês, *404 error – Not found*). O *mod_CC* analisa uma situação de erro 404 e verifica se o *hyperlink* procurado é acessado com freqüência. Em caso positivo, o módulo cria a referência não encontrada, tornando o *hyperlink* válido. O autômato adaptativo responsável pela tomada de decisão é apresentado na Figura 6.

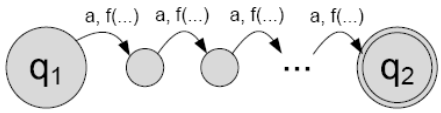


Figura 6 Autômato Adaptativo do Módulo de Criação de Conteúdo.

De acordo com a Figura 6, o autômato verifica se o recurso não encontrado *a* deve ou não ser criado. Entre os estados q_1 e q_2, existe um caminho de estados (representados por estados menores) que define a ocorrência da procura pelo *hyperlink* procurado e não encontrado *a*; à medida que tal recurso *a* é acessado, a função adaptativa *f* remove um dos estados intermediários, até que exista uma única transição q_1 q_2. Quando o estado final q_2 F é atingido, isto significa que recurso não encontrado *a* já possui um número de ocorrências suficientemente grande para que tal recurso seja criado e disponibilizado no servidor web.

O *mod_CC* avalia se o estado final q_2 do autômato adaptativo foi alcançado e, em caso positivo, cria o recurso e o disponibiliza no espaço de aplicações do servidor web. A Figura 7 ilustra uma situação de erro em relação à tentativa de exibição do recurso *new.html*.

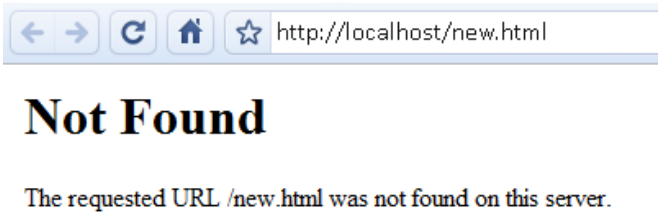


Figura 7 Erro lançado pelo servidor web durante a tentativa de acessar um recurso inexistente.

Suponha que a procura pelo recurso inexistente *new.html* tenha atingido um número considerável de ocorrências. Dessa forma, ao tentar novamente exibir tal recurso, o *mod_CC* verificará esse número de ocorrências e criará o recurso, conforme ilustra a Figura 8.

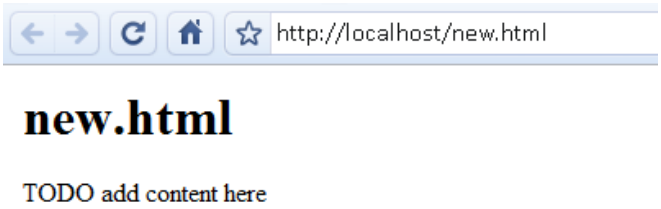


Figura 8 Recurso recém-criado pelo Módulo de Criação de Conteúdo.

No momento da escrita deste artigo, o *mod_CC* ainda possui algumas limitações. O conteúdo criado limita-se apenas às páginas web, sem nenhum contexto válido; as páginas web recém-criadas contêm apenas um texto informando que não há conteúdo efetivo definido. Espera-se que, no futuro, este módulo seja capaz de obter dados oriundos de diversas fontes e popular tais conteúdos com informações válidas e pertinentes ao domínio da aplicação. Testes preliminares com sistemas de *question answering* [20] mostraram-se promissores para popular os novos recursos; por exemplo, o recurso recém-criado *futebol.html* poderia gerar uma pergunta a um sistema de *question answering* – "*O que é futebol?*" – e este se encarregaria de buscar a melhor resposta e acrescentá-la como conteúdo válido ao novo recurso.

D. Módulo de Auto-Correção de Hyperlinks

O *Módulo de Auto-Correção de Hyperlinks* (*mod_ACH*) é responsável pela verificação dos *hyperlinks* e uma eventual auto-correção dos mesmos. Em algumas situações do cotidiano, o erro 404 não é proveniente apenas de referências não existentes, mas também de erros de digitação dos *hyperlinks*. Por exemplo, a palavra *downloads* pode ser grafada erroneamente de várias formas, entre elas, *downkloads* (proximidade das letras K e L no teclado), *downlaods* (inversão das letras O e A), e *dowloads* (ausência da letra N). Esse tipo de erro inevitavelmente lançará um erro de recurso não encontrado, apesar de não ser este o caso (grafia errada).

O *mod_ACH* analisa o *hyperlink* da requisição e verifica se existe algum erro de grafia. Em caso positivo, tenta identificar qual é o *hyperlink* correto mais provável para a requisição e o retorna para o cliente. Para cada *hyperlink* a ser analisado, o *mod_ACH* cria um autômato adaptativo correspondente. Por exemplo, o autômato adaptativo responsável pela identificação do *hyperlink downloads* é apresentado na Figura 9.

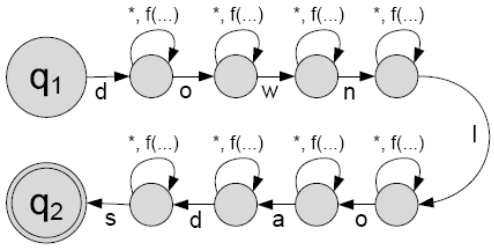


Figura 9 Autômato Adaptativo do Módulo de Auto-Correção de *Hyperlinks*, responsável pela identificação do *hyperlink downloads*.

O autômato da Figura 9 tenta identificar possíveis variações do *hyperlink downloads*. Caso a grafia seja diferente de *downloads*, o autômato tenta alterar sua topologia através da função adaptativa *f*. Se a cadeia for aceita, o *hyperlink* grafado erroneamente corresponde ao *hyperlink downloads*. O módulo *mod_ACH* então tratará a requisição como se fosse originalmente ao *hyperlink downloads*. Caso o autômato não aceite a cadeia, o erro 404 será lançado.

A Figura 10 apresenta a situação onde o *hyperlink downloads* foi grafado erroneamente (*downkloads*). O *mod_ACH* corrigiu o erro e informou o usuário acerca de seu erro, através de um *layer* adicionado ao código-fonte da página web.

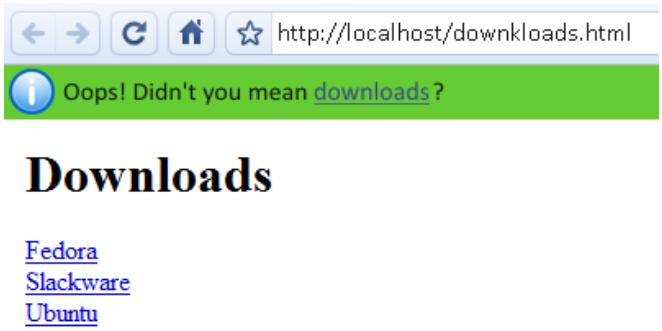


Figura 10 *Hyperlink* corrigido através do Módulo de Auto-Correção de *Hyperlinks*.

O *mod_ACH* pode ser utilizado tanto para verificação automática de *hyperlinks* com erros de grafia nas páginas web, quanto para a digitação manual dos mesmos na barra de endereços do navegador. Em caso de lançamento do erro 404, o Módulo de Criação de Conteúdo pode trabalhar conjuntamente para evitar problemas oriundos de recursos não encontrados ou mal grafados.

E. Módulo de Navegação Adaptativa

O *Módulo de Navegação Adaptativa* (*mod_NA*) é responsável pela análise da navegação dos usuários e eventual alteração do mapa de *hyperlinks*. Em outras palavras, a

navegação pode ser facilitada, reduzindo-se o número de cliques necessários até alcançar algum recurso em particular.

Um exemplo de uma instância do autômato adaptativo do módulo *mod_NA* é apresentado na Figura 11. As possíveis configurações do autômato refletem os diversos mapas de navegação possíveis.

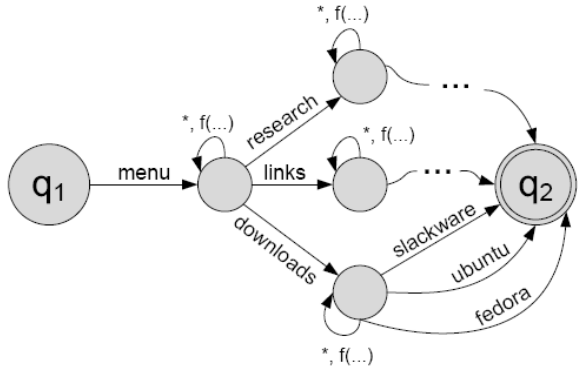


Figura 11 Autômato Adaptativo do Módulo de Navegação Adaptativa.

De modo simplificado, o autômato da Figura 11 pode ser analisado como um mapa de navegação por *hyperlinks*, onde cada caminho possível representa a ordem de cliques até atingir um determinado recurso. A função adaptativa *f* é responsável por encurtar um caminho, de acordo com a ocorrência de caminhos similares obtidos a partir de um histórico de navegação.

Por exemplo, suponha que o *hyperlink Fedora*[12] seja um dos mais acessados, e que o caminho *Página inicial* *Menu* *Downloads* *Fedora* possua um grande número de ocorrências. O módulo *mod_NA* poderá promover o *hyperlink Fedora*, reduzindo em um nível o caminho até ele. Dessa forma, o caminho mais curto passará a ser *Página inicial* *Menu* *Fedora*, conforme ilustrado na Figura 12.

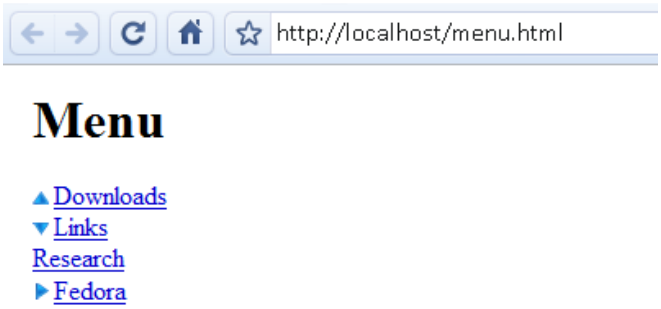


Figura 12 Promoção do *hyperlink Fedora*, reduzindo em um nível o caminho até ele.

Da mesma forma que o Módulo de Promoção de *Hyperlinks* (*mod_PH*), o *mod_NA* também faz uso de anotações especiais nas páginas web, inseridas em comentários no código-fonte, que são interpretadas em tempo de execução:

```
<!-- process-browsing -->
<a href="fedora.iso">Fedora</a>

<!-- process-browsing -->
<a href="slackware.iso">Slackware</a>

<!-- process-browsing -->
<a href="ubuntu.iso">Ubuntu</a>
```

As anotações assinalam quais *hyperlinks* podem participar de alterações na navegação. Esse tipo de marcação faz-se necessário devido ao fato de que alguns *hyperlinks* são imutáveis na página web, ou seja, não podem – e não devem – mudar de posição ao longo do tempo. Assim, optou-se por anotar explicitamente quais *hyperlinks* podem efetivamente sofrer alterações na navegação.

A personalização de conteúdo sob o aspecto de uma navegação adaptativa não atua no código-fonte original da página web armazenada no servidor web, mas na instância a ser exibida em função de uma determinada requisição. O *mod_NA* encarrega-se de construir o autômato, analisá-lo e alterar o código-fonte da página web antes de enviá-la.

F. Módulo de Gerenciamento de Processos Adaptativo

O servidor Apache HTTPD utiliza um *pool* de processos em *prefork* para reduzir o atraso de tempo e custos que são associados à criação de novos processos. Existe um processo principal que monitora a combinação de portas e *sockets* nas quais as requisições via TCP/IP são recebidas. O processo principal nunca manipula tais requisições, mas as distribui para os processos subordinados. Cada um desses processos atua como um servidor em série, tratando cada cliente por vez. Quando um processo termina seu trabalho, volta para o *pool* [18].

Além de ser responsável pela distribuição de carga, o processo principal também é responsável pelo ajuste do número de processos subordinados. Esse ajuste é extremamente importante, pois poucos processos subordinados atrasam o atendimento aos clientes, enquanto que muitos processos desperdiçam recursos do sistema.

O *Módulo de Gerenciamento de Processos Adaptativo* (*mod_GPA*) tem como objetivo utilizar um autômato adaptativo para o gerenciamento do *pool* de processos. Este módulo ainda é experimental e apresenta alguns desafios de implementação.

O autômato adaptativo utilizado pelo *mod_GPA* é apresentado na Figura 13.

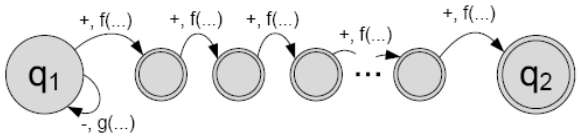


Figura 13 Autômato Adaptativo do Módulo de Gerenciamento de Processos Adaptativo.

[12] *Fedora* é uma distribuição Linux muito popular para *desktop*. Sua página oficial é: http://www.fedoraproject.org .

De modo simplificado, o autômato adaptativo do *mod_GPA* (Figura 13) tem o mesmo comportamento do processo principal do Apache HTTPD, no qual distribui as requisições para os processos subordinados (função adaptativa *f*, removendo estados intermediários) e gerencia os processos que retornam ao *pool* (função adaptativa *g*, inserindo estados intermediários). Os estados intermediários do autômato representam os processos subordinados disponíveis.

No momento da escrita deste artigo, o *mod_GPA* ainda possui limitações. Espera-se que, no futuro, este módulo seja capaz de gerenciar plenamente o *pool* de processos de modo consistente e eficiente.

V. Implementação e Implantação

Para a implementação de um servidor web adaptativo, foi utilizado o Apache HTTPD em sua versão 1.3.41[13]. As funcionalidades apresentadas na seção anterior foram escritas em C como módulos do Apache e carregadas junto ao servidor web durante sua inicialização.

De acordo com a proposta de arquitetura para o servidor web adaptativo apresentada na Seção III-B (Figura 1), seria interessante utilizar um banco de dados para realizar a persistência dos dados obtidos e tratados pelos módulos. Como primeiro protótipo, optou-se pela utilização do *engine* de banco de dados *SQLite*[14], devido à sua simplicidade.

O servidor web adaptativo foi implantado em um computador dedicado e populado com uma quantidade considerável de páginas web, documentos e conteúdo multimídia. O computador em questão foi utilizado exclusivamente para atender requisições.

VI. Experimento e Análise

Para verificar as funcionalidades do servidor web adaptativo, foi realizado um experimento com 30 alunos de graduação e pós-graduação de instituições de ensino superior da região de São Carlos. A primeira parte do experimento consistiu na definição das tarefas que cada aluno realizaria durante o período de quatro semanas. A Tabela I apresenta as tarefas definidas.

Tabela 1
TAREFAS DEFINIDAS PARA O EXPERIMENTO

Índice	Descrição da tarefa
t_1	Navegar aleatoriamente pelos websites
t_2	Verificar os *hyperlinks* disponíveis
t_3	Procurar por determinado conteúdo disponível
t_4	Procurar por conteúdo inexistente
t_5	Errar o nome de *hyperlinks* propositalmente
t_6	Realizar diversas requisições simultaneamente
t_7	Aceitar ou não as recomendações sugeridas
t_8	Acessar conteúdo multimídia disponível

Cada aluno poderia realizar qualquer tarefa apresentada na Tabela I quantas vezes desejasse, e sem uma ordem definida. A única exigência do experimento foi que todas as tarefas apresentadas fossem realizadas.

A segunda parte do experimento consistiu na interação dos alunos com o ambiente. O experimento durou quatro semanas, com tarefas diárias realizadas pelos alunos. Dois servidores contendo as mesmas aplicações web foram disponibilizados: o Apache HTTPD convencional e sua versão adaptativa. Os 30 alunos participantes foram divididos em dois grupos, e cada grupo interagiu com um dos servidores.

Durante todo o experimento, ao longo de quatro semanas, a navegação de cada aluno e o comportamento de cada servidor foram monitorados. A partir destas informações, os resultados foram obtidos.

A Figura 14 ilustra o número de ocorrências do erro 404 lançado pelos servidores durante as quatro semanas do experimento.

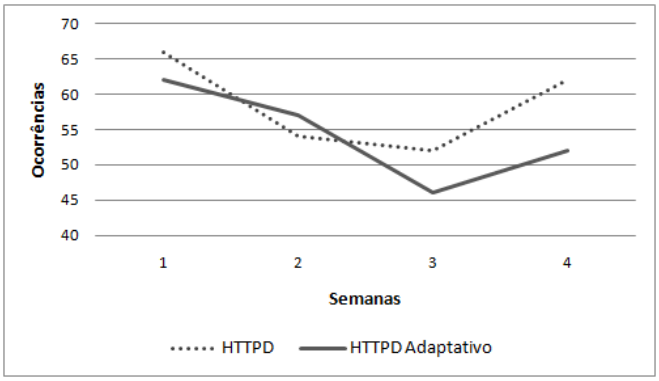


Figura 14 Número de ocorrências do erro 404 lançado pelos servidores durante as quatro semanas do experimento.

Algumas das tarefas consistiam em gerar propositalmente erros 404 (tarefas t_4 e t_5, Tabela I). De acordo com a Figura 14, ao longo das quatro semanas, o servidor web adaptativo apresentou um número relativamente menor de erros 404. Essa diminuição de erros deve-se aos módulos que tratam da criação de conteúdo e da auto-correção de *hyperlinks* (*mod_CC* e *mod_ACH*, respectivamente). Tais módulos minimizam alguns problemas inerentes à navegação dos usuários.

A Figura 15 ilustra o total de cliques processados pelos servidores durante as quatro semanas do experimento.

[13] Nota de lançamento disponível em http://tinyurl.com/rdvtg .
[14] Página oficial disponível em http://www.sqlite.org .

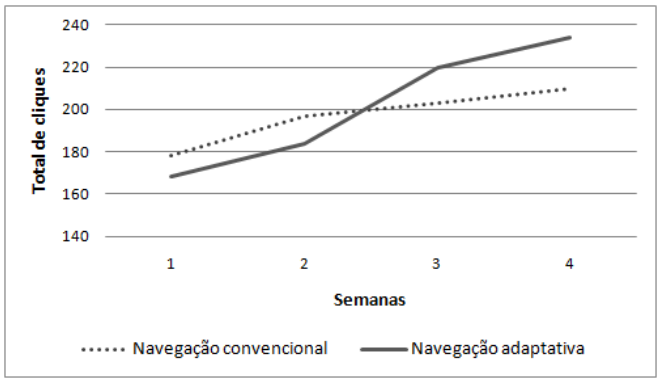


Figura 15 Total de cliques processados pelos servidores durante as quatro semanas do experimento.

De acordo com a Figura 15, é possível observar que, ao longo do tempo, a navegação adaptativa proporcionou um número maior de cliques nos *hyperlinks* disponíveis nas páginas. O módulo de navegação adaptativa (*mod_NA*) proporcionou uma navegação mais inerente aos interesses dos usuários durante o acesso ao conteúdo disponível.

A Figura 16 apresenta o total de cliques dos *hyperlinks*, promovidos (na versão adaptativa) ou não (na versão convencional), durante as quatro semanas do experimento.

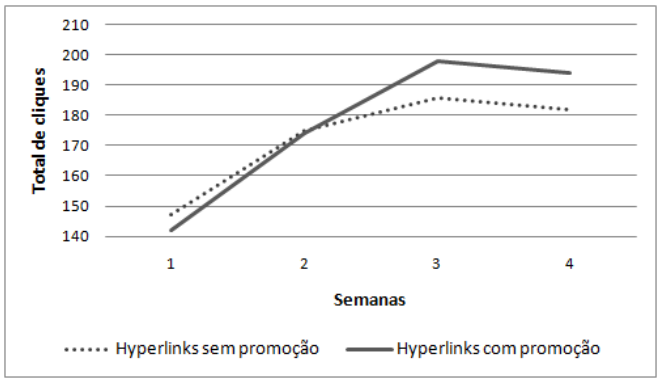


Figura 16 Total de cliques dos *hyperlinks* promovidos durante as quatro semanas do experimento.

É possível observar, de acordo com a Figura 16, que os *hyperlinks* promovidos proporcionaram um número maior de cliques do que seus equivalentes não promovidos. O módulo de promoção de *hyperlinks* (*mod_PH*) proporcionou ao usuário os *hyperlinks* mais relevantes à sua navegação, aumentando portanto o número de cliques.

A Figura 17 apresenta o total de cliques dos recursos recomendados durante as quatro semanas do experimento.

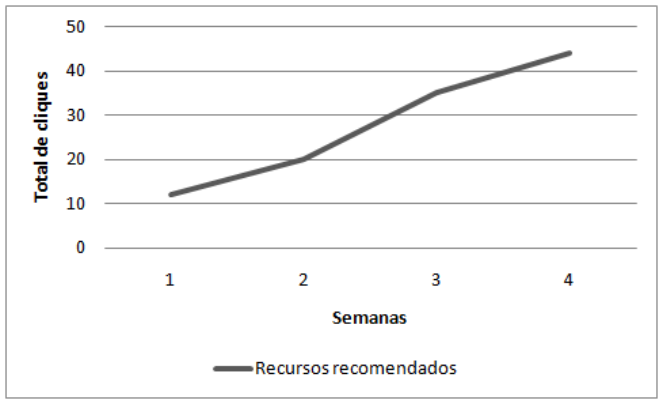


Figura 17 Total de cliques dos recursos recomendados durante as quatro semanas do experimento.

De acordo com a Figura 17, o total de cliques dos recursos recomendados aumentou ao longo das quatro semanas do experimento, ilustrando a importância da recomendação de recursos para os usuários. Com o tempo, as recomendações tornaram-se mais consistentes, o que justifica o aumento mais significativo nas duas últimas semanas. O módulo de recomendação de recursos (*mod_RR*) proporcionou uma navegação direcionada aos interesses do usuário, apresentando recursos semelhantes e sugerindo sua visualização.

VII. DISCUSSÕES

Esta seção trata das discussões acerca da utilização de um servidor web com características de adaptatividade para oferecer melhorias na navegação do usuário. A seguir, são apresentados alguns questionamentos.

A motivação para a proposta de um servidor web adaptativo iniciou-se com uma análise do comportamento do usuário ao navegar na Internet e as implicações no ambiente web. Constatou-se que a utilização de serviços personalizados e um tratamento mais adequado aos erros inerentes à navegação, da forma como são tratados atualmente, estão intimamente relacionados com as técnicas de implementação e suporte da aplicação web disponível. Em outras palavras, a qualidade da navegação do usuário está relacionada com a qualidade do serviço disponibilizado na aplicação web.

Na tentativa de remover a dependência de serviços de personalização e recomendação da aplicação web, propôs-se que o servidor web assumisse essa tarefa. Apesar de remover essa dependência da aplicação web, surgiram questionamentos referentes à viabilidade do servidor web como componente para tratar dos requisitos apresentados na Seção III-B. É importante destacar que o Apache HTTPD, mesmo tendo como objetivo principal e função básica atender requisições, devida à sua arquitetura modular, permite que tais requisições possam ser tratadas nos mais diversos níveis de abstração, desde a manipulação de pacotes de rede até interpretação de arquivos disponíveis no espaço de aplicações web. Essa possibilidade de tratamento em vários níveis motivou a utilização do servidor web para tratar das características de personalização e recomendação das aplicações web hospedadas em seu domínio.

O desenvolvimento de módulos Apache se mostrou interessante porque permitiu que as funcionalidades desejadas

fossem disponibilizadas de forma consistente no servidor web. O Apache HTTPD invoca os módulos em vários momentos durante a configuração e em vários pontos durante o processamento da requisição, mas apenas nas atividades nas quais os módulos tenham sido registrados. Por exemplo, o Apache HTTPD permite que um módulo atue na fase de manipulação de conteúdo (também conhecida como *content handling*), utilizando atributos de geração ou de transformação. Essa característica de gerenciamento permitiu que cada módulo atuasse em seu devido momento e realizasse sua tarefa específica.

Os resultados obtidos a partir do experimento apresentado na Seção VI são importantes, pois indicam que o servidor web adaptativo proposto neste artigo realmente ofereceu uma melhoria significativa na experiência de navegação do usuário. Entretanto, ainda não foi possível realizar testes de desempenho do servidor web nas mais diversas situações – aumento de demanda, latência de rede, latência de módulos, entre outros – e verificar seu comportamento. Tais testes podem permitir uma análise mais detalhada e fornecer subsídios para otimizações no modelo arquitetural. A obtenção destas métricas poderá ratificar o servidor web adaptativo como uma opção viável para melhoria da navegação do usuário.

VIII. Conclusões

Este artigo apresentou uma proposta de um servidor web adaptativo que utiliza autômatos adaptativos em seus módulos de controle. O autômato adaptativo, devido à característica de auto-modificação, é uma escolha interessante para auxiliar na tarefas de personalização, recomendação e otimização.

A adaptatividade em servidores web pode tornar-se uma tendência muito interessante, pois ao mesmo tempo em que fornece uma experiência nova de navegação ao usuário, também torna o processo de disponibilização e atualização de conteúdo já existente quase que transparente para o administrador do servidor. A utilização da tecnologia adaptativa na área de servidores web proporciona soluções consistentes que atendem às necessidades inerentes a esse domínio de aplicação. Além disso, o usuário beneficia-se com um conteúdo mais direcionado aos seus interesses.

Agradecimentos

O autor agradece ao prof. José de Oliveira Guimarães, do Departamento de Computação da Universidade Federal de São Carlos, *campus* de Sorocaba, pelo auxílio na revisão deste artigo.

Referências

[1] R. E. Grande, "*Sistema de integração de técnicas de proteção de privacidade que permitem personalização*", Dissertação de Mestrado, Departamento de Computação, Universidade Federal de São Carlos, 2006.

[15] R. Kimball and R. Merz, *Data Webhouse: construindo o data warehouse para a Web*. Rio de Janeiro: Campus, 2000.

[16] S. D. Zorzo, R. A. Gotardo, P. R. M. Cereda, B. Y. L. Kimura, R. A. Rios and R. E. Grande, "Web privacy controlled by user: an approach to treat the user's preferences about personal data". In *European Computing Conference 2007*, 2007.

[17] S. D. Zorzo and P. R. M. Cereda, "*Fatores de privacidade e confiança em websites*," Revista de Computação e Tecnologia da PUC-SP, vol. 1, pp. 35–43, 2009.

[18] P. R. M. Cereda and S. D. Zorzo, "Formalismo com autômato adaptativo em mecanismo de privacidade e personalização," in *XXXIII Conferencia LatinoAmericana en Informatica*, vol. 1, San Jose, Costa Rica, 2007.

[19] A. L. Montgomery, "*Using clickstream data to predict www usage*", University of Maryland, Tech. Report, 2003.

[20] A. L. B. Nogueira and L. R. Oliveira Jr., *Uma análise da aplicabilidade de Data Warehouse em ambientes empresariais*, Faculdade Ruy Barbosa, Salvador, 2004.

[21] R. A. Gotardo, C. A. C. Teixeira and S. D. Zorzo, "Predicting user's interests in web-based educational systems using a collaborative filtering weighted method". In *11th IEEE International Conference on Computational Science and Engineering – Workshop on Computational Science and Engineering – CSE 2008*, 2008.

[22] S. Sae-Tang and V. Esichaikul, "Web personalization techniques for e-commerce". In *Active Media Technology: 6th International Computer Science Conference, 2005.*

[23] P. R. M. Cereda, R. A. Gotardo, and S. D. Zorzo, "Resource recommendation using adaptive automaton," in *16th International Workshop on Systems, Signals and Image Processing*, Chalkida, Greece, 2009.

[24] W. Aiello and P. McDaniel, *Lecture 1, Intro: Privacy*, Stern School of Business, NYU, 2004.

[25] J. J. Neto, "*Contribuições à metodologia de construção de compiladores*", Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[26] _____, "Adaptive automata for context-dependent languages", *SIGPLAN Notices*, vol. 29, no. 9, PP. 115-124, 1994.

[27] _____, "Solving complex problems efficiently with adaptive automata," in *CIAA '00: Revised Papers from the 5th International Conference on Implementation and Application of Automata*. London, UK: Springer-Verlag, 2001, pp. 340–342.

[28] P. R. M. Cereda, "*Modelo de controle de acesso adaptativo*", Dissertação de Mestrado, Departamento de Computação, Universidade Federal de São Carlos, 2008.

[29] R. L. A. Rocha and J. J. Neto, "Autômato adaptativo, limites e complexidade em comparação com máquina de Turing". In *Proceedings of the Second Congress of Logic Applied to Technology – LAPTEC*. São Paulo, Brasil: Faculdade SENAC de Ciências Exatas e Tecnologia, 2001.

[30] J. J. Neto, "*Um levantamento da evolução da adaptatividade e da tecnologia adaptativa*," IEEE Latin America Transactions, vol. 5, no. Num. 7, pp. 496–505, Novembro 2007.

[31] N. Gray, *Web Server Programming*. Wiley, 2003.

[32] P. R. M. Cereda, R. A. Gotardo, and S. D. Zorzo, "Recomendação de recursos utilizando autômato adaptativo," in *Terceiro Workshop de Tecnologia Adaptativa*, São Paulo, Brasil, 2009.

[33] Lehnert, *The Process of Question Answering*. Erlbaum, 1978.

Paulo Roberto Massa Cereda obteve o mestrado em Ciência da Computação pela Universidade Federal de São Carlos. Tem experiência na área de Ciência da Computação, atuando nas áreas de Inteligência Artificial, Privacidade e Personalização de Serviços, Teoria da Computação e Tecnologia Adaptativa.

Controle de Concorrência Reconfigurável para o Ambiente Móvel

A. G. de A. L. Pierre, M. B. F. de Toledo, T. da Rocha

***Resumo*— Dentre as tecnologias emergentes, a computação móvel tem a sua posição de destaque. Apesar de atrativa, existe a limitação de recursos e nesse contexto, as transações representam um papel importante. Diversos modelos de transações têm sido apresentados na literatura, mas apesar de muitas idéias interessantes, alguns modelos não permitem que a adaptação seja realizada durante a execução de uma transação e quando permitem, eles realizam grandes reconfigurações arquiteturais. Motivado por essas questões, esse artigo propõe a configuração do mecanismo de controle de concorrência, no início da transação, e a reconfiguração da propriedade transacional, nível de isolamento, durante a execução da transação. O modelo de componentes OpenCOM foi utilizado para o desenvolvimento de um protótipo de um sistema de vendas. A configuração do controle de concorrência é uma contribuição inovadora desse artigo, pois em muitos trabalhos existentes não é possível a configuração desse mecanismo transacional.**

***Palavras chave*— Computação móvel (*Mobile computing*), sistemas de computação adaptativos *(adaptive computing systems*), reconfiguração dinâmica (*dynamic reconfiguration*), OpenCOM, controle de concorrência (*concurrency control*), nível de isolamento (*isolation level*).**

I. INTRODUÇÃO

Os dispositivos móveis estão cada vez mais presentes na vida das pessoas e contendo aplicações mais sofisticadas e semelhantes às executadas em computadores pessoais.

Um dos exemplos mais recentes é o surgimento da plataforma aberta para dispositivos móveis, chamada Android [1], criada pela empresa Google que promete revolucionar o desenvolvimento de aplicações nesse ambiente. Outro exemplo é o uso crescente da internet em celulares, permitindo que os usuários acessem diversos tipos de aplicações, tendo grande parte delas interação com banco de dados.

A computação móvel traz desafios ao desenvolvedor [2], pois ele deve considerar os recursos limitados tais como largura de banda, conectividade e o alto custo da obtenção de dados.

Nesse contexto, as transações representam um importante papel de garantir que o dinamismo do ambiente da computação móvel não comprometa a confiabilidade das aplicações. Porém, algumas aplicações não podem ser implementadas considerando o modelo de transações tradicional, pois ele não foi projetado para lidar com obstáculos de ambientes móveis – desconexões inesperadas, atrasos na comunicação e mobilidade.

Diante das limitações do modelo tradicional, diversos modelos de transações têm sido propostos na literatura. Alguns exemplos são: MobileTrans [3], CATE [4] , AMT [5], ReflecTS [6] e SGTA [7]. Esses modelos permitem que transações sejam personalizadas de acordo com os requisitos das aplicações e com as características específicas do ambiente móvel em questão. Porém, eles não permitem que essa adaptação seja realizada durante a execução da transação – característica essencial para lidar com aspectos dinâmicos do ambiente móvel.

Nesse contexto, esse artigo propõe uma arquitetura adaptável para o mecanismo de controle de concorrência de transações voltadas ao ambiente móvel. Esta arquitetura permite tanto configuração estática – para atender requisitos das aplicações – quanto dinâmica – para se adaptar às variações dinâmicas nos recursos do ambiente. O nível de isolamento é a propriedade transacional que poderá ser reconfigurada durante a transação e o controle de concorrência é o mecanismo que garantirá o isolamento entre as transações e poderá ser configurado no início da transação.

Um modelo de componentes denominado OpenCOM [8] foi o escolhido para a implementação da arquitetura proposta. Como esse modelo utiliza a técnica de reflexão computacional [9], ele provê uma facilidade de adaptação da arquitetura dos componentes diante de certas mudanças do ambiente.

A fim de verificar a arquitetura proposta, um protótipo de um sistema de vendas foi desenvolvido.

O artigo está organizado conforme descrito: A seção II apresenta a arquitetura proposta de controle de concorrência reconfigurável para dispositivos móveis. A relação entre a computação reconfigurável e a tecnologia adaptativa é mostrada na seção III. A implementação do protótipo e dos estudos de casos são apresentados na seção IV e a seção V termina o artigo apresentando as conclusões.

II. ARQUITETURA PROPOSTA

Considerando que aplicações que utilizam serviços de transações podem possuir requisitos distintos, este artigo apresenta uma nova arquitetura capaz de ser configurada com um controle de concorrência pessimista ou com um controle de concorrência otimista.

Além de permitir a configuração do tipo de controle de concorrência desejado, esta arquitetura foi projetada para prover mecanismos de adaptação dinâmicos para transações. Neste sentido, esta arquitetura permite que uma transação

Esse artigo é parte da dissertação de defesa de Mestrado da aluna A. G. de A. L. Pierre orientada pela professora M. B. F. de Toledo, do Instituto de Computação da UNICAMP e com contribuição do professor T. da Rocha.

mude o seu nível de isolamento dinamicamente como reação às mudanças ocorridas em tempo de execução.

Desta forma, optou-se por permitir que a troca de controle de concorrência ocorra somente de forma estática, ou seja, antes do processo de execução de transações. A possibilidade de reconfiguração dinâmica foi destinada somente ao âmbito das propriedades transacionais como, por exemplo, os níveis de isolamento que causam menos mudanças estruturais.

A arquitetura de serviços de transações distribuídos normalmente é estruturado em duas partes: (i) uma parte cliente - localizado na máquina onde a aplicação cliente executa; (ii) uma parte servidor – localizada em cada ponto da rede onde a transação cliente acessa dados transacionais remotos.

A Fig. 1 mostra a arquitetura do lado cliente proposta neste artigo enquanto a Fig. 2 mostra a arquitetura do lado servidor.

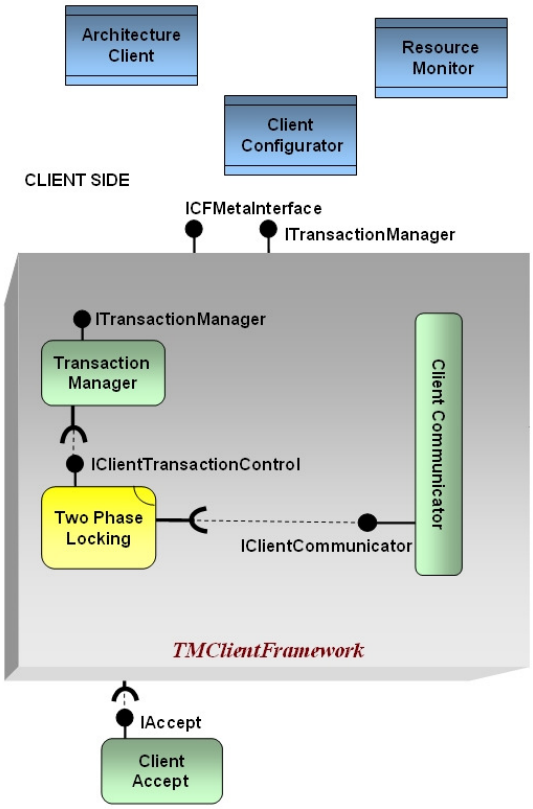


Fig. 1. Arquitetura proposta – lado Cliente

As convenções utilizadas para representar o desenho arquitetural são descritas a seguir:

 representa um framework de componentes [10] que é definido no modelo OpenCOM como uma sub-arquitetura de componentes sobre a qual pode ser verificado um conjunto de propriedades arquiteturais desejadas. Essa sub-arquitetura também pode ser vista como um componente complexo que é composto por outros componentes interconectados.

A função do framework de componentes é impor restrições ao processo de configuração e reconfiguração da plataforma de forma que a integridade da mesma possa ser mantida.

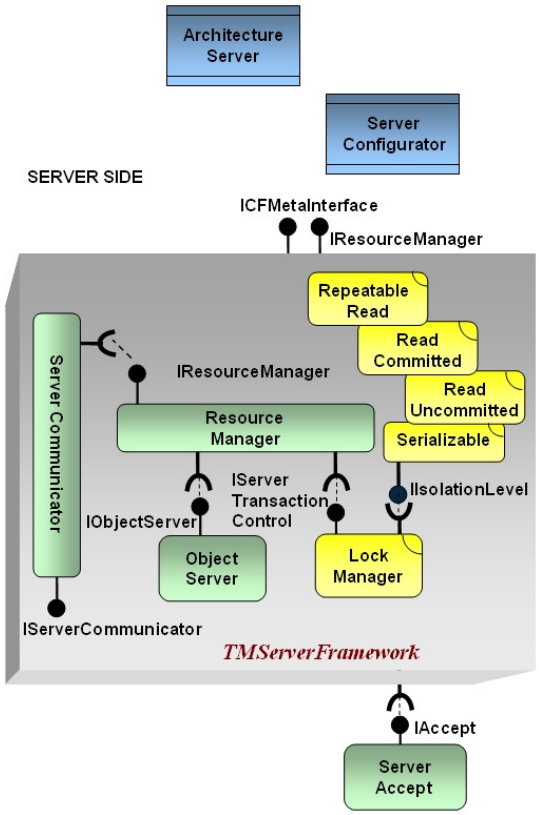


Fig. 2. Arquitetura proposta – lado Servidor

 representa uma interface: serviços providos pelo componente. Uma interface estabelece uma espécie de contrato que é obedecido por uma classe. Ela só expõe o que uma classe que a implementa deve fazer. Ela não possui atributos, nem métodos com implementação. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface.

 representa um receptáculo: serviços requeridos por um componente. Eles podem ser simples ou múltiplos. Um receptáculo simples só aceita a conexão de uma interface por vez. Já o receptáculo múltiplo permite a conexão de mais de uma interface ao mesmo tempo.

 representa uma classe para um conjunto de objetos com características comuns. Ela possui atributos e métodos. É necessário instanciá-la para criar os objetos.

 representa um componente OpenCOM. Estes são componentes fixos da arquitetura, ou seja, não podem ser mudados no processo de reconfiguração.

 também representa um componente OpenCOM.

Esses componentes estão destacados na arquitetura porque podem ser substituídos dinamicamente conforme o tipo de controle de concorrência e nível de isolamento escolhidos.

representa a conexão entre uma interface e um receptáculo. Para um que componente Y utilize serviços de outro componente Z, um receptáculo de Y precisa ser conectado a uma interface de Z.

Na arquitetura proposta foram considerados os controles de concorrência pessimista e otimista [11], [12]. Basicamente, um controle de concorrência é chamado pessimista quando ele evita previamente que ocorram conflitos entre transações em execução. Para isso ele normalmente impõe restrições de acesso aos dados às transações. Já um controle otimista, não impõe restrições de acesso aos dados durante a execução das transações, pois os conflitos só são detectados e resolvidos no final da execução das transações concorrentes. Controles pessimistas normalmente são baseados em trancas, enquanto que os otimistas são baseados em controle de versões sobre os dados acessados. Os componentes *TwoPhaseLocking*, *LockManager*, *ReadUncommitted*, *ReadCommitted*, *Repeatable Read* e *Serializable* fazem parte do controle de concorrência pessimista, conforme mostrado anteriormente nas Fig. 1 e Fig. 2.

No controle de concorrência otimista, no lado Cliente, o componente *TwoPhaseLocking* é substituído pelo componente *VersionControl*. Já no lado servidor, o componente *LockManager* é substituído pelo componente *VersionManager*. Os componentes que implementam os níveis de isolamento *Read Uncommitted*, *ReadCommitted*, *Repeatable Read* e *Serializable* foram incluídos somente na arquitetura para o controle de concorrência pessimista. Os componentes que são alterados para o controle otimista estão destacados nas Fig. 3 e Fig. 4.

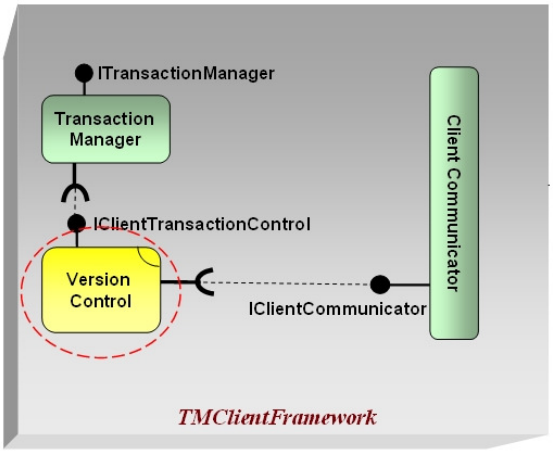


Fig. 3. Controle otimista – lado Cliente

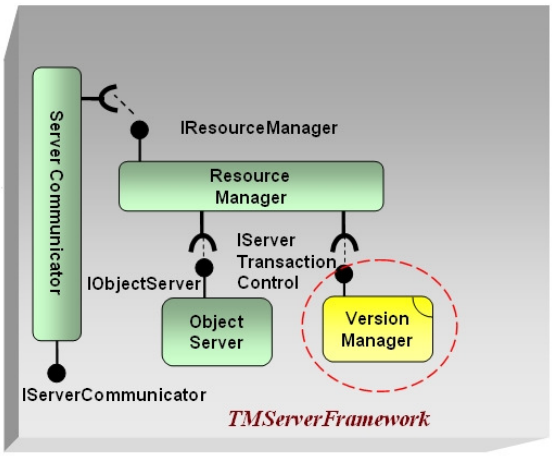


Fig. 4. Controle otimista – lado Servidor

A seguir, cada elemento da arquitetura será detalhado dentro do contexto arquitetural (eles estão listados em ordem alfabética):

- ***ArchitectureClient*** e ***ArchitectureServer***: montam a arquitetura OpenCOM (definindo os controles de concorrência e níveis de isolamento) respectivamente do lado cliente e servidor, fazendo a ligação entre os componentes.

- ***ClientAccept*** e ***ServerAccept***: componentes que implementam as regras capazes de impor as propriedades arquiteturais desejadas para a arquitetura interna dos frameworks *TMClientFramework* e *TMServerFramework*. Para evitar que estas configurações e reconfigurações comprometam a integridade da plataforma, cada mudança realizada é comparada a um conjunto de arquiteturas válidas pré-definidas. As operações de inspeção dos frameworks incluem: a recuperação de componentes internos, a recuperação da lista de conexões de um dado componente e a recuperação da lista de interfaces e da lista de receptáculos de componentes internos.

- ***ClientCommunicator***: componente responsável pela comunicação com o lado servidor através de mensagens via RMI (*Remote Method Invocation*)[13]. Implementa a interface *IClientCommunicator*.

- ***ClientConfigurator***: classe que implementa a política de adaptação do lado cliente, como por exemplo, o que acontece com o controle de concorrência e com o nível de isolamento se um determinado recurso do dispositivo móvel está limitado. Ela se comunica com a classe *ServerConfigurator* através da tecnologia RMI para notificar que uma mudança ocorreu na arquitetura do lado cliente e a mesma deve ser realizada no lado servidor.

- ***ServerCommunicator***: responsável pela comunicação com o lado cliente, através de mensagens via RMI. Implementa a interface *IServerCommunicator*.

- ***ServerConfigurator***: classe que define o controle de concorrência e o nível de isolamento do lado servidor especificados no início da aplicação ou após a notificação da classe *ClientConfigurator*.

- ***TMClientFramework***: framework de componentes OpenCOM do lado cliente.

- ***TMServerFramework*:** framework de componentes OpenCOM do lado servidor.

