Memórias do WTA 2012

VI 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
2012

Ficha catalográfica

Workshop de Tecnologia Adaptativa (6: 2012: São Paulo)
Memórias do WTA 2012. – São Paulo; EPUSP, 2012. 121p.

ISBN 978-85-86686-66-5

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

Apresentação

A sexta edição do Workshop de Tecnologia Adaptativa realizou-se em São Paulo, Brasil, nos dias 26 e 27 de Janeiro de 2012, nas dependências da Escola Politécnica da Universidade de São Paulo. As contribuições encaminhadas na forma de artigos relacionados à Tecnologia Adaptativa, nas seguintes áreas, abrangeram, de forma não exclusiva, os tópicos abaixo:

Fundamentos da Adaptatividade

- Modelos de computação, autômatos, gramáticas, grafos e outros dispositivos automodificáveis, suas notações, sua formalização, complexidade, propriedades e comparações com formalismos clássicos.

Tecnologia Adaptativa – Técnicas, Métodos e Ferramentas

- Aplicação dos conhecimentos científicos relativos à adaptatividade e dos dispositivos adaptativos como fundamento para a formulação e para a resolução de problemas práticos.
- Ferramentas, técnicas e métodos para a automatização da resolução de problemas práticos usando técnicas adaptativas.
- Programação adaptativa: linguagens, compiladores e metodologia para o desenvolvimento, implementação e validação de programas com código adaptativo.
- Meta-modelagem de software adaptativo.
- Engenharia de Software voltada para a especificação, projeto, implementação e desenvolvimento de programas automodificáveis de qualidade.
- Adaptatividade multinível e outros conceitos introduzidos recentemente: avanços teóricos, novas idéias para aplicações, sugestões de uso prático.
- Linguagens de alto nível para a codificação de programas automodificáveis: aplicações experimentais e profissionais, práticas e extensas, das novas idéias de uso de linguagens adequadas para a codificação de programas adaptativos e suas metodologias de desenvolvimento.

Aplicações da Adaptatividade e da Tecnologia Adaptativa

- Inteligência computacional: aprendizagem de máquina, representação e manipulação do conhecimento;
- Computação natural, evolutiva e bio-inspirada;
- Sistemas de computação autonômica e reconfigurável;

- Processamento de linguagem natural, sinais e imagens: aquisição, análise, síntese, reconhecimento, conversões e tradução;
- Inferência, reconhecimento e classificação de padrões;
- Modelagem, simulação e otimização de sistemas inteligentes de: tempo real, segurança, controle de processos, tomada de decisão, diagnóstico, robótica;
- Simulação, arte por computador e jogos eletrônicos inteligentes;
- Outras aplicações da Adaptatividade, nas diversas áreas do conhecimento: ciências exatas, biológicas e humanas.

Comissão Organizadora

- André Riyuiti Hirakawa (São Paulo, SP, Brasil)
- Amaury Antônio de Castro Junior (Ponta Porã, MS, Brasil)
- João José Neto, Chair (São Paulo, SP, Brasil)
- Ricardo Luis de Azevedo da Rocha (São Paulo, SP, Brasil)

Apoio

- Escola Politécnica da Universidade de São Paulo
- IEEE – Institute of Electrical and Electronics Engineers
- PCS – Departamento de Engenharia de Computação e Sistemas Digitais
- São Paulo Convention & Visitors Bureau
- SBC – Sociedade Brasileira de Computação
- SPC – Sociedad Peruana de Computación
- Universidade de São Paulo

Memórias do WTA 2012

Esta publicação do Laboratório de Linguagens e Técnicas Adaptativas do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo é uma coleção de textos produzidos para o WTA 2012, o Sexto Workshop de Tecnologia Adaptativa, realizado em São Paulo nos dias 26 e 27 de Janeiro de 2012. A exemplo da edição de 2011, 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.

Introdução

Com muita satisfação compilamos neste documento estas memórias com os artigos apresentados no WTA 2012 – Sexto Workshop de Tecnologia Adaptativa, realizado na Escola Politécnica da Universidade de São Paulo nos dias 26 e 27 de Janeiro de 2012.

Esta edição contou com 21 trabalhos relacionados à área de Tecnologia Adaptativa e aplicações nos mais diversos segmentos. O evento foi registrou uma participação efetiva de uma centena de pesquisadores durante os dois dias, constatando-se o sucesso do mesmo.

Esta publicação

Estas memórias espelham o conteúdo apresentado no WTA 2012. A exemplo do que foi feito no ano anterior, todo o material referente aos trabalhos apresentados no evento estará acessível no portal do WTA 2012, 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.

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 2015.

Agradecimentos

Às instituições que apoiaram o WTA 2012, à comissão organizadora, ao pessoal de apoio e a tantos colaboradores voluntários, cujo auxílio propiciou o êxito que tivemos a satisfação de observar. Gostaríamos também de agradecer às seguintes pessoas pelo valioso auxílio para a viabilização das necessidades locais do evento:

- Daniel Assis Alfenas
- Daniel Costa Ferreira
- Edson de Souza
- Leia Sicília
- Luís Emilio Cavechiolli Dalla Valle
- Nilton Araújo do Carmo
- Rafael Barbolo Lopes
- Ricardo Henrique Gracini Guiraldelli

De modo especial, agradecemos ao Departamento de Engenharia de Computação e Sistemas Digitais e a Escola Politécnica da Universidade de São Paulo pelo inestimável apoio para a realização da sétima edição do Workshop de Tecnologia Adaptativa.

São Paulo, 27 de Janeiro de 2012

João José Neto
Coordenador geral

Sumário

Lista de autores

Parte I

Submissões

Avaliação de Políticas Abstratas na Transferência de Conhecimento em Navegação Robótica

R. L. Beirigo, F. A. Pereira, M. L. Koga, T. Matos, V. F. da Silva, A. H. Reali Costa

***Abstract*— This paper presents a new approach to the problem of solving a new task by the use of previous knowledge acquired during the process of solving a similar task in the same domain, robot navigation A new algorithm, Qab-Learning is proposed to obtain the abstract policy that will guide the agent in the task of reaching a goal location from any other location in the environment, and this policy is compared to the policy derived from another algorithm, ND-TILDE. The policies are applied in a number of different tasks in two environments. The results show that the policies, even after the process of abstraction, present a positive impact on the performance of the agent.**

***Keywords*— relational representation, abstract policy, reinforcement learning, inductive logic programming, robotics.**

I. Introdução

ATUALMENTE robôs móveis encontram-se presentes nos mais diversos locais, auxiliando em serviços de entrega e limpeza em hospitais, escritórios e residências. Entretanto, a difusão de seu uso e completa aceitação só se dará, de fato, se os robôs forem capazes de rapidamente aprenderem novas tarefas e de reutilizarem o conhecimento previamente adquirido em novas atividades, ou seja, se demonstrarem boas autonomia e capacidade de adaptação a novas situações. Apesar da diversidade das soluções encontradas na literatura para a navegação autônoma de robôs móveis [1, 2, 3], elas tipicamente não consideram soluções previamente encontradas para problemas similares. Isso impede que o robô tenha a habilidade de melhorar seu desempenho por meio do aprendizado decorrente de experiências passadas em tarefas similares. Além disso, os planos de navegação gerados por estas soluções não são facilmente comunicados a humanos, tornando menos eficiente a interação entre humanos e robôs. Representações baseadas em Lógica de Primeira Ordem podem facilitar tal comunicação [4].

Tem havido um grande e recente interesse em abordar o problema de adaptação em tarefas de navegação e planejamento em robôs móveis, utilizando uma linguagem mais expressiva. Diversas técnicas têm sido exploradas, desde a construção de descrições relacionais de macroações (planos parciais) para transferência e reuso no aprendizado de novas políticas de atuação [5] até a utilização de aprendizes simbólicos proposicionais para generalizar a política ótima para outros problemas [6].

Este artigo tem por objetivo propor uma técnica de abstração do conhecimento adquirido por um robô na solução de um determinado problema, para permitir sua transferência e uso em um outro novo problema, similar ao anterior. Além disso, visa avaliar comparativamente esta nova técnica com uma proposta anteriormente na literatura [7] [8], em aplicações de navegação robótica em ambientes internos de edifícios.

O artigo está organizado da seguinte forma. A seção II apresenta os conceitos de Processos Markovianos de Decisão e Lógica de Primeira Ordem. As seções III e IV descrevem duas técnicas para se aprender políticas abstratas, ND-TILDE e Qab-Learning. A seção V descreve como se dá a aplicação das políticas em um caso de navegação robótica e os resultados obtidos dos experimentos são discutidos na seção VI. Finalmente, na seção VII estão as conclusões.

II. Representação Relacional

Uma abordagem tradicional de formalização para problemas estocásticos de decisão sequencial consiste no uso de Processos Markovianos de Decisão (MDP – *Markov Decision Process*), que podem ser expandidos para MDPs relacionais, ou RMDPs (*Relational Markov Decision Process*), a fim de permitir descrições através de Lógica de Primeira Ordem.

Um MDP pode ser definido formalmente pela quádrupla <S, A, T, R> [4, 9], onde S e A são conjuntos finitos de estados do ambiente e ações que o agente pode realizar, respectivamente, sendo o conjunto de ações aplicáveis no estado $s \in S$ denotado por A(s). T: $S \times A \times S \rightarrow [0, 1]$ é a função de transição de estado, com $T(s_t, a_t, s_{t+1})$ definindo a probabilidade de realizar a transição do estado s_t para o estado s_{t+1} através da execução da ação a_t e R: $S \times A \times S \rightarrow \Re$ é a função de recompensa, com $r_t = R(s_t, a_t, a_{t+1})$. Dessa forma, uma transição do estado $i \in S$ para $j \in S$, decorrente da execução de alguma ação $a \in A(i)$, ocorre com probabilidade T(i, a, j), e então uma recompensa R(i,a,j) é recebida.

Em um MDP, pode-se utilizar Lógica de Primeira Ordem na descrição de estados e ações, aplicando variáveis para evidenciar as relações entre os objetos, transformando-o em um RMDP [10, 11]. Na lógica relacional, expressões da forma $p(t_1, t_2, ..., t_m)$ são átomos, com p sendo um símbolo de relação entre os termos t_i. Por exemplo, *con*(r_1, r_2) indica que os

R. L. Beirigo, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, rafaelbeirigo@usp.br.

F. A. Pereira, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, fernando.pereira2@poli.usp.br.

M. L. Koga, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, mlk@usp.br.

T. Matos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, tiago.matos@poli.usp.br.

V. F. da Silva, EACH, Universidade de São Paulo, São Paulo, SP, Brasil, valdinei.freire@usp.br.

A. H. Reali Costa, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, anna.reali@poli.usp.br.

cômodos r_1 e r_2 estão conectados (são adjacentes e mutuamente acessíveis). Um termo pode consistir em uma variável (iniciando com letra maiúscula) ou uma constante (iniciando com letra minúscula, como r_1). Conjunções de átomos lógicos representam estados. Assim, um exemplo de estado $z \in S$ seria:

$$z \equiv in(r_1), con(r_1, r_2), room(r_1), room(r_2).$$

indicando que o robô está no cômodo r_1, o qual é conectado ao cômodo r_2. Da mesma forma, define-se o conceito de estados abstratos como conjunções de átomos lógicos que contêm variáveis. Por exemplo, o estado abstrato

$$Z \equiv in(R), con(R,R'), room(R), room(R')$$

representa estados nos quais o robô está no cômodo R, o qual é conectado a um outro cômodo R'. Neste caso, o estado z seria uma instância do estado abstrato Z. Uma substituição θ atribui termos às variáveis de forma que $Z\theta \subseteq z$. Assim, para o exemplo acima, a instância z seria alcançada com a substituição $\theta = \{R/r_1, R'/r_2\}$. Um estado é denominado concreto se não contiver variáveis. Conjunções com variáveis são consideradas existencialmente quantificadas. Uma conjunção Z é subsumida pela conjunção W, isto é, $Z \leq_\theta W$ se existir θ tal que $W\theta \subseteq Z$. Analogamente, ações abstratas podem ser representadas por átomos tendo variáveis como argumentos. Assim, *goto*(R,R') é uma ação abstrata que indica que o robô navega de um cômodo R para um cômodo R', e *goto*(r_1, r_2) é uma ação concreta com a substituição $\theta = \{R/r_1, R'/r_2\}$.