Os componentes e as classes que implementam o gerenciamento de transações são descritos a seguir:

- ***ObjectServer***: componente que representa o repositório de objetos acessados pelas transações.

- ***ResourceManager***: componente que gerencia o recurso (*ObjectServer*) executando as operações necessárias (inclusão, atualização e consulta).

- ***ResourceMonitor***: classe responsável por monitorar os recursos do ambiente do dispositivo móvel que possam sofrer variações como por exemplo, bateria e sinal da conexão. Ela também é responsável por notificar as transações das variações do ambiente quando estas solicitam tal monitoramento.

- ***TransactionManager***: componente que gerencia a transação do lado cliente.

Os componentes relacionados com o controle de concorrência pessimista são descritos a seguir:

- ***LockManager***: componente pertencente ao controle de concorrência pessimista do lado servidor. Ele mantém uma lista de objetos bloqueados (uma cópia desses objetos) durante uma transação.

- ***ReadCommitted***: componente que representa o nível de isolamento *Read Committed*.

- ***ReadUncommitted***: componente que representa o tipo de isolamento *Read Uncommitted*. É o nível de isolamento menos restrito.

- ***Repeatable Read***: componente que representa o tipo de isolamento *RepeatableRead*.

- ***Serializable***: componente que representa um tipo de isolamento *Serializable*. É o nível de isolamento mais restrito.

- ***TwoPhaseLocking***: componente pertencente ao controle de concorrência pessimista do lado cliente. Implementa o protocolo de bloqueio em duas fases [11].

Os componentes relacionados com o controle de concorrência otimista são descritos a seguir:

- ***VersionControl***: componente que faz parte do controle de concorrência otimista do lado cliente e é responsável por implementar as operações (inclusão, atualização e consulta) sobre os objetos da transação. As alterações são armazenadas em *cache* (memória local de acesso rápido) antes de serem efetivadas.

- ***VersionManager***: componente que faz parte do controle de concorrência otimista do lado servidor. Ele controla a versão dos objetos utilizados durante uma transação implementando as operações de validação referentes ao controle de concorrência.

A. Reconfiguração de componentes

A reconfiguração da transação é feita através da troca de componentes: componentes podem ser conectados ou desconectados de acordo com a configuração desejada, não necessitando remontar toda a arquitetura definida.

Como o foco do modelo proposto é o controle de concorrência, a idéia é que o modelo permita a configuração estática do controle de concorrência e a reconfiguração dinâmica do nível de isolamento baseado em políticas de adaptação pré-definidas.

O controle de concorrência é configurado no início da transação conforme a situação dos recursos disponíveis no ambiente móvel, como, por exemplo, bateria, largura de banda ou conectividade.

O mecanismo de controle de concorrência padrão é o pessimista com nível de isolamento *Serializable*. Se a situação de um recurso for alterada extrapolando alguns limites pré-estabelecidos pela transação, algumas propriedades podem mudar: se a transação estiver no início, o controle de concorrência poderá ser alterado, mas se ela estiver em execução, apenas o nível de isolamento poderá ser alterado.

Nesse contexto, a transação pode reagir às mudanças do ambiente conforme a política de adaptação definida para ela. A lógica dessa política é definida pelo programador. Ela é escrita utilizando uma linguagem de programação e contém as regras que deverão ser seguidas caso o monitor de recursos verifique que os valores pré-estabelecidos pela aplicação foram extrapolados.

A reconfiguração é feita pelas classes *ArchitectureServer* e *ArchitectureClient*. O monitor de recursos verifica os recursos disponíveis no ambiente do dispositivo móvel cujo monitoramento foi requisitado pela aplicação. Se o limite for atingido, ele notifica a classe *ClientConfigurator* que contém a política de adaptação do lado cliente.

Se uma das regras definidas na política de adaptação for satisfeita, a arquitetura deverá ser alterada, portanto, a classe *ArchitectureClient* será chamada para realizar a adaptação dinâmica. A reconfiguração é realizada desconectando e conectando componentes.

Como o cliente e o servidor devem estar sincronizados, ou seja, com o mesmo tipo de controle de concorrência, o cliente notifica a classe *ServerConfigurator*

indicando que a sua arquitetura também deve ser alterada já que a arquitetura do cliente foi mudada. O *ServerConfigurator* contém a política de adaptação do lado servidor e caso alguma regra inserida nele seja cumprida, a classe *ArchitectureServer* é chamada para também fazer a reconfiguração do lado servidor.

A arquitetura que foi apresentada na Fig. 2 contém quatro níveis de isolamento, porém novos níveis de isolamento poderiam ser adicionados.

III. TECNOLOGIA ADAPTATIVA E COMPUTAÇÃO RECONFIGURÁVEL

Segundo [14], a "tecnologia adaptativa" é o conjunto das aplicações práticas do conceito fundamental da adaptabilidade, sempre que esta for utilizada como instrumento na resolução de problemas oriundos das mais variadas. E a adaptabilidade é a propriedade que apresenta um sistema, dispositivo ou processo computacional, que lhe permite sem a interferência de agentes externos – mesmo do próprio operador – tomar a decisão de modificar dinamicamente, de forma autônoma, seu próprio comportamento, em resposta apenas à sua configuração corrente e ao estímulo de entrada recebido.

A adaptabilidade é uma das idéias que viabilizam a computação móvel, pois mecanismos de adaptação devem ser projetados para que os recursos disponíveis sejam melhor utilizados.

Relacionado à adaptabilidade está o conceito de reflexão computacional [9] que implica em inspeção e adaptação. O sistema inspeciona o estado corrente e a adaptação faz com que o comportamento do mesmo seja alterado para melhor se ajustar ao ambiente.

A principal vantagem da reflexão computacional é permitir que o sistema seja alterado em tempo de execução, sendo essas alterações comportamentais ou até mesmo estruturais. Na computação móvel essa técnica é essencial para prover inspeção e adaptação do sistema diante da variação dos recursos disponíveis.

Na arquitetura apresentada, a adaptabilidade do controle de concorrência e do nível de isolamento foi implementada através do modelo de componentes OpenCOM, que implementa a reflexão computacional, e de regras pré-definidas pelo programador nos componentes *ClientConfigurator* e *ServerConfigurator*, permitindo que o sistema se modificasse dinamicamente sem a interferência do usuário, adaptando-se às mudanças do ambiente.

IV. PROTÓTIPO E ESTUDOS DE CASOS

A. Protótipo

Um protótipo foi desenvolvido para comprovar a viabilidade da arquitetura proposta. Ele foi desenvolvido utilizando a versão 1.3.5 (Java) do OpenCOM [15] e a linguagem Java [16]. O motivo da escolha da linguagem Java é por ela ser orientada a objetos e independente de plataforma.

O protótipo evidencia a configuração dos controles de concorrência e a reconfiguração dos níveis de isolamento. Nele foram implementados, dois níveis de isolamento: *Serializable* (mais restrito) e *Read Uncommitted* (menos restrito).

Para avaliar a solução proposta, um estudo de caso na área de vendas foi escolhido.

B. Estudo de casos: Vendas

Considere uma aplicação de vendas destinada aos gerentes e representantes de vendas de uma distribuidora de medicamentos. Ambos utilizam PDA's (*Personal Digital Assistant*) para acessarem os módulos da aplicação e realizarem as tarefas de sua função. Um PDA é conhecido como um computador de bolso, por ter dimensões reduzidas.

A aplicação possui dois módulos: Pedido de venda de produtos e Relatório da média das vendas do dia.

O representante de vendas efetua o pedido de venda do produto através de um PDA. Ele comunica-se diretamente com o servidor em tempo real, fazendo seus pedidos através do computador, ganhando dessa forma, agilidade. O pedido do produto, após a venda efetivada, é enviado para o serviço de entrega, que já inicia o seu trabalho.

O gerente consulta a média das vendas realizadas no dia também através de um PDA. Para tomar alguma decisão com maior rapidez, ele precisa ser informado sobre as vendas realizadas no dia e o desempenho de sua equipe de vendedores.

Os dois módulos se conectam a um servidor para efetuar as vendas dos produtos e realizar as consultas necessárias.

Existem vários representantes de vendas em lugares diferentes que utilizarão a aplicação ao mesmo tempo.

Como esses usuários estão utilizando dispositivos móveis, eles encontram restrições de alguns recursos como, por exemplo, bateria e conectividade. Sendo assim, em situações de escassez de energia ou nível de bateria limitado, o gerente necessita visualizar a média das vendas considerando até os produtos que já foram registrados pelos representantes, mas cuja venda ainda não foi concluída. Essa informação o ajudará a tomar decisões como criar promoções e agilizar a entrega de alguns produtos, aumentando a margem de lucro e a satisfação do cliente.

Para esse tipo de aplicação em que existe concorrência, é necessário haver um gerenciamento de transações.

O protótipo dessa aplicação de Vendas utilizará a arquitetura mencionada anteriormente. Três casos de uso serão demonstrados para explicar o funcionamento da arquitetura, a configuração dos controles de concorrência no início da transação e a reconfiguração dos níveis de isolamento durante a transação.

1) Caso de uso: tipo de controle pessimista

Para um melhor entendimento dos casos de uso, será mostrada, primeiramente, a aplicação e posteriormente, algumas informações arquiteturais do protótipo implementado.

A aplicação de vendas contendo os dois módulos (Pedido de venda de produtos e Relatório de média das vendas do dia) representa o lado Cliente na arquitetura proposta. O lado

servidor contém o banco de dados com as informações do estoque de produtos e das vendas realizadas.

O servidor está pronto aguardando a conexão de algum cliente. O tipo de controle de concorrência padrão é o pessimista.

Um representante de vendas chamado José inicia o seu dia de trabalho. Ele está nas ruas, utilizando seu PDA.

José visita uma farmácia, verifica os medicamentos que necessitam ser solicitados à distribuidora e começa a registrar a solicitação na aplicação de vendas em seu PDA. Ele seleciona a opção "*Order*" (representada pelo ícone do carrinho) e escolhe o produto "p1", quantidade igual a 2 e seleciona a opção "*Confirm*", conforme a Fig. 5.

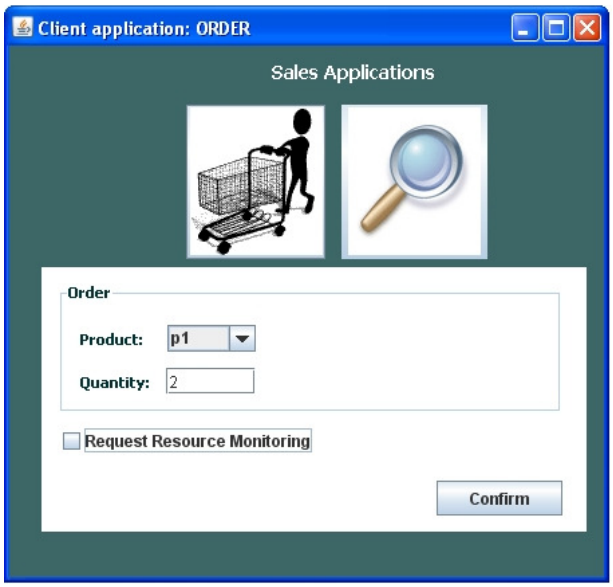


Fig. 5. Pedido solicitado pelo representante de vendas José

A transação inicia executando as operações de inserir o produto na tabela de vendas e atualizar o estoque. Antes de a transação do representante de vendas José ser efetivada, outro representante de vendas chamado Manuel estava executando, ao mesmo tempo, a aplicação em outra farmácia. Ele solicitou o mesmo produto "p1" com quantidade igual a 3. Manuel não consegue realizar a compra do produto porque a transação do representante de vendas José está bloqueando os respectivos objetos da tabela de vendas. Como o controle de concorrência é pessimista, a transação do representante de vendas Manuel segue apenas até a primeira operação de inserir o produto na tabela de vendas.

O representante de vendas José decide cancelar o pedido do produto "p1" porque a farmacêutica descobriu que havia ainda dois medicamentos "p1" no estoque, porém eles estavam armazenados no lugar incorreto. José seleciona a opção "*Cancel*" e com isso as operações de inserção do produto na tabela de vendas e a atualização da quantidade na tabela de estoque são desfeitas.

Portanto, nenhum produto foi vendido. O representante de vendas Manuel poderia ter efetivado a sua compra, mas devido ao tipo de controle de concorrência ser pessimista, ele perdeu a venda.

Diante disso, nunca deveria ser utilizado o controle de concorrência pessimista? A resposta seria: depende. O controle de concorrência pessimista é necessário quando existe a grande probabilidade de conflitos e no ambiente de computação móvel, quando os recursos como conexão e bateria não estão limitados. Mas como saber se os recursos estão limitados? Para isso, a arquitetura contém o componente *ResourceMonitor*. Na Fig. 5, José poderia ter optado por monitorar os recursos do dispositivo móvel, como por exemplo, o sinal da conexão com o servidor.

Como os representantes usam dispositivos móveis, o sinal de conexão poderia estar ruim e a conexão com o servidor cair. Sendo assim, os objetos ficariam bloqueados até a conexão voltar, impedindo novas vendas de um determinado produto. Diante desse contexto, seria apropriado alterar o tipo de controle de concorrência para otimista conforme o próximo caso de uso.

2) *Caso de Uso: alteração para o tipo de controle otimista*

Suponha que o representante de vendas José tenha optado por monitorar os recursos do dispositivo móvel no primeiro caso de uso. Conforme a Fig. 6, ele pode solicitar ao monitor de recursos que verifique o nível do sinal de conexão ou da bateria. Nesse caso, ele solicitou monitorar o nível do sinal de conexão selecionando a opção "*Request Resource Monitoring*" e "*Signal Strength*". O limite mínimo deve ser escolhido também. Entre as opções, existem: "*No Connectivity*", "*Limited*", "*Good*", "*Very Good*" e "*Excellent*", ou seja, opções indicando desde nenhum sinal até um nível de conexão excelente. Esse limite indica qual o nível mínimo do recurso monitorado será suportado pelo dispositivo móvel.

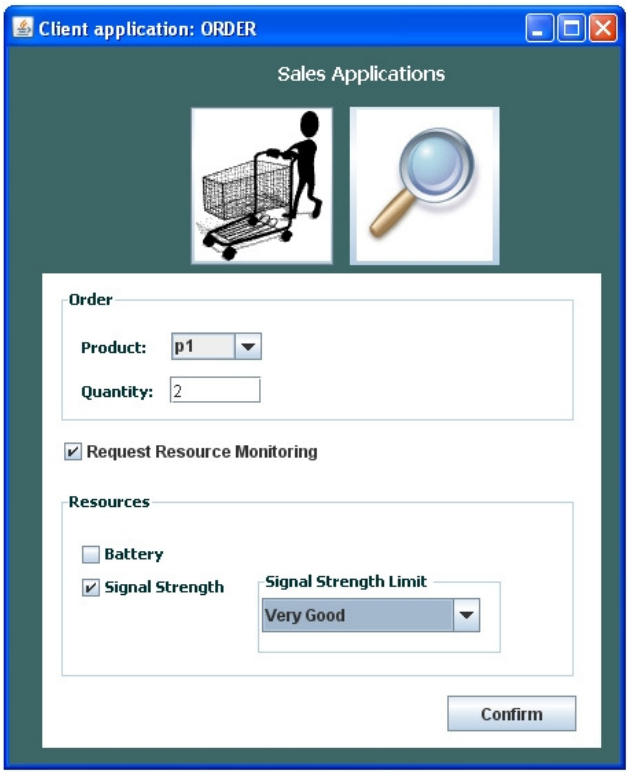


Fig. 6. Solicitação de monitoramento de recursos do dispositivo móvel

Nesse exemplo, a opção escolhida foi "*Very Good*". Isto significa que, se o nível do sinal de conexão do dispositivo estiver abaixo do limite "muito bom", o monitor de recursos notificará a transação que o limite está baixo do solicitado e a política de adaptação definida será aplicada.

Após selecionar a opção "*Confirm*", antes de iniciar a transação, o sistema verificará que a opção de monitoramento de recursos foi selecionada. O componente *ResourceMonitor* é chamado para monitorar o nível de sinal de conexão.

A transação será iniciada no dispositivo móvel de José. Como no caso de uso anterior, o controle de concorrência seria o pessimista. Porém, como ele solicitou o monitoramento do sinal de conexão, o monitor de recursos constantemente verifica se o sinal está dentro do limite pré-estabelecido. A conexão com o servidor estava excelente, porém de repente o sinal ficou limitado, antes do início da transação.

O outro representante de vendas Manuel, ao mesmo tempo, executa a aplicação cliente também solicitando o monitoramento de recursos do seu dispositivo móvel. O recurso escolhido também é o sinal de conexão e o limite tem que ser no mínimo "muito bom". O dispositivo móvel de Manuel está com uma conexão excelente com o servidor.

Antes da execução da operação de inserir o produto na tabela de vendas, a conexão com o servidor é alterada e se torna limitada.

Neste momento, o monitor de recursos percebe que o um dos recursos foi alterado. Ele compara o nível do sinal medido com o limite estabelecido pelo cliente da aplicação. Como o limite do recurso monitorado é menor que o limite esperado, o monitor de recursos notifica a transação. A transação então avalia as regras da política de adaptação para tomar alguma decisão diante da mudança.

Uma das regras definidas na política de adaptação é que, se um dos recursos monitorado estiver abaixo do limite estabelecido pelo cliente da aplicação antes do início da transação, o controle de concorrência deve ser alterado de pessimista para otimista.

No controle de concorrência otimista, os objetos não são bloqueados. Os objetos da transação são copiados para a máquina local do cliente (mantidos em *cache*) e todas as operações são efetuadas sobre a cópia local. Portanto, se a conexão com o servidor cair, a transação pode continuar.

O dispositivo móvel do representante de vendas Manuel estava com a conexão limitada. Prevendo uma possível desconexão, os objetos da transação foram copiados e as operações de inserção na tabela de vendas e de atualização na tabela de estoque foram executadas localmente no dispositivo móvel. No final da transação, as alterações são validadas para verificar se a versão de cada objeto foi alterada. As versões dos objetos de vendas e estoque permaneceram a mesma, portanto a venda do produto foi efetivada e o estoque atualizado.

O controle de concorrência do dispositivo móvel de José também foi alterado para otimista já que o nível do sinal estava limitado e ele havia estabelecido um limite de nível de conexão "muito bom". Ele conseguiu executar localmente as operações sobre os objetos. Porém, quando ele foi efetivar a transação do lado servidor, como o representante de vendas Manuel já havia alterado os objetos de vendas e estoque, as versões estavam diferentes (a versão dos objetos do repositório com a versão desses objetos do cache do lado cliente). Dessa forma, José não conseguiu efetivar o seu pedido.

No caso de uso acima, os dois clientes iniciaram a transação utilizando o controle de concorrência otimista, devido ao sinal de conexão limitado. Supondo que a transação do dispositivo de José inicie utilizando o controle pessimista (o sinal de conexão não está limitado) e que antes de a transação ser efetivada, Manuel também solicita o mesmo produto. O nível de conexão do dispositivo móvel dele está limitado e, portanto, o controle é alterado para otimista do lado cliente. Quando o servidor receber uma solicitação para alterar o controle de concorrência, ele deve verificar se existe alguma transação iniciada com algum cliente em outro modo de concorrência. Se existir, ele mantém o modo de concorrência e recusa a nova solicitação. Nesse caso, José conseguiria efetivar a transação, Manuel não conseguiria mudar o modo de controle de concorrência e teria que prosseguir no modo pessimista ou abortar a transação. O motivo disso é evitar que cliente e servidor tenham tipos de controles de concorrência diferentes.

3) Caso de Uso: reconfiguração do nível de isolamento

Os casos de uso anteriores demonstraram a configuração do controle de concorrência antes do início da transação. Esse caso de uso demonstrará a reconfiguração do nível de isolamento durante uma transação.

Suponha que os seguintes pedidos já tenham sido efetivados:

Produto = "p2" com quantidade = 1
Produto = "p2" com quantidade = 2
Produto = "p2" com quantidade = 3

Porém, um outro pedido é solicitado, mas não é efetivado (as operações de inserir o produto na tabela de vendas e atualizar o estoque são executadas, mas os dados não são gravados no banco de dados). Esse pedido é descrito abaixo:

Produto = "p1" com quantidade = 50

Ao mesmo tempo, o gerente da equipe de vendas, para tomar uma decisão, necessita consultar a média das vendas realizadas no dia. Como ele está fora do escritório, ele utiliza o seu PDA para obter esse resultado. Ele seleciona a opção "*Sales report*" e solicita o monitoramento de recursos do dispositivo móvel. O recurso a ser monitorado será a bateria. A aplicação é exibida na Fig. 7.

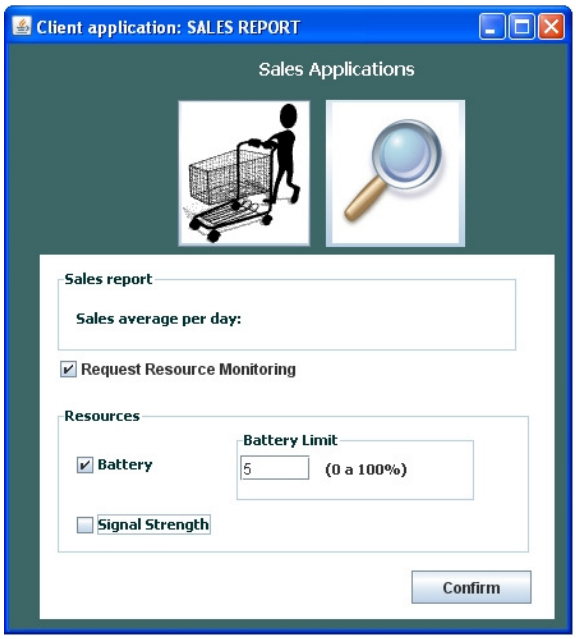


Fig.7. Opção selecionada: relatório das vendas do dia

Supondo que o nível da bateria do dispositivo móvel esteja 100%, a aplicação exibirá apenas a média das vendas efetivadas. Portanto, no exemplo acima, a média é igual a 10 considerando-se que:

Produto = "p2" com quantidade = 1 (1 x 10.00) = 10.00
Produto = "p2" com quantidade = 2 (2 x 10.00) = 20.00
Produto = "p2" com quantidade = 3 (3 x 10.00) = 30.00

No cálculo acima, o produto "p1", com quantidade = 50, não foi considerado porque o controle de concorrência definido é o pessimista e por padrão o nível de isolamento é *Serializable*. Esse nível não permite que os resultados de transações não efetivadas sejam visualizados.

Porém, conforme o gerente utiliza o dispositivo, ele percebe que o nível da bateria está diminuindo rapidamente. Como ele optou por um limite de 5%, se o nível da bateria estiver abaixo, o monitor de recursos notificará a transação e a política de adaptação será executada. A transação iniciada tem a política de adaptação de alterar o nível de isolamento de *Serializable* para *Read Uncommitted* se o nível de bateria estiver abaixo do limite estabelecido.

Como a transação já foi iniciada, apenas o nível de isolamento pode ser alterado. Se alguma transação estiver em andamento com outro nível de isolamento, ela não será afetada. O servidor permite a execução de diferentes níveis de isolamento em paralelo.

Enquanto isso, a transação do gerente de vendas continua e como o nível de isolamento é menos restrito, as vendas não efetivadas serão visualizadas pelo gerente. Portanto, a última venda não concluída será considerada no cálculo, logo o resultado mostrado ao gerente não será mais 10.00 e sim 179.64. Foram consideradas as seguintes vendas:

Produto = "p2" com quantidade = 1 (1 x 10.00) = 10.00
Produto = "p2" com quantidade = 2 (2 x 10.00) = 20.00
Produto = "p2" com quantidade = 3 (3 x 10.00) = 30.00
Produto = "p1" com quantidade = 50 (50 x 200.00) =10000.00

Se a última venda for concluída pelo representante de vendas, a média de vendas realizadas exibida para o gerente estará correta. Porém, o representante de vendas desistiu do produto, ou seja, a última venda não foi efetivada. Sendo assim, a média das vendas seria 10.00 e não 179.64. Mas o gerente tinha necessidade de ver a informação rapidamente, pois o nível da bateria estava diminuindo e logo o dispositivo móvel poderia deixar de funcionar. Além disso, o resultado exibe uma média geral. A preocupação nesse tipo de aplicação não é a consistência dos dados e sim o desempenho. O gerente sabe que algumas vendas podem não ser efetivadas, mas diante da situação do ambiente, ele opta por considerar essas informações para tomar uma decisão.

C. Detalhes da arquitetura do protótipo implementado

No primeiro caso de uso, o servidor aciona a classe *ArchitectureServer* para montar a arquitetura com os componentes OpenCOM do lado servidor. Do lado cliente, a classe *ArchitectureClient* é a responsável pela montagem.

Como o controle de concorrência definido como padrão é o pessimista, o componente *LockManager* é conectado ao componente *ResourceManager*. Os componentes referentes ao nível de isolamento são conectados ao *ResourceManager* também. O nível de isolamento padrão é o *Serializable*. O outro nível isolamento (*ReadUncommitted*) permanece desconectado da arquitetura. O componente *TwoPhaseLocking* é conectado ao componente *TransactionManager*.

Como a interface exposta pelo framework do cliente é a *ITransactionManager*, todas as operações pertencentes a ela podem ser utilizadas pelo cliente da aplicação. Seguindo o modelo de transação, as operações chamadas são: *begin, execute*, *commit* e *rollback*. A primeira operação chamada do componente *TransactionManager* é *begin* que indica o início da transação. Seguindo a arquitetura, a operação chama os componentes *TwoPhaseLocking* e *ClientCommunicator*. No *ClientCommunicator*, o servidor é procurado no servidor de nomes RMI, realizando dessa forma a conexão cliente-servidor.

Do lado servidor, o componente *ServerCommunicator* recebe a requisição do cliente e chama o componente *ResourceManager*. Na operação de *begin* dentro desse componente, uma cópia dos objetos a serem utilizados pela transação (tabela de vendas e tabela de estoque) é inserida na lista de objetos bloqueados. O responsável por esse controle é o componente *LockManager*.

A próxima operação da transação é o *execute*. Nesse exemplo, duas operações de execução são realizadas: inserir o pedido solicitado na tabela de vendas e atualizar a tabela de estoque.

As duas operações são executadas de forma atômica, ou seja, se o produto for inserido na tabela de vendas, a tabela

de estoque deve ser atualizada. Se uma das operações falhar, a outra deve ser desfeita.

Na execução das operações, o componente *ResourceManager* pede para o componente *LockManager* verificar se o objeto está bloqueado por outra transação. Caso exista outra transação utilizando o mesmo objeto, a transação corrente não pode continuar. Outra possibilidade seria bloquear a transação corrente, mas só a primeira opção foi implementada. Em seguida, se a requisição de bloqueio teve sucesso, o componente inclui o pedido na tabela de vendas e atualiza o estoque.

Para efetivar a transação, a operação *commit* precisa ser executada.

Para que o controle de concorrência seja alterado dentro da arquitetura proposta, o monitor de recursos notifica a classe *ClientConfigurator* que contém a política de adaptação. Essa classe solicita para o componente *ArchitectureClient* que ele altere o controle de concorrência de pessimista para otimista. Esse componente, por sua vez, desconecta o componente *TwoPhaseLocking* do componente *TransactionManager* e conecta o componente *VersionControl*.

Até esse momento, o controle de concorrência foi alterado apenas do lado cliente. O mesmo deve acontecer do lado servidor. Para isso, a classe *ClientConfigurator* após solicitar a alteração do controle de concorrência do lado cliente, manda uma notificação para a classe ServerConfigurator através da tecnologia RMI. A classe *ServerConfigurator* ao receber a notificação, solicita ao componente *ArchitectureServer* a alteração do controle de concorrência de pessimista para otimista. Esse componente, por sua vez, desconecta o componente *LockManager* do componente *ResourceManager* e conecta o componente *VersionManager*.

No controle de concorrência otimista, os objetos não são bloqueados. Os objetos da transação são copiados para a máquina local do cliente (mantidos em *cache*) e todas as operações são efetuadas sobre a cópia local. Portanto, se a conexão com o servidor cair, a transação pode continuar. Essa é a fase de preparação. O responsável por esse processo é o componente *VersionControl*. Apenas no final da transação, nas fases de validação e escrita, o cliente necessita se conectar ao servidor.

Por último, no terceiro caso de uso, o monitor de recursos notifica a classe *ClientConfigurator* que verifica se a transação já começou. Como a transação já foi iniciada, apenas o nível de isolamento pode ser alterado. Se alguma transação estiver em andamento com outro nível de isolamento, ela não será afetada. Como do lado cliente não existe alteração, a classe *ClientConfigurator* notifica a classe *ServerConfigurator* através da tecnologia RMI. A classe *ServerConfigurator* solicita para o componente *ArchitectureServer* a alteração do nível de isolamento. Esse componente, por sua vez, desconecta o componente *Serializable* do componente *ResourceManager* e conecta o componente *Read Uncommitted*.

V. CONCLUSÕES

O propósito desse artigo foi demonstrar como algumas propriedades transacionais e alguns mecanismos para garanti-las, podem ser configurados e reconfigurados dinamicamente utilizando um modelo de componentes. Neste contexto, a configuração do tipo de controle de concorrência foi uma proposta inovadora em comparação aos modelos de transações existentes, permitindo a demonstração do conceito de adaptabilidade.

O modelo proposto baseou-se na idéia do modelo SGTA [7], porém utilizando o modelo de componentes OpenCOM, ao invés de CORBA [17] e também utilizando o controle de concorrência como foco principal da arquitetura. Além disso, foi implementado um protótipo para analisar os impactos da reconfiguração durante uma transação.

Dois motivos fizeram o nível de isolamento ter sido escolhido como mecanismo de adaptação: permitir pequenas reconfigurações arquiteturais durante a transação e propiciar ganhos de desempenho. Quanto mais baixo o nível de isolamento, maior o ganho de desempenho. Como esse ganho pode muitas vezes ocasionar perda de consistência, apenas em algumas situações isso seria desejável. Portanto, essa é uma boa propriedade para ser alterada durante a execução da transação. Além disso, como mostrado num dos casos de uso da seção IV, em algumas aplicações e em situações de recursos limitados, pode-se optar por melhor desempenho ao invés de nível de consistência restrito como a seriabilidade.

Notou-se que cada controle de concorrência tem a sua vantagem. Em determinadas situações, como por exemplo, quando a conexão está limitada, o controle otimista pode ser mais adequado, pois o cliente pode trabalhar com os dados localmente. Já em situações em que largura de banda e a conexão são satisfatórias, o controle pessimista pode ser utilizado, pois evita a cópia de dados o que acarretaria em largura de banda e custo de comunicação excessivos.

Porém o tipo de controle de concorrência só pôde ser alterado antes de a transação começar. A razão disso é porque quando a transação já iniciou com um determinado controle de concorrência, algumas ações já foram executadas tanto no cliente quanto no servidor. Por exemplo, se o cliente inicia a transação utilizando o controle pessimista, os objetos utilizados na transação ficam bloqueados até o final da mesma. Já o controle otimista exige que sejam feitas cópias locais dos objetos envolvidos na transação. Se durante a transação o controle de concorrência for alterado para otimista, uma cópia dos objetos será feito no cliente e o processo de validação será realizado como determinado nesse tipo de controle, porém os objetos não serão desbloqueados.

REFERÊNCIAS

[1] Android. http://www.android.com. Acessado em 01/2010.
[2] E. A. Nassu. Consultas sobre "aqui" em Sistemas de Bancos de Dados em Ambientes de Computação Nômade. Tese de Doutorado - IME-USP, Orientador: Prof. Dr. Marcelo Finger, Junho de 2003.
[3] N. Santos and P. Ferreira. Making Distributed Transactions Resilient to Intermittent Network Connections. Instituto Superior Técnico de Lisboa, 2006.
[4] R. Rouvoy , P. S. Alvarado and M. Philippe. Towards Context-Aware Transaction Services, 2006.
[5] P. S. Alvarado, C. Roncancio, M. Adiba and C. Labbé. Adaptable Mobile Transactions and Enviromnet Awareness, 2003.

[6] A. Arntsen and R. Karlsen. ReflecTS; A Flexible Transaction Service Framework. University of Tromsoe, 2005.
[7] T. Rocha and M. B. F. Toledo. Um Sistema de Transações Adaptável para o Ambiente de Computação Móvel. Simpósio Brasileiro de Redes de Computadores, 2003.
[8] OpenCOM. Computer Departament of Lancaster University. http://www.comp.lancs.ac.uk/computing/research/mpg/reflection/opencom.php. Acessado em 01/2010.
[9] B. Smith. Reflection and Semantics in a Procedural Language. PhD thesis, Cambridge, Massachusetts, 1982.
[10] F. Backmann et. al. Volume II: Technical Concepts of Component-Based Software Engineering, 2ª edição, 2002.
[11] P. Bernstein, V. Hadzilacos and N. Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.
[12] G. Coulouris, J. Dollimore and T. Kindberg. Distributed Systems: Concepts and Design. Third edition. Addison-Wesley, 2001.
[13] Remote Method Invocation Home. Java Sun Microsystems. Acessado em 01/2010. http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
[14] http://www.pcs.usp.br/~lta/ Página web do LTA – Laboratório de Linguagens e Técnicas Adaptativas.
[15] The OpenCOMJ Handbook. Lancaster Univerty. Febreary, 2007. http://www.mirrorservice.org/sites/download.sourceforge.net/pub/sourceforge/g/gr/gridkit/OpenCOMJHandbook.pdf
[16] Java Sun Microsystems. http://java.sun.com/. Acessado em 01/2010.
[17] CORBA - Common Object Request Broker Architecture. http://www.corba.org. Acessado em 01/2010.

Allyn Grey de A. Lima Pierre é Engenheira de Computação pela Pontíficia Universidade Católica de Campinas (2002) e Mestre em Ciência da Computação pela Universidade Estadual de Campinas (2009). Atualmente trabalha no Centro de Inovação Tecnológica.

Maria Beatriz Felgar de Toledo possui graduação e mestrado em Ciência da Computação pela Universidade Estadual de Campinas e doutorado em Sistemas Distribuídos pela Universidade de Lancaster, Inglaterra (1992). Atualmente é professora associada na Universidade Estadual de Campinas. Tem interesse nas áreas de gestão de processos de negócio, serviços Web, Web semântica, bibliotecas digitais e museus.

Tarcisio da Rocha possui doutorado em Ciência da Computação pela Universidade Estadual de Campinas (2008) com um período sanduíche na Universidade de Tromso (Noruega). Atualmente é professor Adjunto da Universidade Federal de Sergipe. Suas áreas de interesse incluem sistemas distribuídos, middleware, computação móvel e transações.

An Adaptive Maximum Entropy Approach for Modeling of Species Distribution (Janeiro 14, 2010)

E. S. C. Rodrigues, F. A. Rodrigues, R. L. A. Rocha, P. L. P. Corrêa

***Abstract* – One of the most used methods in modeling biological species geographical distribution is the Maximum Entropy Algorithm. This paper presents an Adaptive Maximum Entropy (AME) approach for modeling biological species, which aims to reduce the amount of features used in the construction of models. The approach presented here is an alternative to the classical algorithm. Instead of using the same set of features at each iteration, the AME approach tries to find a new set of features that converge faster. The preliminary experiments were well performed. However, statistical evaluations and comparisons with other algorithms are beyond the scope of this paper. These needed studies are proposed as future works.**

***Index Terms* – Adaptive systems, Maximum Entropy methods, biological system modeling.**

I. Introduction

The world is suffering the consequences of the lack of environmental conservation. Several environmental problems need attention and efficient strategies for its solution. The modeling of environmental systems can assist in the decision making process to solve these problems. There are several Artificial Intelligence (AI) techniques that are being used for modeling environmental systems. AI techniques have large potential to solve inference complex problems because they try to simulate some human characteristics, such as reasoning and learning.

Some examples of AI techniques used for modeling environmental systems are: (a) Case-Based Reasoning applied to forest fire management [1], air quality prediction of urban areas [2], reduction of environmental impact caused by chemical process [3] and cyclone forecasting [4]; (b) the Rule-Based Systems can be employed in pests diagnostic [5], [6], [7] and diseases [8]; (c) Artificial Neural Networks can be applied to water resources management [9], water quality prediction [10], [11], ozone concentrations prediction [12] and modeling of biological species geographical distribution [13], [14]; (d) Genetic Algorithms are a very known and used technique in modeling biological species geographical distribution [15], but it can also be applied in predictions of air quality [16], water quality models calibration [17] and water management in irrigated agriculture [18]; (e) Cellular Automata can be applied to species migration modeling [19], landscape modeling [20] and modeling of seismic activity [21]; (f) Fuzzy systems can be employed in water management [22], soil erosion prediction [23] and forest fire risk estimation [24]; (g) Multi-Agents Systems has been applied in natural resource management [25], [26] and forest management [27]. Other techniques such as Swarm Intelligence, Reinforcement Learning and hybrid systems have also been used in environmental systems modeling. In [28] there is an overview about these AI techniques with additional references.

Since Brazil is the country with the richest flora and fauna on Earth, about one sixth of the total, the motivation and support to researches directed to natural resources management and sustainable development are very important. The modeling of biological species geographical distribution can assist decision-making processes, planning and accomplishing actions aiming at environmental conservation.

There are some free tools available for modeling biological species geographical distribution and several algorithms with different approaches were already implemented for this aim. OpenModeller is a framework for modeling biological species geographical distribution with several resources [29]. One of the algorithms available in openModeller is a Maximum Entropy (MaxEnt) algorithm. This is one of the most used algorithms by researchers of biological species modeling.

The MaxEnt algorithm implemented in openModeller chooses a feature from a set at each iteration and updates its parameters. This choice is based on a minimization function with all features. Thus, features influence the algorithm performance, that is, inserting or removing features can improve the parameters fitting and convergence can be reached faster. However, in the algorithm implemented in openModeller, the set of features used along the learning process is static. The approach proposed here inserts features in the set or removes features from it along the learning algorithm. Adaptive Devices were used here to represent this dynamic characteristic. Adaptive Devices can modify their own structure without external interferences. The only algorithm available in openModeller tool that has an adaptive version is GARP [30].

An Adaptive Maximum Entropy (AME) approach for modeling biological species geographical distribution is presented, which aims to reduce the amount of features used in the construction of models. The AME approach was based on the developed MaxEnt algorithm available in openModeller. The proposed approach uses an ambitious strategy because it

Thanks to CAPES for support the first two authors.
E. S. C. Rodrigues and F. A. Rodrigues are students of the Doctoral Program in Electrical Engineering of Polytechnic School of the University of São Paulo (e-mail: elisangela.rodrigues@poli.usp.br; fabricio.rodrigues@poli.usp.br).
R. L. A. Rocha and P. L. P Corrêa are professors of Department of Computer and Digital Systems Engineering of Polytechnic School of the University of São Paulo (e-mail: luis.rocha@poli.usp.br; pedro.correa@poli.usp.br).

tries to find out a set of features that seems to be the best at each iteration instead of using the same set of features all along the model learning.

Section II describes the modeling process of biological species geographical distribution as well as the openModeller tool. Section III presents the main idea of adaptive devices. Section IV describes the Maximum Entropy Principle and its application to modeling biological species. Section V presents the Adaptive Maximum Entropy approach. The methodology used in the experiments is presented in Section VI, as well as the results obtained. Final discussion and proposals for future works are presented in Section VII.

II. MODELING BIOLOGICAL SPECIES

The amount of available data about biological species is increasing and it is becoming largely disseminated in the World Wide Web. In [32] there are several websites addresses that supply data on species distribution. Modeling tools can process these data, for example, and the results can generate a lot of information to assist environmental conservation planning.

The set of ecological conditions necessary for a species to keep populations is known as species ecological niche [33]. This is the main concept related to modeling of biological species geographical distribution. The fundamental ecological niche is the set of all conditions that allow the survival of species for a long period of time. A niche-based model, produced by a modeling tool, is an approximation of the species fundamental ecological niche.

A modeling tool uses two kinds of data to produce a niche-based model: occurrence data and environmental data. The occurrence data are georeferenced points – latitude and longitude – recorded where the species were observed. These occurrence records are also called presence points. The species occurrences are determined by the environmental conditions in the region where it occurs. Sometimes, there are records of the species absence, indicating its inexistence in a given region. However, absence data are rarely available [34]. Environmental data are also known as environmental layers and they represent the species ecological niche [35]. All environmental layers should be in the same geographic area and they are georeferenced, too [34]. Some examples of environmental layers commonly used in modeling are temperature and precipitation.

The main purpose of a modeling tool is to find a probability function that represents the relation between the suitable environmental conditions for the species and the given environmental layers [34]. Some modeling tools are freely available on the web, such as DesktopGarp [36], MaxEnt [37] and openModeller [27], [38], in which this work is inserted. In these tools, several algorithms have been applied to modeling biological species, such as GARP (Genetic Algorithm for Rule-set Production) [39], [40], [41], [15], Maximum Entropy [34], [42], SVMs (Support Vector Machines) [43], [44], Neural Networks [14], [45] and others [46].

[1] openModeller

The openModeller is a framework developed to support all the modeling process. It offers several functionalities, such as search and preparation of data, pre-analysis modules, modeling algorithms and visualization of results [29]. OpenModeller is an open source tool, written in C++, which runs in different platforms and provides several modeling algorithms [38].

Each algorithm in the openModeller tool has specific input parameters that can be changed by users. Although a little knowledge about these parameters is desirable, all algorithms have default input parameters that were widely tested. The input data are the same for all algorithms: an occurrence data set and an environmental data set. It allows the user to make different experiments with a variety of algorithms but with the same data, making the result analysis and the comparison among them easier.

Figure 1 shows the modeling process used by openModeller. The niche points are environmental layer values at each georeferenced point where the species were recorded. Thus, the occurrence points are transformed into points in the environmental space. The algorithm receives the niche points as input data and, after processing, gives a probability function that maps the environmental suitability for the species to a domain in the environmental layers space. The probability function given as output by the algorithm represents a niche-based model. The model generated is projected in a geographical area, producing a georeferenced map with the probability function of the species. All areas in the map satisfying the environmental conditions of its fundamental niche represent the potential distribution of a species.

In a modeling with the openModeller tool, the user must specify the input occurrence data, the input environmental layers set and the algorithm(s) that will be used. The occurrence data must be in a file containing the record identifier, the species name, the coordinates (longitude and latitude) where the species was observed and the abundance, that assume value 1 when the record is a presence point and 0 when the record is an absence point. Each georeferenced point must be in a new line and a tab separates the fields. Each environmental layer in a GIS format (standardized codification of geographical information) must be stored in a different file.

Besides that, the user must choose the modeling algorithm and set its parameters or keep the default one. There are several help appliances and ways to use XML configuration files [38]. One of the most used algorithms in the environmental modeling community is the Maximum Entropy-based one. This was one of the motivations for the study of a new approach of a MaxEnt method. Another motivation for the development of this work was that both Adaptive Devices and MaxEnt methods tend to be largely applied to modeling and AI problems.

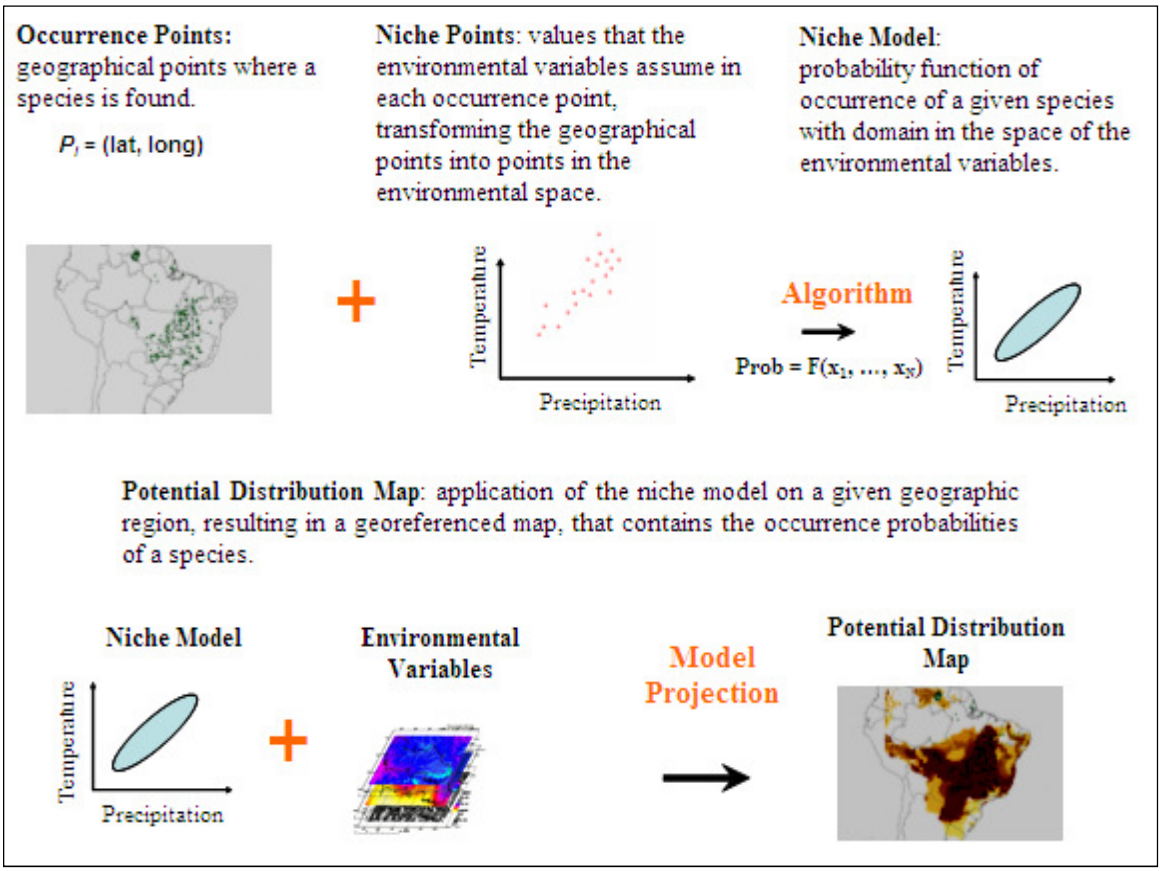


Fig. 1. Modeling Process used in openModeller (adapted from [35])

III. Adaptive Devices

Adaptivity is the capacity that a system has to modify its own structure without external interferences [47]. Adaptive Devices are abstract descriptions of problems that have dynamic behavior. These descriptions are associated with non-adaptive subjacent devices that represent problems with static behavior [48].