Resolver um problema modelado por um (R)MDP consiste em computar uma política π que especifica quais ações $a \in A$ devem ser executadas quando o agente está no estado $s \in S$. Se a política for não determinística, tem-se que $\pi: S \rightarrow 2^A$; entretanto, se a política for determinística, tem-se que $\pi: S \rightarrow A$, ou seja, π é determinística se leva cada estado em S a uma única ação em A. Dada a política π e um fator de desconto $\gamma \in [0, 1]$ que desconta recompensas futuras, a função de valor de estado $V^\pi: S \rightarrow \Re$ representa o valor de estar em um estado seguindo a política π, isto é, representa as recompensas esperadas. A política ótima π^* é a política que satisfaz a condição: $V^{\pi^*} \geq V^{\pi'}$, $\forall s \in S$ e $\forall \pi'$. A função valor ótima é denotada por V^*. Assim, o interesse consiste em computar uma política que melhor aproxime, ou, idealmente, iguale a política ótima π^*.

Da mesma forma que se definiu estados e ações abstratas no RMDP, pode-se definir uma política abstrata π_a [11] como uma lista ordenada de regras de ações abstratas na forma i: $\sigma \Rightarrow \{\alpha\}$, sendo σ um estado abstrato; $\{\alpha\}$, um conjunto de ações abstratas e i um inteiro representando a ordenação da regra.

Existem diversas formas para encontrar a política que soluciona um (R)MDP. Um exemplo é a utilização de algoritmos de planejamento de atividades, como o SPUDD [12], ou técnicas de aprendizado que utilizam repetidas iterações do agente com o ambiente, como no aprendizado por reforço [9]. Por sua vez, uma política abstrata pode ser obtida tanto por meio da generalização de uma política concreta conhecida quanto pelo aprendizado direto. Neste artigo, essas duas formas são exploradas. A primeira utiliza uma política concreta conhecida para gerar exemplos, que são agregados por meio de aprendizado supervisionado usando o algoritmo ND-TILDE [7], descrito na seção III. Já a segunda forma utiliza aprendizado por reforço por meio da abordagem proposta nesse artigo, descrita na seção IV.

III. Aprendendo políticas abstratas com aprendizado indutivo.

Esta seção descreve um método de obtenção de uma política abstrata através da indução de uma árvore de decisão em Lógica de Primeira Ordem [8] a partir de exemplos gerados por uma política concreta conhecida.

Dado um conjunto de exemplos E, onde cada exemplo consiste em um par estado-ação obtido durante a execução de uma política concreta conhecida, pode-se aplicar o algoritmo ND-TILDE, descrito em pseudo-código no Algoritmo 1. Ele induz a política abstrata a partir de E, gerando uma representação através de uma árvore binária de decisão em Lógica de Primeira Ordem [7].

Algoritmo 1: Algoritmo ND-TILDE

```
1:  função ND-TILDE (E: cj. de exemplos): retorna uma
    árvore binária de decisão
2:      T ← GERAR_TESTES(E)
3:      τ ← DIVISAO_OTIMA(T,E)
4:      E_e ← TESTE_VERDADEIRO(E,τ)
5:      E_d ← TESTE_FALSO(E,τ)
6:      se CRITERIO_PARADA(E,E_e,E_d) então
7:          retorna folha(INFO(E))
8:      senão
9:          t_e ← ND-TILDE(E_e)
10:         t_d ← ND-TILDE(E_d)
11:         retorna nó_interno(τ,t_e,t_d)
12:     fim se
13: fim função
```

ND-TILDE utiliza um conjunto de exemplos de treinamento E para criar uma árvore. Os candidatos aos testes de decisão são criados a partir de um conjunto de operadores de refinamento [8] definidos previamente por um especialista (passo 2 do Algoritmo 1). Cada um desses operadores gera um conjunto de literais de primeira ordem que é candidato para a divisão do conjunto de exemplos E fornecido. O melhor teste que pode ser aplicado para dividir o conjunto E é aquele que apresenta a maior redução de entropia. O teste ótimo é escolhido utilizando-se a taxa de ganho padrão. Este teste particiona o conjunto E de exemplos em dois outros conjuntos: E_e que contém os exemplos onde o teste τ é verdadeiro (passo 4); e E_d que contém os exemplos onde o teste τ é falso (passo 5).

Se a partição induzida em E indica que a divisão deva

parar (procedimento CRITERIO_PARADA no passo 6), uma folha contendo sentenças atômicas que representam ações abstratas é então criada na árvore (passo 7). Caso mais de uma sentença atômica representando uma ação esteja associada aos exemplos remanescentes na folha, todas elas são adicionadas pelo algoritmo a esta folha, gerando uma política não-determinística. Dessa forma, todas as ações abstratas indicadas pelos exemplos de E presentes nessa partição são associadas ao estado abstrato que corresponde à folha gerada. Se a partição induzida em E não indica que o processo deva parar, então para cada um dos elementos da partição (conjuntos E_e e E_d), a função ND-TILDE é chamada recursivamente (passos 9 e 10). Um nó interno é então criado (passo 11) usando o teste ótimo τ como teste e tendo como filhos as duas sub-árvores (τ_e e τ_d) criadas em cada chamada de ND-TILDE.

Ao término do algoritmo, cada caminho na árvore de decisão gerada da raiz até uma folha (excluindo esta) determina um estado abstrato, e cada folha possui um conjunto de ações abstratas relacionadas àquele estado. Percorrendo essa árvore sistematicamente seguindo a estratégia de busca em profundidade, obtém-se a sequência de regras ordenadas que compõem a política abstrata. A Seção V apresenta um exemplo de política abstrata para o problema da navegação em robótica obtida através da aplicação do ND-TILDE.

IV. Aprendendo políticas abstratas com aprendizado por reforço.

Nessa seção é proposto o uso de aprendizado por reforço (RL – *Reinforcement Learning*) como uma alternativa para a determinação da política abstrata que generaliza o conhecimento da solução do problema defrontado pelo robô. No RL, o aprendizado se dá por meio da interação direta do agente com o ambiente, em intervalos de tempos discretos contendo ciclos alternados de percepção e ação. Tido como o mais popular algoritmo de RL, o algoritmo Q–Learning é um algoritmo de aprendizado livre de modelos, com o qual uma política ótima π^* é aprendida iterativamente quando o modelo do sistema não é conhecido, i.e., as funções T e/ou R do (R)MDP são desconhecidas [9]. Fazendo $V^*(s) = \max_a Q^*(s,a)$, o algoritmo Q–Learning iterativamente aproxima a estimativa de $Q^*(s,a)$, da seguinte forma:

$$Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha) Q_t(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q_t(s_{t+1}, a) \right), \quad (1)$$

sendo s_t o estado atual, a_t a ação realizada em s_t, r_t o reforço recebido após realizar a_t em s_t e atingir s_{t+1}, s_{t+1} o novo estado, $\gamma \in [0, 1]$ o fator de desconto e $\alpha \in]0, 1]$ a taxa de aprendizagem. O aprendizado aqui conduzido segue uma política ε-*greedy*, que consiste em aplicar uma ação totalmente aleatória quando o valor de uma variável aleatória for menor que ε (correspondendo à fase de exploração), e aplicar a política *greedy*, isto é, $\pi(s) = \arg\max_a Q(s, a)$, no caso contrário (correspondendo à fase de explotação).

Algoritmo 2: Qab-Learning

1: **função** Qab-Learning (**σ**: cj. de estados abstratos, **α**: cj. de ações abstratas, ε: fator exploração): retorna uma política abstrata
2: INICIALIZAR Qab_0
3: Observar o estado concreto s_t
4: Determinar o estado abstrato σ_t que subsume s_t, isto é, $s_t \leq_{\theta t} \sigma_t$
5: **repetir até** CONDIÇÃO_PARADA
6: **se** um número aleatório $\in [0,1] \leq \varepsilon$ **então**
7: Determinar a ação concreta a_t = RANDOM($A(s_t)$)
8: **senão**
9: Determinar ação abstrata: $\alpha_t \leftarrow \arg\max_\alpha Qab(\sigma_t, \alpha)$
10: Determinar o cj. de ações concretas subsumidas por α_t com a substituição θ_t: $\{a_t\} \leq_{\theta t} \alpha_t$
11: Fazer a_t = RANDOM($\{a_t\}$)
12: **fim se**
13: Executar a ação concreta a_t
14: Observar o novo estado concreto s_{t+1} e o reforço recebido r_t
15: Atualizar $Qab(\sigma_t, \alpha_t)$ usando a equação (1), com r_t e σ_{t+1} que subsume s_{t+1}
16: $s_t \leftarrow s_{t+1}$; $\sigma_t \leftarrow \sigma_{t+1}$
17: **fim repetir**
18: **laço para cada** $\sigma \in$ **σ faça**
19: $\pi_{abs}(\sigma) \leftarrow \arg\max_\alpha Qab(\sigma, \alpha)$
20: **fim laço**
21: **retornar** (π_{abs})
22: **fim função**

O Algoritmo 2 apresenta o pseudo-código do algoritmo Qab-Learning, que é uma modificação do algoritmo Q-Learning para aprender diretamente uma política abstrata. O algoritmo Qab-Learning consiste basicamente no Q-Learning tradicional usando a política ε-*greedy*, porém o aprendizado se dá no nível abstrato. Observa-se o estado concreto atual (passo 3) e determina-se o estado abstrato que subsume esse estado concreto e a correspondente substituição θ_t (passo 4). O ciclo de percepção e ação prossegue até atingir uma condição de parada (passo 5). Se na iteração atual uma exploração deve ser feita (passo 6), então deve-se determinar uma ação aleatória concreta do conjunto de ações aplicáveis no estado concreto atual (passo 7). Se não, fazer explotação determinando a ação abstrata recomendada pela política *greedy* para o estado abstrato que subsume o estado concreto atual (passo 9) e determinar o conjunto de ações concretas subsumidas por esta ação abstrata (passo 10). Escolher aleatoriamente uma ação concreta deste conjunto (passo 11). Executar a ação definida pela fase de exploração ou de explotação (passo 13), observar o reforço recebido e o novo estado concreto (passo 14). Determinar o estado abstrato que subsume o novo estado concreto observado e atualizar a tabela Qab no nível abstrato com esses dados (passo 15). Fazer o estado concreto e o correspondente abstrato atual como sendo o estado observado (passo 16) e repetir o ciclo. No final, retornar a política aprendida no nível abstrato. Na

próxima seção este algoritmo é aplicado no problema de navegação robótica, gerando uma política abstrata que busca generalizar a solução do problema.

V. Aplicando na Navegação Robótica

Considere agora um problema de navegação de robôs, cujo ambiente é o ilustrado na Fig. 1 (doravante ambiente 1).

Os estados deste problema podem ser descritos de acordo com o seguinte conjunto de predicados:
{*corridor*/1, *room*/1, *center*/1, *nearDoor*/1, *in*/1, *con*/2}, e o conjunto de ações com: {*gotoRDRD*/2, *gotoCDCD*/2, *gotoCCCC*/2, *gotoRCRD*/2, *gotoRDRC*/2, *gotoRDCD*/2, *gotoCDRD*/2, *gotoCCCD*/2, *gotoCDCC*/2}.

O significado de cada predicado é literal: por exemplo, se o agente estiver ocupando a localização 1 no mapa (representado como objeto l1), pode-se descrever o estado como:
s_{l1} = {*in(l1), corridor(l1), center(l1), con(l1,l5), corridor(l5), nearDoor(l5)*}

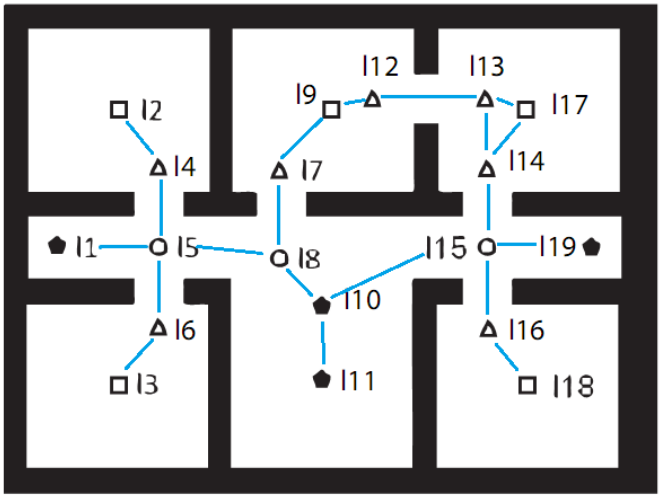


Figura 1 – Mapa do ambiente usado no aprendizado de políticas (ambiente 1). Centros e portas de cômodos são representados por quadrados e triângulos, respectivamente; círculos representam locais que estão perto de portas em corredores e pentágonos pretos representam locais nos corredores que não estão próximos de portas. As adjacências entre lugares estão indicadas por um grafo sobreposto no mapa, onde os lugares são vértices e as arestas indicam adjacências.

Se o agente faz a escolha de ir da localização *l1*, que é *corridor(l1)* e *center(l1)*, para a localização *l5*, que é *corridor(l5)* e *nearDoor(l5)*, descreve-se essa ação como *gotoCCCD(l1,l5)*. As abreviações CC e CD indicam *corridor* e *center*, *corridor* e *door*, respectivamente, e as demais seguem o mesmo padrão.

A precondição de *gotoXXYY*(Li,Lj) é (*in(Li)* ^ *con(Li,Lj)*) e ambas localizações envolvidas na ação são representadas pelas variáveis Li e Lj, as quais atendem às condições *XX* e *YY*.

A. Definindo as Políticas Abstratas

O problema que o agente deve resolver é episódico e consiste em atingir um estado-meta. Esse estado-meta foi modelado como sendo o único estado para o qual o agente recebe um reforço positivo, independentemente da posição inicial.

Dado este domínio, a tarefa utilizada para a indução das duas políticas abstratas foi a de ir de qualquer local no ambiente 1 para a localidade *l2*. Ou seja, o objetivo é chegar a *l2*, partindo de qualquer outro lugar no mapa. A primeira política, π_{ND}, foi criada com o algoritmo ND-TILDE, fornecido na Seção III (Algoritmo 1), e a segunda, π_{Qab}, com o algoritmo Qab-Learning, fornecido na Seção IV (Algoritmo 2).

Para a política π_{ND}, em primeiro lugar descobriu-se a política ótima concreta para o ambiente 1. Dada essa política conhecida, aplicou-se o ND-TILDE para a extração da política abstrata π_{ND} em forma de árvore de decisão. A árvore pode também ser descrita como um conjunto de regras ordenadas, que estão ilustradas na Tabela I.

Nota-se que existem 8 estados abstratos nessa política, pois esses foram os estados que o algoritmo julgou serem as mais relevantes partições do problema. Além disso, cada estado aponta para um conjunto de ações abstratas. Na implementação dos experimentos deste artigo, descritos na Seção VI, a escolha entre uma das ações deste conjunto é feita aleatoriamente.

Já a segunda política, π_{Qab}, foi construída de forma diferente. Cada descrição de estado concreto (a descrição de um estado só contempla conexões com localidades vizinhas) foi transformada em um estado abstrato simplesmente substituindo por variável cada constante presente nos termos dos predicados, resultando em 12 estados abstratos diferentes, que generalizam os 19 estados concretos do problema. Em posse desse conjunto de estados abstratos e do conjunto de ações abstratas, o algoritmo Qab-Learning (Algoritmo 2) foi usado para gerar uma política abstrata π_{Qab}. Para o estado abstrato em que i=7, foram encontradas duas ações abstratas ótimas, sendo os experimentos executados com a aplicação de ambas. O resultado pode ser verificado na Tabela II.

Tabela I
Política Abstrata construída com o ND-TILDE

i	Estado Abstrato	Ações Abstratas
1	*in(A), con(A,B), center(B), corridor(B), center(A), corridor(A), con(A,C), nearDoor(C), corridor(C)*	*{gotoCCDC(A,C)}*
2	*in(A), con(A,B), center(B), corridor(B), center(A), corridor(A)*	*{gotoCCCC(A,B)}*
3	*in(A), con(A,B), center(B), corridor(B), con(A,C), nearDoor(C), corridor(C)*	*{gotoCDRD(A,D), gotoCDCD(A,C)}*
4	*in(A), con(A,B), center(B), corridor(B)*	*{gotoCDCC(A,B)}*
5	*in(A), nearDoor(C), corridor(C), center(A,) corridor(A)*	*{gotoCCCD(A,C)}*
6	*in(A), nearDoor(C), corridor(C)*	*{gotoRDRC(A,D) gotoRDCD(A,C)}*
7	*in(A), room(A), nearDoor(A)*	*{gotoRDRC(A,D), gotoRDRD(A,D)}*
8	*in(A)*	*{gotoRCRD(A,D)}*

TABELA II
POLÍTICA ABSTRATA CONSTRUÍDA COM QAB-LEARNING

i	Estado Abstrato	Ações Abstratas
1	*in(A), con(A,B), con(A,C), con(A,D), con(A,E), nearDoor(A), corridor(A), center(B), corridor(B), nearDoor(C), room(C), nearDoor(D), room(D), nearDoor(E), corridor(E)*	*gotoCDRD(A,D)*
2	*in(A), con(A,B), con(A,C), con(A,D), con(A,E), nearDoor(), corridor(A), center(B), corridor(B), nearDoor(), room(C), nearDoor(D), room(D) , center(E), corridor(E)*	*gotoCDCC(A,B)*
3	*in(A), con(A,B), con(A,C), con(A,D), nearDoor(A), corridor(A), nearDoor(B), corridor(B), nearDoor(C), room(C), center(D), corridor(D)*	*gotoCDCD(A,B)*
4	*in(A), con(A,B), con(A,C), con(A,D), center(A), corridor(A), nearDoor(B), corridor(B), center(C), corridor(C), nearDoor(D), corridor(D)*	*gotoCCCD(A,B)*
5	*in(A), con(A,B), nearDoor(A), room(A), center(B), room(B), con(A,C), nearDoor(C), room(C), con(A,D), nearDoor(D), room(D)*	*gotoRDRD(A,C)*
6	*in(A), con(A,B), nearDoor(A), room(A), center(B), room(B), con(A,C), nearDoor(C), room(C), con(A,D), nearDoor(D), corridor(D)*	*gotoRDCD(A,D)*
7	*in(A), con(A,B), nearDoor(A), room(A), center(B), room(B), con(A,C), nearDoor(C), corridor(C)*	*gotoRDRC(A,D) ou gotoRDCD(A,D)*
8	*in(A), con(A,B), con(A,C), center(A), room(A), nearDoor(B), room(B), nearDoor(C), room(C)*	*gotoRCRD(A,B)*
9	*in(A), con(A,B), con(A,C), nearDoor(A), room(A), center(B), room(B), nearDoor(C), room(C)*	*gotoRDRC(A,B)*
10	*in(A), con(A,B), center(A), corridor(A), nearDoor(B), corridor(B)*	*gotoCCDC(A,B)*
11	*in(A), con(A,B), center(A), room(A), nearDoor(B), room(B)*	*gotoRCRD(A,B)*
12	*in(A), con(A,B), center(A), corridor(A), center(B), corridor(B)*	*gotoCCCC(A,B)*

B. Aplicando as Políticas Abstratas

O objetivo da construção dessas políticas abstratas é possibilitar seus usos em outros problemas similares, avaliando a extensão da transferência desse conhecimento adquirido. Para essa aplicação em robótica móvel, as políticas geradas foram testadas no mesmo ambiente 1, porém para tarefas distintas, e também em um outro ambiente (ambiente 2), maior que o anterior e cujo mapa pode ser observado na Fig. 2. Para a escolha deste problema "similar", foram consideradas duas características para serem mantidas fixas entre os problemas: o conjunto de predicados que descrevem os estados e o conjunto de ações abstratas.

A aplicação da política abstrata em um problema ocorre da seguinte forma: primeiro, observa-se o estado concreto s_t no qual o agente se encontra. Busca-se nas regras que definem a política abstrata em uso (π_{ND} ou π_{Qab}), na ordem, qual estado abstrato σ_t (antecedente da regra) que subsume o estado s_t, isto é, $s_t \leq_\theta \sigma_t$, definindo a regra e a respectiva substituição θ, obtendo-se, assim, o correspondente conjunto de ações abstratas, já com a respectiva substituição θ aplicada. Em posse deste conjunto de ações abstratas instanciadas com a substituição θ, uma delas é escolhida aleatoriamente (chamada aqui de α_θ). Caso esta ação α_θ escolhida já seja uma ação concreta (devido à substituição aplicada), esta ação concreta a_t é executada em s_t; caso contrário, ela definirá um conjunto $A\alpha_\theta$ de ações concretas que poderão ser aplicadas em s_t (substituindo adequadamente todas as variáveis restantes em α_θ), com $A\alpha_\theta \subseteq A(s_t)$, e uma escolha aleatória é feita neste conjunto $A\alpha_\theta$ para definir a ação concreta a_t a ser executada em s_t. Para o caso de aplicar-se em um outro ambiente a política abstrata gerada para um determinado ambiente, é possível que o estado concreto atual s_t não seja subsumido por nenhum estado abstrato da política abstrata em uso; neste caso, a implementação corrente escolhe aleatoriamente uma ação a_t no conjunto $A(s_t)$ e executa a_t em s_t. Este processo repete-se para o novo estado s_{t+1} alcançado com a aplicação de a_t em s_t, até que s_{t+1} seja um estado-meta.