Non-adaptive devices have their behavior defined by a static set of rules. A non-adaptive subjacent device is improved by the addition of a set of adaptive actions. These actions characterize the operations needed for making the system behavior adaptive [47].

Any system that has its behavior defined by a set of rules and that has dynamic behavior can use adaptive devices as its abstract description. There are several adaptive formalisms that can be used as descriptions, such as adaptive automata [49], [50], [51], grammars, state charts, Markov chain, decision tables and decision trees [47].

Adaptive actions allow alterations in the set of rules that defines the system. There are three elementary actions in adaptive devices: searching, erasing and inserting [52]. Searching actions try to find a rule according to some pattern. These actions do not modify the set of rules. Erasing actions remove from the set of rules all rules matching a given pattern. Inserting actions add rules with a given pattern in the set of rules. Adaptive actions can be executed before or after the application of the underlying non-adaptive rule that they are associated.

In AME approach proposed here, the underlying device is the structure of the method. The adaptive actions will modify the number of environmental variables used along the learning process.

IV. Maximum Entropy Principle

The Maximum Entropy principle originates from statistical mechanics [53], [54] and has been successfully applied in several research areas, such as Natural Language Processing [55] and modeling of biological species geographical distribution [34], [42]. However, it is possible to produce better results and to find solutions using Adaptive Devices.

The main idea of the MaxEnt principle is: from a probability distribution set that satisfies some constraints, to find the one that has maximum entropy. This principle can be considered a constrained optimization problem, that is, the aim is to find a solution maximizing or minimizing a function.

Entropy (1) is a very important concept in Information Theory. It measures the amount of uncertainty associated to the possible states of an event. The amount of information of an event is inversely proportional to its probability of occurrence. The entropy is defined as [34]

$$H(p) = -\sum_{k=1}^{N} p_k \log p_k \qquad (1)$$

where p is the probability distribution over the set of possible states of an event, N is the total number of possible states of the event and p_k is the occurrence probability of the *k-th* state.

A. Maximum Entropy to Modeling of Species Distribution

The main advantage of applying Maximum Entropy to modeling of biological species geographical distribution in comparison with other methods is that it needs just presence data, besides the environmental layers. Furthermore, it is possible to use both categorical and continuous layers [34].

In modeling biological species geographical distribution, suppose that the finite set of pixels representing the area of interest is X. The set of points $x_1,...,x_m$ pertaining to X represents the presence points of a species. The aim is to find a probability distribution p^* that approximates p, the potential distribution of the species.

The environmental layers are treated as *features*, that is, a set of functions $f_1,...,f_n$, so that f_j: $X \rightarrow \mathbb{R}$. These features can also be functions derived from the environmental layers, such as the square of a continuous environmental layer or a product of two continuous environmental layers. Thus, each feature sets a real value $f_j(x)$ to each point in X. The constraints are that the expectations of each feature matches its empirical average, denoted by $\tilde{p}[f_j]$, where

$$\tilde{p}[f_j] = \frac{1}{m}\sum_{i=1}^{m} f_j(x_i). \tag{2}$$

The MaxEnt probability distribution can be proved to be equivalent to the Gibbs distribution, that is, an exponential distribution with a vector of feature weights that parameterizes it [56]. This probability distribution is defined as

$$q_\lambda(x_i) = \frac{\exp(\sum_{j=1}^{n} \lambda_j f_j(x_i))}{Z_\lambda} \tag{3}$$

where λ is a vector of the feature weights, with real values, and Z_λ is a normalizing constant that guarantees that the probability distribution sums to one over the area of interest. The MaxEnt probability distribution is also equivalent to minimize the log loss, that is the negative log likelihood [34], [42], [56]. The log loss is

$$-\lambda_j \cdot \tilde{p}[f_j] + \log(Z_\lambda) \tag{4}$$

Thus, both Gibbs distribution and log loss are used as objective functions.

B. Maximum Entropy in the openModeller

There are several algorithms to estimate the Maximum Entropy parameters, such as Generalized Iterative Scaling, Improved Iterative Scaling and limited memory variable metric [57]. The estimating algorithm available in the openModeller tool is similar to the sequential algorithm used in the MaxEnt program [34], [42]. It was proved to converge to Maximum Entropy probability distribution in [58].

This algorithm is called sequential because it chooses one feature at each iteration and adjusts its parameters. This procedure is executed until either the convergence is reached or the number of iterations is reached. This estimating method was implemented in openModeller because it has shown to be suitable for modeling biological species geographical distribution [34], [42]. However, there is a similar estimating algorithm that updates all feature weights at each iteration [58]. Figure 2 shows the high-level algorithm to estimate the Maximum Entropy parameters implemented in openModeller. The update consider

$$\delta = \log\frac{(\tilde{p}[f_j](1-q_\lambda[f_j]))}{((1-\tilde{p}[f_j])q_\lambda[f_j])} \tag{5}$$

and, in the second *for*, j is the feature's label that will be updated and j' is the current label.

The algorithm has four input parameters: number of iterations, number of background pixels and convergence. The default values of each parameter defined empirically are: number of iterations = 500, number of background pixels = 10,000 and convergence = 10^{-5}. Background pixels are georeferenced points that are used to delimit the area of study, but these points are not interpreted as pseudo-absences, as in other techniques [34]. Currently, only the linear feature is implemented, that is, the raw environmental layer values.

V. Adaptive Maximum Entropy Approach

The underlying device used here is the structure of the method presented in Figure 2. The adaptive actions do not change the program code but they change the method structure.

Input: feature functions $f_1,\ldots,f_n$

occurrence points $x_1,\ldots,x_2$

Output: vector of feature weights, λ.

for $j = 1$ to n

$\lambda_j = 1$;

for $k = 1, 2, \ldots$

1. **Let $(j, \delta) = \arg \min_{(j,\delta)} F_j(\lambda_{k,j}, \delta)$, where**

   $F_j(\lambda_{k,j}, \delta)$ is the log loss (4) and

   δ is the expression in (5);

2. **if ($j = j'$)**

   $\lambda_{k+1,j'} = \lambda_{k+1,j} + \delta$;

   else

   $\lambda_{k+1,j'} = \lambda_{k+1,j'}$

Fig. 2. High-level algorithm to estimate MaxEnt parameters (adapted from [58]).

Figure 3 presents the general view of the training procedure. This procedure can be applied to any maximum entropy algorithm because the changes occur in the number of features used at each iteration.

The algorithm begins with all features chosen by the user and all feature weights are set to 1. Instead of just choosing the best feature to adjust its parameters, the adaptive approach searches for the best set of features based on the log loss of each possible set.

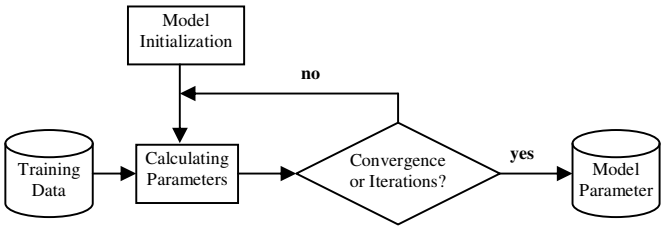


Fig. 3. General view of the training procedure.

Figure 4 shows how the procedure chooses the best set of features at each iteration. For all possible subset of features, F_j is calculated and the minimum value is chosen to update the respective parameter. At each moment of time, there is a set of feature weights that produces a probability distribution. The log loss of every subset of features is calculated and the subset with the smallest value is chosen to be the new set of features. This procedure is repeated until the convergence or the number of iterations is reached. Thus, the features can be inserted or removed from the set according to the log loss. This variable set of features characterizes the dynamic behavior of the task, which tries to find the probability distribution with maximum entropy as soon as possible.

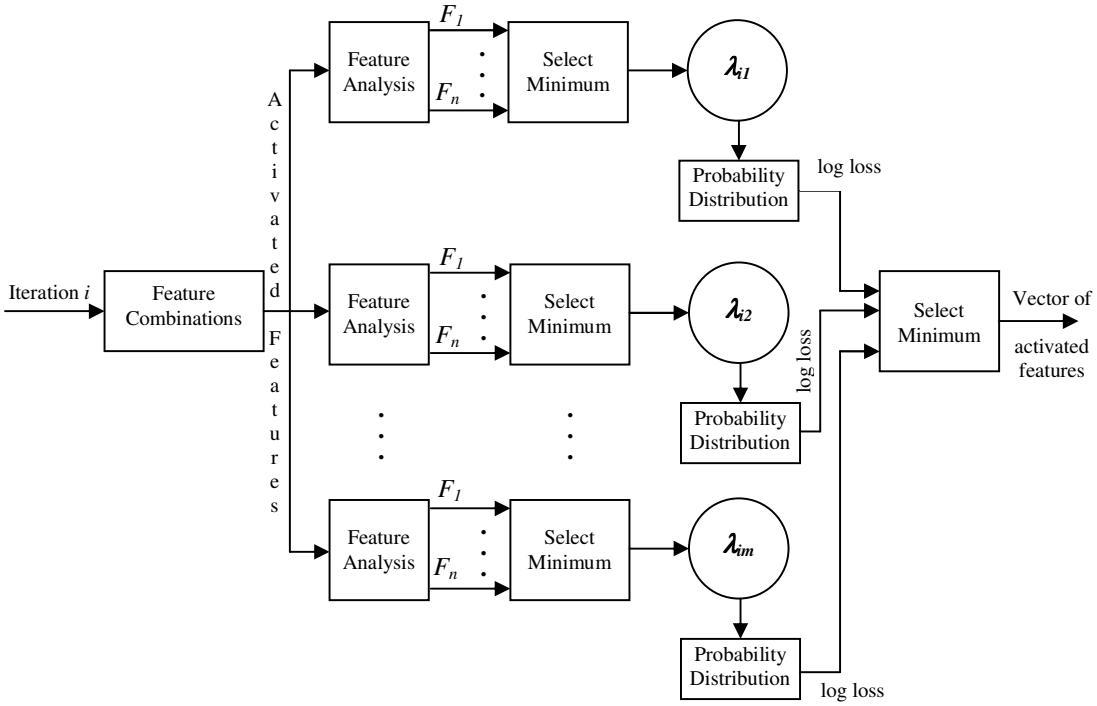


Fig. 4. Adaptive Maximum Entropy approach.

VI. Experimental Methodology and Results

A. Occurrence Data

Occurrence data of two species were used for tests here: *Byrsonima intermedia* and *Xylopia aromatica*. The first one is a shrub of the family *Malpighiaceae*. It is popular known as small-murici and is a native medicinal species of Brazilian Cerrado. The second one is a small tree of the family *Annonaceae*. It is popular known as malagueto and is found in Brazilian Cerrado. It is commonly used for firewood.

The occurrence data are derived from SinBiota – environmental information system for the program Biota/Fapesp [59]. The tests were carried out with 38 records of *Byrsonima intermedia* species and 33 records of *Xylopia aromatica* species. These species were chosen because they had been used in other experiments in openModeller.

B. Environmental Data

A biologist suggested the environmental layers. The models were generated for the Sao Paulo state, Brazil. Thus the environmental layers were from the same region with spatial resolution of 30 arc-second (approximately 1 km^2) provided by WorldClim – Global Climate Data [60]. The layers used were annual mean temperature, mean diurnal range, maximum temperature of warmest month, minimum temperature of coldest month, annual precipitation, precipitation of wettest month and precipitation of driest month, totaling 7 layers.

C. Experiments

The aim of the experiments was to validate the AME approach. All experiments were carried out in a computer with Core 2 Duo Intel processor of 1.66 GHz and 2 GB of RAM. The Operational System used in this architecture was Ubuntu 7.04, a Linux distribution. In all experiments, the *time* command available in Linux was used. This command was used aiming to evaluate the impact of the AME approach in the openModeller performance.

The *time* command measures the execution time of the application, the time spent by the system functions during the application execution, the total time from the beginning until the end of the execution, the CPU percentage that the application got (application time + system time/total time), number of files read and written by the process and the number of page faults during the process execution. There are other output options that can be activated through the command line [61]. Here, only the total time of the application execution was considered.

Besides the execution time, the accuracy of all the models generated was considered to evaluate if the AME approach had a better performance than the classical algorithm. Both AME and the classical algorithms were run 5 times for each species and the considered values were the average of the recorded values.

D. Results

AME approach spent 9.47 seconds in average for *Xylopia aromatica* species and 9.57 seconds in average for *Byrsonima intermedia* species, whereas the classical approach spent 14.25 seconds for the first species and 14.38 seconds for the second one.

The model average accuracy was 45.45% with the AME approach for *Xylopia aromatica* species and 56.31% for *Byrsonima intermedia* species. The classical approach generated models with 42.42% of accuracy in average for the first species and 33.68% for the second one.

The difference between the results of each run was insignificant. It is because the Maximum Entropy algorithm is deterministic. Thus, it is not necessary to run the algorithm more than once. However, the algorithms were run 5 times for each species because a different set of background pixels is generated at each run. This different set of background pixels can generate small differences in the algorithm's statistics.

Since the AME approach tests all possible combinations of features for choosing the best one, it is an algorithm computationally expensive. Its complexity is exponential because as the number of features increase, the number of combinations grows exponentially. However, AME approach searches for the best combination at each iteration, removing or inserting features. Therefore, AME approach reaches the convergence faster than classical approach. That is why it ran faster than the classical one.

A significant reduction in the execution total time of the AME approach can be observed, approximately 33.5% faster than the classical algorithm, in average. The accuracy was about 6.7% better with AME approach for *Xylopia aromatica* species than the classical algorithm. However, the accuracy for *Byrsonima intermedia* species with AME approach was 40.19% better than the classical one. These results indicate that the proposed strategy showed to be adequate for modeling species geographical distribution.

Figures 5 and 6 show a distribution model for the *Xylopia aromatica* and *Byrsonima intermedia* species, respectively, generated by the AME algorithm.

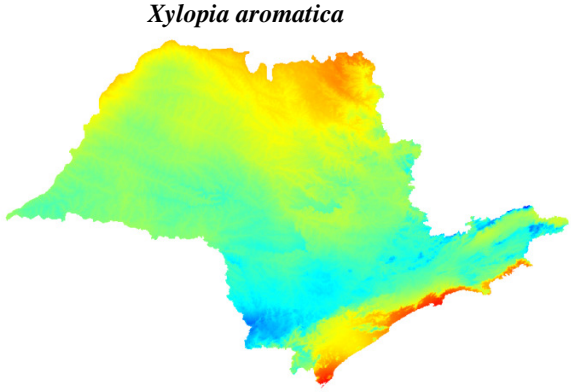


Fig. 5. Distribution model of *Xylopia aromatica*.

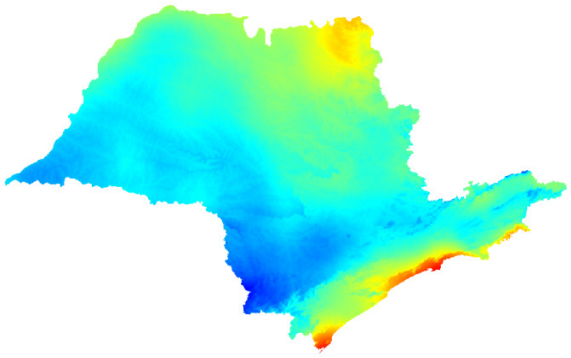


Fig. 6. Distribution model of B*yrsonima intermedia*.

The hot colors in the map represent more suitable environmental conditions for the species. Both distribution models are similar because the species are from the same biome. Thus, the environmental conditions for the species survival are the same. However, the biological analysis of the models generated is beyond the scope of this paper.

VII. Final Discussion and Future Works

The aim was to present a new Adaptive Maximum Entropy approach for modeling biological species geographical distribution. Besides the AME approach integrated to the openModeller tool, some experiments were carried out to show that this approach works as well as the classical algorithm.

With the experiments carried out, it was possible to validate the use of an Adaptive Maximum Entropy approach for modeling biological species geographical distribution, since the results obtained were very significant. It was observed that, for this data set, the AME approach was about 33.5% faster than the classical algorithm, in average. Besides this, the accuracy had an increasing of 6.7% for *Xylopia aromatica* species and 40.19% for *Byrsonima intermedia* species.

These results obtained motivate the continuity of the researches in this area. Since the increasing in accuracy was very different for the two used species, one of the future works will be to test the proposed approach with a larger set of species.

Besides the improvement of the developed approach, another future work is the parallelization of this algorithm. The openModeller project has a cluster with 80 cores and there are just a few algorithms running there. The MPI library (Message Passing Interface) will be used for this implementation. Some algorithms available in the openModeller tool were already parallelized using this library. Thus, the same strategy will be used to guarantee the

Byrsonima intermedia

compatibility. In the parallelization of AME approach, several processes will work at the same time computing the log loss of a different set of features.

Acknowledgment

Thanks to biologist Marinez Ferreira de Siqueira from the Botanical Garden Research Institute of Rio de Janeiro for suggesting the most suitable environmental layers for the studied species.

References

[1] P. Avesani, A. Perini and F. Ricci. Interactive case-based planning for forest fire management. Applied Intelligence 13, pp. 41–57, 2000.

[2] E. Kalapanidas and N. Avouris. Short-term air quality prediction using a case-based classifier. Environmental Modeling and Software, Volume 16, Number 3, pp. 263-272, 2001.

[3] J. M. P. King, R. Bañares-Alcántara and Z. A. Manan. Minimising environmental impact using CBR: An azeotropic distillation case study. Environmental Modelling and Software, Volume 14, Number 5, pp. 359–366, 1999.

[4] J. S. Pedro, F. Burstein and A. Sharp. A case-based fuzzy multicriteria decision support model for tropical cyclone forecasting. European Journal of Operational Research, 160, pp. 308–324, 2005.

[5] B. D. Mahaman, H. C. Passam, A. B. Sideridis and C. P. Yialouris. DIARES-IPM: a diagnostic advisory rule-based expert system for integrated pest management in Solanaceous crop systems. Agricultural Systems, Volume 76, Number 3, pp. 1119–1135, 2003.

[6] E. El-Azhary, H. A. Hassan and A. Rafea. Pest control expert system for tomato (PCEST). Knowledge and Information Systems, Volume 2, Number 2, pp. 242–257, 2000.

[7] B. D. Mahaman, P. Harizanis, I. Filis, E. Antonopoulou, C. P. Yialouris and A. B. Sideridis. A diagnostic expert system for honeybee pests. Computers and Electronics in Agriculture, Volume 36, Number 1, pp. 17–31, 2002.

[8] F. Zetian, X. Feng, Z. Yun and Z. XiaoShuan. Pig-vet: a web-based expert system for pig disease diagnosis. Expert Systems with Applications, Volume 29, Issue 1, pp. 93–103, 2005.

[9] L. S. Iliadis and F. Maris. An Artificial Neural Network model for mountainous water-resources management: The case of Cyprus mountainous watersheds. Environmental Modelling & Software, Volume 22, Issue 7, pp. 1066–1072, 2007.

[10] Z. Qing and S. J. Stanley. Forecasting raw-water quality parameters for the North Saskatchewan River by neural network modeling. Water Research, Volume 31, Number 9, pp. 2340–2350, 1997.

[11] D. Pozdnyakov, R. Shuchman, A. Korosov and C. Hatt. Operational algorithm for the retrieval of water quality in the Great Lakes. Remote Sensing of Environment, Volume 97, Issue 3, pp. 352–370, 2005.

[12] S. I. V. Sousa, F. G. Martins, M. C. M. Alvim-Ferraz and M. C. Pereira. Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. Environmental Modelling & Software,

Volume 22, Number 1, pp. 97–103, 2007.

[13] H. R. Maier, G. C. Dandy and M. D. Burch. Use of artificial neural networks for modeling cyanobacteria *Anabaena* spp. In the River Murray, South Australia. Ecological Modeling 105, pp. 257–272, 1998.

[14] F. A. Rodrigues, A. O. Avilla, E. S. C. Rodrigues, P. L. P. Corrêa, A. M. Saraiva and R. L. A. da Rocha. Species distribution modeling with neural networks. e-Biosphere 2009, London-UK, 2009. Retrieved August 27, 2009, from http://www.e-biosphere09.org/assets/files/e-Biosphere%20Abstracts%20Volume%20-%20FINAL.pdf.

[15] L. Persona, P. L. P. Corrêa and A. M. Saraiva. Environmental Niche Modeling in Biodiversity with Genetic Algorithms. In: 2nd International Information and 68 Telecommunication Technologies Symposium, Florianópolis. Proceedings of the IEEE - I2TS'2003, 2003. (In Portuguese).

[16] E. Kalapanidas and N. Avouris. Feature selection for air quality forecasting: a genetic algorithm approach. AI Communications, Volume 16, Issue 4, pp. 235–251, 2003.

[17] G. J. Pelletier, S. C. Chapra and H. Tao. QUAL2Kw – a framework for modeling water quality in streams and rivers using a genetic algorithm for calibration. Environmental Modelling & Software, Volume 21, pp. 419–425, 2006.

[18] A. V. M. Ines, K. Honda, A. das Gupta, P. Droogers and R. S. Clemente. Combining remote sensing-simulation modeling and genetic algorithm optimization to explore water management options in irrigated agriculture. Agricultural Water Management, Volume 83, Number 3, pp. 221–232, 2006.

[19] B. Schönfisch and M. Kinder. A fish migration model. Lecture Notes in Computer Science, Volume 2493, pp. 210–219, 2002.

[20] S. El Yacoubi, A. El Jai, P. Jacewicz and J. G. Pausas. LUCAS: an original tool for landscape modeling. Environmental Modelling and Software, Volume 18, Number 5, pp. 429–437, 2003.

[21] I. G. Georgoudas, G. C. Sirakoulis, E. M. Scordilis and I. Andreadis. A cellular automaton simulation tool for modeling seismicity in the region of Xanthi. Environmental Modelling & Software, Volume 22, Issue 10, pp. 1455–1464, 2007.

[22] M. Schluter and N. Ruger. Application of a GIS-based simulation tool to illustrate implications of uncertainties for water management in the Amudarya river delta. Environmental Modelling & Software, Volume 22, Issue 2, pp. 158–166, 2007.

[23] G. Metternicht and S. Gonzalez. FUERO: foundations of a fuzzy exploratory model for soil erosion hazard prediction. Environmental Modelling & Software, Volume 20, Issue 6, pp. 715–728, 2005.

[24] L. S. Iliadis. A decision support system applying an integrated fuzzy model for long-term forest fire risk estimation. Environmental Modelling & Sofrtware, Volume 20, Issue 5, pp. 613–621, 2005.

[25] F. Bousquet and C. Le Page. Multi-agent simulations and ecosystem management: a review. Ecological Modelling 176, pp. 313–332, 2004.

[26] D. Borri, D. Camarda and A. De Liddo. Mobility in environmental planning: an integrated multi-agent approach. Lecture Notes in Computer Science, Volume 3675, pp. 119–129, 2005.

[27] H. Purnomo, G. A. Mendoza, R. Prabhu and Y. Yasmi. Developing multi-stakeholder forest management scenarios: a multi-agent system simulation approach applied in Indonesia. Forest Policy and Economics, Volume 7, Issue 4, pp. 475–491, 2005.

[28] S. H. Chen, A. J. Jakeman and J. P. Norton. Artificial Intelligence techniques: An introduction to their use for modelling environmental systems. Mathematics and Computers in Simulation, Volume 78, Issue 2–3, pp. 379–400, 2008.

[29] M. E. S. Muñoz, R. Giovanni, M. F. Siqueira, T. Sutton, P. Brewer, R. S. Pereira, D. A. L. Canhos and V. P. Canhos. openModeller: a generic approach to species' potential distribution modeling. GeoInformatica. DOI: 10.1007/s10707-009-0090-7. 2009.

[30] C. B. Pariente, J. J. Neto and F. S. Santana. Towards an Adaptive Implementation of Genetic Algorithms. I Taller Latinoamericano de Informática para la Biodiversidad (INBI) - CLEI 2007, San José, Costa Rica, 9-12 Octubre, 2007.

[31] F. A. Rodrigues, E. S. C. Rodrigues, L. M. Sato, E. T. Midorikawa, P. L. P. Corrêa and A. M. Saraiva. Parallelization of the Jackknife Algorithm Applied to a Biodiversity Modeling System. Proceedings of the 7th International Information and Telecommunication Technologies Symposium – I2TS'2008, pp. 58–65, 2008.

[32] C. H. Graham, S. Ferrier, F. Huettman, C. Moritz and A. T. Peterson. New developments in museum-based informatics and applications in biodiversity analysis. Trends in Ecology & Evolution, Volume 19, Number 9, pp. 497–503, 2004.

[33] G. E Hutchinson, Introduction to Ecology of Populations. Barcelona, Editorial Blume, 492p., 1981. (In Spanish)

[34] S. J. Phillips, R. P. Anderson and R. E. Schapire, Maximum entropy modeling of species geographic distributions. Ecological Modelling 190: 231 – 259, 2006.

[35] M. F. Siqueira, Use of Fundamental Niche Modeling in the Pattern Evaluation of Vegetal Species Geographic Distribution. PhD Thesis. Department of Ambient Engineering of University of Sao Carlos. Sao Carlos, Sao Paulo, 2005. (In Portuguese)

[36] R. S. Pereira. DesktopGarp, 2002. Retrieved August 27, 2009, from http://www.nhm.ku.edu/desktopgarp/index.html

[37] S. J. Phillips, M. Dudík and R. E. Scaphire. Maxent, 2009. Retrieved August 27, 2009, from http://www.cs.princeton.edu/~schapire/maxent

[38] T. Sutton, R. Giovanni and M. F. Siqueira. Introducing openModeller – a fundamental niche modeling framework.

OSGEO Journal, Vol.1., 2007. Retrieved August 27, 2009, from http://www.osgeo.org/ojs/index.php/journal/article/viewFile/109/90.

[39] D. Stockwell and D. Peters. The GARP modelling system: problems and solutions to automated spatial prediction. International Journal of Geographical Information Science, vol. 13, no. 2, 143–158, 1999.

[40] D. R. B. Stockwell and I. R. Noble. Induction of sets of rules from animal distribution data: a robust and informative method of data analysis. Mathematics and Computers in Simulation, 32, 249–254, 1991.

[41] D. R. B. Stockwell. Machine learning and the problem of prediction and explanation in ecological modelling. Doctoral Thesis, Australian National University, Australia, 1992.

[42] S. J. Phillips, M. Dudík and R. E. Schapire. A Maximum Entropy Approach to Species Distribution Modeling. Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[43] V. Vapinik. The Nature of Statistical Learning Theory. Springer-Verlag, 1995.

[44] A. C. Lorena, M. F. Siqueira, R. Giovanni, A. C. P. L. F. Carvalho and R. C. Prati. Potential Distribution Modelling Using Machine Learning Classifiers. In: The Twenty First International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, 2008, Wroclaw. Lecture Notes in Artificial Intelligence - Proceedings The Twenty First International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems. v. 5027. p. 255–264, 2008.

[45] S. Haykin. Neural Networks and Learning Machines. Prentice Hall, 3rd edition, 2008.

[46] J. Elith, C. H. Graham, R. P. Anderson, M. Dudík, S. Ferrier, A. Guisan, R. J. Hijmans, F. Huettmann, J. R. Leathwick, A. Lehmann, J. Li, L. G. Lohmann, B. A. Loiselle, G. Manion, C. Moritz, M. Nakamura, Y. Nakazawa, J. McC. Overton, A. T. Peterson, S. J. Phillips, K. S. Richardson, R. P. Scachetti, R. E. Schapire, J. Soberón, S. Williams, M. S. Wisz and N. E. Zimmermann. Novel methods improve predition of species' distributions from occurrence data. Ecography 29, pp. 129–151, 2006.

[47] J. J. Neto. A Survey of the Adaptivity Evolution and Adaptive Technology. IEEE Magazine – Latin America, Vol. 5, Num. 7, ISSN: 1548–0992, pp. 496 – 505, 2007. (In Portuguese)

[48] J. J. Neto. A Glossary about Adaptivity. Memories of WTA'2009: third Workshop of Adaptive Technology. ISBN 978-85-86686-51-1. São Paulo: s.n., 2009. (In Portuguese)

[49] J. J. Neto. Contributions to Methodology of Compilers Construction. Thesis of Free Professor, Engineering School of USP, São Paulo, 1993. (In Portuguese)

[50] R. L. A. Rocha and J. J. Neto. Adaptive Automata, limits and complexity in comparison with Turing Machine. In: Proceedings of the Second Congress of Logic Applied to Technology – LAPTEC'2000, São Paulo, pp. 33 – 48, 2001. (In Portuguese)

[51] J. J. Neto. Solving Problems Efficiently with Adaptive Automata. CIAA 2000 – Fifth International Conference on Implementation and Application of Automata, London, Ontário, Canadá, July 2000.

[52] J. J. Neto. Adaptive Rule-Driven Devices – General Formulation and Case Study", Lecture Notes in Computer Science, Watson, B. W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol. 2494, Pretoria, South África, Springer Verlag, pp. 234 – 250, July 23 – 25, 2002.

[53] E. T. Jaynes. Information theory and statistical mechanics. Physical Review, vol. 106, pp. 620–630, 1957a.

[54] E. T. Jaynes. Information theory and statistical mechanics II. Physical Review, vol. 108, pp. 171–190,1957b.

[55] A. L. Berger, V. J. D. Pietra and S. A. D. Pietra. A maximum entropy approach to natural language processing. Computational Linguistic, Volume 22, Issue1, pp. 39–71, 1996.

[56] S. J. Phillips and M. Dudík. Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. Ecography 31: 161–175, 2008.

[57] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002), pp. 49–55, 2002.

[58] M. Dudík, S. J. Phillips and R. E. Schapire. Performance Guarantees for Regularized Maximum Entropy Density Estimation. In Proceedings of the 17th Annual Conference on Computational Learning Theory, ACM Press, New York, pp. 655–662, 2004.

[59] SinBiota – Environmental Information System for Biota. [on-line] available in: http://sinbiota.cria.org.br/

[60] R. J. Hijmans, S. E. Cameron, J. L. Parra, P. G. Jones and A. Jarvis. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25: 1965 – 1978.

[61] MANPAGE OF TIME. Searched in 12/11/2007. [on-line] available in http://www.ncsa.uiuc.edu/~gbauer/time.1.html.

BIOGRAPHIES

Elisângela S. C. Rodrigues obtained her degree in Computer Science from the Federal University of Pelotas in 2000 and her master's degree in Computer Science from the Federal University of Campina Grande in 2002. She initiated her teaching activities in 2002, at the Faculty of Technology and Sciences of Feira de Santana/BA, where she acted until 2005. In the same year, she moved to College ÁREA1 in Salvador/BA, where acted until 2007. In 2006, she participated as researcher (supported by Petrobrás) in the Automated Well Management System Project (SGPA), at the Federal University of Bahia. Currently, she is a regular

doctorate candidate at the Engineering School of University of São Paulo.

Fabrício A. Rodrigues obtained his degree in Information Systems from the Potiguar University in 2000 and his master's degree in Computer Science from the Federal University of Campina Grande in 2002. He initiated his teaching activities in 2002, at the Faculty of Technology and Sciences in Feira de Santana/BA, where he acted until 2005. In the same year, he moved to College ÁREA1 in Salvador/BA, where taught until 2007, besides acting as the coordinator of the Computation Nucleus. Currently, he is a regular doctorate candidate at the Engineering School of University of São Paulo.

Ricardo L. A. da Rocha was born in Rio de Janeiro / RJ in 29/05/1960. He obtained his degree in Electronics Engineering, focus on Digital and Computer Systems, from the Pontifícia Universidade Católica do Rio de Janeiro - PUC-RJ, in 1982. He was granted his MSc (1995) and PhD (2000) in Computing Engineering from the Engineering School of the University of Sao Paulo - EPUSP. He is currently a professor at the Engineering School, and his research interests are Adaptive Techniques and Technologies, Computational Models, Intelligent Systems, Kolmogorov Complexity and Foundations of the Theory of Computation. Dr. Rocha has been a member of the Association for Computing Machinery – ACM since 2001, and also of the Brazilian Computing Society - SBC since 2000.

Pedro L. P. Corrêa obtained his degree in Computer Science in 1987 - ICMSC / USP / Brazil, his master's degree in Computer Science in 1992 - ICMSC / USP / Brazil, his Ph.D. in Electrical Engineering in 2002 - EPUSP / Brazil. He worked as consultant in Development Systems with companies in the financial areas, through the Interchange Services (company of EDI - Electronic Data Interchange - Brazil) and with the coordination of teams and systems design for the electronic government Secretary of the State of São Paulo, through the UNDP (UN Development Program) from 1998 to 2002. He is currently a professor at the Engineering School - University of São Paulo, at the Department of Computer Engineering. Areas of interest: Distributed Database, Information Systems , Biodiversity Informatics, Service Oriented Achitecture and Electronic Government.

Investigação empírica dos comportamentos de um conjunto de autômatos finitos adaptativos

(21 Janeiro 2010)

Nicolau Leal Werneck

***Resumo*— Este artigo relata um experimento realizado com autômatos finitos adaptativos. Realizou-se uma varredura em um conjunto destes autômatos onde seus comportamentos individuais foram analisados. Cada autômato produziu uma sequência de números, e estas foram classificadas de acordo com suas regras de produção aritméticas. As estatísticas revelam quais comportamentos são os mais comuns neste conjunto, e quais são os mais complexos. Especial atenção foi dada para uma série específica que aparenta não possuir uma fórmula fechada, e possui apenas uma relação de recursão relativamente complexa. O artigo conclui com a proposta de pontos de partida para novos estudos, como a investigação dedicada de autômatos finitos adaptativos com máquinas anulares, e buscar responder questões mais teóricas, como a da possibilidade de prever-se o comportamento de um autômato sem executar sua simulação.**

***Palavras chave*— Autômatos finitos adaptativos (*adaptive finite automata*), séries numéricas, teoria dos números, sistemas dinâmicos.**

I. INTRODUÇÃO

ESTE artigo apresenta resultados de uma investigação empírica das possibilidades oferecidas por autômatos finitos adaptativos (AFAs). Este é um estudo realizado de um ponto de vista próximo ao de sistemas dinâmicos e teoria dos números, similar ao que também é feito no estudo de outros temas da ciência da computação, como os autômatos celulares.

Uma grande inspiração para esta pesquisa foi justamente o estudo de autômatos celulares realizado por Stephen Wolfram nos anos 1980. Uma das investigações realizadas por este pesquisador foi a varredura de todas as possíveis regras para autômatos celulares unidimensionais, binários, e com vizinhança de uma única célula para cada direção. Foram realizados testes para cada um dos autômatos possíveis, e a análise dos dados produzidos permitiu identificar o comportamento causado por cada diferente regra [1].

Muitos autômatos apresentaram apenas um comportamento trivial ou pouco interessante, como regimes estacionários ou ciclos muito curtos, porém alguns autômatos como as regras nomeadas 110 e 90 possuem comportamentos de características notáveis. Em 2004 foi apresentada uma prova de que a regra 110 possui poder de computação universal [2]. Este autômato é uma das entidades mais simples que se conhece a apresentar esta característica.

Este artigo apresenta resultados de uma investigação semelhante realizada com autômatos finitos adaptativos. Assim como nos trabalhos de Wolfram e outros similares, foi criada uma forma de associar definições de AFAs com números inteiros, de forma que a seqüência de zero até um limite varresse um conjunto de máquinas com características semelhantes. O resultado da execução da máquina partindo de uma mesma configuração inicial foi então investigado para revelar que tipos de comportamentos podem ser encontrados, e com que frequência.

O alfabeto de entrada dos autômatos estudados é constituído por um único símbolo. Conforme a cadeia vai sendo consumida, uma máquina pode retornar ou não para o estado final, reconhecendo a cadeia naquele ponto. Se olharmos para o número de símbolos consumidos a cada momento que a máquina reconhece a entrada, podemos considerar que a máquina está reconhecendo números que pertencem a uma série, que estão sendo escritos em base unária na entrada da máquina.

Estas séries de números reconhecidas por cada máquina são o alvo principal das análises neste trabalho. Entre estas séries encontramos basicamente polinômios, expressões com exponenciais, e ainda séries periódicas. Algumas séries, no entanto, não podem ser descritas de maneira trivial. Apresentaremos com detalhes uma destas séries, para a qual foi determinada uma expressão apenas para a função de sua geração por recursão.

II. METODOLOGIA

Os testes foram realizados com um programa criado a partir da biblioteca Adaptóide [3]. Os autômatos que esta biblioteca é capaz de simular possuem as seguintes características:

i. Transições da máquina podem ou não possuir uma função adaptativa associada, ativada no momento em que se verificam as condições para a transição.

ii. Cada função adaptativa pode ser aplicada antes ou depois da transição se realizar. Ou seja, podem ser definidas como do tipo "pré" ou "pós".

iii. Funções adaptativas são seqüências de ações adaptativas realizadas em ordem e de efeito imediato.

iv. Cada ação adaptativa comum é uma redefinição do destino de uma certa transição da máquina, e da função

N. Werneck é membro do LTI da Escola Politécnica da USP, e recebe bolsa de doutorado da CAPES (e-mail: nwerneck@gmail.com).

associada a esta transição.

v. Os estados só podem ser referenciados pelas ações adaptativas de forma relativa ao estado atual da execução da máquina. Para isto utiliza-se a máquina como um autômato finito comum. Listas de símbolos especificadas são consumidas partindo-se do estado atual da máquina, e o estado atingido ao final da lista é o estado referenciado.

vi. A criação de um novo estado é a segunda forma de função adaptativa. O último estado criado também pode ser referenciado nas ações adaptativas.

Estas características dos autômatos que podem ser simulados pela biblioteca Adaptóide foram ainda complementadas por mais restrições criadas para a realização dos testes. Os programas que foram estudados são todos os que possuem as seguintes características:

i. Exatamente duas funções adaptativas são definidas, rotuladas como 'F' e 'G'. Elas podem ser tanto de aplicação anterior quanto posterior à transição. Suas ações adaptativas são a criação de um estado novo ao princípio, seguida de duas definições de transição.

ii. Apenas quatro estados diferentes podem ser referenciados pelas ações adaptativas: o estado recém criado, o estado da máquina onde a função foi ativada, ou ainda os estados atingidos a partir deste por uma ou duas transições. As novas transições definidas podem ainda ser ou não associadas a uma das duas funções adaptativas.

iii. A execução principia com uma máquina de apenas um estado, que é um estado de aceitação, com uma transição para si mesmo associada à função adaptativa 'F'.

Existem no total $4\times4\times3$ ações adaptativas que podem ser realizadas. O quadrado deste número multiplicado por dois é o número de funções adaptativas possíveis, e o quadrado deste é o número de máquinas analisadas nesta pesquisa: 21.233.664. Em teste realizado com um computador de mesa moderno de potência considerável foi possível realizar uma análise superficial das máquinas a uma taxa de aproximadamente 40 por segundo. A varredura de todas as máquinas possíveis levaria portanto em torno de uma semana de execução ininterrupta.

III. COMPORTAMENTOS OBSERVADOS

Nesta seção serão descritos os diferentes tipos de comportamento observados na análise das séries numéricas produzidas pelas máquinas estudadas.

A. Regimes estacionários

Um grande número das máquinas estudadas converge rapidamente para o estado inicial, aceitando assim todos os números ao menos a partir de um limite inferior, ou ainda deixa de retornar ao estado inicial, e não parece aceitar novamente qualquer outra entrada. Foram observadas nos testes apenas 100 transições, porém não há ainda prova formal de que este valor seria o suficiente para determinar com certeza se estas máquinas realmente entraram em estado estacionário apesar de não se observarem mais mudanças.

Algumas máquinas apresentaram transitórios relativamente longos antes de entrar em estado estacionário. Foram encontradas 4.787 máquinas em que o estado inicial foi visitado pela última vez após 10 transições, e antes de 20. Por outro lado, todas as máquinas encontradas que convergem para o estado inicial o fizeram em no máximo 10 transições.

B. Osciladores

Um grande número de máquinas oscila entre o estado inicial e outros, criando reconhecedores de números pares ou ímpares. Note que estamos sempre desconsiderando um possível período transitório inicial. Osciladores de três tempos também são bastante comuns, tanto com uma visita ao estado final a cada período, quanto com duas.

Máquinas estacionárias e osciladores de 2 e 3 tempos formam uma boa parte das máquinas estudadas. De todos os milhões de máquinas estudadas, menos de 215.000 não eram nem estacionárias nem osciladores de 2 ou 3 transições por período.

Foram encontrados ainda osciladores de 4 até 8 tempos, e ainda alguns de 10 e 16 tempos. A Tab. 1 apresenta quantas máquinas periódicas foram encontradas para cada período de oscilação, e em quantos diferentes tipos.

Período	Tipos	Quantidade
4	3	28.177
5	3	4.815
6	4	1.803
7	3	700
8	1	533
10	2	90
16	1	49

Tab. 1. Distribuição de osciladores encontrados.

A Fig. 1 apresenta o funcionamento do oscilador de 8 tempos. O quadrado pontilhado indica o estado atual da máquina em cada instante. Note como no quinto e no nono instantes um par de estados acaba sendo desligado da máquina. Estes estados permanecem na memória, mas jamais são acessados novamente. O nono instante corresponde ao primeiro, porém os estados excedentes sendo desligados no primeiro instante foram omitidos na Fig. 1 para fins didáticos.

C. Sequências polinomiais

Todas as máquinas encontradas que reconhecem séries aperiódicas funcionam de uma maneira semelhante. Após um período inicial as máquinas adquirem uma estrutura circular, e novos estados vão sendo acrescidos a este laço com o passar do tempo. Cada vez que a máquina retorna ao estado inicial, um número é reconhecido, que é o número de transições que ocorreram. A Fig. 2 ilustra a estrutura circular destas máquinas.

Uma técnica que se mostra sempre útil no estudo de séries numéricas é analisar as diferenças progressivas entre os números, ou os incrementos necessários para formar os números sucessores a partir de cada número da série. Nestes autômatos estudados estas diferenças muitas vezes possuem uma contrapartida digna de nota: elas se relacionam com tamanho em que se encontra a máquina ao fim de cada volta.

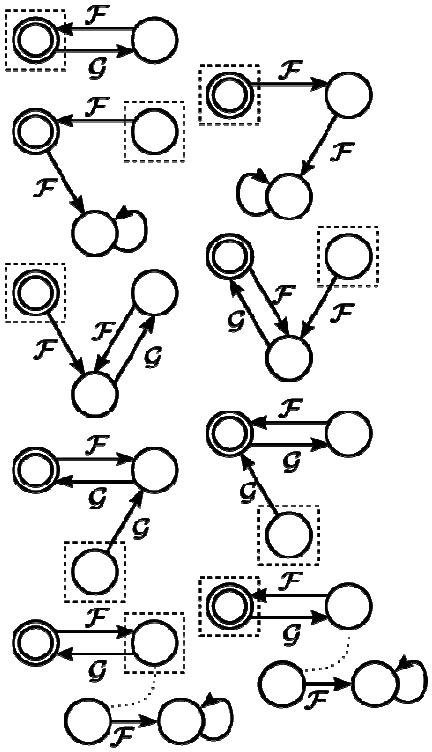


Fig. 1. Funcionamento do AFA oscilador de 8 tempos.

A classe de séries numéricas aperiódicas mais simples que foram encontradas foi a de polinômios. Uma das máquinas mais comuns é a capaz de reconhecer números triangulares, caracterizados por $f(n) = (n^2 + n)/2$, ou $f(n) = f(n-1) + n$. A cada volta completa, ocorre nestas máquinas apenas o acréscimo de um único estado ao anel. Foram encontradas 69.366 máquinas que reconhecem séries triangulares, ou com variações simples tal como a adição de um valor constante.

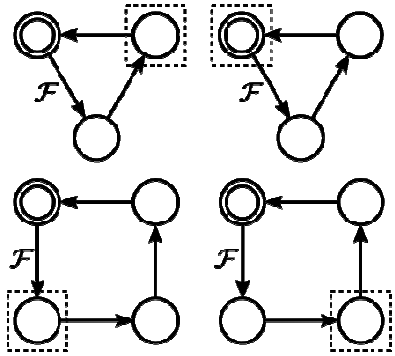


Fig. 2. Exemplo de AFA ligado a séries polinomiais. Um reconhecimento devido ao número triangular 6 ocorre no segundo momento.

Outras duas máquinas reconhecedoras de séries polinomiais foram também encontradas. Uma única máquina foi encontrada que reconhece os números da série dada por $f(n)=(n^2+7n)/2$. Ela também reconhece adicionalmente o número 1. Três máquinas também foram encontradas que reconhecem valores de polinômios de terceira ordem. Duas reconhecem o polinômio $(n^3+5n)/6$, e uma variante reconhece os valores subtraídos por 1.

D. Seqüências exponenciais

Outra classe de séries de números inteiros que foi encontrada com bastante frequência nesta pesquisa foi a de expressões envolvendo exponenciais. Máquinas que reconhecem as potências de 2 são bastante comuns, tendo sido encontradas 67.939 vezes na varredura. Algumas outras séries exponenciais encontradas foram séries como a integração dupla de 2^n acrescida de constantes, exponenciais na base 2 com cofatores como 3 e 10, e ainda exponenciais acrescidas de termos polinomiais.

Duas interessantes séries exponenciais encontradas merecem destaque. Uma delas é reconhecida por duas máquinas, que diferem pela função G ser do tipo pós em uma e pré em outra. Ela é dada pela fórmula

$$\left[\left(1+\sqrt{2}\right)^{n+2} - \left(1-\sqrt{2}\right)^{n+2}\right] \Big/ \sqrt{2} - 1$$

Outras séries exponenciais interessantes encontradas foram $F(n)-n$ onde $F(n)$ indica a série de Fibonacci [4], e ainda esta mesma série com os valores incrementados por 1 [5].

E. Seqüências inexatas

Algumas das séries encontradas possuem valores que se aproximam de funções polinomiais ou exponenciais, porém são construídas de maneiras que impedem que valores exatos possam ser calculados. Dois tipos de séries com fórmulas inexatas foram encontrados.

O tipo mais comum são séries que possuem similaridades com máquinas de séries exponenciais ou polinomiais, porém há uma espécie de chaveamento que alternadamente habilita e desabilita o comportamento que fabricaria uma série correta. Isto pode ser verificado analizando-se as diferenças sucessivas dos valores da série. Ao invés de encontrar valores típicos, encontramos os valores típicos com repetições.

Uma outra classe de séries inexatas encontrada possui valores que se aproximam muito de uma função exponencial, porém o erro de arredondamento encontrado possui um aspecto errático, e não pode ser pré-determinado facilmente. Foram encontrados alguns milhares de máquinas que produzem tal tipo de comportamento, porém apenas uma foi estudada de forma mais aprofundada até o momento.

A Fig. 3 apresenta algumas das transições da máquina estudada. Cada cadeia nesta figura representa um instante de desenvolvimento da máquina. A estrutura é sempre circular, e a cadeia mostra a máquina aberta a partir do estado atual. Os quadrados indicam a função adaptativa associada a cada transição, e as arestas entre eles são os estados da máquina. O estado de aceitação é representado por uma aresta cortada.

A série correspondente a este AFA inicia com os valores:

0 2 5 9 15 24 38 59 90 137 207 312 470 707 1062...

Suas diferenças sucessivas são dadas por:

2 3 4 6 9 14 21 31 47 70 105 158 237 355

É possível ajustar uma função exponencial para aproximar estes valores, obtendo-se a fórmula $73/40 \times (3/2)^n$. Os valores obtidos precisam porém ser arredondados para efetivamente produzir os valores de interesse.

Até o momento não foi encontrada uma fórmula fechada para

realizar o cálculo dos números aceitos pela máquina, mas apenas relações de recursão. Pode-se apenas calcular o tamanho da máquina pela exponencial com arredondamento para todas iterações até o ponto de interesse, e então somar.

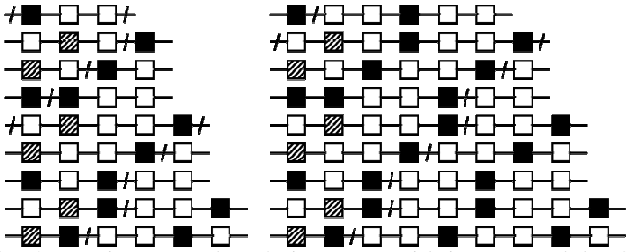


Fig. 3. Funcionamento da máquina exponencial inexata estudada. Os quadrados representam as funções adaptativas de cada transição. A aresta cortada indica um estado final.

O programa a seguir em Python calcula o tamanho da máquina a cada ciclo utilizando apenas com números naturais:

```
len,knot,n,k=(3, 3, 0, 0)
while True:
  if (len==knot):
    print len
    k+=1
  if (n%3==0): len+=1
  if (knot==1): knot = len
  elif (n%3!=0 ): knot-=1
  n+=1
```

Apenas incrementos e testes simples com álgebra modular são utilizados neste algoritmo. Apesar de ser mais sólida e possuir um aspecto mais elegante do que a fórmula com exponenciação e arredondamento, esta técnica exige não só que todos os valores do tamanho da máquina em cada volta sejam calculados para calcular a série, mas exige também que sejam calculados em ordem.

IV. CONCLUSÕES

Este artigo apresentou análises do comportamento de um grande grupo de autômatos finitos adaptativos que está sendo estudado exaustivamente. As máquinas foram geradas a partir de um gabarito em comum, variando-se parâmetros dentro de todas suas possibilidades. Cada máquina foi simulada a partir de um mesmo estado inicial, até a centésima transição, produzindo uma série de números inteiros característica que são os números das transições em que a máquina retornou ao estado inicial.

As séries encontradas até o momento puderam ser classificadas entre classes tradicionais que se estudam em Teoria dos Números, como séries periódicas, polinomiais e exponenciais. Foram ainda encontradas séries que não possuem fórmulas fechadas para seu cálculo, mas apenas fórmulas recursivas. Algumas destas fórmulas recursivas ainda se mostraram bastante complexas.

Entre as máquinas estudadas encontra-se um grande número poderiam ser facilmente reconhecidas como equivalentes. Por exemplo, toda a família de máquinas em que a função F jamais cria transições atreladas à função G independe da definição de G. Esta forma de independência se reflete justamente no número de máquinas diferentes que implementam um mesmo comportamento, dando uma noção do quão complexo aquele comportamento é. Máquinas mais simples podem ser implementadas de mais maneiras diferentes. Funções com poucas possibilidades de implementação possuem menos parâmetros independentes, e estão portanto explorando mais as possibilidades oferecidas pelo gabarito.

As análises realizadas aqui podem servir de ponto de partida para estudos maiores, com outras formas de gabarito, ou ainda estudos mais aprofundados das máquinas já avaliadas. O formato típico dos autômatos aperiódicos com anéis que crescem ao longo do tempo foi algo evidenciado nesta pesquisa e que poderia receber agora um estudo dedicado. Para tal seria útil determinar uma forma de gabarito que produza apenas máquinas com este comportamento, ou ainda um teste que determine *a priori* se uma a máquina é desta classe.

Outras melhorias ainda podem ser feitas nos próprios programas utilizados nas análises desta pesquisa. É preciso automatizar a classificação das séries entre estacionárias, periódicas, polinomiais, exponenciais e séries inexatas, e buscar igualmente maneiras de tentar prever este comportamento sem efetivamente emular o funcionamento do autômato. Estes desejos ainda suscitam naturalmente questionamentos de cunho teórico, como da própria possibilidade de se realizar este tipo de classificação *a priori*, e ainda da possibilidade de se construir automaticamente máquinas para reconhecer séries encomendadas.

REFERENCIAS

[1] S. Wolfram, "Statistical Mechanics of Cellular Automata.", Rev. Mod. Phys. 55, 601-644, 1983.

[2] M. Cook, "Universality in Elementary Cellular Automata", Complex Systems 15 (1) : p.1-40, 2004.

[3] N. Werneck, "Biblioteca para autômatos adaptativos com regras de transição armazenadas em objetos feita em C++", Segundo Workshop de Tecnologia Adaptativa, pp. 22–26, 2008.

[4] N. J. A. Sloane, Ed. (2008), The On-Line Encyclopedia of Integer Sequences, published electronically at www.research.att.com/~njas/sequences/, Sequence A001924

[5] N. J. A. Sloane, Ed. (2008), The On-Line Encyclopedia of Integer Sequences, Disponível em: http://www.research.att.com/~njas/sequences/, Sequence A065220

MODELO *NEURO-FUZZY* ADAPTATIVO PARA REPRESENTAÇÃO DE MOVIMENTOS DE UMA PRÓTESE DO SEGMENTO MÃO-BRAÇO (30 NOVEMBRO 2009)

M.M.G. Barreto, A. Balbinot

Resumo— Esse artigo descreve de forma breve um aparato experimental para controle de uma prótese de mão. Em seguida, apresenta um sistema de inferência neuro-*fuzzy* adaptativo como modelo teórico representativo do aparato experimental descrito. O sistema se mostra válido para determinado conjunto de valores compatível com o escopo definido para as variáveis e suas funções de pertinência, não apresentando margem de erro significativa. O sistema possui todas as regras lógicas apresentadas nos fluxogramas, correspondentes ao modelo formal do modelo experimental descrito, para os movimentos da prótese e outras regras para movimentos adicionais não descritos nos fluxogramas, que permitem a flexibilização dos movimentos da prótese.

Palabras clave— Controle Fuzzy; Engenharia Biomédica; Lógica Fuzzy; Modelagem de Sistemas fuzzy; Sistemas Adaptativos; Sistemas de Controle Biológicos; Sistemas Fuzzy.

Key words- Adaptative Systems; Biological Control Systems; Biological System Modeling; Biological Engineering; Fuzzy Control; Fuzzy Logic; Fuzzy System.

1 Introdução

O desenvolvimento de tecnologia assistiva para integrar partes do corpo humano, com dispositivos eletromecânicos, é um dos grandes desafios da ciência. Doenças neuromusculares, atrofia muscular, distrofia, amputações e paralisias dificultam o desempenho físico de muitas pessoas. Em benefício dessa parcela importante da população é importante o desenvolvimento de tecnologia assistiva para facilitar, auxiliar, controlar e permitir a comunicação com o ambiente a que estão inseridas. A área relacionada ao controle de próteses, por biosinais é parte da grande área de pesquisa HCI (*humam-computer interfacing*). Este tipo de sistema tem por função manipular os biosinais para controlar alguns sistemas externos com o objetivo de melhorar a vida de pessoas com desordens
ou distúrbios neuromusculares, tais como, lesões na coluna vertebral, amiotropia lateral, esclerose, paralisia do tronco cerebral [1].

Segundo dados estatísticos de 2000, existem cerca de 6,5 milhões de pessoas portadoras de deficiência motora no mundo. Principalmente amputações, lesões medulares, paralisia cerebral, entre outras, que atingem, sobretudo, jovens e adultos em idade produtiva. Assim sendo, a protetização é parte fundamental no esforço por uma assistência integral às pessoas portadoras de deficiência, ou seja, com o desenvolvimento de pesquisas em Tecnologia Assistiva, de novas tecnologias, de sistemas adaptativos, de sistemas de menor custo, de sistemas com tecnologia nacional são elementos chaves para o reingresso rápido desse indivíduo ao meio social. No Brasil, segundo dados do IBGE, pessoas com necessidades especiais correspondem a 24,6 milhões da população [2]. Portanto, a aplicação e disseminação de Tecnologia Assistiva deveria ser promovida e viabilizada para beneficiar parcela significativa da nossa população.

Esse é um campo em aberto e fecundo, onde diferentes soluções têm sido avaliadas e muitos resultados são considerados ainda parciais como é mostrado em [3], [4] e [1]. Lógica *fuzzy*, conjuntos *fuzzy*, sistemas *fuzzy*, e redes neurais *fuzzy* constituem hoje técnicas avançadas de inteligência computacional. Existem aplicações efetivas de métodos *fuzzy* em diagnóstico em cardiologia e de doenças do coração, em análise de eletromiogramas, em análise de eletroencefalogramas. Especificamente com relação à caracterização de sinais eletromiográficos e ao desenvolvimento de próteses, muitos resultados parciais também têm sido apresentados e o uso da lógica *fuzzy* se torna cada vez mais freqüente como se pode verificar em [1] e em [5]

A lógica *fuzzy* parece, assim, constituir uma metodologia eficiente para caracterização de sinais biológicos em sua complexidade. A Teoria do Raciocínio Aproximado, que constitui a base teórica da lógica *fuzzy* é um ambiente formal adequado ao desenvolvimento de raciocínios incertos, ambíguos, que contenham lacunas, potenciais inconsistências e principalmente complexidade, que parece ser o caso dessas áreas. A esse respeito ver [6], [7] e [8].

Por outro lado, o desenvolvimento de modelos formais que reproduzam experiências reais é um dos objetivos da inteligência computacional. Nesse aspecto os sistemas adaptativos exercem um importante papel por sua capacidade

M.M.G. Barreto trabalha no Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas, UFABC, 09210-971, Avenida dos Estados, 5001 – Bloco B – Sala 929 - Bairro Bangu, Santo André, SP. E-mail: mara.barreto@ufabc.edu.br.
A. Balbinot trabalha na Escola de Engenharia, no Departamento de Engenharia Elétrica, Laboratório de Instrumentação Eletro-Eletrônica, UFRGS 90035-190, Avenida Osvaldo Aranha, 103 – Sala 206, Porto Alegre, RS. E-mail: alexandre.balbinot@ufrgs.br.

de ajuste e auto-modificação. Os sistemas adaptativos neuro-*fuzzy* são capazes de representar modelos complexos da realidade com uma grande capacidade de variação.

Este artigo descreve de forma breve o sistema experimental que realiza os movimentos de abertura/fechamento da mão e rotação de punho (sentidos anti-horário e horário) de uma protese experimental do segmento mão-braço, gerenciados por biosinais do sistema muscular, denominados de sinais mioelétricos, apresenta e valida um sistema de inferência neuro-*fuzzy* adaptativo desenvolvido para representar esse modelo experimental e reproduzir os *outputs* referentes aos movimentos esperados.

2 Aparato Experimental

O sinal mioelétrico é proveniente do potencial de ação que percorre a fibra muscular levando-a a contração (Ortolan, 2002). A eletromiografia (EMG) é considerada o estudo das funções musculares através da captação do sinal mioelétrico através de equipamentos (chamados de eletromiógrafos) que detectam, amplificam e apresentam o sinal mioelétrico em forma gráfica. A Figura 1 apresenta o digrama de blocos do sistema desenvolvido em [2].

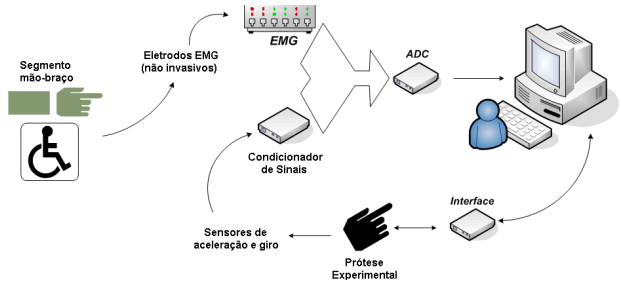


Figura 1. Diagrama de blocos do sistema experimental desenvolvido.

Eletrodos não invasivos são posicionados sobre um dos grupos musculares responsável pelo movimento da mão: flexor superficial dos dedos. Esses eletrodos recolhem os sinais mioelétricos que são amplificados e filtrados pelo eletromiógrafo de oito canais de configuração bipolar. O eletromiógrafo é um equipamento constituído de amplificadores de instrumentação e um banco de filtros adequados a aplicação. A placa de aquisição de dados (denominada de ADC) é a PCI-6024E com dezesseis canais de entrada com 12bits de resolução e duas saídas analógicas (denominada de Interface) com 12bits de resolução - atendendo assim todas as necessidades do sistema completo. Sensores para caracterização de aceleração (inclinação) e giro são utilizados como referência para o posicionamento espacial da prótese experimental. Para testar o sistema experimental foram realizados diversos ensaios de movimentos com relação à mão aberta, aperto leve e mão fechada.

Os sinais derivados do eletromiógrafo são armazenados durante um intervalo pré-determinado e filtrados digitalmente. Posteriormente, alguns parâmetros estatísticos, como por exemplo, valor r.m.s., a integral do sinal mioelétrico bruto (iEMG), entre outros, são obtidos para em seqüência controlar os movimentos da prótese de mão desenvolvida. A Figura 2 mostra algumas situações experimentais demonstrando o acionamento da prótese experimental em função dos movimentos realizados por um braço humano.

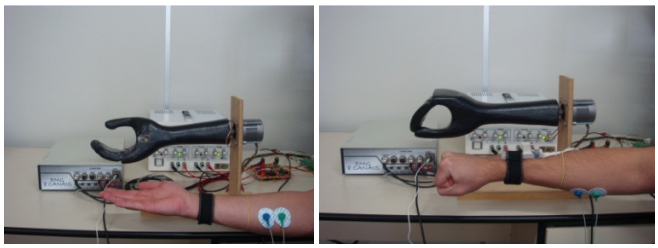


Figura 2. Ensaio demonstrando o funcionamento da prótese experimental em função dos movimentos espaciais do segmento mão-antebraço.

As Figuras 3 e 4 mostram os fluxogramas lógicos do aparato experimental desenvolvido.

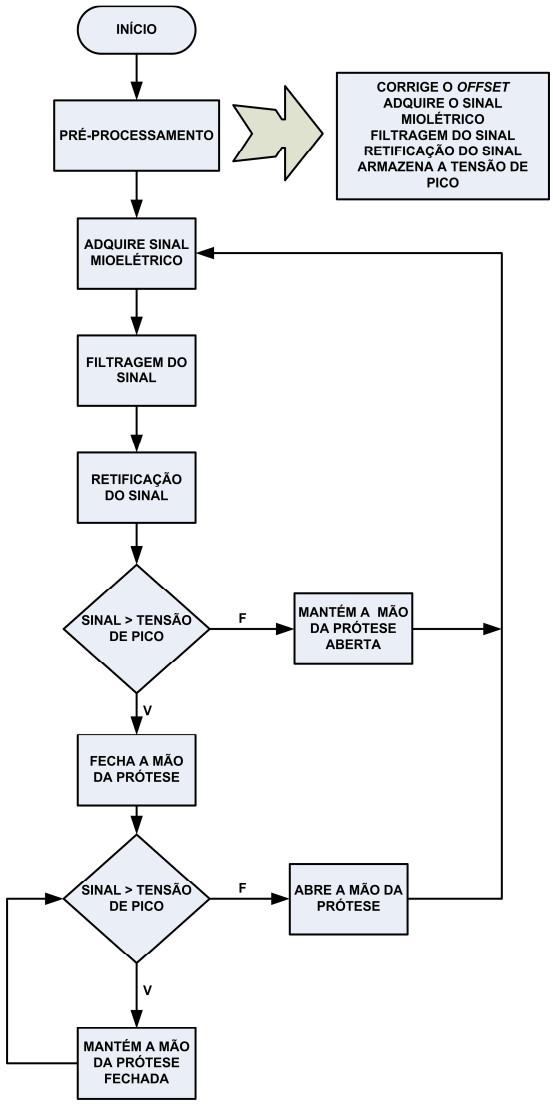


Figura 3. Fluxograma do processo – desde a aquisição do sinal mioelétrico, processamento do sinal bruto e controle simplificado da mão.

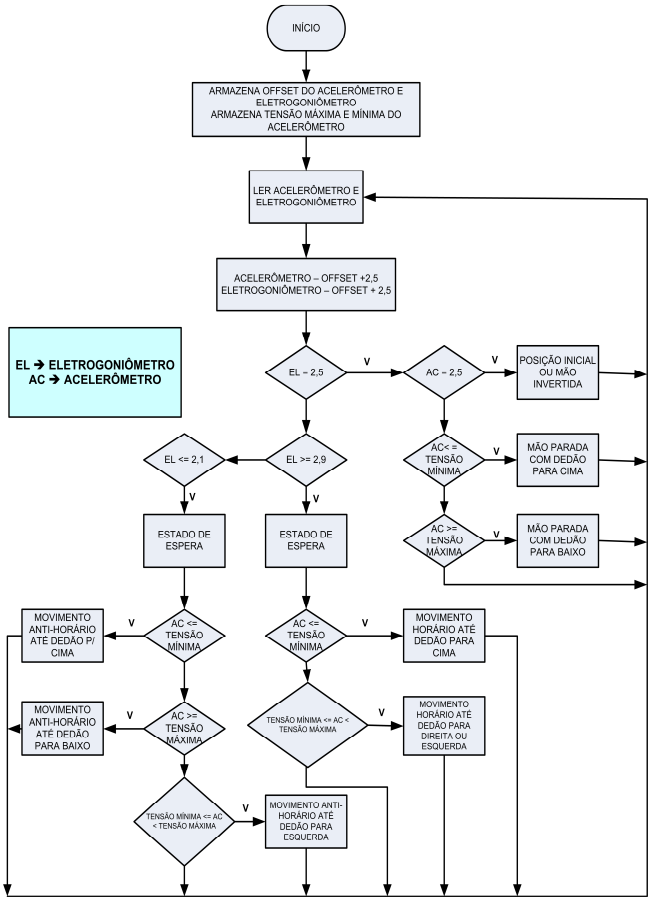


Figura 4. Fluxograma dos sensores que servem para auxiliar o posicionamento espacial da mão.

Os fluxogramas mostrados nas Figuras 3 e 4 serviram de base para o desenvolvimento do sistema de inferência neuro-*fuzzy* adaptativo descrito a seguir. Como pode ser observado, as variáveis de *input* e os parâmetros para determinação dos *outputs* correspondentes aos movimentos de abertura e fechamento da mão e de rotação do braço, estão presentes nos dois fluxogramas e se remetem aos dados obtidos na experimentação descrita. Assim, o sistema *fuzzy* desenvolvido para representar esse modelo deve conter as condições descritas nos fluxogramas para determinação dos movimentos.

3 Descrição do Sistema Neuro- *Fuzzy* Adaptativo

O sistema de inferência neuro-*fuzzy* adaptativo apresentado a seguir foi desenvolvido a partir dos fluxogramas anteriores e modela o posicionamento espacial do segmento mão-antebraço e o controle simplificado de abertura e fechamento da mão. O sistema foi implementado no Matlab®; seu nome é *prótese.mão*; o mecanismo de inferência utilizado na implementação é do tipo sugeno; os métodos *and*, *or*, e de defuzzificação são mim, max e wtaver, respectivamente. O sistema usa um algoritmo de aprendizado híbrido para identificar parâmetros de sistemas de inferência *fuzzy* do tipo sugeno.

O sistema possui três variáveis de *input*, 1 variável de *output* e dezesseis regras lógicas. As variáveis de *input* são *el*, *ac* e *rms* para eletrogoniômetro, acelerômetro e sinal, respectivamente, conforme os fluxogramas (Figuras 3 e 4). Foi definido um conjunto de valores para as variáveis de *input*. Foram definidas para cada variável de *input* três funções de pertinência: *baixo*, *médio* e *alto* para os possíveis valores a serem atribuídos com representação gaussiana. Foram definidos parâmetros constantes para a variável de *output prótese*. A Tabela 1 mostra os valores definidos para as variáveis de *input* e *output*.

Tabela1: Valores estabelecidos e alcançados na validação do sistema de inferência *fuzzy* proposto.

Valores estabelecidos para *el*	**Valores estabelecidos para *ac***	**Valores estabelecidos para *rms***	***Outputs esperados para prótese***
2.5	2.5	2.7	1
2.5	2.5	2.3	2
2.9	1.7	3.4	3
2.9	1.7	2.1	4
2.1	1.7	3.4	5
2.1	1.7	2.1	6
2.1	5.0	3.4	7
2.1	5.5	2.5	8
2.1	3.8	3.2	9
2.1	3.5	2.5	10
2.9	3.9	3.2	11
2.9	3.3	2.4	12
2.5	1.6	3.2	13
2.5	1.7	2.2	14
2.5	5.0	3.0	15
2.5	4.8	2.5	16

Com a rotina de treinamento *anfis* do Matlab®, a partir dos valores e das definições acima, foi construído o sistema neuro-*fuzzy* *prótese.mão*, com as seguintes adaptações de parâmetros para as funções de pertinência:

- a variável de *input el* varia no intervalo [2.1 2.9] e tem as seguintes funções de pertinência: *baixo* (para valores < = 2.1), com representação gaussiana e parâmetros [0.1149 2.081]; *médio* (para valores em torno de 2.5), com representação gaussiana e parâmetros [0.131 2.488]; *alto* (pra valores > = 2.9), com representação gaussiana e com parâmetros [0.1106 2.92];
- a variável de *input ac* varia no intervalo [1.6 5.5] e tem as seguintes funções de pertinência: *baixo* (para valores < = 1.7), com representação gaussiana e parâmetros [0.7843 1.567]; *médio* (para valores 1.7 < *ac* < 5.0) e em torno de 2.5), com representação gaussiana e parâmetros [0.795 3.514]; *alto* (para valores > = 5.0), com representação gaussiana e parâmetros [0.8264 5.494];
- a variável de *input rms* varia no intervalo [2.1 3.4] e tem as seguintes funções de pertinência: *baixo* (para valores < = 2.5, que é a tensão de pico), com representação gaussiana e parâmetros [0.1469 2.038];

médio (para valores em torno de 2.5), com representação gaussiana e parâmetros [0.2285 2.733]; *alto* .(para valores > 2.5), com representação gaussiana e parâmetros [0.2587 3.406];

- a variável de *output prótese* varia no intervalo [-18.91 26.41] e possui vinte e sete funções de pertinência, representadas pelas letras do alfabeto latino de *a* a *z* e *z2* e representação triangular; e cada uma delas, se refere a um possível movimento da prótese. Os movimentos descritos nos fluxogramas e que são mantidos pelo sistema neuro-fuzzy adaptativo *prótese. mão* são: *a* para representar *movimento em sentido anti-horário até o dedão ficar para cima e mão aberta; c* para representar *movimento em sentido anti-horário até o dedão ficar para cima e mão fechada; d* para representar *movimento em sentido anti-horário até o dedão ficar para a esquerda e mão aberta; f* para representar *movimento em sentido anti-horário até o dedão ficar para a esquerda e mão fechada; g* para representar *movimento em sentido anti-horário até o dedão ficar para baixo e mão aberta; i* para representar *movimento em sentido anti-horário até o dedão ficar para a esquerda e mão fechada; j* para representar o movimento *mão parada com o dedão para cima e mão aberta*; *l* para representar *mão parada com o dedão para cima e mão fechada; m* para representar *posição inicial sem movimento e mão aberta*; o para representar *posição inicial sem movimento e mão fechada*; *p* para representar *mão para da com dedão para baixo e mão aberta; r* para representar *mão para da com dedão para baixo e mão fechada;* s para representar *posição inicial sem movimento e mão aberta; u* para representar *mão para da com dedão para baixo e mão fechada; v* para representar *movimento em sentido horário até o dedão ficar para a direita ou esquerda e mão aberta; x* para representar *movimento para a direita ou esquerda e mão fechada; y* para representar *movimento em sentido horário até o dedão ficar para baixo em sentido horário até o dedão ficar e mão aberta; z* para representar *movimento em sentido horário até o dedão ficar para baixo e mão semi-aberta; z2* para representar *movimento em sentido horário até o dedão ficar para baixo e mão fechada.*

Os movimentos não descritos nos fluxogramas e que são adaptados pelo sistema neuro-*fuzzy* adaptativo *prótese. mão* são: *b* para representar *movimento em sentido anti-horário até o dedão ficar para cima e mão semi- aberta; e* para representar *movimento em sentido anti-horário até o dedão ficar para a esquerda e mão semi-aberta; h* para representar *movimento em sentido anti-horário até o dedão ficar para a esquerda e mão semi-aberta; k* para representar *mão parada com o dedão para cima e mão semi- aberta; n* para representar *posição inicial sem movimento e mão semi-aberta; q* para representar *mão para da com dedão para baixo e mão semi-aberta; t* para representar *mão para da com dedão para baixo e mão semi-aberta; w* para representar *movimento em sentido horário até o dedão ficar para a direita ou esquerda e mão semi-aberta.* Como se pode observar, as regras adaptadas pelo sistema são referentes à possibilidade de uma posição intermediária de semi-abertura para a mão. As vinte e sete regras lógicas são mostradas na Tabela 2.

Tabela 2: Regras lógicas do sistema neuro-*fuzzy* adaptativo.

Regra	**Descrição da Regra**
R1	Se *el* é baixo e *ac* é *baixo e rms* é *baixo* então *a;*
R2	Se *el* é *baixo* e *ac* é *baixo* e *rms é médio* então *b;*
R3	Se *el* é *baixo* e *ac* é *baixo rms* é *alto* então *c;*
R4	Se *el* é baixo *o* e *ac* é *médio* e *rms* é *baixo* então *d*;
R5	Se *el* é *baixo* e *ac* é médio e *rms* é *médio* então *e*;
R6	Se *el* é *baixo* e *ac* é *médio* e *rms* é *alto* então *f*;
R7	Se *el* é *baixo* e *ac* é *alto* e *rms* é *baixo* então *g;*
R8	Se *el* é baixo e *ac* é *alto* e *rms* é *médio* então *h*;
R9	Se *el* é *baixo* e *ac* é *alto* e *rms é alto* então *i*;
R10	Se *el* é *médio* e *ac* é *baixo* e *rms* é *baixo* então *j*;
R11	Se *el* é *médio* e *ac* é *baixo* e *rms* é *médio* então *k*;
R12	Se *el é médio* e *ac* é *baixo* e *rms* é *alto* então *l*;
R13	Se *el* é *médio* e *ac* é *médio* e *rms* é baixo então *m*;
R14	Se *el* é *médio* e *ac* é *médio rms* é *médio* então n;
R15	Se *el* é *médio* e *ac* é *médio* e *rms* é *alto* então *o*;
R16	Se *el* é *médio* e *ac* é *alto* e *rms* é *baixo* então *p*.
R17	Se *el* é *médio* e *ac* é *alto e* e *rms* is *médio* então *q*;
R18	Se *el* é *médio* e *ac* é *alto* e *rms* é *alto* então *r*;
R19	Se *el* é *alto* e *ac* é *baixo* e *rms* é *baixo* então *s*;
R20	Se *el* é *alto* e *ac* é *baixo rms* é m*édio* então *t;*
R21	Se *el* é *alto* e *ac* é *baixo* e *rms* é *alto* então *u;*
R22	Se *el* é *alto* e *ac* é *médio* e *rms* é *baixo* então *v*.
R23	Se *el* é *alto* e *ac* é *médio* e *rms* é *médio* então *w;*
R24	Se *el* é *alto* e *ac* é *médio* e *rms* é *alto* então *x;*
R25	Se *el* é *alto* e *ac* é *alto* e *rms* é *baixo* então *y*;
R26	Se *el* é *alto* e *ac* é *alto* e *rms* é *médio* então *z;*
R27	Se *el* é *alto* e *ac* é *alto* e *rms* é *alto* então *z2*;

A Figura 5 mostra a estrutura do sistema neuro-*fuzzy* adaptativo *prótese.mão*.

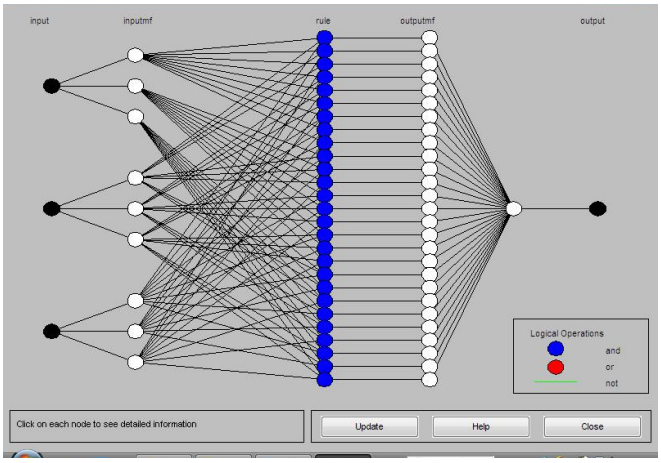


Figura 5. Estrutura do sistema adaptativo neuro- *fuzzy* para controle de uma prótese de mão.

Os primeiros três círculos pretos representam os *inputs*; os nove primeiros círculos brancos representam as funções de pertinência para cada *input*; os círculos azuis representam as vinte e sete regras lógicas; os vinte e sete círculos brancos ao lado das regras lógicas representam os *outputs* para cada regra especificamente; o último círculo branco, os *outputs* agregados e o último círculo preto representa o *output* final, que determina o movimento da prótese.

O sistema de inferência neuro-*fuzzy* adaptativo possui setenta e oito nós e quarenta e oito parâmetros divididos em vinte e sete parâmetros lineares para as funções de pertinência das variáveis de *input* e dezoito parâmetros não lineares para as funções de pertinência da variável de output.

4. Resultados

O sistema de inferência neuro-*fuzzy* foi treinado para os valores estabelecidos inicialmente (ver Tabela 1) através do comando *training data*. Em seguida foi feita a validação do sistema através dos comandos *testing data* e *checking data.* O comando *testing data* testou a capacidade de generalização do sistema, a partir de seu treinamento, para um segundo conjunto de valores atribuídos às variáveis, próximos aos valores para os quais foi treinado. O comando *checking data* verificou se o conjunto de dados utilizados pelo *testing.data* para validar o sistema é um conjunto adequado ao modelo. Para o processo de validação, foram utilizados os valores para *el*, *ac*, *rms* e *prótese*, mostrados na a Tabela3. Ambos os conjuntos de valores usados para treinamento e validação do sistema estão dentro do escopo das variáveis de *input,* descrito nos fluxogramas mostrados nas Figuras 3 e 4 que refletem o modelo lógico do aparato experimental.

Tabela 3: Valores testados para ver a capacidade de generalização do sistema *neuro-fuzzy*

Valores estabelecidos para *el*	**Valores estabelecidos para *ac***	**Valores estabelecidos para *rms***	***Outputs obtidos para prótese***
2.5	2.5	2.6	1
2.5	2.5	2.2	2
2.9	1.7	3.3	3
2.9	1.7	2.1	4
2.1	1.7	3.3	5
2.1	1.7	2.1	6
2.1	5.0	3.38	7
2.1	5.5	2.51	8
2.1	3.8	3.17	9
2.1	3.5	2.48	10
2.9	3.9	3.1	11
2.9	3.3	2.45	12
2.5	1.6	3.21	13
2.5	1.7	2.1	14
2.5	5.0	3.25	15
2.5	4.8	2.51	16

A Figura 6 mostra o resultado do treinamento do sistema neuro-*fuzzy* adaptativo *prótese.mão* para o conjunto de dados constantes na Tabela 1.

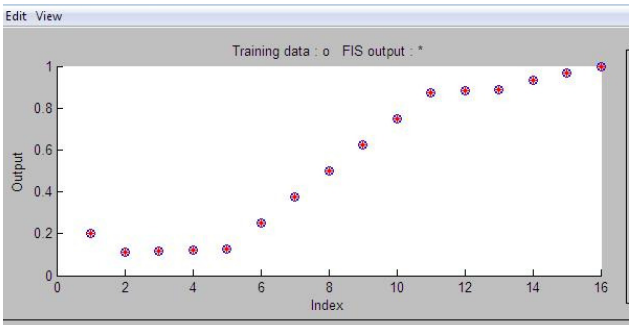


Figura 6: Treinamento do sistema neuro-*fuzzy* adaptativo *prótese.mão.*

O gráfico da Figura 6 mostra que o sistema foi adequadamente treinado para os dados considerados para as variáveis de *input* e *output*. A Figura 7 mostra o resultado do teste do sistema, a partir de seu treinamento, que se mostrou adequado, com relação á sua capacidade de generalização para ajuste de parâmetros para o novo conjunto de valores, apresentados na Tabela 3.

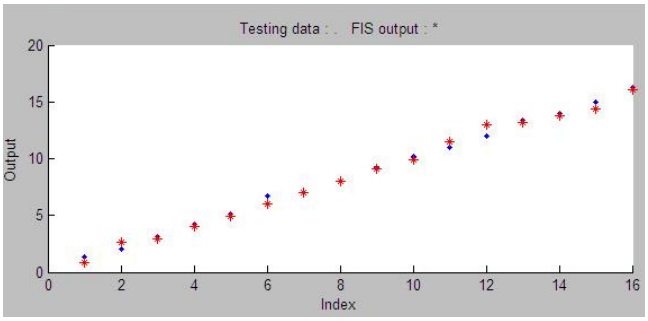


Figura 7: Teste da capacidade de generalização do sistema neuro-*fuzzy* adaptativo *prótese.mão.*

O gráfico da Figura 7 mostra que o sistema neuro-*fuzzy* adaptativo possui boa capacidade de generalização a partir de seu treinamento para dados novos semelhantes, ou com valores aproximados, como pode ser conferido pelas Tabelas 1 e 3. Como já foi visto, a Tabela 1 exibe os valores para os quais o sistema foi treinado; a Tabela 3 exibe os novos dados usados para testar sua capacidade de generalização. A Figura 8 compara os conjuntos de dados usados para treinar o sistema e o conjunto de dados usados para testar o sistema e verifica se esse segundo conjunto de dados é adequado para validar o modelo, de acordo com a margem de erros que apresenta.

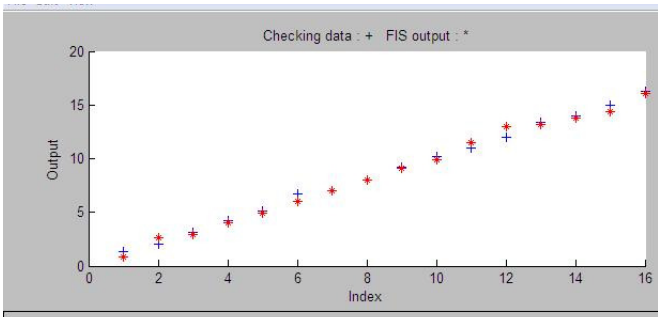


Figura 8: Comparação dos conjuntos de dados usados para treinamento e validação do sistema neuro-*fuzzy* adaptativo.

O gráfico da Figura 8 mostra que o conjunto de dados utilizados para teste da capacidade de generalização do sistema neuro-*fuzzy* adaptativo é satisfatório com relação ao conjunto de dados usados para treinamento do sistema.

6. Discussão

O método utilizado no desenvolvimento do sistema proposto foi o ANFIS (*Adaptive neuro—fuzzy inference system*) que utiliza um algoritmo de aprendizado híbrido para identificar parâmetros em sistemas de inferência fuzzy do tipo Sugeno. Ele constitui uma combinação do método *Backpropagation* e do método *Least-squares* para treinar os parâmetros das funções de pertinência para emular conjuntos de dados.

O trabalho apresenta uma aplicação desse método à representação computacional dos movimentos de abertura, fechamento e rotação de uma prótese do segmento mão-braço. O sistema foi testado para valores referentes ao escopo definido para cada variável a partir de testes realizados com o aparato experimental descrito. O sistema foi testado também para valores de *input* relativamente diferentes dos definidos a partir do aparato experimental. Para esses valores, o sistema redefiniu os parâmetros para as funções de pertinência das variáveis de input, mostrando-se assim, auto-modificável. A esses novos parâmetros foram relacionadas novas possibilidades de movimentos com relação à abertura e o fechamento da prótese, não previstos na experimentação.

7. Conclusões

O sistema neuro-*fuzzy* adaptativo foi adequadamente treinado, mostrou boa capacidade de generalização e o conjunto de dados utilizado para testar a capacidade de generalização do sistema é satisfatório com relação ao conjunto de dados utilizado no treinamento do sistema.

O conjunto de regras do fluxograma, que dizem respeito à experimentação, foi adaptado, permitindo a possibilidade de mais movimentos para a prótese, como se pode comparar nos fluxogramas apresentados nas Figuras 3 e 4, na Tabela 2 e na descrição do sistema apresentado anteriormente, na página 4.

O sistema neuro-*fuzzy* adaptativo *prótese.mão* consegue reproduzir o modelo experimental, uma vez que possui todas as regras dos fluxogramas referentes à movimentação da prótese. Além disso, expande as possibilidades e movimentos da prótese porque possui regras que relacionam os parâmetros adaptados a novos movimentos não previstos nos fluxogramas apresentados.

O trabalho apresenta assim, um resultado parcial, mas significativo para o desenvolvimento de tecnologia assistiva nacional, com o uso de inteligência computacional, através do uso de métodos neuro-*fuzzy* no controle de movimentos de próteses.

Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) pelo apoio financeiro.

Referências Bibliográficas

[1] Begg, R., Lai, D.T.H. e Palaniswami, M. (2008). *Computational Intelligence in Biomedical Engineering,* CRC Press

[2] Balbinot, A., Tomaszewski, J. R., e Marques, P. (2007). Eletromiografia, acelerômetro, eletrogoniômetro e tecnologia openGL para controle de próteses experimentais. *II Encontro Nacional de Biomecânica, Évora. Actas do II Encontro Nacional de Biomecânica*. Lisboa - Portugal: IST Press, 2007. v. I. p. 381-386.

[3] Teodorescu, H.L., Kandel, A. e Lain. L.C. (Editors) (1998). Fuzzy and Neuro-Fuzzy Systems in Medicine. *International Series in Computational Intelligence*, CRC.

[4] Steimann, F. (2001). On the Use and Usefulness of Fuzzy Sets in Medical AI. *Artificial Intelligence in Medicine*, Volume 21, Issues 1-3, p.131-137.

[5] Yahud, S., e Abu Osman, N. A. (2007). Prosthetic Hand for the Brain-computer Interface System. In: Ibrain, F., Abu Osman, N. A. e Usman, J., Kadri, N.A. (Editors), *Biomed 06*, IFMBE *Proceedings*, 15, 643-646.

[6] Zadeh, L. A. (1965). Fuzzy Sets, *Information and Control*, 8, 338-353.

[7] _____. (1969). Biological Applications of the Theory of Fuzzy Sets and Systems. In: *Biocybernetics of the Central Nervous System*. Little Brown & CO., p.199-212.

[8] _____. (1975). The Concept of Linguistic Variables and its Application to Approximate Reasoning – I, II, III. *Information Sciences.8*, 199-249, 8, 301-357, 9, 43-80.

Mara M. G. Barreto nasceu no Rio de Janeiro, Brasil, em 4 de setembro de 1964. É Bacharel em Filosofia pelo Instituto de Filosofia e Ciências Sociais (IFCS)/UFRJ; Mestra em Engenharia de Produção pela COPPE/UFRJ; Doutora em Engenharia Civil, na área de concentração Sistemas Computacionais, COPPE/UFRJ. Possui Pós-doutorado no Instituto de Estudos Avançados da USP, na área Lógica e Teoria da Ciência, desenvolvido parcialmente no Departamento de Fisiologia da Escola Paulista de Medicina (EPM)/UNIFESP. Atualmente Professora Adjunta II do Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas (CECS) da Universidade Federal do ABC (UFABC), atuando nas seguintes áreas: modelagem lógica - fundamentos e aplicações; desenvolvimento de modelos fuzzy e neuro-fuzzy para engenharia; tecnologia assistiva; desenvolvimento de interface cérebro-computador (BCI) fuzzy; sistemas de apoio à decisão em diagnóstico médico; lógica e argumentação.