Para acelerar a execução da tarefa, quando o agente se depara com um estado no qual ele já passou anteriormente no mesmo episódio, todas as escolhas aleatórias citadas são feitas excluindo-se a(s) escolha(s) anterior(es). Ainda, caso o agente volte a um mesmo estado após ter tentado todas as ações possíveis indicadas pela sua política, então a política aleatória é empregada, para que o agente possa assim encontrar sua meta.

Para exemplificar este procedimento, considere que o ambiente seja o ambiente 1 (Fig.1), que esteja sendo aplicada a política abstrata π_{ND} (Tabela I) e que o agente se encontre no local *l8*, sendo

s_t = { *in(l8), con (l8,l10), center (l10), corridor (l10), con(l8, l5), nearDoor (l5), corridor (l5), con(l8, l7), nearDoor (l7), room(l7)*}.

Aplicando a política π_{ND}, verifica-se que o estado abstrato que subsume s_t é o estado da terceira regra da Tabela 1:

σ_t = { *in(A), con (A,B), center (B), corridor (B), con(A,C), nearDoor (C), corridor (C)*}

com substituição θ = { *A/l8, B/l10, C/l5*} e o conjunto de ações abstratas definido pela política π_{ND} é {*gotoCDRD(A,D), gotoCDCD(A,C)*}. Supondo que a escolha aleatória escolheu a ação abstrata *gotoCDRD(A,D)* que, com θ, resulta em $A\alpha_\theta$ = { *gotoCDRD(l8,D)* } = { *gotoCDRD(l8,l7)*}, já que *D=l7* é a única substituição possível neste caso. Assim, a ação concreta a_t = *gotoCDRD(l8,l7)* é executada em s_t, e o agente então observa o novo estado e repete o procedimento, até atingir a meta.

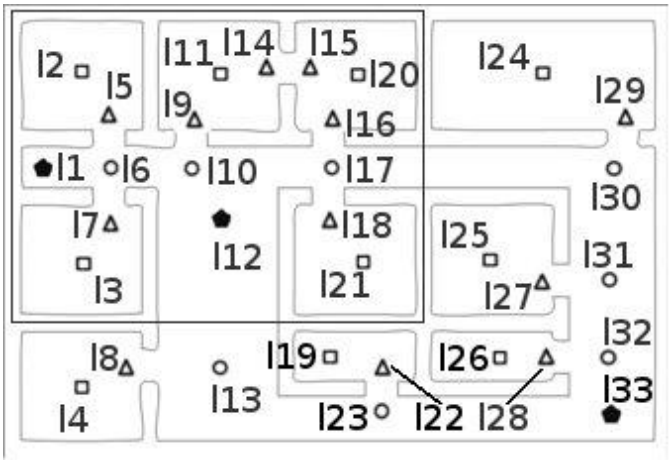


Figura 2 - Mapa do ambiente usado para o novo problema (ambiente 2). Quadrados, triângulos, círculos e pentágonos têm o mesmo significado que o explicado na Fig.1. A região marcada, por ser similar ao ambiente 1, foi utilizada nos experimentos em que a solução de um subproblema já resolvido auxiliava ou atrapalhava o agente na execução de uma nova tarefa.

VI. Experimentos e Análise comparativa.

Para avaliar as políticas abstratas aprendidas, uma série de experimentos foi realizada, usando tanto o ambiente 1, onde as políticas abstratas π_{ND} e π_{Qab} foram geradas, quanto o ambiente 2.

As três políticas utilizadas nos testas são: (i) uma política totalmente aleatória, que serve como referência, (ii) a política abstrata π_{ND} aprendida com um aprendizado supervisionado usando o ND-TILDE (Algoritmo 1) e (iii) a política abstrata π_{Qab} obtida através de aprendizado por reforço utilizando o Qab-Learning (Algoritmo 2). Os testes foram realizados com o objetivo de avaliar o desempenho das políticas na resolução de diversas tarefas.

Um episódio corresponde a iniciar o robô em um local qualquer e aplicar a política que está sendo avaliada até que ele atinja a meta ou que um limite máximo de 1.000 passos seja atingido, a partir do qual se considera que o episódio não teve sucesso. Em todos os experimentos foram executados 10.000 episódios por meta, e a média entre os conjuntos de episódios de cada meta é que é apresentada em diversos gráficos. O eixo das abscissas em cada gráfico representa o número de passos **n**, enquanto o eixo das ordenadas representa a fração acumulada de episódios, dentre os 10.000, nos quais o robô atingiu a meta (em outras palavras, se definirmos que **N** é o número de passos usados para atingir a meta, essa fração representa a distribuição acumulada de N, i.e., P(N ≤ n)). Ou seja, supondo n=100 passos no gráfico da Fig. 3, tem-se que, usando a política aleatória, uma fração de cerca de 60% dos episódios alcançaram a meta em até 100 passos, dentre os episódios que tiveram sucesso (isto é, atingiram a meta em 1.000 passos ou menos) dos 10.000 episódios executados; usando a política π_{ND} e π_{Qab}, as frações foram respectivamente 78% e 80%.

O primeiro experimento consistiu em apenas aplicar as políticas abstratas (e aleatória) no próprio problema para o qual elas foram geradas, ou seja, o mesmo ambiente (ambiente 1) com a mesma meta (alcançar o local *l2*). Isso foi feito para verificar os efeitos e perdas causados pela abstração. Os resultados estão ilustrados na Fig. 3, onde se observa que o Qab-Learning mostrou melhor desempenho nesse caso. Ainda no mesmo ambiente, outro experimento avaliou a execução das políticas na tarefa de, saindo de qualquer local no ambiente 1, atingir um dos centros de cômodo (locais *l2*, *l3*, *l9*, *l17* ou *l18*), apresentando a média alcançada pela execução de todas estas tarefas (cada uma consistindo em 10.000 episódios com no máximo 1000 passos cada). O resultado deste segundo experimento está ilustrado na Fig. 4, onde se verifica que o Qab-Learning e ND-TILDE apresentam resultados similares e melhores do que a política aleatória.

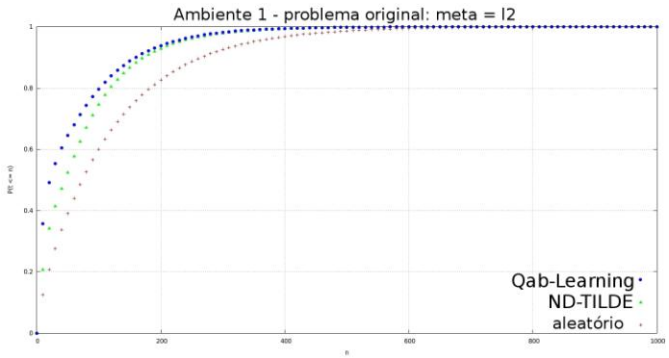


Figura 3 – Resultados da aplicação das políticas abstratas e aleatória no problema original: no ambiente 1, sair de qualquer lugar (exceto o local *l2)* e chegar a *l2*.

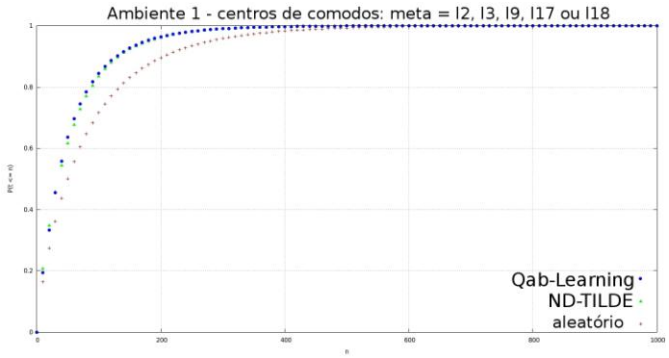


Figura 4 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 1 para a tarefa de, alcançar cada um dos centros de cômodos do ambiente 1 a partir das demais localizações que não a que se pretende alcançar. O gráfico apresenta a média dos resultados obtidos tendo como metas em *l2*, *l3*, *l9*, *l17* e *l18*, em que cada experimento relacionado a cada uma das metas consistiu em 1000 episódios.

Outros três experimentos foram realizados no mapa da Fig. 2 com o objetivo de avaliar o desempenho das políticas abstratas definidas para o ambiente 1. Nesses experimentos,

buscou-se avaliar a capacidade de transferência do conhecimento adquirido em um problema para um novo problema. O experimento cujo resultado está ilustrado na Fig. 5 mostra o desempenho das políticas abstratas e aleatória no ambiente 2, quando a tarefa é atingir *l2* partindo de cada uma das demais localizações. A Fig. 6 mostra o desempenho das políticas em uma tarefa similar, na qual a meta é *l25* a partir das demais localizações do ambiente 2. Já a Fig. 7 mostra o desempenho apresentado nas tarefas de, saindo de qualquer localização a menos da meta, atingir cada um dos centros das salas do ambiente 2.

Analisando as figuras, pode-se observar que, tanto ND-TILDE quanto Qab-Learning geram políticas que apresentam um melhor desempenho que o resultado pelo uso da política aleatória.

O desempenho superior da política π_{Qab} em relação à π_{ND} para a solução do problema que foi utilizado na obtenção de ambas (Fig. 3), sugere que π_{Qab} foi capaz de reter mais elementos necessários à solução do problema original do que $\pi_{ND,}$ isto é, a política π_{Qab} é mais especializada que π_{ND}. Entretanto, quando a tarefa é mais geral (Fig. 4), ou seja, atingir uma localidade que, apesar de possuir a mesma natureza (centro de cômodo), possui uma localização diferente e, consequentemente, uma caracterização topológica dada por sua vizinhança também diferente (Fig. 1), ambas as políticas apresentam um desempenho similar, superando a aleatória.

Esse resultado, aliado ao que foi discutido anteriormente, poderia apoiar uma interpretação positiva em relação à π_{Qab}, sugerindo que essa política, além de reter um relativo bom desempenho na solução do problema que foi treinada para resolver, possui também características que poderiam ser explicadas por uma capacidade de generalização, também apresentada pela π_{ND}. Isso lhes permitiria atingir um bom desempenho em uma tarefa distinta da presente em seus treinos, mas que pertence ao mesmo domínio.

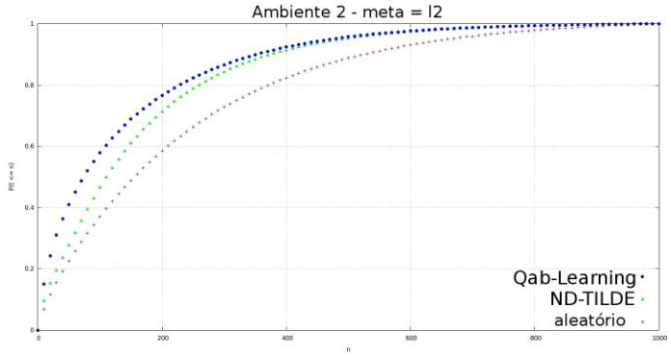


Figura 5 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 2, na tarefa de, saindo de qualquer uma das demais localidades, chegar a *l2*.