Alexandre Balbinot nasceu no Rio Grande do Sul, Brasil, em 27 de março de 1970. É Engenheiro Eletricista pela PUCRS; Mestre em Processamento Digital de Sinais pela PUCRS; Mestre em Engenharia Biomecânica pela UFRGS e Doutor em Engenharia Biomecânica pela UFRGS. Atualmente é Professor Adjunto I da Escola de Engenharia, Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul (UFRGS) atuando nas seguintes áreas: instrumentação biomédica, instrumentação biomecánica, desenvolvimento de dispositivos de tecnología assistiva; desenvolvimento de interfaces cerebro-computador (Brain Computer Interfacing), desenvolvimento e avaliação de sistemas para caracterização de movimentos do corpo humano: sua interface com realidade virtual e realidade aumentada e aplicações de inteligência computacional.

Uma definição simplificada para o estudo das propriedades dos Autômatos Finitos Adaptativos

(18 de Janeiro de 2009)

Diego Queiroz

***Resumo*— Autômatos Finitos Adaptativos são máquinas capazes de reconhecer quaisquer linguagens que possam ser reconhecidas por quaisquer outros modelos computacionais conhecidos. No entanto, embora seu funcionamento seja intuitivo e de fácil entendimento, poucos materiais têm explorado uma maneira de representação simplificada dessa classe de autômatos para a utilização em aplicações menos especializadas sobre o assunto. Deste modo, este artigo tem o intuito de propor uma representação alternativa para os mesmos de modo a facilitar a sua descrição e seu estudo em projetos futuros.**

Palavras chave*— Autômatos *(Automata),* Máquina de Turing *(Turing Machine),* Teoria da Computação *(Computation Theory).

I. Introdução

Os Autômatos Adaptativos [2] são uma classe de autômatos com um poder computacional superior a maioria das outras classes de autômatos. Sua característica peculiar reside na capacidade deles poderem modificar a sua própria estrutura durante sua execução. Essas alterações são realizadas através de ações adaptativas associadas a algumas das transições do autômato que permitem que sejam incluídas ou removidas transições entre seus estados.

A partir destes, é possível definir uma especialização através dos Autômatos Finitos Adaptativos (AFA). Estes autômatos possuem as mesmas características dos Autômatos Adaptativos com a imposição de algumas restrições que os tornam muito semelhantes aos Autômatos Finitos. De fato, os AFA podem ser definidos como uma generalização dos Autômatos Finitos, uma vez que, removendo as suas ações adaptativas, temos um Autômato Finito [1], no entanto, seu poder computacional é muito maior: devido a sua característica de adaptatividade, pode ser provado que os Autômatos Finitos Adaptativos são equivalentes, em poder computacional, a Máquinas de Turing [4]. A figura 1 representa a relação entre estas classes de autômatos.

Essa característica, embora dê grandes poderes aos Autômatos Adaptativos, traz também grandes consequências: todas as condições que cercam as Máquinas de Turing, especialmente sobre incomputabilidade, também cercam os Autômatos Adaptativos (caso contrário, bastaria que se transformasse o problema na forma de Máquina de Turing em outro na forma de Autômato Adaptativo para resolver diversos problemas que até o presente momento não possuem solução).

No entanto, as definições sobre Autômatos Adaptativos em uso são demasiadamente complexas para aplicações simples, portanto, nota-se uma carência de uma definição que contemple grande parte das características dessa classe de autômatos de uma forma prática e intuitiva.

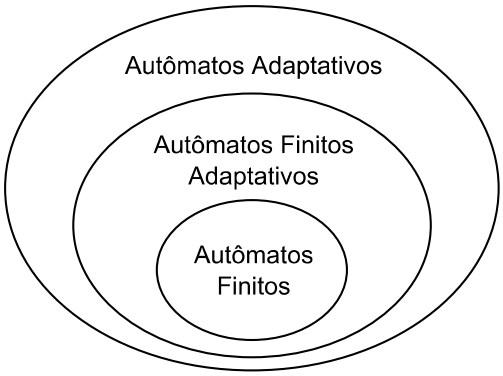


Fig. 1. Relação entre as diferentes classes de autômatos apresentados.

Com base nisso, este trabalho tem o intuito de sugerir uma formulação alternativa e detalhada para a representação de Autômatos Adaptativos com base na formulação original em [3], [11] e no modelo utilizado em [1], com o intuito de futuramente poder estudar as propriedades dos mesmos.

II. Autômatos Finitos Adaptativos

A definição mais simples de um autômato é através da sua representação como uma máquina de estados finitos, ou seja, uma máquina que, partindo de um estado inicial e consumindo uma cadeia de entrada, transita entre seus estados de acordo com um conjunto de regras de transição pré-determinadas.

Os autômatos podem possuir um ou mais estados denominados estados finais. Portanto, se após consumir uma cadeia de entrada, um autômato para sobre um destes estados, diz-se que a cadeia é reconhecida pelo autômato. Ao conjunto de cadeias que a máquina reconhece dá-se o nome de linguagem, de modo que cada tipo de máquina reconhece apenas uma classe de linguagens específica e nada mais. Essas classes são geralmente definidas com base na hierarquia de Chomsky que determina, para cada classe de linguagem, um conjunto de regras gramaticais que, quando aplicadas, geram apenas cadeias daquela classe.

Existem diversos tipos de representações diferentes de autômatos, desde as mais simples, contendo apenas os elementos básicos de uma máquina e uma quantidade finita de

Este trabalho foi financiado pelo Conselho Nacional de Desenvolvimento Científico (CNPq).

estados (Autômatos Finitos), como máquinas mais complexas que dispõem de memória auxiliar (Autômatos de Pilha) ou que incorporam funções diferenciadas [6], [7].

Embora haja equivalência entre diversos tipos de máquinas, cada classe de linguagem necessita de um tipo especializado de máquina para fazer o seu reconhecimento. No entanto, como as linguagens são definidas com base em uma hierarquia, se um tipo de máquina reconhece uma classe de linguagem também reconhece todas as classes de linguagens que estão abaixo dela hierarquicamente.

Para o reconhecimento das linguagens do topo da hierarquia, isto é, as mais gerais, chamadas de linguagens recursivamente enumeráveis, foi proposta por Turing uma "máquina de computação" [5], [6], [7] que atualmente é conhecida como Máquina de Turing. Além da simples interpretação das máquinas como reconhecedores de linguagens, acredita-se que as máquinas de Turing são versões formais de algoritmos e que, portanto, elas exprimem o limite de tudo o que pode ser computado, isto é, se algo pode ser computado, então pode ser representado por uma máquina de Turing e vice-versa.

Os Autômatos Finitos Adaptativos são máquinas de estados capazes de computar a mesma classe de linguagens que as Máquinas de Turing [4]. Sua formação é intuitivamente compreendida como um autômato finito com a capacidade de auto-modificação. Todas as suas modificações são realizadas através de ações adaptativas anexadas às transições, dessa forma, à medida que um símbolo é lido o autômato tem a habilidade de se "adaptar" para reconhecer a cadeia.

Portanto, se explorarmos esse conceito ao considerar que o autômato antes da execução da ação adaptativa é diferente do autômato após a execução da mesma [1], chegamos à conclusão que a capacidade de auto-modificação do autômato finito adaptativo faz com que este sempre possa se transformar em outro autômato à medida que transita através de estados. Desse modo, a capacidade computacional do autômato finito adaptativo reside no fato dele reunir todos os autômatos finitos em um único autômato, dispondo, indiretamente, de uma memória finita, porém, ilimitada, assim como a Máquina de Turing, estando essa, no entanto, incorporada à sua estrutura.

O modelo formal dos autômatos adaptativos pode ser representado de diversas formas. Em [2], [3], [9] os autômatos adaptativos são definidos com base em dispositivos adaptativos dirigidos por regras cujo dispositivo subjacente é um autômatos; em [1], [11] são utilizadas formalizações mais aplicadas e semelhantes às definições originais de autômatos; e, em [12], os mesmos são estudados na forma de gramáticas.

Neste artigo será apresentada uma notação simplificada dos Autômatos Finitos Adaptativos Determinísticos com o objetivo de generalizar suas funções e permitir que suas propriedades possam ser analisadas separadamente.

III. Autômatos Finitos Adaptativos Determinísticos

Diversos trabalhos destacam a importância de se incorporar o determinismo a autômatos [9], e diversos outros trabalhos já realizaram estudos comparativos sobre a complexidade de autômatos determinísticos e suas contrapartes não-determinísticas [13], [14].

Para uma máquina de estados finitos, definimos o determinismo como um requisito que faz com que haja apenas uma regra de transição para cada configuração possível.

Portanto, um Autômato Finito Adaptativo Determinístico ($AFAD$) é uma 5-tupla definida por $(Q_0, \Sigma, \delta_0, q_0, F)$ tal que:

- Q_0 é o conjunto inicial de estados;
- Σ é o alfabeto de entrada;
- δ_0 é a função de transição adaptativa inicial ($\delta_t: Q_t \times \Sigma \rightarrow Q_\infty \times \Omega^\star$) [15];
- q_0 é o estado inicial ($q_0 \in Q_0$);
- F é o conjunto de estados finais ($F \subseteq Q_0$).

com

$$\Omega = Q_\infty \times \Sigma \times Q_\infty \times \Omega^\star$$

$$Q_\infty = \bigcup_{t=0}^{\infty} Q_t \cup \{d\}$$

Em cada instante t, o $AFAD$ assume uma configuração $c_t = (Q_t, q, w, \delta_t, \bar{q})$ que determina o seu conjunto de estados no instante t, seu estado atual, o conteúdo da cadeia a ser processada, uma função de transição adaptativa e uma referência para o estado recém referenciado pela função adaptativa. Assim, sendo w uma cadeia de entrada sobre o alfabeto Σ ($w \in \Sigma^\star$), ao ser iniciado ($t = 0$) a configuração do $AFAD$ é $c_0 = (Q_0, q_0, w, \delta_0, q_0)$.

Sob essas condições, a função δ_t fornece as regras para a mudança de estado e fornece um mecanismo para a execução das ações adaptativas. Estando o $AFAD$ no estado $q \in Q_t$ e possuindo o símbolo σ na cabeça da fita de entrada, a função $\delta_t(q, \sigma) = (q_n, \psi)$ define o novo estado do $AFAD$ e a sequência de ações adaptativas a serem executadas.

Definimos "d" como sendo um estado implícito que garante o determinismo do autômato. Embora este seja geralmente omitido para facilitar a notação, este estado está presente em grande parte das definições não-determinísticas de autômatos. De um modo geral, ele é um estado especial não final ($d \notin F$) tal que a função de transição mapeia ele mesmo para todos os símbolos do alfabeto, isto é, $\delta(d, \sigma) = d$, $\forall \sigma \in \Sigma$. Desse modo, este pode ser considerado um estado terminal de *erro* do autômato finito adaptativo determinístico, pois faz este esgotar a cadeia de entrada e parar em um estado que não corresponde aos critérios para que a cadeia seja reconhecida.

Diferente da definição original dos autômatos finitos adaptativos, onde são previstas as ações de inserção e remoção [9], esta definição prevê para o autômato finito adaptativo determinístico apenas a ação de *alteração*. Essa ação adaptativa tem influência sobre a configuração posterior da máquina e faz com o autômato, ao consumir um símbolo em um estado especificado, transite para um novo estado,

[15] $A^\star$ representa o fecho reflexivo e transitivo sobre o conjunto A.

também especificado. Dessa forma, a ação adaptativa $\omega \in \Omega$ é representada pela sequência (q_i, σ, q_j, ψ) tal que:

- q_i é o estado de origem da transição que será alterada ($q_i \in (Q_\infty \cup \{\alpha,\star\}) - \{d\}$);
- σ é o rótulo da transição que receberá o novo destino;
- q_j é o novo estado de destino da transição que será alterada ($q_j \in Q_\infty \cup \{\alpha,\star\}$);
- ψ é a sequência de ações adaptativas a ser alocada a transição ($\psi \in \Omega^\star$).

Assim sendo, a ação adaptativa $\omega = (q_i, \sigma, q_j, \psi_\omega)$ fará com que a transição que sai do estado q_i, consumindo o símbolo σ da cadeia de entrada, aponte para o estado q_j associando a sequência de ações adaptativas ψ_ω. Isto é, se δ_{t-1} é a função de transição contida na configuração c_{t-1} que corresponde ao estado da máquina antes da execução da ação adaptativa e δ_t é a função de transição adaptativa contida na configuração c_t que corresponde ao estado da máquina após a execução da ação adaptativa, ω fará com que $\delta_t(q_i, \sigma) = (q_j, \psi_\omega)$ e $\delta_t(q, x) = \delta_{t-1}(q, x), \forall q \in Q_t$ e $\forall x \in \Sigma$.

Nessa formulação, dois símbolos especiais foram introduzidos: "α" e "$\star$". O símbolo "α" representa o estado atual do autômato, isto é, o estado que o autômato atingiu antes de executar a sequência de ações adaptativas. Já o símbolo "$\star$" denota uma referência a um novo estado (se $q_j = \star$) ou ao último estado criado (se $q_i = \star$). Note que $\{\alpha,\star\} \notin Q_\infty$.

Com isso, a relação binária $\vdash_{AFAD}$ entre duas configurações c_j e c_k é verdadeira se for possível passar de uma configuração para outra em apenas um movimento. Portanto, um movimento $(Q_j, q_j, w_j, \delta_j, \bar{q}_j) \vdash (Q_k, q_k, w_k, \delta_k, \bar{q}_k)$ ocorre se, e somente se, todas as condições descritas nas expressões (1), (2), (3), (4) e (5) são verdadeiras simultaneamente:

$$w_j = \sigma \circ w_k \tag{1}$$

$$\delta_j(q_j, \sigma) = (q_k, \psi) \tag{2}$$

$$\begin{gathered}(\bar{q}_j = \bar{q}_k) \\ \text{ou} \\ \left[(\omega = (q, x, \star, \psi')) \text{ e } (\bar{q}_k \notin Q_j) \text{ e } (\bar{q}_k \in Q_k)\right] \\ \exists \omega \in \psi, \exists q \in Q_j, \exists x \in \Sigma, \exists \psi' \in \Omega^\star\end{gathered} \tag{3}$$

$$\begin{gathered}(q, x, \star, \psi') \in \psi \rightarrow q' \in Q_k \\ \exists q \in Q_j, \exists x \in \Sigma, \exists \psi' \in \Omega^\star, \exists q' \notin Q_j\end{gathered} \tag{4}$$

$$Q_j \subseteq Q_k \tag{5}$$

Concluímos assim que, uma cadeia w é reconhecida por um $AFAD$ se, e somente se, através de movimentos sucessivos é possível esgotar a cadeia e chegar a um estado final, ou seja, $(Q_j, q_j, w_j, \delta_j, \bar{q}_j) \vdash^\star (Q_k, q_k, \varepsilon, \delta_k, \bar{q}_k)$ e $q_k \in F$ (ε é a cadeia de comprimento zero).

O exemplo 1 demonstra a utilização dessa formulação para a construção de um autômato que reconhece uma linguagem simples.

Exemplo 1: Autômato Adaptativo que reconhece todas as cadeias que contém a mesma quantidade de a's e b's, com nenhum b antes do a.

$$AFAD_1 = (Q_0, \Sigma, \delta_0, q_0, F)$$

$$Q_0 = \{q_0, q_1\}$$

$$\Sigma = \{a, b\}$$

$$\delta_0 = \{(q_0, a, q_1, \psi), (q_1, b, q_0, \emptyset)\}$$

$$F = \{q_0\}$$

$$\psi = \{(\alpha, a, \star, \psi), (\star, b, \alpha, \emptyset)\}$$

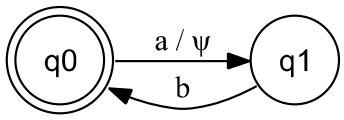


Fig. 2. Diagrama do AFAD do Exemplo 1.

IV. Análise comparativa

A fim de comparar a representação da definição aqui apresentada, serão descritas, a fim de comparação, algumas outras formas de representação de Autômatos Adaptativos.

Seja a linguagem L_1 definida por $L_1 = \{x^{2n}y^{2n}e | n \geq 0\} \cup \{x^{2n+1}y^{2n+1}o | n \geq 0\}$, para tanto, descrevemos um autômato adaptativo AD, representado na forma de um dispositivo adaptativo dirigido por regras [2], [10], que reconhece esta linguagem[16] (figura 3a):

$$AD_0 = (C_0, AR_0, S, c_0, A, NA, BA, AA)$$

$$C_0 = ((1, w), (3, w))$$

$$AR_0 = \left\{\begin{gathered}\left[\left(\varepsilon, c_i, x, c_j, \varepsilon, A(e, o)\right) | c_i = (1, xw) \text{ e } c_j = (1, w)\right], \\ \left[(\varepsilon, c_i, e, c_j, \varepsilon, \varepsilon) | c_i = (1, ew) \text{ e } c_j = (3, w)\right]\end{gathered}\right\}$$

$$S = \{x, y, e, o\}$$

[16] Algumas características, como a linguagem, o formato do autômato e o nome dos estados, foram mantidas inalteradas, conforme descrição original em [10].

$$c_0 = (1, w)$$

$$A = \{(1, \varepsilon), (3, \varepsilon)\}$$

$$NA = BA = \{\varepsilon\}$$

$$AA = \{\varepsilon, A(m,n)\}$$

$$A(m,n) = f, g * \begin{Bmatrix} -[(f,m) \to 3], \\ +[(f,y) \to g], \\ +[(g,n) \to 3], \\ -[(1,x) \to 1, A(m,n)], \\ +[(1,x) \to 1, A(n,m)] \end{Bmatrix}$$

Por motivos práticos, esta descrição foi resumida definindo a cadeia w sobre o alfabeto S. Portanto, todas as ocorrências sobre este símbolo devem ser interpretadas como sendo válidas $\forall w \in S^{\star}$.

Analogamente aos autômatos finitos adaptativos determinísticos, a cada passo k, os estados são representados pelas configurações possíveis C_k e as transições pelo conjunto de regras adaptativas $AR_k = BA \times C \times S \times C \times NA \times AA$, que permitem que o dispositivo modifique a sua configuração e altere o conjunto de regras. Deste modo, partindo de uma configuração inicial c_0, aplicam-se as regras sucessivamente até que o dispositivo atinja uma configuração de aceitação $c \in A$. A configuração deste dispositivo é baseada na do dispositivo subjacente que, neste caso, é o de um Autômato Finito, ou seja, o par (q, w) representando o estado atual e o conteúdo não lido da cadeia de entrada, respectivamente.

Cada vez que uma regra é aplicada, um conjunto de ações adaptativas pode ser aplicado através da execução de uma função adaptativa. Esta função pode ser executada antes e/ou depois de modificar a configuração do dispositivo. Neste caso, o efeito da execução das ações adaptativas se reflete sobre o conjunto de regras adaptativas e diz-se que o dispositivo avançou para o passo seguinte.

O grande interesse aqui reside no conjunto de possíveis funções adaptativas a serem executadas após a modificação da configuração, definido pelo conjunto AA. Neste exemplo, existe uma função vazia (ε) e outra função chamada A, com dois parâmetros.

O conteúdo da função A define a sequência de ações adaptativas a serem executadas. Dentro do seu escopo, também são definidas variáveis locais (f), utilizadas para armazenar a referência a estados/símbolos consultados; e geradores ($g*$), utilizados para armazenar a referência aos estados criados.

Por simplicidade, aqui foram descritas apenas as ações "+" e "−", que representam as ações de inserção e remoção de estados, respectivamente. Além destas, existe também a função de consulta, que fica subentendida por ser a responsável por resolver todas as variáveis não definidas nas chamadas das ações "+" e "−".

É importante notar que, para a descrição se adequar a formulação proposta, são utilizadas apenas as funções adaptativas que se executam após a modificação da configuração do autômato, pois os $AFAD$s sempre executam as ações adaptativas após efetuar a transição entre os estados.

Segundo Pistori [11], "a utilização exclusiva de ações adaptativas anteriores não prejudica a capacidade expressiva do modelo resultante, pois existem artifícios, usando transições vazias, que podem ser empregados na simulação de ações adaptativas posteriores". Como a mesma afirmação se aplica de modo inverso, isto é, transições em vazio podem ser inseridas para a simulação de ações anteriores, esta adaptação não traz nenhuma implicação.

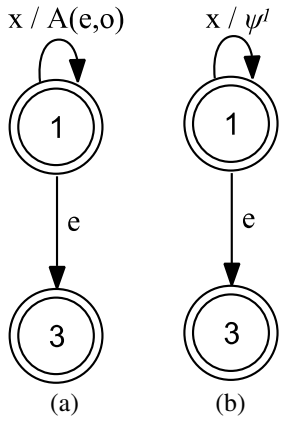


Fig. 3. Diagrama das duas formulações de autômatos apresentadas para o reconhecimento da linguagem L_1. (a) é o diagrama do dispositivo baseado em regras e (b) é do autômato finito adaptativo determinístico.

Os conjuntos NA e BA representam o conjunto de símbolos de saída e o conjunto de funções adaptativas disponíveis para serem executadas antes da mudança de configuração da máquina. Neste exemplo, ambas contém valores nulos devido ao fato deste estudo não possuir nenhum interesse específico nessas funções.

Para servir de parâmetro à comparação, descrevemos a seguir um Autômato Finito Adaptativo Determinístico $AFAD_2$ que também reconhece a linguagem L (figura 3b):

$$AFAD_2 = (Q_0, \Sigma, \delta_0, 1, F)$$

$$Q_0 = \{1,3\}$$

$$\Sigma = \{x, y, e, o\}$$

$$\delta_0 = \{(1,x) \to (1, \psi^1), (1,e) \to (3, \emptyset)\}$$

$$F = \{1,3\}$$

$$\psi^1 = \begin{Bmatrix} (\star, e, d, \emptyset), \\ (\star, y, \star, \emptyset), \\ (\star, o, 3, \emptyset), \\ (1, x, 1, \psi^2) \end{Bmatrix} \quad \psi^2 = \begin{Bmatrix} (\star, o, d, \emptyset), \\ (\star, y, \star, \emptyset), \\ (\star, e, 3, \emptyset), \\ (1, x, 1, \psi^1) \end{Bmatrix}$$

Como se pode notar, as descrições sobre os autômatos finitos adaptativos determinísticos compartilham uma série de características comuns com a representação dirigida a regras

utilizada para representar o Autômato Adaptativo:

- O alfabeto Σ do $AFAD$ é equivalente ao conjunto S de estímulos válidos do AD;
- O conjunto de estados Q_0 é semelhante ao conjunto de configurações válidas C_0;
- O estado inicial q_0 é semelhante à configuração inicial c_0.
- O conjunto de estados finais F é semelhante ao conjunto de configurações de aceitação A.

Uma das diferenças entre esses modelos apresentados está no fato dos dispositivos baseados em regras utilizarem um contador iniciado em zero e que incrementa apenas quando uma ação adaptativa é executada, isto é, a cada alteração, enquanto o $AFAD$ proposto aqui se baseia em uma variável temporal que incrementa sempre que uma transição de estados é efetuada pelo autômato.

Logicamente, esta diferença não exerce grande influência, uma vez que, se um $AFAD$ passa de uma configuração c_i para a configuração c_j sem executar nenhuma ação adaptativa, o conjunto de estados Q_i e a função de transição δ_i permanecem inalterados na nova configuração e apenas o símbolo correspondente da cadeia de entrada é consumido. Com base nisso, pode-se estabelecer uma equivalência entre os modelos considerando que todos os autômatos presentes entre duas ações adaptativas de um $AFAD$ corresponde a um passo do contador do dispositivo dirigido a regras.

Já nas ações adaptativas do $AFAD$, nota-se uma facilidade em relação às do dispositivo dirigido por regras, pois, através do uso do símbolo especial "$\star$" nas mesmas, ele faz com que a variável que armazena o estado consultado e o estado gerado não sejam necessárias.

Na prática, enquanto o dispositivo dirigido por regras implementa o conceito de variáveis locais associadas às funções adaptativas, a definição do $AFAD$ impõe que o último elemento de sua configuração armazene o conteúdo do último estado gerado (ou o estado inicial, caso nenhum tenha sido gerado ainda), funcionando como uma espécie de variável global.

Com relação ao diagrama dos dois autômatos (representados na figura 3), não é possível notar uma grande diferença, uma vez que o diagrama apresenta apenas o modo como os estados estão relacionados entre si, que é a mesma para ambos os autômatos.

V. Conclusão

Este trabalho apresentou uma definição simplificada para a representação de Autômatos Adaptativos.

Dentre as vantagens dessa representação, destacam-se:

- Modelagem simplificada bastante semelhante com a definição dos Autômatos Finitos;
- Inexistência, por definição, de não-determinismo (embora essa ainda possa ser facilmente acrescentada ao modelo);
- Disponibilidade de um método para referência dos estados atual e recém-criado;
- Possibilidade de recursão das ações adaptativas.

Dadas essas características, espera-se que este trabalho colabore para que o desenvolvimento de sistemas envolvendo Autômatos Finitos Adaptativos ocorra de uma maneira mais fácil, prática e intuitiva.

Em trabalhos futuros espera-se explorar com mais detalhes sobre as características da notação proposta a fim de compará-las com outros modelos de computação adaptativa compatíveis, assim como estudar suas propriedades e suas limitações.

Referências

[1] Bó. I. G. L.; Sichman, J. S. Strategy representation complexity in an evolutionary n-Players Prisoner's Dilemma model. In: SEMINARIO DE COMPUTACAO, 16. Blumenau. SC. Anais. Blumenau: 2007. p. 112-124.

[2] Neto, J. J. Adaptive Rule-Driven Devices - General Formulation and Case Study. Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol.2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250.

[3] Neto, J. J. Contribuições à metodologia de construção de compiladores. Tese de Livre Docência, EPUSP, São Paulo, 1993.

[4] Rocha, R. L. A. e Neto, J. J. Autômato adaptativo, limites e complexidade em comparação com máquina de Turing. In: Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, p. 33-48, 2001.

[5] Turing, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, vol. s2-42. p. 230–265. 1936. [doi:10.1112/plms/s2-42.1.230] and; Turing, A. M.. On Computable Numbers, with an Application to the Entscheidungsproblem: A correction. Proceedings of the London Mathematical Society. vol. s2-43. p. 544–546. 1937. [doi:10.1112/plms/s2-43.6.544].

[6] Sipser, M. Introdução à Teoria da Computação: Tradução da 2ª edição norte-americana (trad. Ruy José Guerra Barreto de Queiroz) Thomson Learning, 2007.

[7] Lewis, H. R. & Papadimitriou, C. H. Elementos de Teoria da Computação (trad. Edson Furmankiewicz) Bookman, 2000.

[8] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. Introduction to Algorithms MIT Press, 2002.

[9] Castro JR., A. A.; Neto, J. J.; Pistori, H. Determinismo em Autômatos de Estados Finitos Adaptativos. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 515-521).

[10] Ramos, M. V. M.; Neto, J. J. & Vega, Í. S. Linguagens Formais: Teoria, Modelagem e Implementação. Porto Alegre, Bookman, 2009, 656p.

[11] Pistori H.; Neto J. J. Tecnologia adaptativa em engenhara de computação: Estado da arte e aplicações, Tese de Doutorado, Universidade de São Paulo, São Paulo, Brasil, 2003.

[12] Pariente, C. A. B. Gramáticas Adaptativas - Estado Atual da Pesquisa Aplicada em Autômatos Adaptativos. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 506-514)

[13] Hromkovic, J.; Karhumaki, J.; Klauck, H.; Schnitger, G.; Seibert, S. Measures of nondeterminism in finite automata, in Proc. ICALP 2000, Lecture Notes in Computer Science, vol. 1853, 2000, pp. 199-210.

[14] Stearns, R. E. Deterministic versus nondeterministic time and lower bound problems. Journal of the ACM, vol. 50, no. 1, pp. 91-95, 2003.

Avaliação de uma estratégia para vencer o jogo de "pedra, papel e tesoura" usando técnicas adaptativas

F. S. Santana, C. Barberato, A. M. Saraiva

***Resumo* — O jogo "pedra-papel-tesoura", PPT, sempre foi usado para tomar decisões pouco importantes. Recentemente, porém, ele tem sido aplicado para modelar problemas em computação evolucionária, biodiversidade, aprendizado e tomada de decisão em áreas como economia, psicologia e neurociência. Existem, também, competições oficiais de PPT, promovidas por associações profissionais. Resultados matemáticos afirmam que jogar aleatoriamente é a "melhor" estratégia, mas alguns profissionais propõem outras. Este trabalho avalia uma delas, que considera padrões de comportamento e aspectos emocionais nas decisões dos jogadores. A estratégia foi modelada por uma máquina de estados, onde tabelas de decisão adaptativas foram incluídas para tratar o estado emocional dos jogadores. Experimentos sucessivos podem modificar a estratégia original, para fortalecê-la, enquanto desvios probabilísticos nos resultados dos jogos podem indicar sua eficácia. Como resultado adicional, este pode ser considerado um estudo de caso para avaliar uma possibilidade de identificar padrões de comportamento, aplicáveis na definição de métodos confiáveis de modelagem.**

***Palavras chaves* — Sistemas adaptativos, técnicas adaptativas, máquinas de estado, jogo "pedra-papel-tesoura".**

I. NOMENCLATURA

PPT – Refere-se ao jogo "Pedra-Papel-Tesoura".

II. INTRODUÇÃO

Pedra, papel e tesoura, PPT, é um jogo bastante comum e usado no quotidiano para resolver conflitos que envolvem a tomada de decisões de pouca importância (McCannon, 2007). Embora seja de origem desconhecida, este jogo é muito popular no Japão desde o século XIX, por isso também é conhecido como "jan-ken-pon" ou "janken-po" (algumas das tentativas de "pronúncia" ou "tradução" pelos ocidentais). No ocidente, o jogo começou a se popularizar a partir do século XX. No uso quotidiano, o PPT equivale aos jogos de "par ou ímpar", "cara ou coroa" e "dois ou um", que talvez sejam mais populares no Brasil, no sentido de que todos são usados para se tomar decisões direcionadas principalmente pelo fator "sorte", pelo menos na teoria. Desta forma, não existe um responsável direto pela tomada de decisão.

Nos últimos anos, no entanto, com a evolução dos estudos em teoria dos jogos e sua consequente aplicação em diversas outras áreas do conhecimento, o PPT tem sido utilizado como base para a modelagem e o estudo de diversos problemas relevantes (McCannon, 2007). Entre eles, estão resultados em computação evolucionária (Hofbauer; Sigmund, 2003), colaboração e comportamento de agentes em sistemas de aprendizado em inteligência artificial (Namatame), estudos em biodiversidade e reprodução de espécies (Alonzo; Sinervo, 2001) e diversas aplicações que utilizam modelos quantitativos para descrever o processo de tomada de decisão em diversas áreas da ciência, incluindo economia (McCannon, 2007), psicologia e neurociência (Lee; McGreevy; Barraclough; 2005).

Outro fato pouco conhecido é o de que existem competições de caráter local e internacional de PPT, promovidas por sociedades internacionais compostas por jogadores profissionais, como a Sociedade Internacional de PPT (*World RPS Society* [http://www.worldrps.com/]) e a Liga Oficial Americana de PPT (*USARPS League* [http://www.usarps.com/]). Estas sociedades promovem competições periódicas, com regras oficiais e bem definidas para se jogar PPT, como o "*World Rock Paper Scissors Championship*", promovido pela Sociedade Internacional de PPT e o "*USARPS chAMPionships*", promovido pela Liga Oficial Americana de PPT.

Existem resultados matemáticos que afirmam que a "melhor" estratégia para se vencer o jogo de PPT é sempre jogar aleatoriamente (van den Nouweland, 2007). Porém, nas competições oficiais ocorre a distribuição de altos prêmios em dinheiro para os vencedores. É natural, portanto, que os jogadores profissionais busquem constantemente por estratégias eficazes para aumentar as probabilidades de vencer este jogo.

Este trabalho avalia uma das estratégias não aleatórias para se vencer o PPT, proposta por jogadores (Walker, G.; Walker, D, 2004) da Sociedade Internacional de PPT. De acordo com os autores desta estratégia, dificilmente o comportamento humano consegue ser de fato aleatório. Diversos aspectos, principalmente os de natureza emocional, influem nas decisões tomadas pelos jogadores, que acabam obedecendo a padrões pré-determinados de comportamento. Portanto, a identificação e o estudo destes padrões permitem a criação e a utilização de técnicas capazes de aumentar as chances de

The authors are grateful to FINEP/MCT Brazil, for the support to the Prosensap project.

F. S. Santana is professor at Universidade Federal do ABC, Center of Research in Mathematics, Computer and Cognition, Rua Santa Adélia, 166, Bloco B, Sala 811. Bairro Bangu. Santo André - SP - CEP: 09210-170 (fabiana.santana@gmail.com).

C. Barberato is professor at FEI, Department of Computer Science. Av Humberto de Alencar Castello Branco, 3972, São Bernardo do Campo - SP. CEP: 09850-901 (claudio.barberato@gmail.com).

A. M Saraiva is full professor at Escola Politécnica da Universidade de São Paulo, Department of Engineering of Computation and Digital Systems, Av. Prof. Luciano Gualberto, travessa 3, número 158 - Cidade Universitária - São Paulo - SP - CEP: 05508-900 (saraiva@usp.br).

vitória no PPT.

Para modelar as regras propostas por esta estratégia, foi construída uma máquina de estados onde algumas das decisões precisam ser tomadas usando tabelas de decisões adaptativas. O uso destas tabelas é essencial, pois elas permitem avaliar com mais profundidade o contexto atual do jogo, considerando diversos potenciais estados emocionais dos jogadores oponentes e diversas condições simultâneas. A partir desta avaliação, as ações que devem ser tomadas em algumas situações precisam ser redefinidas, o que não é possível fazer apenas com a construção de uma máquina de estados.

A realização de sucessivos experimentos a partir da implementação da estratégia aqui proposta, é o método proposto para avaliar a sua eficácia do ponto de vista prático. Os resultados de cada jogada serão armazenados em estruturas de dados apropriadas, enquanto se realizarem os experimentos, a fim de construir um histórico de jogadas.

O primeiro objetivo é permitir a realização dos cálculos estatísticos para avaliar esta estratégia, como proposto. Os resultados dos jogos realizados serão comparados com os resultados que seriam obtidos se a estratégia aleatória estivesse sendo utilizada e também com os valores estatisticamente esperados. A ocorrência de desvios probabilísticos no número real de vitórias em relação a estas duas medidas pode indicar se o uso desta estratégia aumenta de fato as probabilidades de vencer o jogo de PPT, desde que o número de experimentos realizados seja significativo.

Porém, deve-se notar que a análise do histórico de jogadas pode resultar também na eventual descoberta de outros padrões de jogo, não previstos originalmente por (Walker, G.; Walker, D, 2004). Desta forma, estes novos padrões, se existirem, podem ser incorporados à estratégia original, de forma a aumentar as probabilidades de vitória no jogo de PPT.

Como resultado adicional desta pesquisa, caso se comprove a existência e a eficácia de uma estratégia adaptativa para vencer o jogo de PPT, este vai se configurar como um estudo de caso para mostrar que é possível identificar e tratar determinados padrões de comportamento usando técnicas adaptativas. A importância deste fato é, talvez, ainda mais relevante do que a própria possibilidade de se criar uma estratégia estatisticamente vencedora de PPT.

Uma vez identificados os padrões de comportamento, estes podem ser aplicados na construção de modelos ainda mais confiáveis do que os disponíveis atualmente para a tomada de decisão baseados no jogo de PPT. Além disso, este raciocínio possivelmente poderá ser estendido para outros estudos em teoria dos jogos, além do próprio PPT, aumentando ainda mais as aplicações potenciais das tecnologias adaptativas.

III. MATERIAIS E MÉTODOS

A. *O jogo*

PPT é um jogo que se joga com as mãos, fazendo os gestos que representam pedra, papel e tesoura, respectivamente de acordo com as Figuras 1, 2 e 3.

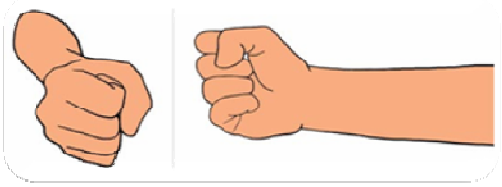


Figura 1 – Representação manual de pedra (extraída de http://worldrps.com/game-basics).

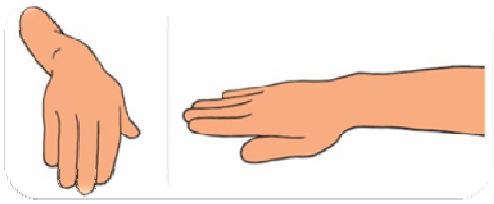


Figura 2 – Representação manual de papel (extraída de http://worldrps.com/game-basics).

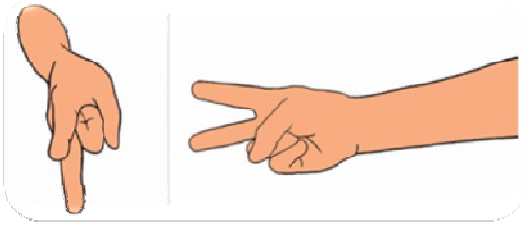


Figura 2 – Representação manual de tesoura (extraída de http://worldrps.com/game-basics).

De acordo com Walker (2006), PPT não é um simples jogo de sorte ou azar e requer a adoção de estratégias específicas. O principal motivo é a incapacidade do ser humano de jogar aleatoriamente. Segundo Walker (2006), mesmo ao tentar de fato jogar aleatoriamente, o comportando do ser humano acaba se tornando previsível ao longo do tempo. Portanto, é fundamental conhecer os fatores, principalmente de caráter psicológico, que motivam as ações do oponente no jogo de PPT. Este conhecimento permite alterar a distribuição de probabilidade, aumentando consideravelmente suas chances de vencer o jogo. Naturalmente, este raciocínio só se aplica quando a decisão do jogo ocorre após rodadas sucessivas, que é o padrão adotado nas competições internacionais [http://worldrps.com/game-basics].

As regras do jogo de PPT são simples (McCannon, 2007):

- Os jogadores devem apresentar os sinais simultaneamente;
- O resultado é determinado pelas seguintes regras, como mostra a Figura 4:
  - Pedra vence tesoura;
  - Tesoura vence papel;
  - Papel vence pedra.
- Os jogadores devem combinar um determinado número de rodadas. Para este trabalho, este número será definido como n, sendo n um número inteiro, finito e ≥ 1.

Figura 4 – As regras básicas do jogo: "pedra" vence "tesoura"; "tesoura" vence "papel" e "papel" vence "pedra".

B. A estratégia

(Walker, G.; Walker, D, 2004) e Walker (2006) estabelecem que, basicamente, existem duas formas de se vencer no PPT. A primeira é retirar uma das opções do adversário, o que significa que, se o adversário for induzido a não escolher "pedra", então é possível escolher com segurança "tesoura", pois nesse caso o pior resultado será um empate. Esta opção parece impossível, a menos que sejam conhecidas formas de se manipular o adversário sem que ele saiba o que está ocorrendo. A segunda maneira é forçar o adversário a fazer um movimento previsível. Novamente, é essencial que o adversário não saiba que está sendo manipulado.

As técnicas propostas por (Walker, G.; Walker, D, 2004) e Walker (2006) a partir destas duas afirmações, e que são computacionalmente implementáveis, são as seguintes:

1. Homens, principalmente os iniciantes, têm a tendência de escolher "pedra" na sua jogada inicial. Isto está relacionado com o fato de que a "pedra" é percebida como uma jogada mais "forte" do que as demais. Portanto, se o adversário for um homem, a primeira jogada deve ser "papel". Esta regra não funciona para mulheres e jogadores experientes.
2. Seguindo o raciocínio feito em 1., "tesoura" deve ser a primeira jogada segura escolhida contra um jogador experiente. Como começar com "pedra" é muito óbvio, um jogador experiente iniciará com "papel", caso em que perde, ou "tesoura", caso em que ocorre um empate.
3. Quando o adversário é inexperiente, verifique se ele repete a mesma jogada duas vezes seguidas. Se isto ocorrer, este jogador tem a tendência a não repetir a mesma jogada uma terceira vez. Portanto, é possível eliminar uma opção e garantir no mínimo um empate, uma das duas opções iniciais de evitar a derrota. Isto ocorre justamente porque as pessoas não querem ser previsíveis e é esta a percepção de quem repete a mesma jogada três vezes seguidas.
4. Quando não souber o que fazer para a próxima jogada e tiver acabado de vencer uma jogada, escolha a opção que o teria feito perder esta jogada. Alguns jogadores, especialmente os menos experientes, têm a tendência a escolher a opção que o teria feito vencer a última jogada, de maneira inconsciente. Por exemplo, se na jogada anterior o adversário apresentou "papel" e perdeu, sua tendência é vir da próxima vez com "tesoura", portanto "pedra" é a melhor opção. Esta opção não é muito boa se o adversário venceu a jogada anterior.
5. Quando não souber o que fazer e não houver nenhuma outra informação disponível, escolha "papel". Isto porque foi observado em competições oficiais que a opção menos escolhida é "tesoura". De acordo com Walker (2006), a probabilidade de se escolher "tesoura" é 29,6%, contra o valor esperado de 33,3%. A vantagem é de apenas 3,7%, mas é melhor do que não ter nenhuma vantagem.

(Walker, G.; Walker, D, 2004) e Walker (2006) ainda propõem outras técnicas, implementáveis apenas através de tabelas de decisão adaptativas, pois elas requerem a análise histórica do oponente.

A estratégia proposta neste trabalho a partir das observações de (Walker, G.; Walker, D, 2004) e Walker (2006), também classifica os jogadores nos níveis "Iniciante", "Médio" e "Experiente". Na primeira vez, o nível deve ser informado. Nas jogadas seguintes, o jogador pode ser reclassificação de acordo com o seu desempenho no jogo. A reclassificação é feita no início de cada jogada.

IV. MODELAGEM ADAPTATIVA DO PPT

Para implementar esta estratégia, a aplicação consiste de um jogo com dois jogadores, sendo um deles a própria aplicação, a partir deste momento chamada de "Jogador", enquanto o outro jogador deve ser obrigatoriamente um ser humano, que será chamado de "Oponente". A estratégia de jogo implementada na aplicação será composta pelas propostas de (Walker, G.; Walker, D, 2004) e Walker (2006), adequada às técnicas adaptativas necessárias.

Embora experimentos não sejam suficientes para provar a eficácia desta estratégia, desvios probabilísticos significativos podem representar um resultado relevante, desde que o número de experimentos realizados seja suficientemente grande. Estes resultados práticos podem ser importantes, considerando a relevância do estudo e da modelagem do jogo de PPT para diversas áreas da ciência nos dias de hoje.

Tabela 1 – Representação do jogo PPT usando tabelas de decisão.

		Jogador 1		
		Pedra	**Papel**	**Tesoura**
Jogador 2	**Pedra**	**0**	**1**	**-1**
	Papel	**-1**	**0**	**1**
	Tesoura	**1**	**-1**	**0**

O PPT tradicional pode ser definido utilizando uma tabela de decisão simples, em uma solução não adaptativa. A Tabela 1 apresenta esta modelagem, sendo que o número "1"

representa a vitória, o número "0" representa o empate e o número "-1" representa a derrota do Jogador 1 em relação ao Jogador 2.

Um jogo é composto por *n* jogadas e estas *n* jogadas devem ser armazenadas em uma estrutura de dados adequada, a fim de permitir acompanhar e analisar o histórico de jogadas. Esta estrutura está apresentada na Figura 5. Além de registrar as jogadas do "Jogador" e do "Oponente", esta estrutura armazena, a cada jogada, a tendência do jogador a escolher uma das opções do jogo de PPT. Esta tendência deve ser recalculada após o término da jogada, para não interferir no resultado da máquina de estados. Portanto, é possível saber, a cada instante, qual a porcentagem de vezes em que cada jogador escolheu "Pedra", "Papel" ou "Tesoura".

Para a realização dos experimentos, uma terceira estrutura exatamente igual a esta será criada para simular outro jogador, chamado de "Jogador Aleatório". Este jogador não participará diretamente do jogo, embora a sua opção de jogada será sorteada e armazenada a cada jogada. O objetivo é a realização das análises estatísticas propostas pelo método de avaliação. Vale lembrar que as análises vão considerar os resultados da estratégia implementada em comparação com os valores estatisticamente esperados e em comparação com os resultados da estratégia aleatória. Desta forma, este terceiro jogador, embora não participe do jogo, é necessário para a avaliação da estratégia.

Um mecanismo para modelar os cinco itens implementáveis da estratégia proposta em Walker (2006) precisa avaliar, a cada jogada *k*, $0 \leq k \leq n$, em qual das situações a aplicação se encontra. Para isto, é necessário conhecer o sexo do jogador oponente, que supostamente é um ser humano. Como as hipóteses propostas consideram o comportamento humano, Walker (2006) supõe que o comportamento entre homens e mulheres é diferente. Este raciocínio, na verdade, é suportado por outros pesquisadores em biodiversidade. Sinervo e Lively (1996), por exemplo, utilizaram o PPT para analisar o comportamento e as estratégias dos machos na teoria da evolução das espécies, em complemento à anterior concepção de se utilizar apenas o "*fitness*" de cada espécie.

A partir destas hipóteses, foi construída uma máquina de estados para implementar esta estratégia, onde algumas decisões devem ser tomadas de acordo com uma tabela de decisões adaptativa.

A máquina de estados está apresentada nas figuras a seguir. A Figura 6 apresenta o início de cada *k*-ésima jogada, onde o Oponente pode ser reclassificado de acordo com a sua pontuação. De acordo com o nível estabelecido para o oponente, ele será direcionado para a máquina de estados apresentada na Figura 7, que trata do Oponente iniciante, na Figura 8, que trata do Oponente de nível médio, ou Figura 9, que trata do Oponente experiente. Esta reclassificação é necessária, pois o Oponente pode informar o nível errado no início do jogo e, desta forma, aumentar as suas chances de vitória.

Tabelas de jogadas

Jogador	Jogadas	0	...	k	...	n
Registro de jogadas	Preencher as colunas com "Pedra", "Papel" ou "Tesoura", onde k é o número da jagada.					
Tendência	Pedra		Papel		Tesoura	
	%		%		%	

Oponente	Jogadas	0	...	k	...	n
Registro de jogadas	Preencher as colunas com "Pedra", "Papel" ou "Tesoura", onde k é o número da jagada.					
Tendência	Pedra		Papel		Tesoura	
	%		%		%	

Figura 5 – Estrutura de dados para armazenar as jogadas de cada jogador.

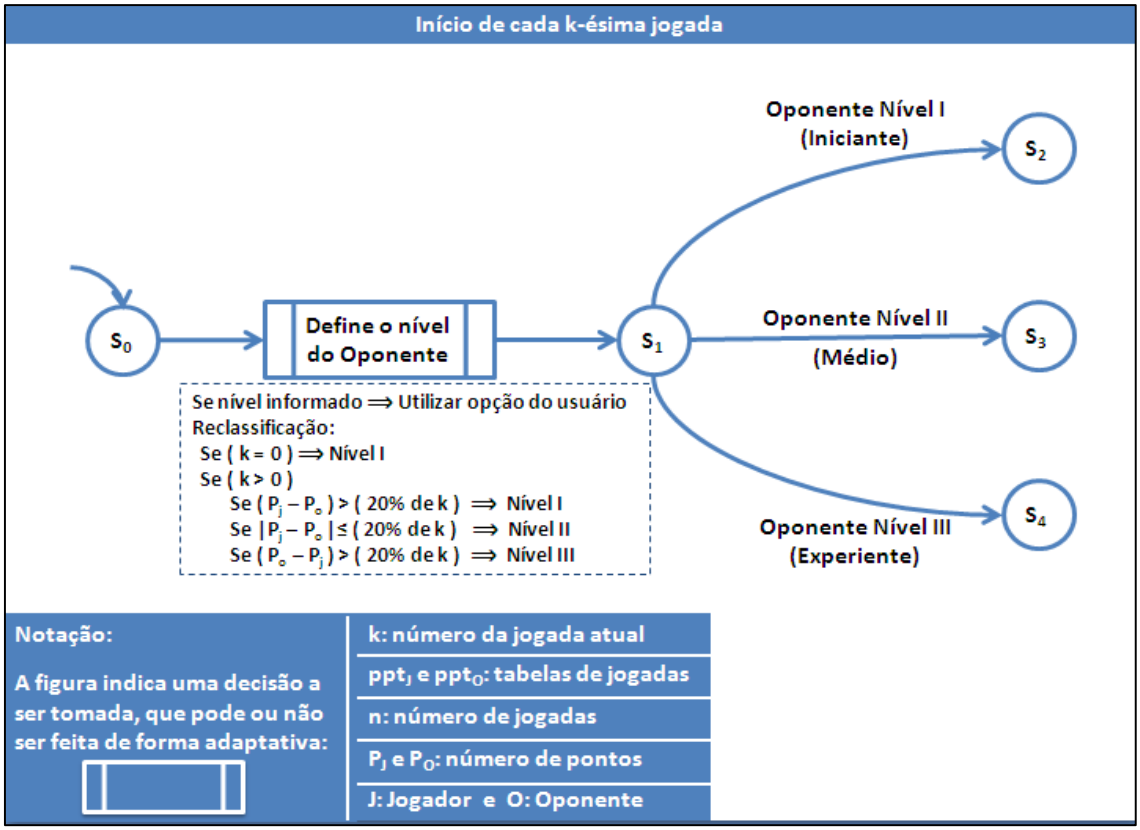


Figura 6 – Máquina de estados: início de cada *k*-ésima jogada.

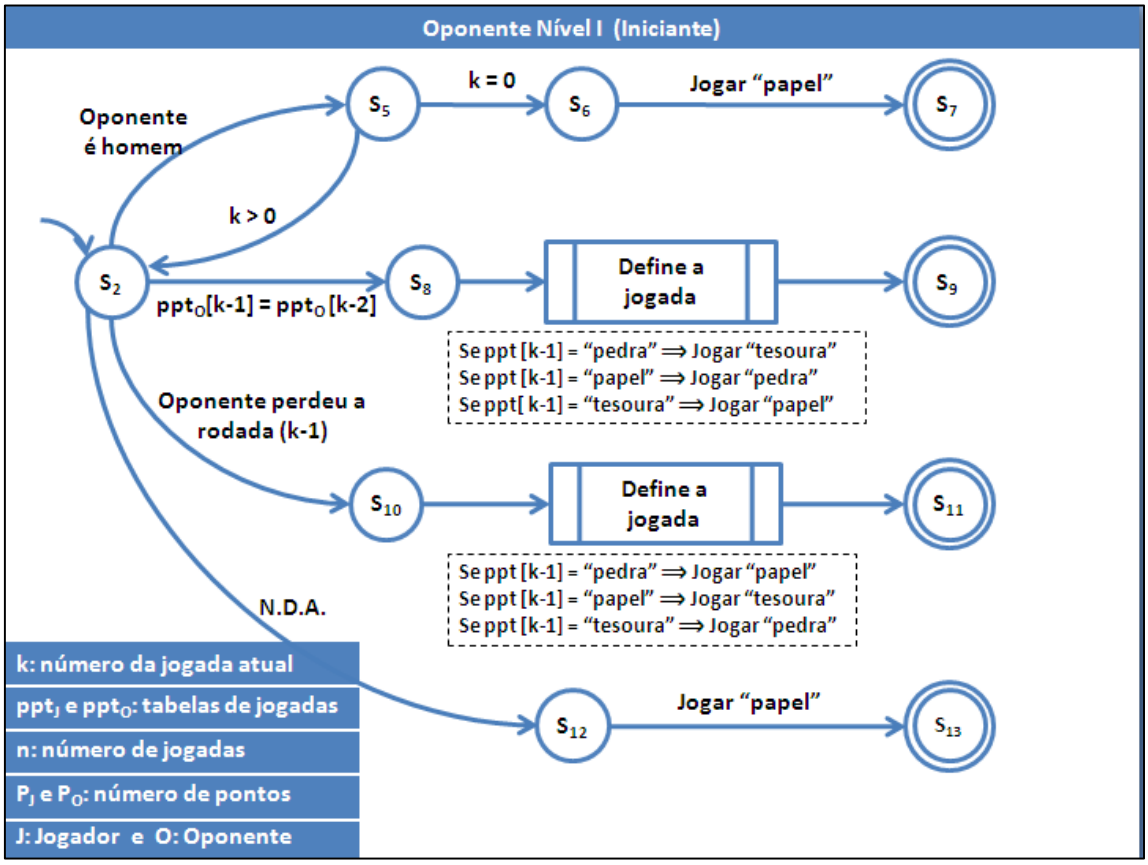


Figura 7 – Máquina de estados: tratamento para o oponente iniciante.

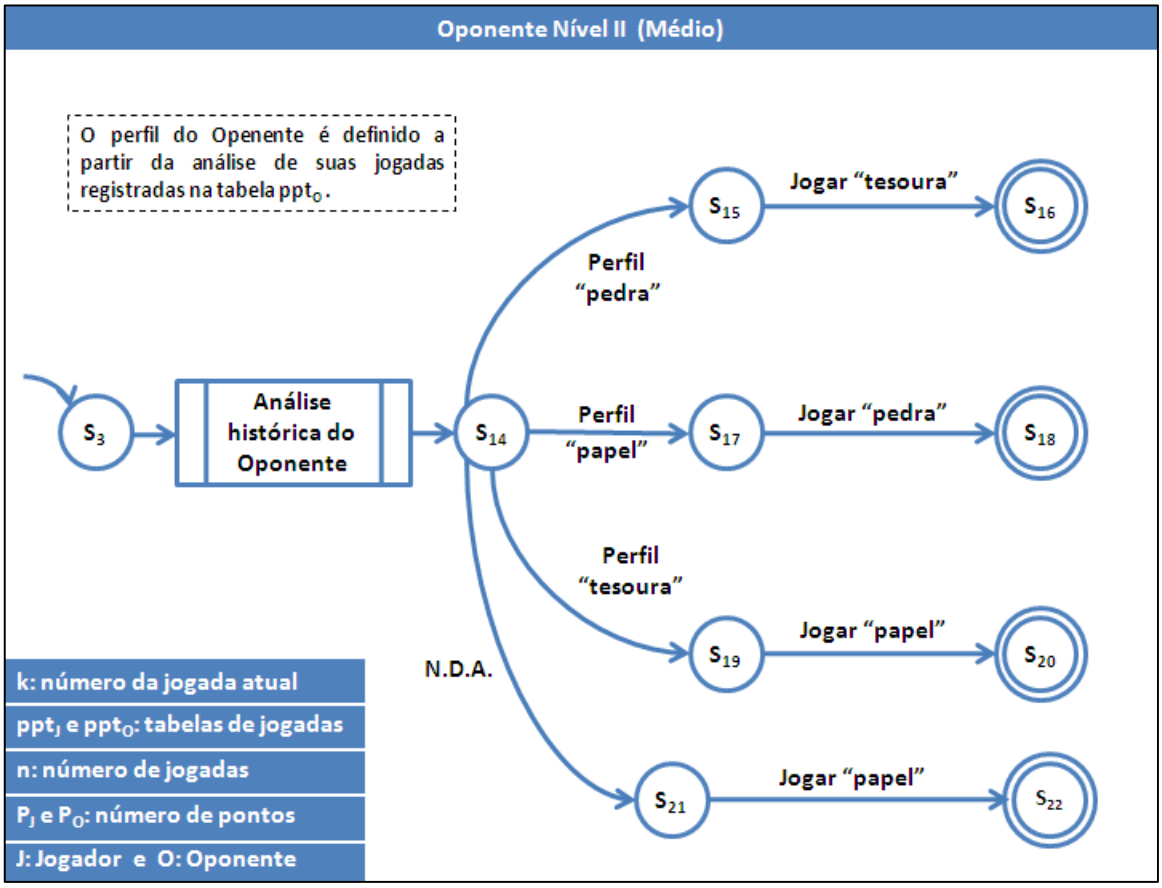


Figura 8 – Máquina de estados: tratamento para o oponente de nível médio.

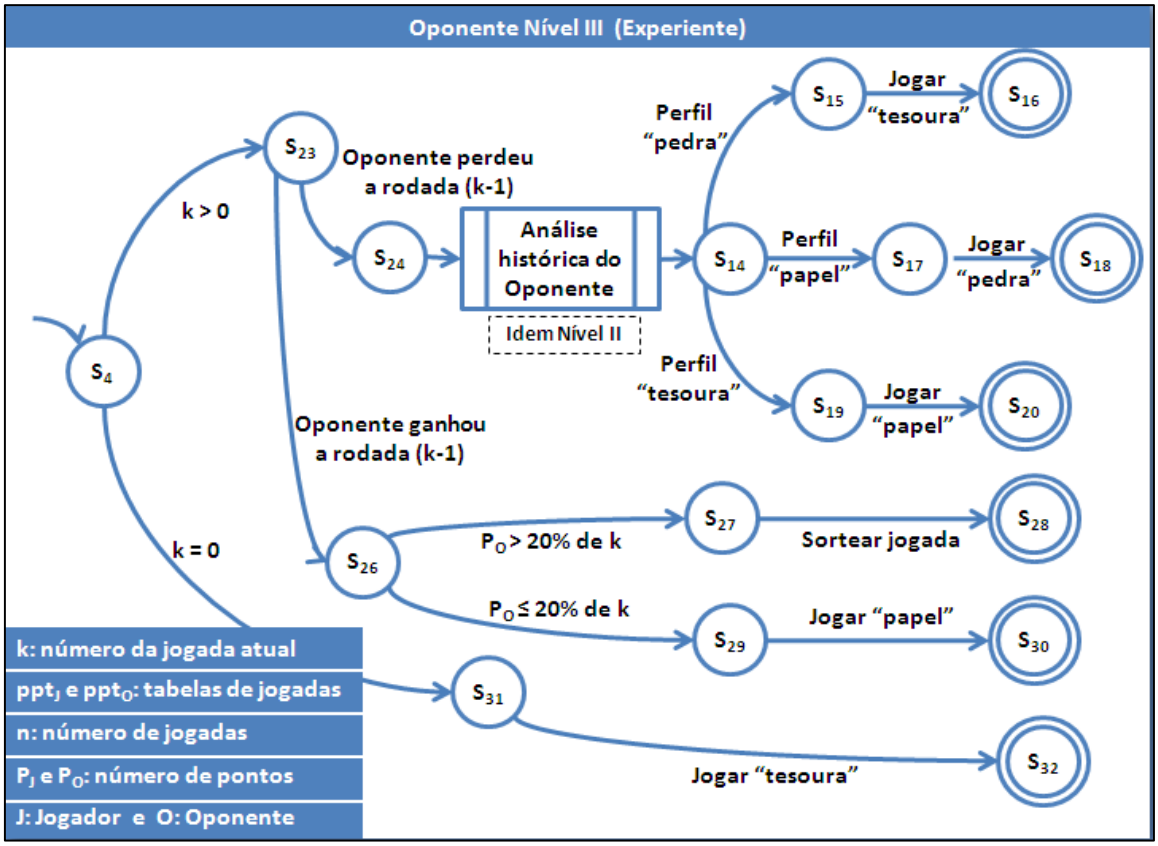


Figura 9 – Máquina de estados: tratamento para o oponente experiente.

Tabela de decisão – Análise Histórica do Oponente para Definição de Perfil

	Jogadas	0	...	k	...	n
Condições	C_1					
	...					
	C_G					
Ações	A_1					
	...					
	A_H					
Funções Adaptativas	F_1					
	...					
	F_I					

Figura 10 – Tabela de decisão adaptativa para análise histórica do oponente e definição do seu perfil.

Como a reclassificação do jogador se baseia no seu desempenho, o nível de jogo e a estratégia aplicada a ele podem variar, conforme ele vai sendo bem sucedido em suas jogadas.

A partir da definição da máquina de estados, o algoritmo é simples:

- O sexo e o nível do Oponente devem ser fornecidos no início do jogo.
- O número *n* de rodadas deve ser estabelecido no início do jogo.
- A cada jogada, digamos *k*, o Jogador, que é a aplicação, obedece a máquina de estados, cujo objetivo é definir se a sua opção de jogo deve ser "pedra", "papel" ou "tesoura" para esta *k*-ésima rodada.
- A cada nova jogada, a pontuação é avaliada para verificar a necessidade de se redefinir o nível do jogador, de acordo com a situação do jogo. Conforme melhora o desempenho do Oponente, a avaliação do seu nível de experiência aumenta.
- A pontuação é definida de maneira simples, sendo que uma vitória soma "1" ponto, enquanto o empate contabiliza "0" pontos. O critério punitivo, que equivaleria a atribuir "-1" ponto, não será aplicado nesta primeira abordagem.

A análise histórica do Oponente é um dos pontos principais da estratégia proposta por (Walker, G.; Walker, D, 2004) e Walker (2006). Porém, esta análise é simples de ser realizada quando se está frente a frente com um adversário, mas não quando deve ser feita de forma "calculada". Para fazê-la, o algoritmo implementa uma tabela de decisão adaptativa, que permite expandir as possibilidades e alterar o comportamento do Jogador de acordo com as alterações e padrões de comportamento do Oponente. Teoricamente, esta tabela deveria prever os potenciais "estados emocionais" do Oponente, conforme a situação do jogo.

Por exemplo, suponha que as seguintes condições e ações estão implementadas na tabela de decisão adaptativa:

1. Suponha que a *Condição 1* seja que o Oponente perde a jogada (k-1). A perda de uma jogada aumenta o nível de frustração do Oponente e uma reação comum, especialmente em jogadores iniciantes ou estressados, é escolher a opção que teria vencido a jogada anterior (k-1). Portanto, a *Ação 1* correspondente deve ser justamente escolher o perfil que teria perdido em (k-1). A definição do perfil permite que a máquina de estados decida o que fazer e, se o Oponente jogar como previsto, no mínimo será possível obter um empate.
2. Suponha que a *Condição 2* seja que o Oponente vence a jogada (k-1). A vitória normalmente aumenta o nível de satisfação do Oponente, deixando-o mais relaxado para escolher livremente qualquer jogada. Neste caso, a *Ação 2* pode ser optar por jogar "Papel" (lembrando que estatisticamente a opção "Tesoura" é a menos escolhida) ou, alternativamente, sortear o perfil. Digamos que a opção escolhida seja a segunda.
3. Suponha que a *Condição 3* seja analisar a tendência do Oponente a escolher com mais freqüência uma das opções entre "Pedra", "Papel" e "Tesoura". Neste caso, a *Ação 3* correspondente é classificar este jogador pelo perfil mais frequente.

Até aqui, temos a definição de condições e ações que são comuns a uma tabela de decisão simples, não adaptativa.

A adaptatividade do algoritmo ocorre no momento em que se inserem funções adaptativas nesta tabela, de forma a alterar as ações correspondentes às condições anteriormente discutidas. Ainda no mesmo exemplo, poderíamos incluir as seguintes funções adaptativas:

1. A função adaptativa F_1 deve acompanhar a tendência do Oponente, de forma semelhante ao que ocorre na *Condição 3*. Porém, caso se identifique

uma tendência muito alta, por exemplo ultrapassando os 50%, para a escolha de uma das opções de "Pedra", "Papel" ou "Tesoura", então tanto a *Ação 1* quanto a *Ação 2* devem ser substituídos pela *Ação 3*. Pois uma tendência muito alta justifica o abandono da estratégia original.

2. A função adaptativa F_2 deve também acompanhar a tendência do Oponente, de forma semelhante ao que ocorre na *Condição 3*. Porém, o objetivo neste caso é verificar se ocorre uma redução nos valores de tendência anteriormente identificados, tendendo para a equalização das opções. Por exemplo, as novas tendências demonstrem que nenhuma das opções de "Pedra", "Papel" ou "Tesoura" é escolhida mais do que 40% das vezes. Neste caso, tanto a *Ação 1* quanto a *Ação 2* podem retornar à estratégia pré-definida.

Da mesma forma que neste exemplo, outras tantas condições, ações e funções adaptativas podem ser implementadas, a fim de se modelar o estágio de frustação ou confiança do Oponente. É possível, por exemplo, analisar o placar, o histórico de jogadas, sequências de vitórias ou de derrotas e assim por diante, definindo funções adaptativas para que o comportamento do Jogador se altere tanto quanto necessário, aumentando desta forma as suas probabilidades de vencer.

Após as n jogadas, vencerá o jogador que tiver o maior número de pontos. Em caso de empate, o jogo deve continuar por mais três rodadas. Esta regra será aplicada de forma repetitiva até que seja estabelecido um vencedor ou o Oponente desista.

O resultado dos jogos deve ser armazenado em arquivo para a posterior realização de cálculos estatísticos para avaliar os experimentos realizados e verificar os eventuais desvios a fim de validar ou evoluir a estratégia de jogo de PPT, a fim de se concluir ou não sobre a possibilidade de construir uma estratégia que, de fato, aumente as probabilidades de vitória de um jogador.

V. Resultados esperados

A implementação desta aplicação e a realização de um número suficientemente grande de experimentos vai permitir uma avaliação estatística e comparativa entre as estratégias de jogo para PPT. É importante, para o sucesso do estudo, que os jogadores não conheçam os detalhes da estratégia proposta neste trabalho, pelo menos na primeira avaliação, para desestimular padrões forçados de jogadas. Posteriormente, experimentos podem ser realizados mesmo com a divulgação destas estratégias.

Caso haja qualquer desvio probabilístico significativo nas avaliações realizadas, de forma a alterar significativamente a esperança de vitória no PPT, este resultado será investigado com maior profundidade, eventualmente envolvendo aplicações em outras áreas do conhecimento.

VI. Conclusão

Este trabalho propôs uma solução adaptativa para implementar uma estratégia, supostamente vencedora, para o jogo PPT. Além disso, foi proposta uma estratégia para avaliar a eficácia desta solução em relação à proposta original que matematicamente é a mais aceita, e que consiste em simplesmente jogar de maneira aleatória.

A justificativa para se fazer esta avaliação foi a afirmação de Walker (2006), autor da estratégia original na qual a solução adaptativa foi baseada, de que é possível identificar facilmente padrões no comportamento humano.

A existência e a eficácia de uma estratégia adaptativa para vencer este jogo pode se configurar como um estudo de caso relevante, onde foi possível identificar e tratar diferentes padrões de comportamento, mesmo que pré-estabelecidos, aplicando tecnologia adaptativa. Esta pode ser a maior contribuição deste trabalho, uma vez que, mais importante do que vencer no jogo de PPT, é a possibilidade de tratar padrões de comportamento para definir estratégias eficazes para a tomada de decisão.

Como o uso da tabela de decisão adaptativa em conjunto com máquinas de estado pode, possivelmente, ser estendida para outras aplicações, esta pode ser uma nova e interessante aplicação potencial das tecnologias adaptativas para a solução de problemas computacionais.

Espera-se que os resultados desta avaliação, neste momento de caráter puramente exploratório, direcionem novas pesquisas neste sentido, oferecendo uma alternativa adicional para a modelagem matemática e/ou quantitativa de problemas encontrados em diversas áreas do conhecimento.

Referências

ALONZO, S. H. E SINERVO, B.. Mate choice games, context-dependent good genes, and genetic cycles in the side-blotched lizard, Uta stansburiana Behav Ecol Sociobiol (2001) 49:176–186. Springer-Verlag 2001.

CHANG, Y-H e KAELBLING, L. P. Playing is believing: the role of beliefs in multi-agent learning. M.I.T. Artificial Intelligence Laboratory.

HOFBAUER, J. e SIGMUND, K.. BULLETIN (New Series) OF THE AMERICAN MATHEMATICAL SOCIETY Volume 40, Number 4, Pages 479-519. S 0273-0979(03)00988-1 Article electronically published on July 10, 2003 EVOLUTIONARY GAME DYNAMICS

LEE, D., MCGREEVY, B. P. e BARRACLOUGH, D. J. Learning and decision making in monkeys during a rock–paper–scissors game Department of Brain and Cognitive Sciences, Center for Visual Science, University of Rochester, Rochester, NY 14627, USA. Cognitive Brain Research 25 (2005) 416 – 430.

MCCANNON, B. C. Rock Paper Scissors. Vol. 92 (2007), No. 1, pp. 67–88. DOI 10.1007/s00712-007-0263-5 Printed in The Netherlands. Journal of Economics.

NAMATAME, A. Emergence of Desired Collectives with Evolving Synchronized Coupling Rules. Dept. of Computer Science, National Defense Academy, Yokosuka, 239-8686, JAPAN.

SINERVO, B. e LIVELY, C. M. The rock-paper-scissors game and the evolution of alternative male strategies. Nature. Vol. 380, March, 1996. Pp. 240-243.

WALKER, G. The Official Rock Paper Scissors Strategy Guide. 2004.

WALKER, G. How to beat anyone at Rock Paper Scissors. 2006. Disponível em [http://www.worldrps.com/rps-news-and-notes/how-to-beat-anyone-at-rock-paper-scissors]

Adaptatividade na Tomada de Decisão Multicritério

A. H. Tchemra

***Resumo* – Este artigo tem como objetivo apresentar as principais características de uma extensão da tabela de decisão adaptativa, denominada Tabela de Decisão Adaptativa Estendida. Este dispositivo adaptativo tem por finalidade apoiar aplicações de tomada de decisão multicritério. Um algoritmo geral de tomada de decisão para o dispositivo adaptativo é implementado, com a incorporação de procedimentos das tabelas de decisão convencionais, dos métodos multicritério e das técnicas adaptativas. O formalismo e a aplicabilidade da Tabela de Decisão Adaptativa Estendida mostram sua viabilidade de uso nos processos decisórios que possuem múltiplos critérios.**

Palavras-chave: **adaptatividade, tabela de decisão adaptativa, tomada de decisão, métodos multicritério.**

I. INTRODUÇÃO

A adaptatividade é uma técnica que permite agregar a um dispositivo dirigido por regras características de automodificação [1]. Essa característica muda o comportamento e a estrutura do dispositivo, de forma dinâmica, automática, e sem a intervenção de agentes externos, atuando principalmente em seu conjunto de regras. Nestas condições, os métodos da Tecnologia Adaptativa podem ser amplamente empregados nos diferentes tipos de processos decisórios.

Processos decisórios, em geral, são processos dinâmicos que buscam soluções para problemas de tomada de decisão. Esses problemas exigem do responsável pela decisão a escolha de apenas uma solução, entre as várias alternativas disponíveis, mas que, simultaneamente, satisfaça as várias restrições ou critérios existentes nos problemas. A presença de múltiplos critérios leva a um conjunto de regras, que podem indicar as possíveis soluções para aquele problema. Segundo [2], nestas circunstâncias, os métodos multicritério de apoio à decisão são os mais adequados para a obtenção da solução.

O objetivo deste artigo é apresentar a Tabela de Decisão Adaptativa Estendida formalizada em [3] e sua potencialidade como dispositivo adaptativo aplicado em processos de tomada de decisão multicritério.

Este artigo apresenta na seção II a formalização da Tabela de Decisão Adaptativa Estendida e suas principais características, seguida pela seção III que descreve a operação da Tabela de Decisão Adaptativa Estendida. A seção IV apresenta os métodos multicritério utilizados nos processos de tomada de decisão. A seção V apresenta uma aplicação da Tabela de Decisão Adaptativa Estendida como ferramenta de apoio em processos que envolvam múltiplos critérios. Finalmente, na seção VI são apresentadas as conclusões e considerações sobre o dispositivo adaptativo apresentado.

II. TABELA DE DECISÃO ADAPTATIVA ESTENDIDA

A Tabela de Decisão Adaptativa ($\mathcal{TDA}$), formalizada por [1], utiliza como dispositivo subjacente uma tabela de decisão, na qual é acrescentada uma camada adaptativa. A tabela de decisão, que é a convencional, é composta por linhas que contêm as condições do problema e por colunas que formam o seu conjunto de regras. A camada adaptativa adicionada é composta por um conjunto de linhas onde são definidas as funções adaptativas, cujas ações adaptativas, quando executadas, modificam o conjunto de regras e, em consequência, a quantidade de colunas da tabela, não havendo alteração na quantidade de linhas. A estrutura geral da $\mathcal{TDA}$ é apresentada em [1] e em [3].

A $\mathcal{TDA}$ utilizada por [1] simula um autômato adaptativo para reconhecer sentenças de linguagens dependentes de contexto. Nela, as regras do processo de reconhecimento são memorizadas e modificadas com a ajuda de um conjunto de funções adaptativas, responsáveis por sua automodificação, de forma independente, apresentando novas informações e novos resultados para o processo em execução.

Pela ampla aplicabilidade da $\mathcal{TDA}$, uma extensão da $\mathcal{TDA}$ é formalizada por [3] para atuar como elemento central de um procedimento de tomada de decisão envolvendo múltiplos critérios simultâneos. Esta extensão é denominada Tabela de Decisão Adaptativa Estendida ($\mathcal{TDAE}$).

A $\mathcal{TDAE}$ tem como objetivo apoiar, por meio de uma formulação expressiva, processos decisórios de problemas

A. H. Tchemra – Universidade Presbiteriana Mackenzie (correspondência: Rua Jorge Tibiriçá, 74 – apto. 64 - São Paulo, SP, Brasil – CEP: 04126-000; e-mail:angela.hum@mackenzie.br).

semi-estruturados, nos quais existem passos conhecidos, mas insuficientes para uma tomada de decisão [4].

Na $\mathcal{TDAE}$ são implementadas técnicas que incorporam algoritmos clássicos de métodos multicritério, e usa técnicas adaptativas para que o sistema se automodifique, tanto na sua estrutura, como no seu comportamento, de forma autônoma, e gere respostas para o processo de tomada de decisão.

Para o usuário/decisor, a $\mathcal{TDAE}$ permite, por exemplo: modelar problemas de decisão semi-estruturados, para seu melhor entendimento; fornecer condições para que o decisor expresse suas preferências e seus julgamentos a respeito das informações do problema, de forma consistente; analisar os critérios impostos às regras do problema e suas combinações, além de apresentar alternativas de solução para situações não previstas anteriormente; interagir com o sistema e assim obter soluções viáveis para os problemas.

A estrutura geral da $\mathcal{TDAE}$ para aplicações de decisões multicritério é apresentada na figura 1.

<table>
<tr><td colspan="2"></td><td>colunas das ações adaptativas</td><td>colunas das regras</td></tr>
<tr><td rowspan="2">Tabela de Decisão Convencional</td><td>linhas dos critérios</td><td rowspan="4">conjunto de ações adaptativas elementares</td><td>valores dos critérios</td></tr>
<tr><td>linhas das alternativas</td><td>ações a serem aplicadas</td></tr>
<tr><td>Conjunto de Funções Auxiliares</td><td>linhas das funções auxiliares</td><td>funções auxiliares a serem chamadas</td></tr>
<tr><td>Camada Adaptativa</td><td>linhas das funções adaptativas</td><td>ações adaptativas a serem executadas</td></tr>
</table>

Figura 1 – Estrutura geral da $\mathcal{TDAE}$

A formulação da $\mathcal{TDAE}$ é dada pela tripla: $\mathcal{TDAE}$=($\mathcal{TDA}$, $\mathcal{FM}$, $\mathcal{M}$), onde $\mathcal{TDA}$ é a Tabela de Decisão Adaptativa clássica, $\mathcal{FM}$ é um conjunto de funções auxiliares e $\mathcal{M}$ é o método multicritério a ser aplicado para um particular problema de decisão, que não é destacado na estrutura, pois o método multicritério permeia e atua sobre todos os outros elementos da $\mathcal{TDAE}$.

O conjunto $\mathcal{FM}$ de funções auxiliares tem por objetivo estabelecer uma comunicação entre a camada adaptativa e o restante da tabela, e operam como uma interface, efetuando operações que seriam demasiadamente extensas se expressas na forma de funções adaptativas convencionais. Essas funções têm a capacidade de calcular e alterar valores utilizados para avaliação das condições que determinam a aplicação das regras.

Cabe observar que as funções em $\mathcal{FM}$ devem ser executadas antes da aplicação da regra que a referencia, porque o funcionamento da $\mathcal{TDAE}$ depende dos valores calculados por elas. A implementação das funções do conjunto $\mathcal{FM}$, a exemplo do que acontece com as alternativas da tabela de decisão subjacente, é definida externamente e sua operação não é explicitada na tabela. Por outro lado, a operação das chamadas dessas funções ocorre de maneira similar ao de uma chamada de função adaptativa, exceto pelo fato de que as funções em $\mathcal{FM}$ devem ser executadas em primeiro lugar.

A Tabela de Decisão Adaptativa clássica $\mathcal{TDA}$ segue o mesmo formalismo definido por [1], e é formada pela dupla $\mathcal{TDA}$ = ($\mathcal{TDN}$, $\mathcal{CA}$), onde $\mathcal{TDN}$ representa a tabela de decisão subjacente (não adaptativa) e $\mathcal{CA}$ o mecanismo adaptativo, que devem ser instanciados de forma tal que implemente de fato os procedimentos pertinentes ao método $\mathcal{M}$ escolhido de decisão multicritério.

Admitindo que o conjunto de critérios do problema de decisão seja $C = \{c_i,\ 1 \le i \le m\}$ finito, a tabela subjacente $\mathcal{TDN}$ é definida por $\mathcal{TDN}$ = ($\mathcal{CT}$, $\mathcal{R}$, $\mathcal{CV}$, t_0, $\mathcal{AT}$, $\mathcal{A}$), onde:

- $\mathcal{CT}$: conjunto de todas as configurações possíveis da tabela de decisão;
- $\mathcal{R}$: conjunto finito de regras de decisão: $\mathcal{R} = \{r_j,\ 1 \le j \le n\}$; cada regra r_j $\mathcal{R}$ é formada por $r_j = (d_{i,j}, x_{k,j})$, onde:
  $d_{i,j}$: representa um valor para o critério c_i na regra r_j;
  $x_{k,j}$: valor que sinaliza a alternativa a_k para a regra r_j;
- $\mathcal{CV}$: conjunto finito dos valores $d_{i,j}$ válidos para os critérios c_i;
- $t_0 \in \mathcal{CT}$ é a configuração inicial da tabela de decisão;
- $\mathcal{AT}$ $\mathcal{CT}$ é o subconjunto de configurações aceitas da tabela de decisão;
- $\mathcal{A}$ é o conjunto finito de alternativas do problema: $\mathcal{A} = \{a_k,\ 1 \le k \le p\}$.

Deve-se observar que uma configuração da tabela de decisão é a imagem da tabela em cada instante, na qual são mostrados os critérios, as alternativas, as regras e as combinações válidas de condições contidas no conjunto de regras. Os valores $d_{i,j}$ e $x_{k,j}$, são sempre binários, indicando critério presente ("S") ou ausente ("N"), e alternativa assinalada ("X").

A camada adaptativa $\mathcal{CA}$ é associada ao conjunto de regras $\mathcal{R}$ e na $\mathcal{TDAE}$ é definida pela dupla $\mathcal{CA}$ = ($\mathcal{FA}$, $\mathcal{RA}$):

- $\mathcal{FA}$: conjunto de funções adaptativas:
  $\mathcal{FA} = \{\mathcal{FAD}_s,\ 1 \le s \le nf\}$, no qual cada função adaptativa $\mathcal{FAD}$ quando instanciada e chamada com os respectivos argumentos, executa as ações adaptativas que a compõe;
- $\mathcal{RA}$: conjunto de regras composto pelas regras da tabela de decisão subjacente e pelas chamadas adaptativas.

Cada função adaptativa pode ser expressa pela ênupla $\mathcal{FAD}$ = (NF, P, V, G, $\mathcal{BA}$, $\mathcal{AD}$, $\mathcal{AA}$), onde cada elemento representa:

- NF: identificação da função;

- P: conjunto de *np* parâmetros P = {p_i, 1 ≤ i ≤ np};
- V: conjunto de *nv* variáveis V = {v_i, 1 ≤ i ≤ nv};
- G: conjunto de *ng* geradores G = {g_i, 1 ≤ i ≤ ng};
- $\mathcal{BA}$: indica ação adaptativa anterior (opcional); $\mathcal{BA}$ $\mathcal{AD}$;
- $\mathcal{AD}$: representa o corpo da função adaptativa e é composto por um conjunto de ações adaptativas elementares de consulta, inclusão e exclusão que modificam o conjunto corrente de regras da tabela de decisão subjacente;
- $\mathcal{AA}$: indica ação adaptativa posterior (opcional); $\mathcal{AA}$ $\mathcal{AD}$.

O corpo $\mathcal{AD}$ associado a cada função adaptativa $\mathcal{FAD}$ pode ser definido por uma lista de ações adaptativas elementares, podendo ser inclusive vazia, que indicam alterações impostas ao conjunto de regras, de acordo com o método multicritério $\mathcal{M}$ adotado.

Considerando $\mathcal{AD}$ = ($\mathcal{AC}$, $\mathcal{AE}$, $\mathcal{AI}$) tem-se:

- $\mathcal{AC}$ ações adaptativas elementares de consulta (?): examinam o conjunto de regras do dispositivo para encontrar regras que satisfaçam os critérios impostos pelo processo;
- $\mathcal{AE}$ ações adaptativas elementares de exclusão (-): eliminam do conjunto de regras alguma regra redundante ou conflitante para o processo; essas ações adaptativas levam em conta como as tabelas de decisão tradicionais trabalham, na qual regras redundantes são as que possuem critérios que são indiferentes à escolha da alternativa, e regras conflitantes são aquelas em que o mesmo conjunto de critérios levam às alternativas diferentes, o que caracteriza uma incoerência;
- $\mathcal{AI}$ ações adaptativas elementares de inclusão (+): permitem adicionar uma nova regra ao conjunto de regras da tabela.

As ações adaptativas de $\mathcal{AD}$ são associadas às regras da tabela subjacente e agem sobre o conjunto de regras $\mathcal{R}$. Essas associações formam na camada adaptativa $\mathcal{CA}$, o conjunto de regras composto por $\mathcal{RA}$ = {ra_j, 1 ≤ j ≤ n}, onde ra_j é definida por ($\mathcal{BA}$, r_j, $\mathcal{AA}$), sendo:

- $\mathcal{BA}$ $\mathcal{AD}$: se houver, indica ação adaptativa a ser executada antes de r_j;
- r_j regra da tabela de decisão subjacente associada à ação adaptativa;
- $\mathcal{AA}$ $\mathcal{AD}$: se houver, é ação adaptativa executada depois de r_j.

É possível associar a cada regra r_j até duas ações adaptativas (opcionais) para formar ra_j, uma para especificar uma ação adaptativa ($\mathcal{BA}$) a ser executada antes que r_j seja realizada, e outra ação adaptativa ($\mathcal{AA}$) para ser aplicada após a execução de r_j. No caso particular em que ra_j é idêntica a r_j, significa que a regra r_j da tabela subjacente não sofre influência de ações adaptativas.

Na $\mathcal{TDAE}$, as funções auxiliares em $\mathcal{FM}$ são executadas para determinar valores que estejam ausentes na regra procurada, como por exemplo, resultados que indiquem quais alternativas devem ser adotadas para a regra procurada. Os valores determinados, portanto, são utilizados na regra, para uma ação adaptativa de inclusão.

III. OPERAÇÃO DA $\mathcal{TDAE}$

A partir de uma configuração inicial predeterminada, a $\mathcal{TDAE}$ opera como uma tabela de decisão convencional, efetuando as buscas das regras procuradas pelo processo decisório, seguindo o método $\mathcal{M}$ definido por um especialista do assunto relacionado ao problema de decisão. O conjunto de regras inicial representa o "conhecimento" prévio contido na tabela de decisão, e é também definido pelo especialista.

Quando o algoritmo de decisão é executado, a $\mathcal{TDAE}$ deduz respostas a situações não previstas no início do processo, gerando outra configuração da tabela com novas informações adquiridas.

Durante a execução do algoritmo, os elementos da tabela de decisão inicial são analisados e julgados pelo decisor, e pelo método multicritério adotado são obtidos os pesos dos critérios e das alternativas, que refletem as preferências do decisor. Em seguida, esses pesos e essas preferências são usados em funções de utilidade associadas às regras, para gerar uma solução ou categorizar as regras iniciais. Deve-se observar que os mesmos valores também são usados pelas funções auxiliares de $\mathcal{FM}$ para definir situações ainda não previstas na tabela de decisão adaptativa.

No caso em que o decisor precisa buscar regras que contenham critérios específicos na $\mathcal{TDAE}$, é possível que uma das seguintes situações aconteça:

- uma regra idêntica é encontrada no conjunto de regras, ou seja, seus critérios são coincidentes aos procurados; neste caso, a regra é apresentada com as alternativas de decisão existentes;
- quando a regra procurada não é encontrada, uma função auxiliar de $\mathcal{FM}$ é chamada para determinar os valores ausentes na regra, seguida pela chamada de uma função adaptativa e ações adaptativas são executadas; essas ações adaptativas são responsáveis pela alteração da estrutura da tabela de decisão adaptativa, modificando sua configuração.

Em seguida, a partir da nova configuração, a $\mathcal{TDAE}$ pode ser utilizada outra vez para diferentes consultas.

A formalização da $\mathcal{TDAE}$ considera apenas casos determinísticos, pois regras redundantes ou conflitantes, como por exemplo, regras que possuem os mesmos valores para todos os critérios, mas apresentam alternativas de solução diferentes, são eliminadas pelas ações adaptativas de exclusão de regras.

IV. MÉTODOS MULTICRITÉRIO

Problemas de decisão multicritério apresentam, em geral, um conjunto de critérios, que podem ser quantitativos ou qualitativos ou ambos, associado a um conjunto de alternativas de solução possíveis, que são analisadas pelo decisor para a escolha de uma dentre elas. Para que a escolha seja feita de forma consistente, vários métodos, denominados métodos multicritério, são usados para apoiar a decisão.

A característica principal dos métodos multicritério é a que permite determinar uma relação de preferências entre as diversas alternativas de solução, uma vez que o decisor pode avaliar e demonstrar preferir uma, oferecendo, desta maneira, maior confiabilidade no seu julgamento. Outra particularidade importante desses métodos, diz respeito à flexibilidade na análise multicritério e à avaliação durante o processo decisório, reduzindo os seus aspectos subjetivos, e aumentando a eficiência da escolha.

De acordo com [5], existem grupos de pesquisa que seguem os conceitos originados na Escola Americana, e outros que seguem a linha da Escola Francesa ou Européia. Os primeiros têm como base a Teoria da Utilidade Multiatributo, cujos modelos de preferência utilizam funções de utilidade, que permitem avaliar a utilidade dos critérios, com a atribuição de pesos, em relação às alternativas do problema, o que, em consequência, resulta no valor da decisão [6]. São exemplos de métodos multicritério desta escola, AHP (*Analytic Hierarchy Process*), SMART (*Simple Multi-Attribute Rating Technique*) e MAHP (*Multiplicative* AHP).

Já as pesquisas na linha européia, utilizam funções de classificação ou categorização das alternativas (*Outranking*), que agrupam as alternativas em níveis de importância, de acordo com a avaliação das preferências do decisor. Fazem parte deste grupo, os métodos multicritério ELECTRE (*Elimination and Choice Translating Reality*), PROMETHEE (*Preference Ranking Method for Enrichment Evaluation*), MACBETH (*Measuring Attractiveness by a Categorical Based Evaluation Technique*), TOPSIS (*Technique for Order Preference by Similarity to Ideal Solution*), entre outros.

Deve-se observar que os métodos multicritério são importantes no apoio à elaboração de uma estrutura, que mostra de forma mais clara os elementos que compõem um problema de decisão, facilitando a escolha da decisão pelo decisor. Para isto, de acordo com [5], os métodos utilizam uma matriz de decisão, apresentada na figura 2:

- $C = \{c_1, c_2, ..., c_i, ..., c_m\}$ conjunto de *m* critérios;
- $\mathcal{A} = \{a_1, a_2, ..., a_j ..., a_p\}$ conjunto de *p* alternativas;
- $w = (w_1, ..., w_i, ..., w_m)^T$ vetor que contém os pesos w_i atribuídos a cada um dos critérios c_i;
- $Z = (z_{i,j})_{mxp}$ matriz de pontuações $z_{i,j}$ para descrever o desempenho da alternativa a_j em relação a cada critério c_i;
- $ax = (ax_1,..., ax_j, ..., ax_p)$ vetor dos valores globais, no qual cada ax_j é associado à correspondente alternativa a_j.

Segundo [5], para a montagem da matriz de decisão, os métodos multicritério, inicialmente, exigem que o decisor julgue cada um dos critérios c_i do problema, para a obtenção do vetor de pesos w. Os pesos refletem a importância relativa de cada critério, segundo o julgamento do decisor, e são obtidos por meio de funções de utilidade que variam de método para método.

		ax_1	.	.	ax_j	.	.	ax_p
		a_1	.	.	a_j	.	.	a_p
w_1	c_1	$z_{1,1}$	.	.	$z_{1,j}$	.	.	$z_{1,p}$
.	.	.	.	.	.	.	.	.
w_i	c_i	$z_{i,1}$	.	.	$z_{i,j}$	.	.	$z_{i,p}$
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
w_m	c_m	$z_{m,1}$	.	.	$z_{m,j}$	.	.	$z_{m,p}$

Figura 2 – Matriz de decisão [5]

De maneira análoga, são obtidas as pontuações $z_{i,j}$ que representam o desempenho da alternativa a_j em relação a cada critério c_i.

A partir da matriz de pontuações $Z = (z_{i,j})_{mxp}$ e do vetor w, cada método multicritério, de acordo com a classe a que pertencem, associará valores globais ax_j para cada uma das alternativas a_j, que são, então, avaliadas para a tomada da decisão.