Os resultados que compõem o cerne da proposta desse trabalho são ilustrados nas figuras 5, 6 e 7 e se referem à aplicação das políticas aprendidas no ambiente 1 na execução de tarefas no ambiente 2. No experimento cujos resultados são ilustrados na Fig. 5, o agente parte de cada uma das localizações e deve atingir a meta *l2* executando, no máximo, 1000 passos. É interessante notar que a localidade *l2* se encontra em uma região similar ao ambiente 1, correspondendo à meta "antiga". Pode-se ver que o desempenho das políticas é superior ao da aleatória, sendo que π_{Qab}, novamente, apresenta desempenho superior ao de π_{ND}. Esse fato corrobora a hipótese discutida acima de retenção de elementos do problema utilizado na obtenção da política, isto é, de maior especialização de π_{Qab} em relação à $\pi_{ND.}$

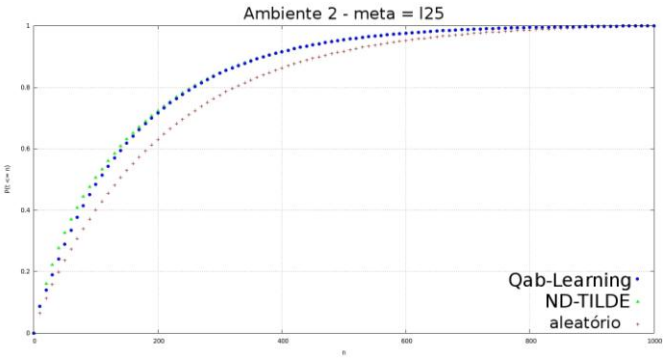


Figura 6 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 2, na tarefa de sair de qualquer lugar e chegar ao local *l25*.

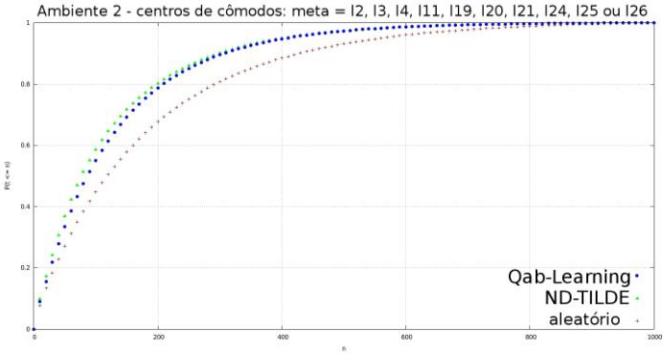


Figura 7 – Resultados da aplicação das políticas abstratas e aleatória no ambiente 2, na tarefa de sair de qualquer lugar e chegar a cada um dos 10 centros dos cômodos. O resultado apresentado é a média dos desempenhos apresentados em cada uma das 10 tarefas.

No experimento cujos resultados são representados na Fig. 6, o agente deveria atingir *l25*, que além de estar fora da região *conhecida* correspondente ao problema original, localiza-se em uma região oposta à *l2*, utilizada como meta na obtenção das políticas. Esperava-se com esse experimento verificar a capacidade de aplicação das políticas quando o problema de navegação consiste em atingir uma localidade que, apesar de possuir a mesma natureza (centro de cômodo), e com isso assemelhar-se à solução do problema original, possuísse uma caracterização topológica que dificultasse a aplicação direta da política na solução do problema (atingir uma meta em região oposta e com vizinhança ainda desconhecida). Nesse experimento, as políticas apresentaram um desempenho similar; entretanto, a π_{ND} foi ligeiramente melhor. Uma possível explicação seria a de que a suposta retenção de elementos do problema original por π_{Qab} seria acompanhada de especialização. Apesar de não ser significativa, pode-se utilizar o fato para analisar as características intrínsecas de obtenção das políticas, além da representação utilizada em cada uma, para que se possa inferir

quais elementos interferem (e com que magnitude) na generalização de uma política concreta obtida na solução de um problema prévio. Uma possível solução para o problema seria desenvolver e aplicar representações que permitissem a generalização da política abstrata para novos problemas, entretanto sem perder a característica desejável de alto desempenho na solução de um problema já resolvido.

Essa análise é respaldada pelos resultados apresentados na Fig. 7, em que o agente deve atingir os centros de cômodos presentes no ambiente 2 a partir de cada uma das demais localidades. A política gerada pelo ND-TILDE apresenta um desempenho ligeiramente superior, e ambas superam a política abstrata na solução do problema.

VII. Conclusões.

Neste artigo, foi proposto o algoritmo Qab-Learning como uma alternativa para a transferência do conhecimento obtido na solução prévia de um problema de navegação e aplicação desse conhecimento na solução de um novo problema de navegação através da geração de uma política abstrata.

A política abstrata gerada pelo Qab-Learning, através de uma modificação do algoritmo Q-learning em que estados e ações abstratos são utilizados no aprendizado, é comparada à política gerada pelo algoritmo ND-TILDE. Ambas as políticas geradas por estes algoritmos possuem um desempenho superior ao de uma política aleatória, i.e., uma escolha aleatória de ações para guiar o robô pelo ambiente. Isso pode ser visto como transferência do aprendizado entre problemas através da utilização das políticas abstratas geradas pelos algoritmos Qab-Learning e ND-TILDE.

Nos experimentos realizados foi possível verificar que o algoritmo proposto, Qab-Learning, criou políticas abstratas que obtiveram desempenho superior ao da política criada com ND-TILDE em determinadas situações, mas que ao mesmo tempo apresentavam desempenho similar ou inferior em outras. Em vista disto, futuramente pretende-se analisar os elementos presentes na obtenção de ambas, buscando verificar quais deles possuem maior e menor impacto na geração de políticas abstratas.

Entretanto, nas situações de superioridade do Qab-Learning, a diferença entre as políticas é notável, principalmente pelo fato do ND-TILDE agregar estados no nível abstrato, no intuito de tornar a política mais geral em detrimento da velocidade, e o Qab-Learning não (utiliza todos os estados abstratos existentes). Certamente, o intuito da construção dessas políticas abstratas não é utilizá-las como únicos guias do agente, mas sim para acelerar o aprendizado de novas tarefas através da transferência do conhecimento prévio. Desse modo, um dos aspectos a se trabalhar no futuro seria a identificação de subtarefas ou partições do espaço de estados nas quais a política abstrata é apropriada para um novo problema. Ou seja, definir quais partes da política apresentam conhecimento relevante e quando aplicá-las.

Além disso, outro aspecto a ser estudado é a definição e avaliação dos predicados que compõem a descrição do domínio. Os predicados possuem papel decisivo na construção da política abstrata e dependendo da forma como são definidos, podem levar à criação de políticas melhores ou piores.

Referências

[1] Dudek, G.; Jenkin, M. *Computational Principles of Mobile Robotics*. New York, Cambridge University Press, 2000.

[2] Choset, H. M; Lynch, K. M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L. E.; Thrun, *S. Principles of robot motion: theory, algorithms, and implementation*. New York, MIT Press, 2005.

[3] Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*. New York, MIT Press, 2005.

[4] Russell, S. J.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, Pearson Education, Inc., 2nd. edition, 2003.

[5] Torrey, L.; Shavlik, J.; Waker, T.; Maclin,R. *Relational macros for transfer in reinforcement learning*. 2008. In: ILP'07 – *Proc. Of the 17th Int. Conf. on Inductive Logic Programming*, 2008.

[6] Madden, M. G.; Howley, T. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, Kluwer Academic Publishers, Norwell, MA, USA, v. 21, p. 375-398, June 2004.

[7] Matos, T.; Bergamo, Y.; Silva, V. F. da; Costa, A. H. R. Stochastic Abstract Policies for Knowledge Transfer in Robotic Navigation Tasks. In: MICAI'2011 – *Mexican International Conference on Artificial Intelligence*, 2011.

[8] Blockeel, H.; Raedt, L.D. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, Volume 101, Issues 1-2, p. 285-297. May 1998.

[9] Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA. 1998.

[10] Kersting,K.; Otterlo,M.V.; and Raedt,L.D. 2004. Bellman goes relational. In *Proc. of 21th Int. Conf. on Machine Learning*, 465–472.

[11] van Otterlo, M. *The Logic of Adaptative Behaviour*, IOS Press, Amsterdam, 2009.

[12] Hoey, J., St-Aubin, R., Hu, A.J., Boutilier, C.: Spudd: Stochastic planning using decision diagrams. In: UAI'99 – *Proc. of Uncertainty in Artificial Intelligence*, Stockholm, Sweden. 1999.

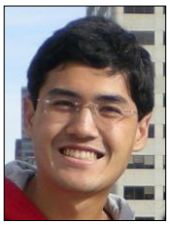
Agradecimentos

Esta pesquisa obteve o apoio da FAPESP (08/03995-5, 09/04489-9, 09/14650-1) e do CNPq (305512/2008-0).

Rafael agradece a Guilherme Renoldi pelo auxílio nas etapas de depuração do simulador e pelas ideias surgidas durante as construtivas discussões.

Rafael Lemes Beirigo é aluno do Curso de Ciências Moleculares, da Pró-Reitoria de Graduação da Universidade de São Paulo (CM-USP) e desenvolve sua pesquisa como aluno de Iniciação Científica no Laboratório de Técnicas Inteligentes (LTI), da Escola Politécnica (POLI-USP). Seus interesses em pesquisa em Inteligência Artificial recaem particularmente sobre o Aprendizado de Máquinas, especificamente Mapas Semânticos e Políticas Abstratas, considerando os processos de abstração de instâncias de problemas de navegação e os elementos envolvidos.

Fernando de Andrade Pereira é aluno do curso de Engenharia Elétrica com ênfase em Computação na Escola Politécnica da Universidade de São Paulo (EPUSP). Seus interesses na área de Inteligência Artificial voltam-se para os ramos do Aprendizado por Reforço e Políticas Abstratas.



Marcelo Li Koga graduou-se em Engenharia Elétrica com ênfase em Computação (2010) e atualmente é estudante de Mestrado em Inteligência Artificial, ambos pela Escola Politécnica da Universidade de São Paulo (EPUSP). Em 2008 atuou como representante discente do PCS (Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP) e em 2011 apresentou trabalho sobre inferência de gramáticas usando adaptatividade no WTA (Workshop de Tecnologias Adaptativas). Atualmente seus interesses de pesquisa se concentram nas áreas de Aprendizado por Reforço e Planejamento Probabilístico.

Tiago Matos possui mestrado em Engenharia Elétrica pela Universidade de São Paulo (2011) e é formado em Engenharia Elétrica - Ênfase Computação pela Escola Politécnica da Universidade de São Paulo (2008). Seus interesses em pesquisa se concentram nas áreas de Aprendizado por Reforço e Reaproveitamento do Conhecimento.

Valdinei Freire da Silva é Professor Doutor do curso de Sistemas de Informação da Escola de Artes, Ciências e Humanidades da Universidade de São Paulo (USP, 2011). Possui Doutorado em Engenharia Elétrica (USP em co-tutela com IST-UTL, 2009) e é formado em Engenharia da Computação (USP, 2002). Seus interesses em pesquisa abrangem vários aspectos de Aprendizado de Máquina e Tomadas de Decisões Sequenciais em ambientes probabilísticos.

Anna Helena Reali Costa é Professora Titular do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo (USP, 2011). Possui Doutorado em Engenharia Elétrica (USP, 1994) e Mestrado em Engenharia (FEI, 1989). É formada em Engenharia Elétrica (FEI, 1983). De 1985 a 1988 e de 1991 a 1992 foi pesquisadora na Universidade de Karlsruhe, Alemanha. Em 1998 e 1999 foi pesquisadora visitante na Carnegie Mellon University, nos Estados Unidos. Seus interesses em pesquisa abrangem um grande espectro da Inteligência Artificial, incluindo particularmente o Aprendizado de Máquinas, Robótica Inteligente e Visão Computacional.

Algoritmos Adaptativos em Detecção de Colisão em Aplicações para a Área de Saúde

A. Zotovici, R. Nakamura

***Abstract*— We present the results of a systematic research about the application of adaptive techniques in collision detection, especially in virtual reality systems. The research showed us that some algorithms have a better performance due to the adaptive technique applied. The aim is apply the adaptive technology in collision detection to improve performance and precision in simulation systems to the health area.**

***Keywords*— Adaptive Techiques, Collision Detection, Virtual Reality.**

I. INTRODUÇÃO

ATUALMENTE, muitas aplicações utilizam a realidade virtual para simulação de elementos com os quais seria difícil ou impossível de interagir. Os elementos podem ser aqueles que existem no mundo real ou fictícios. Há muitas oportunidades de aplicação dessa tecnologia no ensino de diversas disciplinas.

Conforme [1], no ensino de Medicina, a simulação de procedimentos como cirurgia endovascular, que utiliza cadáveres humanos, proporciona alto grau de realidade. Porém não é um processo validado e usa apenas uma vez o material endovascular, o que gera alto custo.

Por outro lado, se é utilizada a realidade virtual, é possível proporcionar alto grau de realismo, simula-se a mudança das especificações do paciente e não há limitação de re-uso do material [1].

No ensino de biologia, os alunos somente podem visualizar células e micro-organismos em laboratório, com auxílio de equipamentos especializados. Um ambiente de realidade virtual poderia ser utilizado para simular essas células e micro-organismos, ampliando-os para que possam ser visualizados sem auxílio dos equipamentos especializados do laboratório e com mais detalhes que as imagens bidimensionais usadas nos livros convencionais.

No entanto, os alunos não aprendem apenas visualizando, é importante interagir com o modelo e, ao explorá-lo, é importante o realismo na reação ao estímulo. Para essa interatividade, faz-se necessária a detecção de colisão, ou seja, o contato entre o objeto de interesse e o usuário (seja através de técnicas de manipulação direta ou indireta).

Para a detecção de colisão de objetos sólidos e não deformáveis, já existem muitas técnicas que aumentam a eficiência da detecção e reduzem o tempo de processamento. Porém, essas técnicas não são eficientes em tempo real quando aplicadas em objetos deformáveis devido ao processo de reestruturação[2].

A revisão sistemática foi realizada com o objetivo de identificar as técnicas de detecção de colisão que utilizam a tecnologia adaptativa sem a restrição de serem específicas para objetos deformáveis. Após a identificação dessas técnicas, será estudado o seu algoritmo para futura aplicação em objetos deformáveis de sistemas de simulação educacional para a área de saúde.

As próximas seções deste artigo estarão organizadas da seguinte maneira: na seção 2 serão discutidos conceitos básicos sobre detecção de colisão, os quais são abordados em trabalhos que utilizam técnicas adaptativas. Na seção 3 serão discutidas as técnicas de detecção de colisão que utilizam adaptatividade. Na seção 4 são discutidas aplicações que utilizam técnicas adaptativas de detecção de colisão em sistemas da área de saúde. Por fim, na seção 5 estarão as considerações finais.

II. DETECÇÃO DE COLISÃO

Esta seção discute algoritmos básicos de detecção de colisão com o objetivo de apresentar seu funcionamento, já que são empregados nas técnicas mais elaboradas de detecção de colisão.

Por motivos de otimização de desempenho quanto ao tempo de execução, o processo de detecção de colisão pode ser dividido em duas fases *broad* e *narrow*. Na primeira fase, denominada *broad*, são identificados os objetos próximos, candidatos a colisão, por meio do teste de intersecção de seus volumes envoltórios. Esses objetos são enviados para a próxima fase, denominada *narrow*, para que seja aplicado um algoritmo mais detalhado para detecção de colisão.

Entre os algoritmos de volumes envoltórios estão incluídos o uso de caixas alinhadas aos eixos, caixas orientadas, esferas e politopos, detalhados a seguir.

No caso de caixas alinhadas ao eixos (Axis Aligned Bounding Box-AABB), os objetos candidatos a colisão são envolvidos por uma caixa na qual a normal de cada face é paralela aos eixos do sistema de coordenadas, como ilustra a Figura 1. Durante a detecção de colisão pode ocorrer a intersecção das caixas sem que os objetos tenham colidido porque esse volume envoltório não fica totalmente ajustado. A vantagem desse volume envoltório é que exige baixo consumo de memória e processamento rápido. Ele pode ser utilizado para verificar objetos que são candidatos à colisão.

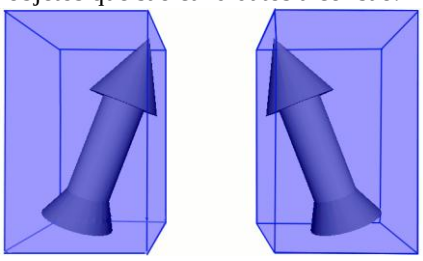


Figura 1 Caixas Alinhadas

Na abordagem utilizando caixas orientadas (Oriented Bounding Box-OBB), o volume envoltório possui a orientação e inclinação de acordo com o objeto que envolve, conforme ilustra a Figura 2. Essa transformação possibilita melhorar a precisão na identificação da intersecção. Porém ainda não é tão justo.

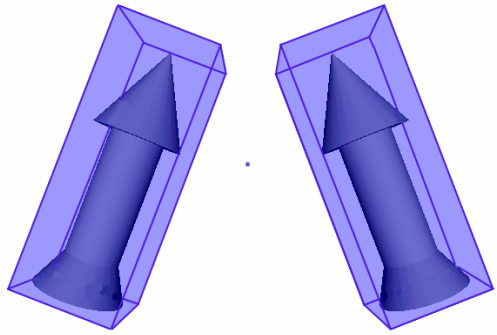


Figura 2 Caixas orientadas

No caso do uso de esferas, os objetos candidatos a colisão são envolvidos por uma esfera, conforme ilustra a Figura 3. Nesse algoritmo também pode ocorrer a intersecção do volume envoltório sem que os objetos tenham colidido. A vantagem do uso desse algoritmo é seu desempenho e pode se usado para identificar os objetos candidatos à colisão.

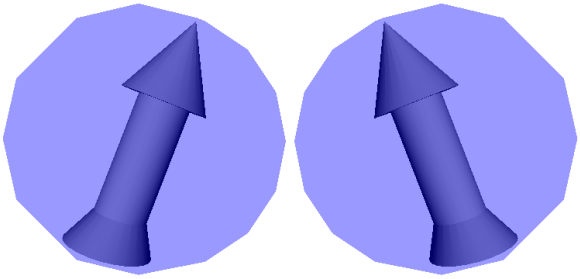


Figura 3 Esferas

Politopos com k orientações discretas(DOP-*Discrete Oriented Polytope*), são objetos convexos definidos por um pequeno conjunto fixo de k direções (D_1, ..., D_k) e uma tupla (d_1,...,d_k) ϵ R^k de escalares[3]. Como por exemplo, para 6DOP há seis direções: D1=(-1,0,0), D2=(0,-1,0), D3=(0,0,-1), D4=(1,0,0), D5=(0,1,0) e D6=(0,0,-1)[3].

Os politopos podem criar um volume envoltório mais justo ao objeto, como ilustrado pela Figura 4. Mas exige mais memória e processamento que os volumes envoltórios discutidos no início da seção.

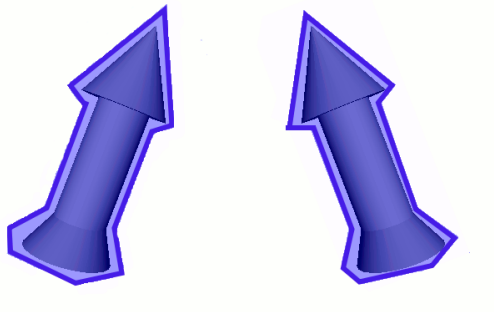


Figura 4 Modelo de Invólucro baseado em Politopo

Outra abordagem para detecção de colisão é a partição hierárquica do espaço que pode ser realizada por meio de uma *octree*.

O volume do espaço de um ambiente virtual tridimensional é dividido gerando uma árvore na qual cada nó pai tem oito nós filhos, o que gera uma divisão similar à Figura 5. Cada nó dessa árvore corresponde a uma partição do ambiente[4].

A divisão ocorre recursivamente até que um critério de parada seja atendido. O critério pode ser um grau máximo para a árvore ou tamanho mínimo para os cubos[4].

Esse processo possibilita mais precisão na detecção da colisão.

Para verificar a colisão, à medida que um objeto se movimenta no espaço, verifica-se a colisão quando há mais de um objeto no mesmo nó[4].

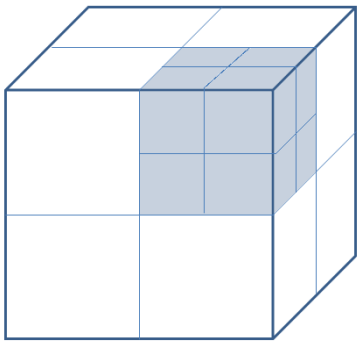


Figura 5 Partição de um cubo

III. Metodologia de Revisão

A metodologia de revisão adotada foi a revisão sistemática que, segundo [5], é uma metodologia rigorosa de pesquisa bibliográfica que difere da revisão de literatura comum pela definição do protocolo de pesquisa. O protocolo de pesquisa de pesquisa documenta critérios definidos durante a fase de planejamento: objetivos da revisão, questão de pesquisa, estratégia de pesquisa e recursos, idioma, palavras-chaves de busca, critério de inclusão de documento para estudo, critério de exclusão de documento, método de seleção de estudos.

Na fase seguinte, a condução da pesquisa, são registrados: a data de busca, a fonte, a palavra-chave utilizada, o título e autores de cada documento encontrado. Os documentos encontrados que atendem os critérios de inclusão são separados para a próxima fase, a análise de resultados.

Durante a análise de resultados, os dados do documento são registrados, realiza-se a síntese dos resultados e a conclusão sobre a pesquisa.

A revisão sistemática foi utilizada na pesquisa sobre técnicas e sistemas discutidos nas seções IV. .

O objetivo dessa revisão foi realizar o levantamento da aplicação de técnicas adaptativas em detecção de colisão para sistemas de realidade virtual.

A questão desta revisão foi "Como são aplicadas as técnicas adaptativas em detecção de colisão nos sistemas de realidade virtual?".

A estratégia para a pesquisa foi busca de artigos que discutam o estado da arte em técnicas adaptativas para detecção de colisão em ambientes que utilizam a realidade

virtual. Os recursos utilizados para a busca foram o Google Acadêmico, publicações do IEEE e ACM.