Para os métodos que seguem a linha de pesquisa baseada na Teoria da Utilidade Multiatributo, as funções de utilidade agregam os pesos w_i dos critérios às pontuações $z_{i,j}$, para determinar os valores de utilidade ax_j. Nesses casos, normalmente, o maior valor de utilidade indica uma maior importância da alternativa associada [7], o que pode assinalar para a alternativa de ação, para alcançar a solução do problema.

Diferentemente, nos métodos da classe *Outranking*, os valores w_i e $z_{i,j}$ são usados para categorizar cada alternativa a_j, agrupando-as em subconjuntos de alternativas viáveis, com o objetivo de reduzir o conjunto inicial de alternativas, ou apontar possíveis ações a serem tomadas [5]. Alguns métodos de categorização levam em conta a preferência de uma alternativa a_i em relação à alternativa a_j, outros consideram o nível de concordância, ou discordância, de uma em relação à outra, tendo como base os critérios do problema.

Diversas são as pesquisas e aplicações dos métodos multicritério de apoio à decisão, o que suscitaram o desenvolvimento de implementações computacionais para apoiar e aumentar a eficiência do processo decisório, principalmente, para reduzir a subjetividade dos julgamentos do decisor.

Pode-se destacar o software *Expert Choice* desenvolvido para o método AHP, considerado um dos mais utilizados a nível internacional [8]. Também são exemplos de sistemas de apoio à decisão multicritério: *VIP Analysis* (*Variable Interdependent Parameters Analysis for Multicriteria Choice Problems*), IRIS (*Interactive Robustness Analysis and Parameters' Inference for Multicriteria Sorting Problems*), SRF, TRIMAP, LinearTri, MOMILP (*Multiple Objective Mixed Integer Linear Programming Package*), InterFractional, e outros.

Cabe observar que não há um método multicritério que seja considerado o melhor, pois cada um tem suas vantagens e desvantagens. Um estudo comparativo entre os métodos, encontrado em [8], mostra que a forma como os critérios são julgados, as funções de utilidade usadas para encontrar e tratar os pesos dos critérios e os valores globais das alternativas, podem influenciar nos resultados finais para um mesmo problema de decisão.

Em [8] são analisados e comparados os métodos multicritério AHP, ELECTRE I e MAHP quanto à sua aplicabilidade, onde são considerados, entre outros fatores, o tipo de entrada de dados que pode ser utilizado, qual saída ou decisão pode ser obtida, a existência de *software* no mercado e sua facilidade de uso. Os resultados do estudo mostram que o método AHP é o que apresenta maior número de publicações e aplicações práticas, devido à facilidade de entendimento, apesar de algumas restrições quanto ao número de julgamentos.

O método AHP (Processo de Hierarquia Analítica) foi desenvolvido por Thomas L. Saaty na década de 70, para apoiar problemas de tomada de decisão com múltiplos critérios. Sua principal característica tem como base a decomposição hierárquica do problema, criando-se uma hierarquia de critérios [9] e convertendo avaliações subjetivas de importância relativa em um conjunto de pontuações ou pesos gerais. A metodologia do modelo AHP consiste de três fases principais: estruturação do problema; julgamentos comparativos e análise das prioridades.

O método propõe ao decisor que o problema de decisão, primeiramente, seja estruturado ou decomposto em partes, representando as partes em níveis hierárquicos, para facilitar a sua compreensão e visualizar a sua estruturação através de um modelo formal. No nível mais alto da estrutura, no topo, é representado o objetivo da decisão, seguido pelos níveis de critérios e subcritérios, caso existam, e finalizando com o nível das alternativas, mostrando as relações entre os elementos. A figura 3 ilustra um problema de decisão e seus níveis hierárquicos.

A metodologia do método AHP é descrita na próxima seção junto ao algoritmo de tomada de decisão, que expressa a operação da $\mathcal{TDAE}$ e uma aplicação.

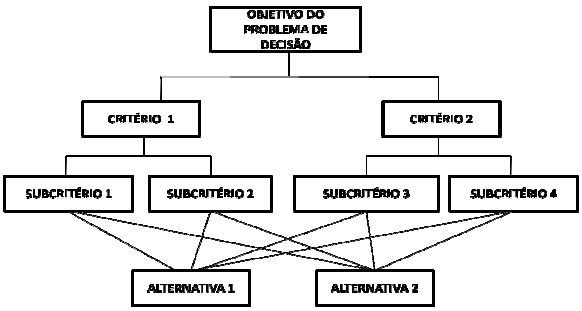


Figura 3 – Exemplo de hierarquia de um problema de decisão

V. APLICAÇÃO DA $\mathcal{TDAE}$ NA TOMADA DE DECISÃO

Primeiramente, deve-se ressaltar que, considerando o formalismo da $\mathcal{TDAE}$, a sua aplicação apoiada em um algoritmo de tomada de decisão pode ser vista sob dois aspectos:

- a $\mathcal{TDAE}$ faz o papel de decisor, onde a solução do problema de decisão é apresentada pelo próprio dispositivo, ao final do processo decisório;
- a $\mathcal{TDAE}$ pode ser utilizada como uma ferramenta para consulta de decisões parciais, deixando que a decisão final seja tomada pelo decisor ou pelo especialista do particular problema de decisão; nesse sentido, o algoritmo pode apresentar um conjunto de possíveis soluções viáveis para o problema, segundo o método multicritério adotado.

Nesta seção é descrita uma aplicação da $\mathcal{TDAE}$, na qual o método multicritério $\mathcal{M}$ adotado é o AHP, escolha motivada pelas razões apresentadas anteriormente. Além disso, o método possibilita uma modelagem hierárquica do problema, na qual os objetivos a serem alcançados, os critérios envolvidos e as alternativas de solução sejam identificados e relacionados. Também suas técnicas de julgamento e suas medidas de consistência tornam menos subjetivas as avaliações do decisor durante o processo de decisão.

A operação da $\mathcal{TDAE}$ tem como base um algoritmo composto por três módulos principais:

- Módulo I: definição da tabela de decisão subjacente, com a inserção dos critérios, das alternativas e das regras do problema de decisão;
- Módulo II: geração da matriz de decisão, cujos valores representam os pesos e as preferências relativas dos critérios e das alternativas, em decorrência dos julgamentos do decisor;
- Módulo III: elaboração da camada adaptativa e uso da $\mathcal{TDAE}$ para consultas e tomada de decisão.

Módulo I da $\mathcal{TDAE}$

Para a definição da tabela de decisão subjacente, é importante que o decisor defina e compreenda com clareza o

problema de tomada de decisão. É necessário que ele identifique os objetivos a serem atingidos, relacionando os critérios que influenciam nas diversas possibilidades ou alternativas de ação, para gerar um modelo que sintetize e represente adequadamente as informações do problema.

O modelo representado numa tabela de decisão convencional atende as características padrão da metodologia de tomada de decisão de múltiplos critérios, onde cada uma das linhas se refere a um critério, e cada uma das colunas, às regras que estabelecem as alternativas a serem executadas para as diversas combinações possíveis de valores assumidos pelos critérios.

Numa aplicação em que o problema de decisão possui *m* critérios, *p* alternativas e *n* regras, a configuração inicial t_0 da tabela de decisão tem o formato apresentado na figura 4, onde o decisor deve preencher os valores $d_{i,j}$ ("S" ou "N") de cada critério c_i a ser satisfeito, ou não, na regra r_j, bem como assinalar as alternativas $x_{k,j}$ ("X" ou deixar em branco) quando a ação deve ser executada, ou não, para cada combinação dos critérios.

	Regras					
	r_1	r_2	...	r_j	...	r_n
Critério c_1						
Critério c_2						
...						
Critério c_i				$d_{i,j}$		
...						
Critério c_m						
Alternativa a_1						
Alternativa a_2						
...						
Alternativa a_k				$x_{k,j}$		
...						
Alternativa a_p						

Figura 4 – Tabela de decisão convencional

Módulo II da *TDAE*

Como no método multicritério AHP, este módulo permite que o decisor compare os critérios aos pares, entrando com valores que reflitam o seu julgamento, mostrando qual a intensidade de importância de cada critério c_i em relação a outro c_j, de acordo com a escala fundamental de Saaty [9], mostrada na figura 5.

A escala, imposta pelo método possibilita traduzir em valores quantitativos as preferências do decisor, que normalmente são expressas de modo verbal quando usa adjetivos, e podem ser revistos quando os resultados apresentam algum conflito entre as preferências.

Escala	Significado
1	importância ou preferência igual: os critérios são igualmente importantes
3	importância ou preferência moderada por um em relação ao outro: um critério é um pouco mais importante que outro
5	importância ou preferência forte ou essencial: o julgamento favorece fortemente um critério em relação ao outro
7	importância ou preferência muito forte ou demonstrada: um critério é muito fortemente importante em relação ao outro
9	extrema importância ou preferência: um critério é extremamente preferível a outro
2, 4, 6, 8	valores usados em julgamentos intermediários

Figura 5 – Escala Fundamental de Saaty

Nessa primeira etapa do módulo é gerada uma matriz *E* de julgamentos (figura 6).

	c_1	...	c_i	c_j	...	c_m
c_1	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
c_i	...	...	$E_{i,i}$	$E_{i,j}$	...	...
c_j	...	...	$E_{j,i}$	$E_{j,j}$	...	...
...	...	...	...	...	...	...
c_m	...	...	...	...	...	...

Figura 6 – Matriz *E* de julgamentos aos pares entre os critérios

Os valores $E_{i,j}$ representam, portanto, os valores de julgamento do decisor ao comparar o critério c_i com o critério c_j, sendo o recíproco do valor $1/E_{i,j}$ estabelecido para a comparação de c_j com c_i. Na comparação do critério c_i com ele mesmo, o julgamento de valor igual a 1 deve ser atribuído.

Para a obtenção do vetor de pesos w, que indicam as importâncias relativas de cada critério, os elementos da matriz *E* são normalizados. Porém, para que os pesos sejam aceitos, [10] observa que é necessário verificar se os julgamentos são consistentes. Para isso, criou uma grandeza denominada razão de consistência (*CR*), cujo valor de aceitação é de até 0.10, ou seja, de 10%. Se o valor *CR* ultrapassar o limite de aceitação, o decisor deve rever os seus julgamentos para que o índice seja reduzido.

A razão de consistência *CR* é melhor compreendida a partir da álgebra matricial [9]. Na expressão $E \cdot w = m \cdot w$, w é definido como um autovetor de *E* e *m* como autovalor.

De acordo com [11], se λ_1, ..., λ_m, números reais ou complexos satisfazem a expressão anterior, são definidos como autovalores de *E*, e se $E_{i,i} = 1$ para todo i, a seguinte somatória é válida:

$$\sum_{i=1}^{m} \lambda_i = m$$

Logo, se $E \cdot \mathcal{W} = m \cdot \mathcal{W}$, então, todos os autovalores são zero, exceto um, que é m. No caso consistente, m será o maior autovalor de E ($\lambda_{max} = m$).

Além disso, se os elementos $E_{i,j}$ de uma matriz recíproca positiva E variarem em pequenos valores, então os autovalores também variarão por pequenas quantidades.

Portanto, a combinação dos dois resultados, e considerando que os valores da diagonal principal $E_{i,i} = 1$ da matriz E, e se E for consistente, pequenas variações de $E_{i,j}$ manterão o autovalor λ_{max} próximo de m, e os autovalores restantes próximos de zero. O pequeno desvio de λ_m a partir de m é uma medida de consistência. Denotando por CI o índice de consistência, tem-se a expressão:

$$CI = \frac{\lambda_{max} - m}{m - 1}$$

Por outro lado, [9] gerou uma tabela de índices que mede a consistência de matrizes recíprocas geradas aleatoriamente, baseadas na escala de 1 a 9, com recíprocas forçadas, para matrizes de ordem 1 até 15. Os índices encontrados na pesquisa, denominados índice de consistência randômico (*ICR*), são usados para determinar a razão de consistência *CR*, que determina a aceitação dos julgamentos do decisor:

$$CR = \frac{CI}{ICR}$$

[9] observa que para matrizes de ordem menor que 3, os julgamentos são sempre consistentes, portanto, não há necessidade de apurar *CR*.

Segundo [10], o número de critérios não deve superar a 7, por representar um bom limite prático, pelo menos em relação à consistência. [10] parte, também, do pressuposto de que para quantidades superiores, os julgamentos tendem a se tornar inconsistentes pelas dificuldades de análise do próprio decisor.

A próxima etapa do módulo II consiste na criação da matriz $\mathcal{Z}$ de desempenho de cada alternativa em relação ao conjunto de critérios (figura 7). Cada valor $z_{i,j}$ descreve a preferência da alternativa a_j em relação a cada critério c_i, de acordo com os julgamentos do decisor. O algoritmo, neste caso, também verifica a consistência dos julgamentos. No caso em que os julgamentos se mostrarem inconsistentes, o decisor deve revê-los.

	c_1	c_2	…	c_j	…	c_m
a_1						
…						
…						
a_i				$z_{i,j}$		
…						
…						
a_p						

Figura 7 – Matriz $\mathcal{Z}$ de desempenho das alternativas em relação aos critérios

A partir da matriz $\mathcal{Z}$, é possível agregar as importâncias relativas dos critérios w_i e os níveis de preferência $z_{i,j}$ de cada alternativa em relação a cada critério, para obter o valor global ax_j de cada alternativa, determinando o vetor:

$ax = \mathcal{Z} \cdot \mathcal{W}$.

Os pesos w_i dos critérios e os valores globais ax_j das alternativas podem, desta maneira, ser utilizados para determinar a melhor solução do problema, ou obter um conjunto de soluções viáveis, segundo o método multicritério adotado pelo decisor.

O algoritmo da $\mathcal{TDAE}$, porém, segue com os valores para ordenar o vetor ax dos valores globais das alternativas, ordenando-o pela alternativa de maior importância para a de menor importância. Em seguida, as regras da tabela de decisão são tratadas, isto é, são ordenadas com o objetivo de mostrar as regras que apresentem melhores possibilidades de solução, segundo os pesos dos critérios e dos valores das alternativas.

Módulo III da $\mathcal{TDAE}$

Com a tabela de decisão ordenada, o decisor pode utilizá-la para buscar regras específicas. É neste módulo do algoritmo que a $\mathcal{TDAE}$ executa os passos definidos na camada adaptativa, para regras não previstas no início do processo decisório.

Para a montagem da camada adaptativa da $\mathcal{TDAE}$, é possível implementá-la quando da escolha prévia do método multicritério, ou definida pelo decisor. Nesta etapa, são especificadas as funções adaptativas e definidas as ações adaptativas elementares de cada função.

Na aplicação particular deste trabalho, duas funções adaptativas $\mathcal{F}1$ e $\mathcal{F}2$ são declaradas:

- $\mathcal{F}1$ (p_1, p_2, p_3, ..., p_i,v_1, v_2, ..., v_j, g_1), onde:
  $\mathcal{F}1$ é a identificação da função adaptativa, composta por parâmetros, variáveis e geradores;
- definição das suas ações adaptativas elementares, na qual *regrap* representa um padrão ("*template*") de regra:
  ação adaptativa elementar de consulta: ? [*regrap*]
  ação adaptativa elementar de inclusão: + [*regrap*]
- $\mathcal{F}2$ (p_{i+1}, p_{i+2}, ..., v_{j+1}, v_{j+2}, ..., g_2):
  $\mathcal{F}2$ é a identificação da função adaptativa, seguida por parâmetros, variáveis e geradores;
- ações adaptativas elementares:
  ação adaptativa elementar de consulta: ? [*regrap*]
  ação adaptativa elementar de consulta: ? [*regrap*]
  ação adaptativa elementar de exclusão: – [*regrap*]

Cabe lembrar que as funções auxiliares $\mathcal{FM}$ são implementadas entre a tabela de decisão subjacente e a camada adaptativa, e dependem do método multicritério $\mathcal{M}$ adotado. Nesta aplicação, as funções auxiliares procuram determinar valores ausentes nas regras procuradas.

Uma vez definida a camada adaptativa, a $\mathcal{TDAE}$ pode ser utilizada pelo decisor para buscar regras específicas na tabela de decisão e obter uma resposta.

O algoritmo solicita, como entrada da busca, os parâmetros da regra que será pesquisada na $\mathcal{TDAE}$, que neste caso, correspondem à uma combinação de critérios que compõem a regra. Seja, portanto, a regra *rcc* (c_1, c_2, ..., c_m) a regra a ser consultada, onde cada c_i é um valor binário ("S" ou "N").

Uma pesquisa é inicialmente realizada na $\mathcal{TDAE}$, ou melhor, o conjunto de regras da tabela de decisão subjacente é percorrido para procurar *rcc*. No caso em que uma regra da tabela tenha os critérios idênticos à da regra *rcc*, significa que a alternativa de solução procurada está indicada na regra encontrada na tabela, e o processo continua para outras consultas.

No caso, porém, em que a pesquisa na tabela subjacente encontra a regra E (End – última regra da tabela), significa que a regra *rcc* não existe. Logo, uma função auxiliar em $\mathcal{FM}$ é chamada na regra E e executada. Nesta aplicação, a função auxiliar determina os valores ausentes na regra *rcc*, que consiste em indicar quais alternativas serão associadas à regra.

Essa função auxiliar, semelhante a uma função de utilidade, tem como fundamento os elementos existentes, tais como, os exemplos da tabela de decisão subjacente, os pesos w dos critérios, ordenados de acordo com a sua importância relativa, e os valores globais ax das alternativas, também ordenados. Portanto, a função auxiliar determina o peso da regra *rcc*, utilizando para isto, os pesos dos critérios w já obtidos pelo método AHP, nas etapas anteriores do algoritmo. Em seguida, a função auxiliar, tendo como base os valores globais ax das alternativas, indica quais alternativas a_1, a_2, ..., a_p serão atribuídas à regra *rcc*.

Quando a função auxiliar em $\mathcal{FM}$ determina o peso da regra *rcc*, é possível definir a posição de *rcc* na $\mathcal{TDAE}$, uma vez que as suas regras estão ordenadas de acordo com os seus pesos. A regra *rcc* pode ser incluída na $\mathcal{TDAE}$ por meio de uma ação adaptativa de inclusão, com a chamada da função adaptativa $\mathcal{F}1$. Essa ação adaptativa gera um número (ou nome) em g_1, que permite o acréscimo de uma coluna na $\mathcal{TDAE}$, que será ocupada por *rcc*.

Neste instante a $\mathcal{TDAE}$ se apresenta numa nova configuração, e a função adaptativa $\mathcal{F}2$ posterior é chamada, para que a ação adaptativa de consulta seja executada. Essa ação adaptativa de consulta inspeciona as regras na tabela de decisão subjacente para verificar se há regras redundantes.

Essas regras são as que têm critérios que são indiferentes, ou seja, os critérios estando presentes ("S"), ou não ("N"), levam às mesmas alternativas, portanto, não interferem nas regras. No caso em que essas regras sejam encontradas, elas são assinaladas e ações adaptativas de exclusão de $\mathcal{F}2$ excluem as regras da tabela. Em seguida, uma regra combinada é gerada por uma função auxiliar $\mathcal{FM}$, com todos os elementos iguais, a menos dos critérios que são indiferentes, para os quais são atribuídos um traço "–" (ou branco). Uma ação adaptativa de inclusão de $\mathcal{F}1$ é executada, para que essa regra combinada seja incluída na tabela. Dessa maneira, uma nova configuração da $\mathcal{TDAE}$ é apresentada.

Deve-se observar que regras combinadas, resultantes da fusão de regras redundantes, reduzem a quantidade de regras da tabela de decisão, melhorando o desempenho do algoritmo de decisão durante a busca de regras.

A partir da configuração corrente da $\mathcal{TDAE}$, novas consultas podem ser realizadas para outras regras.

Opcionalmente, a ação adaptativa de exclusão de regras pode ser usada no início do algoritmo de decisão, após a conclusão do módulo I. A execução desta ação adaptativa pode verificar a existência de regras redundantes ou conflitantes, já na configuração inicial t_0 da tabela de decisão, introduzida pelo decisor.

VI. Conclusões

Neste trabalho foi apresentada a formulação da Tabela de Decisão Adaptativa Estendida ($\mathcal{TDAE}$), que é uma extensão das tabelas de decisão adaptativas. O dispositivo $\mathcal{TDAE}$ tem como origem a tabela de decisão convencional, na qual são agregados os métodos multicritério e as técnicas adaptativas.

O trabalho descreve sua aplicabilidade na tomada de decisão, em particular em problemas de decisão que envolvem múltiplos critérios. Pelo seu caráter genérico, a $\mathcal{TDAE}$ é um dispositivo adaptativo que pode representar mais uma opção de dispositivo para apoiar processos decisórios.

O algoritmo de tomada de decisão da $\mathcal{TDAE}$, como foi visto, integra o uso de métodos multicritério e técnicas adaptativas, que tem como base o conjunto de regras iniciais da tabela de decisão subjacente, na qual, para um particular problema de decisão, o conhecimento do decisor é expresso. A partir deste conjunto de regras, para situações não previstas, no caso de regras não encontradas na tabela, a $\mathcal{TDAE}$ procura apresentar soluções para elas, a partir do conhecimento existente, da execução da lógica interna e das novas informações.

No entanto, a formulação da $\mathcal{TDAE}$ restringe as ações adaptativas para determinados problemas de decisão. Alguns métodos multicritério, tais como ELECTRE e PROMETHEE, até mesmo o AHP, preveem alterações nos conjuntos dos critérios e das alternativas, possibilitando a inclusão ou exclusão desses elementos no conjunto original do problema [12]. As ações adaptativas na $\mathcal{TDAE}$ atuam sobre o conjunto

de regras, alterando o número de regras, agindo, somente, nas colunas da tabela de decisão adaptativa. A adaptatividade da *TDAT* formulada permite, portanto, apenas operações envolvendo regras (colunas) e não considera modificações sobre as linhas da tabela de decisão subjacente, o que representa uma limitação.

Outra observação diz respeito aos métodos multicritério, que apresentam um comportamento que impõe restrições quanto à quantidade de critérios e alternativas do problema de decisão. Isto se deve ao número de comparações que é realizado entre eles, aos pares, que pode crescer muito rapidamente com o tamanho das matrizes, o que influencia na resposta do algoritmo proposto da *TDAT*, tanto em relação ao tempo, quanto ao espaço.

Um estudo de tempo e espaço do algoritmo da *TDAT* pode ser encontrado em [3]. Os resultados mostram que o algoritmo de tomada de decisão da *TDAT*, tanto no custo de consumo de memória, quanto no tempo de execução, são dependentes das entradas da tabela de decisão, ou seja, dependem da quantidade de critérios e alternativas do problema. No entanto, não foi possível relacionar os comprimentos das diversas entradas, pois elas não possuem um padrão de medida comum. Os resultados da análise do custo, também, mostram a dependência entre o número de funções auxiliares, das funções adaptativas utilizadas e da escolha do método multicritério. Além disso, a adição ou eliminação de algum critério ou alternativa pode afetar significativamente os julgamentos do decisor, e em consequência os pesos e a ordem de preferência dos mesmos.

Apesar destas restrições, a formulação da *TDAT* é multidisciplinar, por integrar fundamentos adaptativos, métodos de tomada de decisão multicritério e modelos tradicionais, tais como as tabelas de decisão, o que contribui nos estudos dessas áreas. Sua formulação mostra a consistência entre os conceitos teóricos envolvidos, ao incorporar métodos multicritério e o conceito de adaptatividade para os ajustes necessários na busca de soluções para um problema de decisão.

Para a área da Tecnologia Adaptativa, a *TDAT* representa um importante estudo do uso de técnicas adaptativas em aplicações envolvendo processos decisórios, e se mostra como uma alternativa prática de especificação e análise de problemas de decisão, cujo comportamento é dinâmico e variável.

A formulação da *TDAT* traz uma interessante contribuição, quando inclui o conceito de funções auxiliares *FM*, que operam como interface entre a implementação conceitual das tabelas de decisão adaptativas originais e os métodos de tomada de decisão. As funções auxiliares substituem trabalhosos algoritmos envolvendo elementos básicos da tabela de decisão adaptativa (regras) por algoritmos equivalentes mais eficientes. Essas funções podem ser implementadas ou simuladas usando linguagens de programação convencionais, e integradas aos mecanismos de decisão e de automodificação das tabelas de decisão adaptativas.

Desta maneira, a *TDAT* mostra que as tabelas de decisão adaptativas, originalmente definidas por [1], podem ser estendidas de várias maneiras com vantagens práticas, tornando-se viáveis para outras aplicações de problemas de decisão. Nos problemas de otimização, por exemplo, as tabelas de decisão adaptativas podem modificar a forma de resolução dos problemas, incorporando técnicas adaptativas às funções tradicionais de maximização ou minimização, e alterar, tanto a quantidade de variáveis, quanto o seu conjunto de restrições, de forma autônoma e dinâmica.

A incorporação do conceito de adaptatividade multicamada pode ampliar o uso da *TDAT*. Assim, uma ou mais camadas adaptativas poderiam ser adicionadas sobre a já existente na *TDAT*, para permitir, por exemplo, que ações adaptativas possam excluir critérios e alternativas, ou incluir novos critérios e alternativas ao conjunto inicial, agindo, portanto, sobre a quantidade de linhas da *TDAT*.

Outro aspecto que pode aumentar a capacidade de processamento do algoritmo da *TDAT* é a implementação de estruturas hierárquicas de vários níveis, que o método AHP admite. Para isto, é possível representar os diferentes níveis de decisão em tabelas de decisão individuais. Essas tabelas podem ser vinculadas e controladas por meio de uma tabela de decisão principal, na qual existiriam regras cujas condições executariam as chamadas de tabelas de decisão de níveis subsequentes. As tabelas de decisão vinculadas poderiam ser estáticas, logo, não se alterariam durante o processo, ou adaptativas, para que de forma independente se automodificassem, mudando as regras de cada uma.

A formulação da *TDAT* pode ser estendida e sua camada adaptativa modificada para suportar decisões em grupo. De acordo com [5], uma decisão em grupo envolve diversos decisores com habilidades, experiências e conhecimentos individuais, e o método multicritério deve sintetizar as decisões do grupo, levando em consideração os diferentes perfis de cada um. Os vários métodos multicritério, tais como AHP, ELECTRE e PROMETHEE, já possuem estudos estendidos para suportar decisões em grupo, e as pesquisas nessa área buscam formas de medição, que sintetizem e possam representar, o mais próximo possível, os julgamentos dos decisores. Seria interessante que estudos com uso de técnicas adaptativas pudessem, também, seguir nessa linha de pesquisa, pois parecem bastante promissoras.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] NETO, J. J., *Adaptive Rule-Driven Devices - General Formulation and Case Study*. Lecture Notes in Computer Science. Watson, B.W. and Wood, D. (Eds.): Implementation and Application of Automata 6th International Conference, CIAA 2001, Vol. 2494, Pretoria, South Africa, July 23-25, Springer-Verlag, 2001, pp. 234-250.

[2] ROY, B. Multicriteria methodology goes decision aiding. Kluwer Academic Publishers, 1996.

[3] TCHEMRA, A. H. Tabela de decisão adaptativa na tomada de decisões multicritério. Tese (Doutorado), Escola Politécnica da USP, São Paulo, 2009.
[4] O'BRIEN, J. A. Sistemas de informação e as decisões gerenciais na era da Internet. 2. ed. São Paulo: Saraiva, 2004.
[5] FÜLÖP, J. Introduction to Decision Making Methods. Laboratory of Operations Research and Decision Systems, Computer and Automation Institute. Hungarian: Academy of Sciences, 2005.
[6] GOMES, L. F. A. M.; ARAYA, M. C. G.; CARIGNANO, C. Tomada de decisões em cenários complexos. São Paulo, Pioneira Thomson Learning, 2004.
[7] KEENEY, R.; RAIFFA, H. Decisions with multiple objectives: preferences and value tradeoffs. New York: Willey, 1976.
[8] GUGLIELMETTI, F. R.; MARINS, F. A. S.; SALOMON, V. A. P. Comparação teórica entre métodos de auxílio à tomada de decisão por múltiplos critérios. XXIII ENEGEP - Ouro Preto, Minas Gerais, 2003.
[9] SAATY, T.L. Método de Análise Hierárquica. São Paulo: McGraw – Hill, Makron, 1991.
[10] SAATY, T.L.Fundamentals of decision making and priority theory with the Analytic Hierarchy Process. VI v. Pittsburgh: RWS Publications, 1994.
[11] NICHOLSON, W. K. Álgebra Linear. 2. ed. São Paulo: McGraw-Hill, 2006.
[12] DAGDEVIREN, M. Decision making in equipment selection: an integrated approach with AHP and PROMETHEE. Springer Netherlands: Journal of Intelligent Manufacturing, 19 v., n. 4, 2008.

Angela Hum Tchemra é doutora em Engenharia Elétrica, pelo Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, mestre em Administração de Empresas pela Universidade Presbiteriana Mackenzie e bacharel e licenciada em Matemática pela mesma universidade. Atualmente é professora de cursos de graduação na Universidade Presbiteriana Mackenzie, Faculdade de Engenharia São Paulo e Fundação Armando Álvares Penteado.

Mesa Redonda

***Resumo* — Esta Mesa Redonda, que encerrou os trabalhos do WTA 2010, foi presidida pelo Prof. Dr. Almir Rogério Camolesi, e contou com a participação de mais três convidados: Prof. Dr. Carlos Eduardo Cugnasca (EPUSP/PCS/LAA), Prof. Dr. Edson Satoshi Gomi (EPUSP/PCS/KNOMA) e Profa. Fabiana Santana (UFABC). O tema desta Mesa Redonda do WTA 2010 foi a discussão da utilização da tecnologia adaptativa em diversas áreas de aplicação, e da importância da pesquisa nessa área, de acordo com a visão dos componentes da Mesa.**

Transcreve-se a seguir as falas dos participantes desse trabalho. As siglas seguintes identificam as pessoas que proferiram suas manifestações nessa ocasião: JJN – João José Neto; ARC – Almir Rogério Camolesi; FS – Fabiana Santana; ESG – Edson Satoshi Gomi; CEC – Carlos Eduardo Cugnasca; AACJ – Amaury Antônio de Castro Júnior. Todas as transcrições seguem rigorosamente a seqüência em que aconteceram na ocasião.

VII. INSTALAÇÃO DA MESA REDONDA

(Prof. Almir Rogério Camolesi)

JJN - Neste evento teremos uma mesa redonda, e para coordenar este trabalho eu gostaria de chamar o Prof. Dr. Almir Rogério Camolesi, da FEMA, município de Assis, SP.

ARC - Boa tarde. Gostaria de inicialmente agradecer o convite do prof. João e da comissão organizadora para que eu coordenasse esta mesa redonda, e gostaria de convidar as demais pessoas que comporão a mesa.

- Chamo primeiramente o prof. dr. Carlos Eduardo Cugnasca, do EPUSP-PCS-LAA;
- também gostaria de chamar o prof. dr. Edson Satoshi Gomi, do EPUSP-PCS-KNOMA;
- e a profa. Dra. Fabiana Santana, da UFABC.

VIII. PRIMEIRA PARTE - APRESENTAÇÕES

Inicialmente nesta mesa eu gostaria que cada um dos membros se apresentasse, e falasse um pouco da sua área de trabalho, e também da sua experiência nessa área. Por volta de uns dez minutinhos cada um. Acho que poderemos começar pelas damas.

FS - como sempre (rss) ... Eu sou professora da UFABC, terminei o doutorado recentemente, e o fiz aqui no LAA da Poli, com o prof. Saraiva, e trabalho essencialmente com modelagem de sistemas, e por acaso, em um dos trabalhos de modelagem nós usamos a adaptatividade, e a partir daí eu comecei a me interessar pela área. Essencialmente, trabalho com modelagem para as áreas ambiental, agrícola, ecologia e problemas do gênero. Um dos grandes desafios no trabalho de modelagem é que a quantidade de técnicas que podem ser aplicadas é muito grande, e então depende de muitas situações específicas. Mas é um desafio, bastante grande e muito interessante. Em relação à adaptatividade, nós trabalhamos primeiro construindo um algoritmo adaptativo para modelagem ambiental, e começamos a usar também em outras coisas, e o interesse foi crescendo cada vez mais, como o interesse nos trabalhos que temos acompanhado aqui. Concluindo eu queria agradecer o convite para participar desta mesa, agradecer o prof. João, naturalmente, pela realização do evento, junto com a comissão, e enfim, fazer uma observação geral de que eu acompanhei os trabalhos ontem e hoje, e posso falar com certeza que em relação ao ano passado a evolução foi bastante interessante, tanto nos resultados como nas idéias, então eu acho que a tecnologia adaptativa e o WTA estão realmente dando frutos. Obrigada.

ESG - Boa tarde a todos. Eu vou focar um pouco, talvez puxando a sardinha para o meu lado, porque essencialmente a área que venho estudando inicialmente seria mais relacionada com os aspectos de aprendizagem, e como pesquisador do laboratório de engenharia do conhecimento - KNOMA - tenho ao longo desses anos estudado como é que se pode fazer para que máquinas aprendam. Ah, ultimamente tenho me dedicado também a estudar a representação do conhecimento, que é um dos meus interesses atuais. O que isso teria a ver com a questão dos chamados autômatos adaptativos? Bastante, a meu ver, na medida que, quando falamos em aprendizagem de máquina, isso essencialmente significa passar a máquina de um estado em que ela tem a habilidade e a capacidade de fazer certas coisas, e que ela passe a fazer essas mesmas coisas de maneira melhor ou aprenda a fazer coisas que ela não sabia fazer inicialmente. De certa forma, nos autômatos adaptativos, a idéia que eu tenho no momento é começar com uma descrição mais simples, e à medida que for sendo necessário, ir evoluindo isso para uma representação mais completa. Isso gera umas questões tais como a forma de controlar essas mudanças, ou adaptações. Porque na aprendizagem de máquina não existe de certa forma a preocupação, por exemplo, de se fazer eventualmente alguma coisa errada. Isso é um aspecto importante dentro do processo de aprendizagem, no qual se aprende e se tem uma certa tolerância a erro. Então neste ponto talvez na questão da adaptação talvez seja um ponto a ser considerado, talvez nas aplicações que eu tenha visto seja uma questão a se tomar um certo cuidado. Não sei como é que será a dinâmica da mesa redonda, mas só queria passar esta mensagem de que durante as discussões talvez eu possa contribuir um pouco, colocando a visão de outros formalismos que são tradicionalmente pesquisados, dentro da área de aprendizagem de máquina, sobre o formalismo mais específico aqui, que seriam os autômatos adaptativos.

CEC - Boa tarde a todos. Eu gostaria de agradecer a oportunidade de participar desta mesa redonda. Ontem já estive aqui apresentando o trabalho relativo à pesquisa de um orientado, então para aqueles que não estiveram aqui eu me apresento novamente: sou Carlos Cugnasca, professor do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP, e na área de pesquisa, estou ligado ao LAA, laboratório que originalmente tinha o nome de Automação Agrícola, mas que hoje atua na área da aplicação

da tecnologia da informação no agronegócio e ambiente. Então estamos trabalhando há uns 20 anos nessa área, inicialmente envolvendo automação, depois redes para automação, as chamadas redes de controle, depois redes e sensores com fio, sem fio, tudo isso com o objetivo claro de atender demandas nacionais de pesquisas na área do agronegócio, e também como o agronegócio está muito ligado ao ambiente, essas pesquisas extrapolaram o contexto da agricultura e do agronegócio e também passaram para a área ambiental. Desde o advento do GPS, todas essas questões passaram a ser vistas com maior precisão (talvez o termo certo fosse "acurácia") em função da possibilidade de localização precisa no campo. Então, foi a última fronteira que começou a receber os benefícios da automação, a exemplo do que já havia acontecido na área industrial, de serviços, área comercial, etc. E uma demanda que já temos percebido há algum tempo é tratar as questões associadas às demandas na área agrícola e de ambiente de uma maneira diferente. Os problemas não se apresentam sempre de uma forma muito convencional, muito clara, e a parte da adaptatividade é realmente a chave para se fazer trabalhos realmente interessantes. Ontem, na apresentação do artigo relacionado com o trabalho de um orientado meu, nós focalizamos o uso da adaptatividade aplicada a redes de sensores sem fio, que nós usamos muito em aplicações agrícolas e ambientais, mas focalizando o aspecto de otimização, no sentido de economizar energia, que é a parte crítica da aplicação dessas redes. Mas um outro aspecto de que eu não tratei na apresentação de ontem, e que é alguma coisa que está aberto em relação a como explorar, é como aplicar isso em algo que varia o tempo todo, que é o clima. Se eu conseguir transportar esse tipo de comportamento diferenciado da rede, no sentido de ela poder perceber os momentos certos de captar informação com mais granularidade, evitando tal coleta quando ela não for necessária, uma outra dimensão estará sendo aberta. Então portas estarão sendo abertas para o uso das tecnologias adaptativas, cuja aplicabilidade é muito grande nesse contexto, para se adaptar às necessidades de uso dessas tecnologias na produção vegetal, na produção animal, e no conhecimento e preservação do meio ambiente. Então nossa experiência no uso da tecnologia adaptativa diz respeito à aplicação em vários aspectos ligados a essas áreas que eu comentei.

ARC - Vou então falar agora minha experiência na área. Sou professor na FEMA - Fundação Educacional do Município de Assis, sou filho de pequeno agricultor, fico contente de ver o professor trabalhando e hoje em dia nós vemos lá no campo - meu pai já não atua mais - mas vemos a importância da tecnologia, sem a qual os agricultores estariam perdidos. Na área de tecnologia adaptativa, eu fiz meu doutorado aqui, na Poli, junto ao grupo do prof. João, terminei em 2007, trabalhei na busca de explorar a representação para um modelo geral de tecnologia adaptativa, o modelo geral apresentado no artigo de 2001 no CIAA, e meu trabalho foi na representação intermediária capaz de computacionalmente representar todo aquele modelo formal, e atualmente tenho trabalhado sobre esse modelo, aperfeiçoando-o e buscando o emprego detecnologia adaptativa na área comercial, no dia-a-dia. Estou começando um trabalho com uma pessoa lá em Assis, que pretende fazer seu doutorado nessa linha, e é nesse caminho que pretendo continuar a explorar o uso da tecnologia adaptativa. Creio que isso seja suficiente para situar o pessoal.

Então eu gostaria de verificar se há alguém que deseje fazer alguma pergunta.

IX. SEGUNDA PARTE – INTERAÇÃO COM A PLATÉIA

AACJ - Inicialmente eu gostaria de agradecer a participação dos professores membros desta mesa. Acredito, pelo que conversei com o prof. Ricardo, um dos objetivos seja explorar áreas distintas sobre as quais a tecnologia adaptativa possa ser aplicada, e no nosso caso estamos exatamente buscando desenvolver um pouco mais essas áreas, através de uma troca bastante interessante. Lá no estado de Mato Grosso do Sul nós elaboramos, em conjunto com outros professores de outras instituições, um projeto de pesquisa que envolve estudos da aplicação da tecnologia adaptativa no desenvolvimento de sistema de controle de processos robóticos. A intenção é incluisive trabalhar principalmente com futebol de robôs, por exemplo, que é um abiente bastante dinâmico, que a todo momento se modifica, e tentar explorar a parte de adaptatividade neste tipo de aplicação. Durante o desenvolvimento da minha tese eu trabalhei com o estudo de linguagens para a codificação de programas automodificáveis ou adaptativos. E a gente sabe que o desenvolvimento de sistemas de controle de processos robóticos pode ser uma aplicação muito interessante para esse tipo de linguagens. Isso envolve desde a modelagem de sistemas até aprendizagem de máquina, que o prof. Gomi colocou, e também automação, no caso os robôs; Uma aplicação interessante futura, principalmente para o nosso estado é exatamente a automação agrícola. O estado de Mato Grosso do Sul ainda é um pouco resistente a essas novas tecnologias, mas se alguns bons resultados puderem ser conseguidos como resultado desses trabalhos, acredito que de repente dê para aplicar um pouco melhor dentro do próprio estado. A minha grande pergunta é: como nós conhecemos bastante, ou pelo menos um pouquinho mais a tecnologia adaptativa, pelo nosso histórico, eu gostaria de saber dos especialistas de cada uma das áreas que representam, quais os principais aspectos que poderiam ser explorados durante o desenvolvimento desse trabalho, em que vocês acreditam que a tecnologia adaptativa possa contribuir para direcionar um pouco nossos passos nesse estudo desenvolvimento dessa proposta e nessa pesquisa. Então eu gostaria de saber o que cada um pensa que sejam os pontos das respectivas áreas que poderiam ser explorados para que a gente possa aplicar nesse tema específico a tecnologia adaptativa.

ESG - (acho que não temos uma ordem específica...) A colocação foi no sentido de tentar apresentar idéias diferentes, de como se poderiam expandir as aplicações em formas diferentes de se usar a tecnologia adaptativa. Vou dar uma visão (bem ou mal, eu me considero um leigo no assunto, conheço o prof. João há muitos anos, mas só mais recentemente acabei tendo de fato a curiosidade e este ano em particular pelo fato de ter sido convidado para participar do processo de avaliação de trabalhos submetidos, me vi de certa forma com a necessidade de estudar o assunto mais seriamente, então fiz a metade da lição de casa: ao menos

estudei a teoria, recuperei até mesmo os textos primordiais do prof. João, descobri que essas coisas remontam até a muito mais tempo do que a gente imagina, e isso ajuda a gente a entender um pouco da teoria.) Na essência, considerando-se que fortemente baseados nesses conceitos já consagrados das linguagens formais, teoria dos autômatos, e vendo os trabalhos que pude avaliar, senti a necessidade de se ter um pouco mais de uniformização de uma linguagem em que se possa desenvolver aplicações usando a tecnologia adaptativa. Vendo trabalhos diferentes, que têm um começo digamos muito nos princípios e nos fundamentos. Só que, falando em engenharia, é claro que todos os fundamentos teóricos são muito importantes, mas o que a engenharia faz de bom é pegar aquilo que digamos é a teoria, que no final a gente acaba não tendo que se preocupar no dia a dia, porque os fundamtentos conceituais nos dizem: o que é, quais são os elementos construtivos, desse conceito, o que dá para fazer, o que não dá para fazer, quais são os limites, quais são os critérios que façam ter um desempenho melhor ou pior, mas a partir daí, é preciso criar ferramentas, mais práticas no sentido de acelerar o desenvolvimento das aplicações. Ou seja, estou tentando dizer algo equivalente a toda a tecnologia que foi desenvolvida ao longo do tempo para programação. Se estivéssemos ainda naquela época em que a gente tinha que programar em linguagem de máquina, certamente seria muito mais complicado produzir os programas, os softwares com a complexidade que nós temos hoje. Dá para fazer isso graças ao desenvolvimento de ferramentas, linguagens que permitem as pessoas que, digamos, não estão preocupadas com os fundamentos teóricos, mas estão preocupadas com o lado da aplicação, possam trabalhar e desenvolver sistemas baseados nesse conceito. Talvez este seja um ponto que possa ser investigado, tentar criar frameworks, ambientes que possam facilitar o desenvolvimento de aplicações baseadas nos autômatos adaptativos. Seria esta talvez uma idéia.

Uma outra seria desenvolver alguma coisa, algum sistema que permita fazer simulações sobre a tecnologia adaptativa. Quando falamos de coisas práticas, essas questões de desempenho, problemas na especificação e outros que são comuns a qualquer sistema que a gente vá projetar, certamente começa a aparecer aqui, porque a tecnologia adaptativa por si só não é imune a todos esses problemas que não são problemas do conceito mas que são problemas inerentes a qualquer projeto. E hoje a engenharia tem à sua disposição ferramentas que permitem fazer essas simulações antes que a gente vá para uma implementação mais prática propriamente dita.

Bem, isto que eu coloquei na parte inicial de minha fala, talvez uma curiosidade que eu tenho seja na medida que tiver algum sistema baseado em tecnologia adaptativa, quais são digamos os graus de liberdade que eu posso ter nessa adaptação, ou seja, a adaptação por si só que problemas poderia trazer no desempenho da execução da tarefa propriamente dita para o qual o sistema foi construído? Eu acho que a adaptatividade é essencial para um grande problema que temos hoje, que é a barreira da complexidade. Isso significa que se eu quiser hoje criar um sistema que é duas vezes mais complexo, seja em instruções, seja em memória, seja em processar mais rápido, os métodos de design não dão mais qualidade boa, ou seja, duas vezes mais complexo não significa que eu vou gastar duas vezes mais tempo, duas vezes mais recursos, mas quatro ou cinco vezes mais. O aumento não é mais polinomial, mas exponencial no esforço necessário para produzir alguma coisa duas vezes mais complexa. Então, como é que se ultrapassa essa barreira? Certamente, se tivermos que ficar programando é que não vai se ultrapassar isso. Uma solução seria permitir que as máquinas se adaptem, ou aprendam. Em lugar de resolver o problema, eu ensino a máquina a resolver o problema. É uma forma então de ultrapassar a barreira da complexidade ou escalabilidade no design de sistemas. Só que esta estratégia traz seus problemas: não se sabe exatamente o que a máquina vai fazer no final das contas. Quando eu programo, a máquina vai fazer exatamente aquilo para o qual ela foi programada, tirando-se, é claro, os bugs de programação e considerando que tudo esteja correto, ela vai fazer exatamente aquilo que foi pensado inicialmente. No campo da aprendizagem de máquina, a gente não tem uma garantia do que vai acontecer no final. E a máquina ganha mais graus de liberdade. Isso é bom na medida que eu vou quebrar essa barreira da complexidade. Mas torna complicada a questão de se ter o chamado determinismo. Começamos a ter coisas que não são mais tão determinísticos assim. O problema, mais que a resposta, é mais uma questão que eu apresentaria para as pessoas que estiverem querendo pesquisar esses temas.