O idioma para as palavras-chave da busca foi o inglês. Esse é o idioma das publicações do IEEE e ACM por isso foi o escolhido.

As palavras-chave utilizadas foram: "Adaptive collision detection" "virtual reality".

A chave de pesquisa no Google Acadêmico foi "*adaptive collision detection*" com oito resultados, mas apenas quatro atendiam os critérios de inclusão. Um item do resultado foi uma tese, a partir da qual foram encontradas citações de artigo do mesmo autor que contribuíram no entendimento de seu trabalho. No IEEE, a chave de pesquisa foi ((*adaptive collision detection*) AND *virtual reality*) que retornou quatro artigos, dos quais apenas dois foram selecionados. E na ACM, a chave de pesquisa foi "*adaptive collision detection*" and "*virtual reality*", para a qual não foi obtido resultado.

O critério de inclusão foi escolher artigos e livros que abordam algoritmos e exemplos de aplicação de técnicas adaptativas para detecção de colisão em realidade virtual. As publicações deveriam ter tempo inferior a cinco anos ou que discutisse conceitos fundamentais para detecção de colisão ou técnicas adaptativas que os artigos mais atuais não discutiram.

O critério de exclusão foi dispensar publicações com tempo superior a cinco anos, que não seja conceito fundamental para detecção de colisão e técnicas adaptativas, publicação não acadêmica, artigos em que a técnica adaptativa não é discutida por meio de algoritmos ou exemplos de sua aplicação.

A seleção de estudos foi baseada na seleção preliminar realizada por meio da leitura do *abstract* do artigo para identificar se atende aos critérios de inclusão, caso não fique explícito, o artigo será incluído. E a seleção final foi realizada por meio da leitura completa do artigo e o que possibilitou a definição sobre sua inclusão.

Durante a extração de dados, para cada artigo, foram documentados: dados para a referência do mesmo, objetivo do trabalho, descrição da técnica adaptativa aplicada à detecção de colisão, comentários relacionados à precisão e tempo de detecção de colisão, e conclusões.

A síntese de dados foi baseada na análise a aplicação de técnicas adaptativas em detecção de colisão e comparação das abordagens o que foi relatado na seção VI. .

IV. Adaptatividade em Detecção de Colisão

Esta seção discute algoritmos de detecção de colisão que utilizam técnicas adaptativas com o objetivo de analisar como essas técnicas são aplicadas, isto é, se são aplicadas para adaptar o processo ou na reconstrução da estrutura de dados após alguma deformação.

A adaptatividade proposta por [6] é aplicada no *pipeline* de detecção de colisão paralelo executado em arquitetura multi-core. Essa arquitetura permite dinamicamente adaptar a repartição de paralelismo, eliminando a seqüencialidade tradicional das duas principais fases (*broad* e *narrow*), conforme ilustra o processo paralelo da Figura 6. As fases são definidas como *threads* cuja configuração de repartição pode variar de acordo com a necessidade da aplicação.

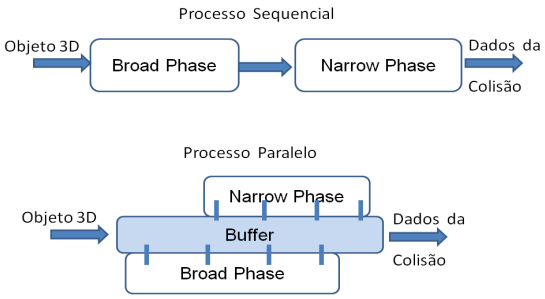


Figura 6 Processo Sequencial X Paralelo[6]

Dois exemplos de configuração de repartição para aplicações diferentes:

- configuração 4-4: quatro *threads* da *Broad* e quatro da *Narrow*;
- configuração 5-1: cinco *threads* da *Broad* e uma da *Narrow*.

No processo paralelo, aplica-se uma estrutura de *buffer* para compartilhar pares de objetos entre threads. Quando a *thread* da fase *Broad* identifica potencial colisão entre um par de objetos, por meio do algoritmo *Sweep and Prune*[1], envia-os para o *buffer*. A *thread* da fase *Narrow* recebe esse par e, de acordo com a forma geométrica, utiliza o algoritmo apropriado para verificar se há colisão. Como por exemplo, o GJK(Gilbert-Johnson-Keerth) para os objetos complexos e o AABB para os objetos não convexos.[6]

Para distribuir as *threads* nas fases do *pipeline* e otimizar a performance do multi-core, [6] propõe uma técnica de balanceamento dinâmico de carga. No início da aplicação, por um curto período, é analisado o comportamento e a performance de cada fase. A configuração das *threads* varia de acordo com a demanda de tempo de cada fase.

A detecção de colisão para malhas de polígonos deformáveis[7]e[8] em simulações interativas, é outro algoritmo adaptativo. O algoritmo está baseado em adaptatividade de hierarquias de volumes envoltórios (*bounding volume hierarchies*-BVH).

Os tipos de modelos considerados nos algoritmos são: malhas de polígonos que são deformadas por reposicionamento arbitrário de vértices, modelos deformados por *morphing* linear com um número fixo de malhas referência e modelos de movimento completamente desestruturado relativo entre as primitivas geométricas[8].

O algoritmo proposto possui duas fases[7]:

1ª. *Broad* é a fase na qual se utiliza o método *Sweep and Prune* para identificar corpos que estão próximos;

2ª. *Narrow* é a fase que utiliza árvore de volume envoltório para determinar a intersecção entre os corpos.

[1] *Sweep and Prune* ou *Sort and Sweep* é um algoritmo que mantém os objetos 3D em alguma ordenação espacial, preferencialmente, em uma lista ligada duplamente encadeada, o que possibilita a atualização da lista com complexidade O(n).[3]

A árvore de volume envoltório é uma árvore binária construída em um estágio de pré-processamento, por repetidas divisões da geometria[7].

Para a construção da árvore, são utilizadas duas estratégias.

A primeira estratégia foi baseada na divisão da forma inicial do corpo, que depende do grau máximo de um nó de árvore, para que se defina a divisão de um nó pai. A divisão pode ocorrer ao longo de um, dois ou três eixos de coordenadas; em dois, quatro ou oito sub-volumes. O ponto médio de cada primitivo geométrico é atribuído a um desses volumes e um nó é criado para cada sub-volume não vazio. Para árvore de grau dois, o pai é dividido ao longo de seu lado mais longo. Se é de grau quatro, a divisão é feita ao longo dos dois lados mais longos. E se é de grau oito, todos os três lados dos pais AABB são separados.[7]

A segunda estratégia calcula a média pelo ponto médio de todos os polígonos e os valores para a divisão do plano são escolhidos a partir desse ponto. [7]

Durante a execução, podem ocorrer mudanças na geometria dos objetos, ou novos objetos podem ser adicionados, o que gera atualização da hierarquia.

A técnica de detecção de colisão de objetos proposta por [9] é destinada a ambientes distribuídos de realidade virtual porque esses ambientes possuem a interferência da latência da rede.

Segundo [9], a posição e orientação de objetos nas máquinas conectadas à rede são enviadas por meio de mensagem ao servidor. O problema é que devido à latência, a mensagem somente chega ao servidor depois de algum tempo que foi enviada, assim esses dados já podem estar desatualizados. A predição adaptativa de colisão é uma técnica de detecção de colisão que identifica objetos em movimento com potencial colisão, calculando o tempo de colisão em função da distância e velocidade dos mesmos. Dois objetos estarão próximos se as esferas que os envolvem estão intersectadas ou se elas irão se intersectar em um certo tempo Δ_T[2].

A abordagem do artigo procura compensar as baixas taxas de atualização. No algoritmo, calcula-se o tempo de colisão para dois objetos em movimento. Se este tempo está abaixo de um limiar, os parâmetros para o protocolo de estimativa da posição atual são atualizados para permitir maior taxa de atualização, assim o limiar baseado em erro é reduzido ao menor valor.

Para dois objetos, verifica-se se suas esferas limitantes se intersectam no momento t=0. Se a verificação for verdadeira, nenhum outro cálculo é realizado, a intersecção foi identificada. Caso as esferas limitantes ainda não tenham se intersectado, sua posição atual e velocidade são usadas para calcular o momento em que as esferas limitantes irão se intersectar.

Quando ocorre a colisão, é possível que dois computadores da rede tenham resultados diferentes. Essa diferença pode ocorrer porque o erro de aproximação pode ser maior para objetos com maior velocidade. Assim, é necessário um processo, denominado resolução de decisão, para escolher qual computador é o responsável pela detecção de colisão. Um meio de tomar essa decisão é utilizar um número único de identificação para o computador, o computador com o menor número é o responsável por detectar as colisões e distribuir os resultados. Outro critério para definir qual computador é o responsável por detectar a colisão, é escolher aquele onde está o objeto com maior velocidade. E também, pode-se decidir com base no campo de visão para evitar inconsistências visuais.

Outra técnica que aplica a adaptatividade é LBG que possibilita a divisão da superfície de objetos 3D em sub-áreas, e em cada uma há uma partícula. As partículas de um objeto 3D são classificadas em relação às partículas de outro objeto 3D candidato a colidir. Essa classificação é realizada pelo cálculo da força de atração entre as partículas, como ilustra a Figura 7.[2]

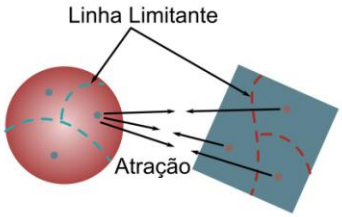


Figura 7 Interação de Força em uma Partícula baseado em [2]

O processo de partição do LBG consiste em determinar a quantidade de partículas usando a quantidade de objetos candidatos à colisão com este. Particionar a superfície, subdividindo-a em áreas, como ilustra a Figura 8. E em seguida, realizar a detecção de colisão, dividida em três partes:

a) Cálculo da força de interação é o processo no qual a força aplicada a cada partícula é calculada de acordo com a distância entre as partículas. A relação da força de interação é que, quanto menor a distância, maior será força de atração.
b) Movimento das partículas é o processo em que cada partícula movimenta-se apenas ao longo dos vértices na direção do vértice vizinho de onde recebeu força máxima de atração.
c) Distância da colisão consiste em encontrar o par de partículas mais próximas.

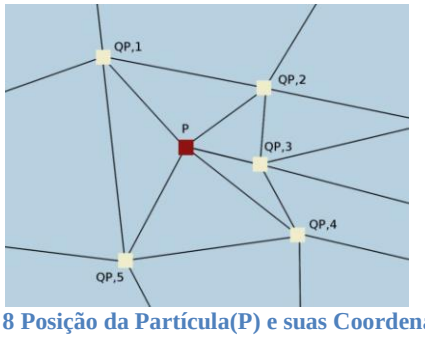


Figura 8 Posição da Partícula(P) e suas Coordenadas Vizinhas(QP) baseada em [2]

Quando ocorre grande deformação de um objeto, é realizado o reparticionamento, de modo que uma partícula não seja o suficiente para detectar a colisão em determinada área[2].

[2] Δ_T é o limiar de proximidade.

V. Sistemas que utilizam a Adaptatividade para Detecção de colisão

Esta seção apresenta dois sistemas[10] que foram encontrados durante as buscas. Os dois utilizam como estrutura básica a *Octree*.