FS - Uma idéia, seguindo um pouco a linha que o prof. Gomi falou no início: se dentro do seu projeto você está pensando em uma linguagem de programação automodificáveis e vocês querem ter um resultado prático, computador, eu acredito que hoje de manhã foi apresentado um trabalho que a gente discutiu um pouquinho eu e o prof João a gente estava ali em cima, que na verdade uma linha que talvez seja mais interessante para se buscar essas linguagens é você fazer como fizeram os pesquisadores aqui da inteligência artificial, que a gente tem o prolog que resolve só um pequeno número de problemas, mas o prolog é implementável, e os problemas que o prolog resolve, ele resolve mesmo. ou quase sempre. Foi feita aqui uma versão simplificada dos autômatos adaptativos, então de repente seguir esta definição ou alguma outra definição, mas tentar fazer uma especificação bem simplificada, mesmo que você tenha um qua a linguagem ou a classe de linguagem que você tenha aí seja menor, mas que você consiga implementar e ter resultados. Acho que isso é bem mais prático do que ... você não precisa chegar na máquina de turing, você pode fazer como o prolog: você tem que reduzir tudo à primeira ordem, um pouquinho menos ainda, mas tem uma ferramenta, e aí tudo começa a acontecer. Então aí nessa linha de linguagens a minha recomendação seria tentar chegar perto usando uma definição simples para linguagem, ou autômatos adaptativos (depende da abordagem que você quer dar ao problema) mesmo que ele não consiga resolver todos os problemas. Tem-se que ser bem realista.

Sobre os outros aspectos, sobre agronegócio e ambiente, aí "puxando um pouco a sardinha" para o nosso lado, sempre a gente tem muitos trabalhos em modelagem ambiental e em modelagem agrícola. E a gente ainda usa pouco a adaptatividade perto do que poderia usar, então emcompensasção a gente tem várias outras soluções para vários outros problemas. então valeria a pena conversar com

um pouco mais de calma sobre os problemas que vocês têm lá, e de repente trazer uns para cá, e fazer o que o prof Carlos fez: vocês têm um problema envolvendo sensores no campo, então trazemos para cá e usamos a adaptatividade para resolver. Então tem que analisar um pouquinho mais, e certamente tem uma gama enorme de coisas para explorar. nem daria para falar agora tamanha a quantidade de problemas. Agora, quanto à linguagem, eu iria pela linha do prof. Gomi, começar pequeno, fazer um compilador, até chegar a um resultado. é uma sugestão.

CEC - Vou começar a falar pegando uma abordagem que o prof. Amaury colocou, que Mato Grosso do Sul ainda tem uma barreira para o uso de novas tecnologias no campo, etc. A tecnologia da informação aplicada em áreas como o agronegócio pode ser vista de várias maneiras. Se formos olhar segundo a visão da computação ubíqua ou pervasiva (como queiram) um desafio quetodos nós temos é tornar o computador invisível ao usuário, porém prestando relevantes serviços, agindo e reagindo aos estímulos que ele recebe, procurando atender as necessidades do usuário, mas sem que este usuário deva estar obrigatoriamente envolvido excessivamente com tecnologia, sem que ele se estresse com essa tecnologia. Na área urbana, nas cidades em que vivemos a maioria das pessoas está caminhando para morar em cidades, é normal que as pessoas estejam cada vez mais familiarizadas com os avanços tecnológicos. No campo isto tem mudado, mas ainda existe um a barreira a grandes mudanças, e as coisas caminham mais devagar no campo. E muitas iniciativas náo progridem, ou têm dificuldade em progredir exatamente porque esquecem aquele que vai exatamente utilizar a tecnologia. A tecnologia não foi projetada para ele, e sim para um especialista, e isso cria uma dificuldade para a sua adoção. A tecnologia adaptativa tem um potencial grande nessa área no sentido de tornar a tecnologia mais simples de ser usada, o mais plug-and-play possível. Ontem na apresentação que citei, "redes de sensores sem fio", já existe um produto comercial, que está sendo lançado exatamente para a agricultura, mas o interessante dele é que praticamente o usuário pouco tem que saber de tecnologia para usar: é ligar, escolher os locais onde ele quer colocar os sensores, e depois, ir buscar os dados e analisá-los. É mais nesse sentido que tudo tem que caminhar: o usuário não tem que ser um especialista. O sistema tem que se adaptar àquilo de que o usuário necessita: sua vontade, sua forma de usar, sua cultura, e tudo o mais. Muito das soluções importadas nessa área não progridem em outros países exatamente porque esquecem que a parte cultural é diferente de um país para outro. Então éuma área que permite muitas soluções próprias, não só pela regionalidade da agricultura, mas pelas especificidades de quem vai usá-la. Então nesse aspecto eu creio que a tecnologia adaptativa pode contribuir. Os sistemas estarem se preparando para se adaptar às necessidades de cada usuário. Eu citei em minha primeira fala a parte de uma amostragem diferenciada, que a gente poderia chamar de amostragem inteligente. Os sistemas atuais eles favorecem a gente a obter montanhas de dados, muito dado mas não necessariamente muita informação: dados redundantes, dados desnecessários, que mais atrapalham que ajudam. Um sistema que percebe quais os momentos que devem ser relevantes para a busca e armazenamento da informação, esses sistemas sem dúvidas vão ter mais sucesso em relação aos usuários; Ao longo de um dia a maior parte do tempo (à exceção de São Paulo, onde temos as quatro estações no mesmo dia), em geral um lugar estável, de agricultura, varia pouco o clima ao longo do dia. Não faz sentido os sistemas como muitos que existem atualmente, guardarem quase que infinitos dados daquele dia, que não correspondem a qualquer conteúdo significativo. Essa é uma parte simples onde, a meu ver, não seria difícil usar um sistema adaptativo que possa perceber quando alguma coisa é relevante, e fazer uma armazenagem diferenciada, não só reduzindo a massa de dados, mas tornando aquilo que for armazenado mais importante.

A forma de os usuários utilizarem as máquinas também é algo que é interessante. Eu já tive a oportunidade de dirigir algumas máquinas agrícolas, e nessas grandes feiras que acontecem pelo Brasil, e em particular no Agri-Show em Ribeirão Preto. Uma máquina moderna mais parece a cabine de um avião do que com um trator. É um equipamento altamente sofisticado, e nem mais tem a direção, é um joy-stick, e eu pilotei aquilo sem nunca ter entrado numa máquina dessas, e fiz cavalo de pau com muita simplicidade nele, porque afinal ele gira as quatro rodas separadamente, então dá para brincar de pião. Mas eu me surpreendi com a extrema simplicidade de uso daquele equipamento. Era muito fácil o usuário se adaptar a ele, embora seja quase tão complexo como a cabine de um avião pela quantidade de tecnologia lá presente. Tambéé uma iniciativa que eu vejo pelo que tem acontecido – eu participo de uma comissão de normatização da ABNT no sentido de padronizar as redes para máquinas agrícolas – é exatamente no sentido de simplificar e tornar o trator um equipamento plug-and-play, ou seja, compra o trator de um fabricante, coloca o implemento do outro, liga e tem um único computador de bordo, e ele se adapta e descobre o que tem de fazer. Então, isso já é realidade, na comunidade européia, e dificilmente alguém vai conseguir exportar máquinas para lá no futuro se não tiver esse tipo de tecnologia. os sistemas se adaptarem para que haja interoperabilidade. Esse é um grande desafio para um país como o Brasil, onde o agronegócio é forte, a fabricação de máquinas é também importante para nós e enquanto não tivermos essa tecnologia nós ficaremos excluídos desse tipo de mercado. Então os equipamentos que são muito flexíveis poderem se adaptar a cada uso, que varia ao longo da safra, é algo que tem um potencial que está começando a ser explorado. Resumindo: tornar a tecnologia de uso mais simples, invisível ao usuário, no caso do agronegócio é chave para o sucesso de novas iniciativas, e essas barreiras irem sendo suplantadas, ou seja, o usuário não percebe em muito detalhe qual é a complexidade da tecnologia que ele está usando, mas ele consegue perceber os benefícios dela.

ARC - Aproveitando também, Amaury, queria acrescentar algumas coisas: trabalhar com tecnologia adaptativa e observar assim no dia-a-dia é a falta também de ter uma linguagem de programação - o salvador está trabalhando nela - como modelar nossas aplicações, como gerar o modelo dessas aplicações para tecnologia adaptativa. uma das coisa que nós facilitaríamos também para o público que vai trabalhar com tecno logia adaptativa -- porque temos paradigma orientado a

objetos, o paradigma procedural, o paradigma funcional, Já venho há tempo conversando com o prof. João,e com você também, e acho que o paradigma adaptativo e alguns conceitos -- hoje o trabalho da manhã que nós vimos essa profa. falou dos autômatos e depois a busca de características em que o adaptativo pode ser empregado eu vejo que isso pode facilitar o pessoal das outras áreas e nos projetos em que eles estão sendo empregados, e a partir daí ter algo que facilitasse a modelagem das aplicações. Às vezes a pessoa pegar algo orientado a objetos e ele está pensando em um modelo orientado a objetos mas a funcionalidade e a implementação dele vai ser feito em adaptativo. Então eu vejo que são algums áreas onde podemos investir e isto estou falando para a comunidade toda, e acho que poderia facilitar. não sei se vocês gostariam de comentar alguma coisa em relação a isso, porque tem esta falta desse tipo de ação relacionada à modelagem e mesmo quanto a onde empregar o adaptativo. Vocês querem acrescentar alguma coisa em relação a isso?

CEC - Quando começamos a considerar o uso dessa tecnologia em nossas aplicações essa foi uma dificuldade que nós enfrentamos: existe uma forma convencional de se tratar os problemas e prepará-los para receber as soluções convencionais, e no caso da tecnologia adaptativa nós não tínhamos uma referência, uma forma mais fácil de começar a introduzir essas téncicas em nossas aplicações, então nós recorremos àqueles que já tinham experiência, que puderam nos dar com essa experiência a suas primeiras ajudas, para conseguir progredir nessa área, isso certamente vai ajudar bastante, novos trabalhos no sentido de nos orientar como modelar nossos problemas para que possam ser tratados usando essa tecnologia.

FS - Uma espécie de UML para adaptatividade, não é isso? Eu acredito que existem aí diversos dispositivos adaptativos. Há um trabalho do prof. João de 2001 que tem uma lista mais ou menos grande deles, mas na hora que você realmente vai usar fica mais ou menos complexo, então acho que vale a pena investir um pouquinho um nível acima, antes mesmo da programação. É claro, já é bastante trabalho pensar em algo implementável, e montar uma linguagem, mas pensar como vai modelar o que vocês conseguirem abordar no projeto, ou fora dele, eu acho que vale a pena, sim.

ESG - Já que estamos nessa seara de linguagem de programação, frameworks de programação, em termos práticos, o que talvez a história nos conta é que aquilo que acaba vingando é aquilo que acaba sendo usado por várias pessoas em várias áreas diferentes. Aproveitar que foi mencionado aqui a linguagem prolog, o prolog tem uma característica no pcs que talvez seja um dos poucos lugares que tem alta concentração per capita de professores que conseguem programar em prolog, mas apesar de ser infame, eu não diria que prolog seja uma linguagem que esteja disseminada, e de fato não vai ser, mas os conceitos que exitem por trás estão de fato incorporados em muitos sistemas. Dizer que prolog é a melhor linguagem de programação do mundo portanto podemos largar java e vamos programar tudo em prolog. Acho que não é por aí. Talvez uma solução boa seja enfocar a criação de compiladores ou bibliotecas que contenham dentro de si a tecnologia adaptativa, e que sejam plugins que possam ser incorporados em linguagens que estejam de fato sendo utilizadas. E com isso a gente ganha então mais chances ou oportunidades de a tecnologia se disseminar, de modo que possa passar a ser utilizada por mais pessoas. Porque em termos estratégicos, se você imaginar as empresas, ou mesmo aqui na academia, a gente faz escolhas em termos de investimento: o que é que eu vou estudar? que linguagem eu vou estudar? Não vou investir em uma coisa que não tenha utilidade ou que não tenha um a sobrevida razoável. Nessa altura do campeonato, Fortran foi uma linguagem que ... Foi, e ainda tem sido bastante utilizada mais em engenharia, mas aqui na poli não se ensina mais Fortran. Talvez o motivo seja este: outras estejam sendo mais utilizadas, então nesta hora só vale a pena o investimento naquilo que de fato vá ser utilizado, aquilo que de fato tem chance de ser aplicado em lugares diferentes, então talvez a estratégia seja incorporar a tecnologia adaptativa em sistemas ou linguagens em que de fato estejamos utilizando seja de fato mais eficaz.

FS - Na verdade, é assim: mencionei o prolog não no sentido de usar o prolog mas no sentido de você buscar uma linguagem que resolve, mesmo que não resolva tantos problemas como uma linguagem tradicional, pelo menos ela seja implementável. Então nesse sentido mais naturalmente você conseguir fazer uma biblioteca em java seja muito mais aproveitável. Mas mesmo que você não resolva todos os problemas acho que vale a pena investir em algo que você consegue de fato implementar. Este ponto é importante.

X. TERCEIRA PARTE – INTERAÇÃO COM A PLATÉIA

ARC - Continuando, gostaria de saber se os membros da mesa gostariam de fazer alguma pergunta em diversas áreas, aproveitando a heterogeneidade? A platéia tem perguntas?

ARC - Bom, eu tenho aqui uma das perguntas, então nós falamos aí das áreas de trabalho, e dessa experiência em tecnologia adaptativa. E vocês têm experiência mais diretamente em trabalhos específicos: a professora falou, e o prof. Gomi comentou alguns trabalhos. Gostaria de perguntar em que área especificamente cada um está atuando nesses trabalhos.

ESG - Essencialmente, dando um caso prático: dada a descrição de um sistema, em uma linguagem mais abstrata, UML, por exemplo. Como se poderia fazer uma transformação para algo que seja um pouco mais "executável"? O pessoal pode talvez imaginar que seja uma especie de programação automática, mas não é bem assim, porque teríamos alguma coisa descrita em UML. No meu entender, se estamos fazendo uma descrição conceitualmente correta, em cada nível de abstração, o primeiro diagrama UML que podemos gerar seria alguma coisa em nível de negócio. São as aplicações tradicionais que as pessoas têm em engenharia de software. No meu tempo, era video-locadora, agora acho que é controle de estoque, agora deve ser locadora de dvd, ou blu-ray, né? mas nesse modelo quais são os objetos, os termos que devem aparecer? são os termos de

negócio: dvd, blu-ray, etc. São essas as coisas que devem aparecer no modelo. Não se fala em bits, não se fala em bytes, não se fala em pilha, nada disso. Os objetos têm de ser objetos do mundo dos negócios. Bom, aí a gente fica imaginando: como é que a partir daí eu posso converter isso em coisas que vão ser executadas? hoje como é que é feito? Eu modelo, e aí alguém vai programar. Uma parte vai ser feita em java, outra em banco de dados, que fica mais em sql, e assim por diante. A partir do diagrama, tenho transformações que me dizem exatamente como eu devo fazer isso. Olhando assim, a pergunta é: com é que uma máquina poderia fazer isso? Se tivermos um mapeamento dos elementos do meu modelo do mundo dos negócios nos elementos em que eu devo transformá-los, no mundo da computação, o problema está resolvido. O problema é que este mapeamento se ele existir ele é imenso. E provavelmente a quantidade de mapeamentos possíveis será infinito. A dificuldade está aí. Outra dificuldade: os elementos do meu modelamento do mundo dos negócios não contém todas as informações daquilo que eu preciso saber para mapeá-lo completamente para o mundo da computação. Existem informações sobre a arquitetura, informações sobre onde isso irá ser executado, informações essas que não aparecem quando eu faço o modelo no nível da aplicação, e que precisam ser inseridas pelos desenvolvedores ao longo do processo de implementação. Então qualquer máquina que tenha como propósito fazer essa transformação terá de possuir a mesma capacidade de inserir essas informações adicionais. Mas se quisermos criar uma máquina dessas para realizar essa operação, vamos esbarrar nessa barreira da complexidade, portanto precisamos dar inteligência e capacidade de adaptação e de aprendizagem de máquina para que ela consiga fazer, a partir de conhecimentos abstratos sobre arquitetura, sobre linguagem, sobre os elementos computacionais de maneira genérica, e começar ela a montar esses bloquinhos e começar a fazer esssa transformação. Então as pesquisas mais recentes seria simulador de diagramas UML em nível de negócios. No fundo é o mesmo problema, só que em vez de deixar a máquina propor a arquitetura alvo, nós fixamos a arquitetura, e assim é possível fazer alguns mapeamentos internamente, e a gente consegue de certa forma sem ter que programar ou desenvolver a partir de um diagrama UML fazer a execução desse diagrama, ao menos para a gente ver que em termos de semântica, aquilo que está descrito no modelo corresponde àquilo que queremos. Aliás esta é uma outra dificuldade que podemos encontrar nas linguagens de programação: é preciso criar linguagens, sejam textuais ou gráficas, que sejam fáceis para a pessoa da área de aplicação utilizar, talvez como essa é a área agrícola, talvez nós vamos ter pessoas que vão entender de agricultura, de máquinas do campo, e não serão provavelmente as pessoas mais versadas em linguagens de programação, e aí talvez venha a utilidade de linguagens gráficas na área industrial, isso já é usado ha muito tempo: quando você tem a representação de circuitos de chaveamento, em vez de ficar representando-as em forma de expressões lógicas o que você faz é utilizar componentes gráficos sobre os quais você faz ligações de forma mais intuitiva. Aí aparece uma questão interessante: será possível criar uma linguagem genérica que atenda todo mundo? Provavelmente não, pois cada área de aplicação vai apresentar uma linguagem mais apropriada para as pessoas dessa área. Essa parte da linguagem e interface. Toda a parte ligada à compilação de tudo isso poderia ser reaproveitada. Talvez juntando várias idéias talvez seja possível determinar elementos comuns a várias aplicações e outras, que seriam específicas para cada área de aplicação.

FS - Bem, eu comentei rapidamente sobre os trabalhos de modelagem - fazer trabalhos de modelagem em biodiversidade é bem mais complicado do que inicialmente parece. Então dentro dos projetos que estamos desenvolvendo hoje em dia, temos sempre buscado integração e interoperabilidade e uso de padrões, porque você precisa de dados, e estes dados podem provir de um portal, cada um de uma origem diferente, e sempre que não se tem padrões de interoperabilidade a integração fica comprometida. Então em termode sistema, a gente sempre procura trabalhar com arquitetura orientada a serviços. Uma outra área de pesquisa, ainda nessa linha que nós precisamos, é são os serviços GIS de informação de gráficos, que fornecem tanto dados de entrada como são resultados de modelos: um mapa agrícola, um mapa de distribuição de uma espécie no campo, etc. Para tudo isso a gente consegue trabalhar usando basicamente tecnologia de serviços. Então a gente procura sempre fazer integração usando SOA - service-oriented architecture, e a gente tem conseguido bons resultados. Mas isso em termos de sistemas. Quando a gente constrói uma plataforma, com ela conseguimos fazer modelagem. Aí a gente cai no problema de fazer a modelagem em si. O que seria então fazer um modelo de biodiversidade, um modelo agrícola? Aqui no Brasil, a respeito de algumas espécies, dispõe-se de pouquíssimas informações. Então às vezes informações sobre a ocorrência da espécie, a partir de uns 20 lugares no Brasil inteiro (20 coordenadas geográficas x-y) eu tenho que inferir em quais lugares do mundo eu posso achar essa espécie. Então são algoritmos assim... a geração dessses modelos é bastante complicada. Então existem desafios de vários lados.

Um deles é a obtenção de dados. Por isso a gente tende sempre a obter as coisas de forma integral, com interoperabilidade.

O segundo é a correção dos dados. Às vezes você pega uma máquina que passou do ponto, e essa máquina obteve dados com seus sensores, e nos tratores também, que colhem informações. Há informações que vêm até de fora da área de tratamento. Torna-se preciso fazer uma limpeza nesses dados, e tudo isso cai na área de algoritmos. Dentre os maiores desafios, além da geração dos modelos em si, o que eu considero mais crítico hoje em dia é a questão da avaliação do modelo. O que eu posso tirar: então eu tenho um algoritmo aqui, e ele é super bom, e eu consigo prever qual vai ser a produtividade da minha colheita no ano que vem, por exemplo, e se eu gerar um modelo para isso,... como vou saber se esse meu modelo é confiável? esta é uma área de pesquisa que ainda é muito forte. Então, dentro desses desafios, que eu chamo de problemas principais da modelagem, a gente acaba usando diversas tecnologias. Temos algoritmos baseados em técnicas de entropia máxima, algoritmo de busca de padrão, fizemos até alguns estudos em ontologias, algoritmos estatísticos, e m´wtods de tentativa e erro. Fazendo um parênteses, quando você chega para um pesquisador, na área ambiental em geral, o que vai acontecer se a terra aumentar a

temperatura em cinco graus? aí tdo mundo coloca os pés para trás: eu não sei. Por quê? Porque o sistema é complexo demais para se modelar, e mais ainda, uma vez modelado, para saber quanto ele stá correto, qual seu grau de acerto, etc. tão bem quanto estas tecnologias, estão dando bons resultados os algoritmos adaptativos. porque por exemplo, um dos algoritmos mais utilizados para o estudo de distribuições de espécies é o GARP, o qual a gente tornou adaptativo, é um algoritmo genético que processa regras, e ele vai fazendo inferências. Com a adaptatividade nós conseguimos introduzir no algoritmo o conhecimento do pesquisador, então mesmo que o algoritmo genético apresente o pesquisador sabe que uma determinada espécie só vai sobreviver de 12 a 25 graus, de temperatura, talvez o algoritmo genético queira baixar a temperatura, e o pesquisador já sabe que a espécie não vai sobreviver, então na hora que você mistura a tecnologia adaptativa com esses outros tipos de algoritmos, você consegue resultados muito bons, por evitar os desvios, então vai adaptando o que você tem em direção a algum resultado mais preciso. Especificamente além da régua-papel=tesoura a teoria dos jogos é outra área de modelagem mas especificamente nós trabalhamos esse tipo de aplicação da adaptatividade na modelagem ambiental. Aí a gente consegue inserir conhecimento do pesquisador e este conhecimento direciona o funcionamento do algoritmo genético. Temos então aí os resultados práticos, que comprovam que a adaptatividade realmente funciona, e permite obter resultados melhores que as abordagens convencionais.

CEC - Dentro deste contexto de modelagem, vou destacar um problema que a gente tem enfrentado, e que estamos tentando suplantar, e que tratamos rapidamente no trabalho apresentado ontem. No nosso contexto, a solução final, qualquer que seja ela, usando adaptatividade ou não, tem que ser implementada num ambiente limitado. O contexto de computação pervasiva, redes sem fio, a idéia é sempre embutir em pequenos dispositivos que possam ser usados em grande amplitude, maciçamente nos ambiente, embutir o máximo possível de capacidade de ele interagir com o ambiente, reagir aos estímulos e processá-los adequadamente, e cumprir seu papel da melhor forma possível, mas esses dispositivos, para serem pequenos, consumir pouco e não ter custo alto, eles têm e apresentam limitações. Então o desafio que temos enfrentado é conseguir soluções ao mesmo tempo eficazes e de implementação não complexa, então isso tem sido um desafio, se não o benefício que teríamos por usar a tecnologia adaptativa seria anulado por outros problemas: lentidão do processamento, ou então estou tentando otimizar consumo de um lado e perco de outro, pelo excessivo processamento, ou então até inviabiliza a aplicação, ela não consegue responder aos estímulos na velocidade pretendida. nós temos em algumas situações a condição de simular, e até certo ponto, sem a necessidade de uma implementação em ambiente real, que é a simulação para comprovar, avaliar se a solução que está sendo dada tem ou não condições de atender as necessidades, de atender as especificações iniciais, e de trazer benefícios, e ela também já dá uma idéia de qual vai ser a complexidade das sua implementação. A partir daí já temos outros desafios: de conseguir simplificar soluções, otimizá-las no sentido de ela poder ser implementada em pequenos ambientes.

ARC - Perguntas?

XI. QUARTA PARTE – FINALIZAÇÃO DA MESA REDONDA

ARC - Não havendo perguntas, podemos encerrar, então. Eu tenho o costume de, quando organizo mesas redondas, finalizar com os membros fazendo uma apresentação final, e falando, por exemplo: tecnologia adaptativa - a que é que ela se resume na sua vida, ou na sua área de pesquisa?

Na minha área de atuação, eu vou eu começar, até dando a idéia, na forma de uma brincadeira, como uma forma para instigar um pouco as áreas - desculpem-me por tomar a liberdade de fazer isso, mas eu acho que é importante - pára mim, a tecnologia adaptativa é importante, e se resume a uma palavra: "início": início de uma amizade com o prof. João, início de minha carreira de pesquisa, e a tecnologia adaptativa também está no início. Início de muitas oportunidades de as pessoas desenvolverem pesquisas nas mais diversas áreas, como estamos vendo.

Então, se vocês quiserem falar e fazer suas considerações finais, acrescentando uma palavra ou uma frase, seria interessante.

FS - Bom, "início" mais ou menos, né? Está quase no meio, vai? (rss) Eu diria o seguinte: em termos de algoritmos, e eu mexo bastante com algoritmos e técnicas algorítmicas, eu diria que a tecnologia adaptativa é realmente uma possibilidade nova. Acho que as últimas evoluções, e o prof. Edson pode me corrigir, mas foram talvez nas lógicas - lógica fuzzy, redes neurais, etc. - em termo de tecnologias novas, a construção de algoritmos para a solução de problemas e a tecnologia adapatativa nesse sentido tem muito a contribuir. Eu não sei o quanto ela está no início ou não, mas eu acho que é preciso investir mais também no formalismo mas realmente pode ter uns efeitos colaterais indesejáveis, como o prof. também falou, que de repente você não sabe o que vai acontecer, ao contrário de um programa que você faz hoje, à exceção do bug você sabe o que ele faz, mas eu acredito que, em termos de solução do problema é uma tecnologia que não deve ser descartada, ela deve ser investigada e considerada sempre, principalmente no caso de problemas complexos. Para terminar, gostaria então de parabenizar o prof. João pelo trabalho todo de carregar essa tecnologia adaptativa nas costas por alguns anos.

ESG - Nesta fala de encerramento da mesa, acho que toda área de pesquisa nasce de uma idéia, aí as pessoas olham e falam: essa idéia é bacana, essa idéia é legal, essa idéia é útil. E se de fato for isso, de repente você tem mais pessoas, muita gente trabalhando nesse assunto. Tipicamente nós podemos ver os assuntos e ter essas fases: em geral a primeira fase é aquela em que se tem um esforço muito grande de resolver os principais desafios teóricos e conceituais. Ultrapassado isso, começa a aparecer uma explosão de aplicações, e isso vai sendo desenvolvido, vai sendo incorporado no nosso dia-a-dia, e se pode perguntar: e de repente, o que acontece? Isso morre? Não é que fisicamente morre, ela fica pervasiva: ela está aqui entre nós mas ninguém percebe. Mas como nós estamos no negócio acadêmico, o pesquisador tem que estar sempre na

evidência é diferente da pervasividade, não quer ficar escondida. tem que aparecer. e acho que para a área se manter ativa ela tem que estar olhando esse aspecto, ou seja, ok, aqui foi resolvido, que tipo de aplicações conseguimos construir, e agora vamos produzir os próximos desafios. então do pouco que estudei dos autômatos adaptativos, existe um limite hoje que é um dos limites teóricos das máquinas conceituais, mais especificamente, os limites da máquina de turing. não estamos falando nada diferente disso. Olhar para a máquina de turing versus autômato adaptativo, que vantagem ou desvantagem tem? a grande vantagem é que eu posso começar com um formalismo mais simples, e a adaptação me dá a capacidade de uma máquina simples começar a fazer coisa mais complicada, cujo limite é chegar na máquina de turing.

mas nesse meio caminho, nessa expansão, provavelmente existam diversos problemas teóricos de que ele seja importante serem estudados. Pela fala do prof. Carlos Cugnasca, me chamou a atenção a outro ponto, quando você fala em aplicaçoes em agronegócio, você fala em aplicações embarcadas, onde de fato você tem essa restrição de espaço de memória, tempo de processamento, mas essa barreira é quebrada à medida que você passe a usar a idéia da computação distribuída. Aí talvez eu tenha outra idéia -- estou jogando verde aqui – os autômatos adaptativos distribuídos. Como é que teria isso, como se comporta, como é que posso criar um modelo onde eu possa estudar quais são os limites, quais são os perigos, quais são as questões de desempenho em cima disso. Não tenho sugestão de "label". Não sei se seria um tema interessante. Então, a visão que eu tenho é essa: área de pesquisa, sai e nasce das idéias, as pesquisas das principais questões teóricas têm que ser vencidas, por meio de aplicações demonstrar que isso tem sua utilidade, e chega uma hora que para não deixar morrer a gente tem que acrescentar coisas mais complicadas, e assim gerar novos desafios, para continuamente levar então o circulo adiante. Agradeço então aqui pela oportunidade de participar, contribuindo com sugestões e idéias.

CEC - Partindo do que o prof. Gomi estava falando, da tecnologia adaptativa distribuída, no trabalho que estamos realizando, por natureza ele é distribuído: rede de sensores por que não dezenas,centenas e até milhares de dispositivos? E cada um, teria embutido nele o comportamento adaptativo, pois pelo comportamento das redes, a topologia é dinâmica, podendo ser alterada a todo momento, e não existe um elemento central para administrar essa mudança: está totalmente distribuído. Nosso trabalho está neste contexto: cada elemento usa a tecnologia adaptativa no sentido de otimizar a comunicação e portanto economizar energia para prolongar a vida útil da rede, então também pegando este gancho escolhendo uma palavra para atender o desafio do prof. Almir, resumindo o que poderíamos falar a respeito da tecnologia adaptativa na nossa área que pudesse espelhar o que nós esperamos ou o que nós estamos encontrando, eu traria a palavra "eficácia": estamos procurando tornar a comunicação eficiente, o uso simplificado e otimizado, então no conjunto nós teríamos uma maior eficácia da solução com mais benefícios que uma solução usual, como é praticada atualmente.

ARC - Mais alguma pergunta? Não? Então eu gostaria de agradecer aos professores, prof. Carlos, prof. Gomi e profa. Fabiana, pela explanação. Eu acho que apresentaram muitas contribuições para a área, muitas oportunidades de pesquisas para quem está iniciando, para quem está pensando em um projeto, ou para a continuidade de seus trabalhos, nas diversas áreas, então nós agradecemos em nome da comissão organizadora. Passo agora a palavra ao prof. João, para que faça o encerramento deste evento. Obrigado.

XII. ENCERRAMENTO DO EVENTO

JJN - Bom, gente, foi um prazer realmente muito grande ver o que aconteceu aqui nesses dois dias. Eu estou muito satisfeito com os resultados deste WTA, apesar do problema da chuva, da mudança de horário, etc, mas estamos aqui, no final desta tarde, com bastante gente na sala ainda.

A mesa colocou uma série de pontos bastante importantes, que foram discutidos aqui, e que refletem mais ou menos a situação em que se encontra esta pesquisa, este trabalho, que não é um trabalho do prof. João, como tanta gente fala por aí, mas é um trabalho de todos nós, que é uma coisa importante, porque se não fossem todos os que participam, isto aqui não existiria. Um sozinho realmente não resolve nada, às vezes a pessoa pode ter uma idéia, pode sugerir alguma coisa, mas quando dois sugerem, dois têm idéias, e quando são quarenta, a coisa vai crescendo, e chega em um ponto crítico em que se "perde o controle", e esse é um ponto importante que o prof. Amaury sempre coloca isso quando ele fala mas essa "perda do controle", o deixar de centralizar é o que a gente busca exatamente com esses eventos que a gente proporciona e tenta promover todos os anos. E as respostas, os resultados, aquilo que a gente viu nos trabalhos apresentados aí, que são realmente trabalhos muito interessantes e muito criativo, a tecnologia adaptativa utilizada em seu jeito, na sua forma mais diversa, em várias áreas, alguns trabalhos com conotação teórica, muitos com conotação de aplicação, muitos de aplicação com conotação didática até, e até fora de área, que a gente viu, tanta gente fora da nossa área específica,,nós demos também contribuições para cá, a gente aprende muito com isso, a gente dá um pouco do que tem, e recebe muito de volta. Eu acho que isto em muito gratifica o trabalho da gente, quando a gente vê esse monte de contribuições, esse monte de coisas novas acontecendo aí todo ano. Este ano em particular eu fiquei realmente muito satisfeito com a resposta dos alunos. Tivemos os alunos de graduação trabalhando e fazendo trabalhos excelentes, quatro trabalhos muito bons dos trabalhos dos alunos de graduação, um dos quais responde a uma das colocações que foram feitas pela mesa: da inexistência de pacotes que possam ser aproveitados e diretamente incorporados nas aplicações, para incluir a tecnologia adaptativa. Então: um dos alunos fez esse pacote um grupo de alunos deste ano fez este pacote e apresentou aqui neste evento. É uma coisa muito interessante, e acrescenta às ferramenta – diversas ferramentas – que a gente já tem também e estão disponíveis que, em particular, são frutos desse histórico da pesquisa que a gente vem desenvolvendo. Certamente tudo começou lá na teoria, mas hoje em dia a gente tem pouca coisa teórica - infelizmente, pouca coisa teórica tem sido feita nos dias de hoje. A gente

tem uma série de problemas teóricos em aberto, que estão precisando de solução, para que se possa avançar, e em desenvolvimento hoje a gente tem pouca coisa caminhando, mas a gente tem várias coisas engatilhadas, e eu acho que para o ano que vem, nós vamos ter algumas novidades interessantes. Outra coisa: a gente tem também uma série de ferramentas que foram desenvolvidas nos anos anteriores, algumas obtidas como resultados de trabalhos de teses e dissertações, uma delas em particular, que foi a tese de doutorado do prof. Almir, que procura criar ambientes para o desenvolvimento de aplicações usando dispositivos adaptativos. Hoje já não se fala mais, praticamente, em autômatos adaptativos. Os autômatos deram origem à área, começou-se a estudar a adaptatividade pelos autômatos, e hoje em dia os autômatos são eventualmente escolhidos porque seriam o melhor formalismo disponível para resolver um determinado problema. Mas existe uma série de outros dispositivos adaptativos que estão hoje em dia sendo usados extensivamente, particularmente, as tabelas de decisão, que são ferramentas muito aceitas pelos programadores, árvores de decisão, também que são estruturas de dados muito flexíveis, que são grafos dinâmicos, que podem ser usados confortavelmente com tecnologia adaptativa, porque são completamente compatíveis. Isso é um ponto importante, que a gente tem que dar um jeito de disponibilizar um pouco mais essas coisas para que as pessoas tenham conhecimento de que elas existem e possam aproveitar e fazer uso delas nas aplicações.

Outro ponto: falou-se de não-determinismo, de ensinar a máquina fazer ou aprender, e no campo de aprendizado, a gente fala muito no determinismo: a gente cria coisas que aprendem, mas que podem errar, e o que eu acho que é importante, quando a gente aplica as técnicas adaptativas com essa finalidade, infelizmente não se tem tido o cuidado de abordar a solução do problema do ponto de vista de teoria de sistemas. Infelizmente a teoria de sistemas pelo fato de ter sido tirada da grade dos nossos cursos de computação, as pessoas da área de computação não a têm usado, como poderiam. E a teoria de sistemas dá uma visão incrivelmente boa, incrivelmente rigorosa das coisas, e poderia ajudar muito a gente a desenvolver coisas, na área, com os pés bem em cima de pedras, bem firmes para poder tomar decisões com base em alguma coisa sólida, e esse é um ponto que eu acho que é uma falha conceitual em nossa grade curricular, que poderia caso fosse diferente, as coisas poderiam caminhar por outros caminhos.

Quanto às linguagens, outro ponto que foi mencionado, nós tivemos quatro trabalhos de pós-graduação que mexeram com esse assunto. Começou com o trabalho do prof. Aparecido, depois tivemos simultaneamente os trabalho do Éder, e do Amaury, trabalhando com linguagem de baixo nível, e linguagem de alto nível, voltadas para implementação de programas adaptativos. Eles de certa forma mostraram a existência, provaram que dá para fazer uma coisa deste tipo. E está em andamento agora mais um mestrado, do prof. Salvador, que está prometendo uma linguagem de alto nível completa, voltada para o desenvolvimento de programas automodificáveis. Portanto daria o mínimo necessário para se trabalhar um pouco mais confortavelmente com problemas dessa natureza. Isso cairia bem para aplicações e desenvolvimentos na área de adaptatividade. Uma coisa que está faltando, e é uma coisa fundamental para as coisas correrem bem, e eu mencionei isso num dos comentários de um dos trabalhos, é o desenvolvimento de um pouco de engenharia de software específica para tratar problemas resolvidos através de técnicas de automodificação, ou seja: tratar um problema adaptativo, ou seja, um programa que foi escrito com tecnologia adaptativa, da mesma forma que a gente trata um programa que foi feito para ter código fixo, não dá certo. As premissas são diferentes, e a gente vai aplicar resultados que dão certo para uma dada premissa, de código fixo, num programa de código variável não vai funcionar. Então existe um choque de interesses aí, que faz com que o desenvolvimento adequado de programas adaptativos, com ou sem linguagens de alto nível envolvidas aí, necessita de uma metodologia de programação necessita de uma série de técnicas, que existem meio informalmente, mas ainda não existem formalizadas, e colocadas no papel, sacramentadas, que possam ser usadas de uma forma mais ampla, mais generalizada. Então está aí uma necessidade muito grande e eu volto a apelar para quem se interessa por esse assunto que trabalhar com isso, porque isso é uma necessidade muito grande e as contribuições que vierem desses interessados certamente vão beneficiar a comunidade inteira. Existe uma primeira tentativa, uma primeira abordagem de engenharia de software e adaptatividade ao mesmo tempo, que foi dada partida nisto pelo prof. Ítalo, num seminário que ele deu aqui dois anos atrás. Ele ministrou um tutorial mostrando como fazer adaptatividade com "óculos" de orientação a objetos, ou seja, ele usou o traquejo que ele tem com a orientação a objetos para mostrar como daria para fazer até código automodificável usando uma boa engenharia de software. Mas parou ali. Não se deu continuidade a esse trabalho, e eu acho que dá para partir daí, porque ele fez bastante coisa, e está registrado em papel tudo isso. Quanto à obtenção de linguagens de uso específico. Linguagens de uso específico hoje em dia são uma necessidade, e como a demanda por linguagens de uso específico é muito grande, cada um quer ter a sua, não é prático fazer um compilador para cada uma, inventar uns 2500 tipos de linguagens, depois fazer um compilador para cada uma delas. É interessante ter uma forma de gerar esses compiladores de uma forma automática ou semi-automática. A tese do prof. Amaury mostra um primeiro caminho para se desenvolver alguma coisa neste estilo. O trabalho dele ele faz uma primeira proposta de como se faz uma linguagem, ainda rudimentar, de desenvolvimento de programas adaptativos, e mostra como gerar automaticamente, usando as técnicas convencionais de computação e um pouco de adaptatividade, o conceito de extensibilidade lingüística na linguagem de programação e ele tem um exemplo que teve sucesso e funcionou e ele implementou, inclusive, na tese dele. O ponto da topologia dinâmica distribuída que o prof. Carlos levantou, pelo fato de a gente estar ainda engatinhando nas parte de paralelismo e esse tipo de coisa, é uma questão de tempo e de pessoas que se interessem pelo assunto. São realmente pontos que foram tocados nos WTAs anteriores, e existe um interesse muito grande de que se trabalhe nisso. Infelizmente hoje a gente está sem muita gente, sem mão de obra, sem pessoas que estejam trabalhando nisso no momento, mas acho que é uma questão conjuntural. Isso tudo é o

resultado de um trabalho que está em desenvolvimento já há algum tempo, e começou quase vinte anos atrás, e a gente está hoje digamos assim, no meio do caminho, no meio da subida. Tem ainda muita coisa para subir, mas já tem muita coisa para colher. Então a gente precisa, neste momento, ter um certo discernimento de não investir tudo na subida e de não investir tudo na colheita. Porque se for só colheita, a pesquisa vai morrer, como foi muito bem colocado aí, se a gente não se lançar desafios, as coisas não vão para frente. A gente para onde está e fica satisfeito: Bom, eu já sei o que eu preciso de teoria para resolver o meu problema então não preciso mais estudar teoria. O que é que vai acontecer? Daqui a uns cinco anos eu vou ter um problema um pouquinho mais complicado e eu não vou ter bases para pisar em uma pedra firme para poder seguir em frente. Então nós precisamos, sim, investir um pouco na teoria, sem descuidar do resto. Eu acho que só uma coisa, ou só a outra, não funciona bem. Pode até funcionar temporariamente enquanto a bateria tem carga. Quando acaba a carga da bateria, não tem jeito: tem que ligar na tomada, se não a coisa não funciona. Então, a nossa tomada é a teoria. A nossa tomada, inclusive, para aplicações em engenharia, é a teoria e aquilo que vem dela. Tudo aquilo que nós fizemos até hoje nós devemos aos estudos teóricos que foram feitos inicialmente e que têm sido feitos, talvez com uma velocidade mais baixa, não tão grande como deveria ser, hoje, mas a gente vê que as colheitas que têm sido feitas são muito grandes. As aplicações que a gente tem obtido, nas quais a gente tem conseguido aplicar essas idéias todas, são muito numerosas e muito interessantes nas mais diversas áreas. Então eu mais uma vez gostaria de agradecer a todos por terem comparecido a este evento, trazido para a gente as contribuições, o próprio interesse, o fato de vocês estarem aqui é algo que anima a gente porque se essa sala estivesse com quatro pessoas agora a gente iria pensar seriamente se valeria a pena ou não fazer de novo o ano que vem. Eu não tenho dúvidas a respeito disso. Acho que, apesar de todos os percalços, temos muitos interessados na área, e outros que não vieram por causa da mudança de data que infelizmente ocorreu não por vontade nossa, mas não houve como evitar. Então eu agradeço muito a todos, um abraço muito forte em cada um, e eu gostaria que vocês continuassem com a gente, ao longo do ano, e participem, escrevam artigos para a gente apresentar no ano que vem, porque se não vierem artigos da nossa comunidade não vêm de fora. A gente sabe um pouco mais dessas coisas que os que estão começando agora. Cabe a nós apresentar material para que dois, ou três ou quatro que vierem de fora o ano que vem tenham a possibilidade de se iniciar, aproveitando a nossa experiência, e assim dar continuidade ao nosso trabalho e continuar motivados para a gente trabalhar nesse assunto tão interessante. Mais uma vez, muito obrigado aos membros da mesa redonda e aproveitando para convidá-los a participarem com a gente não apenas no ano que vem, no próximo WTA, mas ao longo do ano todo, em interações que vão ser certamente muito proveitosas. Então muito obrigado, e até o ano que vem.

ESTA TRANSCRIÇÃO LIVRE, DAS FALAS DA MESA REDONDA, FOI PRODUZIDA PELO PROF. DR. JOÃO JOSÉ NETO A PARTIR DE GRAVAÇÃO EM VÍDEO REALIZADA DURANTE O EVENTO. PROCUROU-SE RESPEITAR FIELMENTE TUDO O QUE FOI PROFERIDO NA OCASIÃO, PORÉM FORAM NECESSÁRIAS PEQUENAS ALTERAÇÕES NAS FALAS, COM A FINALIDADE DE PRIORIZAR O CONTEÚDO TÉCNICO, BEM COMO ELIMINAR REDUNDÂNCIAS E EXPRESSÕES COLOQUIAIS OU OUTRAS, PRÓPRIAS DA LINGUAGEM FALADA, PORÉM INADEQUADAS A UM REGISTRO ESCRITO DESTA NATUREZA.