O sistema apresentado por [10] usa modelos surfels[3], ilustrado pela Figura 9, para renderizar as ferramentas do dentista e modelos de dentes virtuais para obter melhor qualidade visual com menos custo de memória.

O modelo baseado em surfels, utiliza a operação booleana para determinar quais pontos foram adicionados ou removidos do modelo. Esse sistema possui dois modos de uso. Um possibilita a escultura do modelo, o que deforma e gera sensação de força. Enquanto que o outro, não possibilita a deformação do modelo, e também não fornece a sensação de força.

Durante a interação, quando ocorre a colisão da broca com o dente, os pontos são removidos do modelo e a estrutura de dados atualizada. Como o modelo está representado por uma *Octree*, apenas o nó referente à parte removida é atualizado.

Figura 9 Modelos usando Surfels baseado em [10]

A detecção de colisão utiliza tecnologia adaptativa porque durante a busca por colisão na *Octree*, o nível dessa estrutura hierárquica será calculado de acordo com a velocidade do dispositivo *haptic* que representa a broca. Se é movimentada lentamente, a colisão é calculada com o nível máximo da *Octree*. Se é movimentada rapidamente, será calculada com o nível dois.

A abordagem de detecção de colisão, proposta por [11], é para simulação de cirurgia intestinal. Nessa abordagem, pares de pontos próximos são rastreados o tempo todo. Durante este processo há partes do mesmo objeto, com estrutura tubular ou de membrana, colidindo devido ao formato do órgão. Para identificar essas colisões, utiliza o método de amostragem adaptativa para detecção da curvatura do intestino.

Para calcular a detecção da curvatura, encontra-se a curva *spline* por uma aproximação interpolando pontos de amostragem mecânica do intestino. Segundo [11], há dois passos para o cálculo da detecção: um para as extremidades do segmento e outro para as regiões de alta curvatura.

VI. Considerações Finais

Após a síntese dos dados obtidos com a revisão sistemática, foi possível verificar a aplicação da adaptatividade nas técnicas discutidas:

- configuração da repartição de acordo com a demanda nas fases *Broad* e *Narrow*[6];
- reconstrução da estrutura de dados depois da deformação do objeto 3D[7];
- repartição do objeto 3D após grande deformação[2];
- predição adaptativa de colisão[9] foi incluída no trabalho devido a sua abordagem para sistemas em rede;
- reação a um estímulo de colisão, como o da broca com o dente, reorganizando o modelo de acordo com a velocidade desse aparelho[10];
- detecção de colisão com diferentes partes o mesmo objeto[11].

A próxima etapa na pesquisa é testar e comparar o desempenho e precisão dos algoritmos que utilizam a tecnologia adaptativa para aplicação futura em projetos na área de simulação para ensino na área de saúde e biológicas.

[3] Surfel é a combinação de duas palavras *surface* (superfície) e *element* (elemento) .[10]

Referências

[1] R. M. S. Almeida. O cirurgião cardiovascular como intervencionista. Revista Brasileira de Cirurgia Vascular, 24(2), Suplemento: 35s-37s, 2009.

[2] N. Saenghaengtham, P. Kanongchaiyos. Using LBG quantization for particle-based collision detection algorithm. *Journal of Zhejiang University SCIENCE A*, 7(7):p.1225-1232, CN, 2006.

[3] C. Fünfzig, D.W.Fellner. Easy Realiggnment of k-DOP Bounding Volumes. Proceedings 9th Symp. Virtual Reality Software and Technology (VRST'02), pp.121-128, 2002.

[4] C. Ericson. Real-time collision detection. Morgan Kaufmann, 2005.

[5] V. M. Gonçalves, F. L. S. Nunes, M. E. Delamaro. Avaliações de Funções de Similaridade em Sistemas de CBIR: Uma Revisão Sistemática. *6º.Workshop de Visão Computacional*, pp.199-204,2010.

[6] Q. Avril, V. Gouranton and B. Arnaldi, Synchronization-free parallel collision detection pipeline. *20th International Conference on Artificial Reality and Telexistence*, p.22-28, Australia, 2010.

[7] T. Larsson, T. Akenine-Möller, Collision Detection for Continuously Deforming Bodies. *Eurographics*, p.325-333, 2001.

[8] T. Larsson, *Adaptive Bounding Volume Hierarchies for Efficient Collision Queries*. Suécia, Mälardalen University, 2009. 81p. Tese de Doutorado.

[9] J. Ohlenburg. "Improving collision detection in distribuited virtual environment by adaptive collision prediction tracking". *Proceedings of IEEE VR*, pp.83-90, Mar.2004.

[10] H.T Yau, C.Y. Hsu. "Development of a Dental Training System Based on Point-Based Models". *Comput-Aided Design Applications*, pp779-787, 2006.

[11] L. Raghupathi, L. Grisoni, F. Faure, D. Marchal, M. P. Cani e C. Chaillou. "Na Intestinal Surgery Simulator: Real-Time Collision Processing e Visualization". IEEE Transactions on Visualization and Computer Graphics, pp.708-719, vol.10, no.6, 2004.

Andréa Zotovici é Bacharel em Ciência da Computação na Universidade São Judas Tadeu, São Paulo, São Paulo, Brasil, em 1995. Obteve o título de mestre em Engenharia Elétrica pela Universidade de São Paulo (USP), São Paulo, Brasil, em 2003. Atualmente é professora da Universidade São Judas Tadeu (USJT), Fundação Santo André e FATEC. Pesquisadora no Laboratório de Tecnologias Interativas (Interlab) da Escola Politécnica da USP.

Ricardo Nakamura possui graduação em Engenharia Mecânica - Automação e Sistemas pela Universidade de São Paulo (USP), mestrado e doutorado em Engenharia Elétrica também pela USP. Atualmente é Professor Doutor da Escola Politécnica da USP, no departamento de Engenharia de Computação e Sistemas Digitais. Suas principais áreas de interesse são jogo digitais, realidade aumentada e interação humano-computador.

Linguístico: Uma Proposta de Reconhecedor Gramatical Usando Tecnologia Adaptativa

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 Livres de Contexto

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.

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 simples nem tampouco 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

Considerando o processamento da direita para esquerda e de baixo para cima, a frase seria analisada da seguinda forma:

O menino usa o chapéu
1. chapéu = Subst: O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo: O menino verbo SN
5. verbo SN=SV: O menino SV
6. menino = Subst: O subst SV
7. O = Det: Det subst SV
8. Det subst= SN: SN SV
9. SN SV= F: F (Aceita)

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. Por exemplo:

O meninos usa o chapéu
1. chapéu = Subst: O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo: O menino verbo SN
5. verbo SN=SV: O menino SV
6. meninos = Subst: O subst SV
7. O = Det: Det subst SV
8. Det subst= SN: SN SV
9. SN SV= F: F (Aceita)

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. Aplicando este formalismo ao exemplo acima, o erro de concordância seria identificado e a frase não seria aceita:

O meninos usa o chapéu

1. chapéu = Subst masc sing : O menino usa o subst
2. O = Det : O menino usa det subst
3. Det Subst = SN:O menino usa SN
4. usa = verbo pres 3a. Pessoa sing: O menino verbo SN
5. verbo SN=SVsing: O menino SVsing
6. meninos = Subst masc plural: O subst SVsing
7. O = Det: Det subst SVsing
8. Det subst= SNplur: SNplur SVsing
9. SNplur SVsing = Não aceita

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 trabalhos de pesquisas em processamento de linguagem natural para a Língua Portuguesa tais como o desenvolvido pelo 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.

Reconhecedor adaptativo: Suporte teórico Linguístico

A Moderna Gramática Brasileira de Celso Luft [12] foi escolhida como suporte teórico linguístico do reconhecedor aqui proposto. A escolha foi feita em função da forma clara e precisa com que Luft categoriza 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

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.

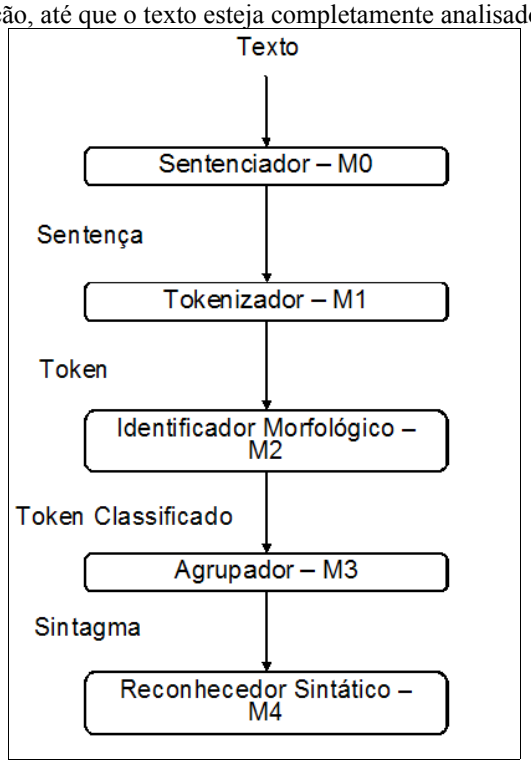


Figura 2. Estrutura do Linguístico

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, palavras abreviadas e palavras compostas, e eliminando aspas simples e duplas. Ao final, o Sentenciador divide o texto em supostas sentenças, para análise individual 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, horas e minutos, numerais arábicos e romanos, palavras compostas, 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, como biblioteca de apoio, os textos pré-anotados do corpus Bosque[13], os verbos, substantivos e adjetivos que fazem parte da base de dados do TeP2.0 – Thesauro Eletrônico para o Português do Brasil [14] e as conjunções, preposições e pronomes disponíveis no Portal

São Francisco[15], cujas informações provém da Wikipedia [16]. 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.

```
#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 TeP2.0 é um um dicionário eletrônico de sinônimos e antônimos para o português do Brasil, que armazena conjuntos de formas léxicas sinônimas e antônimas. É composto por 19.888 conjuntos de sinônimos, 44.678 unidades lexicais, e 4.276 relações de antonímia [14].

O Portal São Francisco apresenta um curso online da Lingua Portuguesa e, entre seus módulos, encontra-se um sumário das classes morfológicas, no qual são encontrados exemplos de palavras e locuções mais comuns de cada classe [15].

Inicialmente, o Identificador Morfológico procura pela classificação morfologica dos tokens no léxico do Bosque, caso não a encontre, então ele procura na base de dados do TeP2.0 e no léxico do Portal São Francisco.

O padrão de etiquetas usado pelo Linguístico é o mesmo do Bosque, que apresenta, além da classificação morfológica, o papel sintático que o token exerce na sentença pré-anotada. Por exemplo, a etiqueta >N+art indica que o token é um artigo que está à esquerda de um substantivo (>N). A notação usa como gramática subjacente a Gramática Constritiva proposta por Fred Karlsson [17].

O léxico do Bosque foi organizado de modo a relacionar todas as classificações de um tokens ordenadas por frequencia em que ocorrem no texto pre-anotado. Por exemplo, o token "acentuada" está classificado com as seguintes etiquetas: N<+v-pcp e P+v-pcp, o que significa que, no texto pré-anotado, ele foi classificado como verbo no particípio (+v-pcp) antecedido de um substantivo (N<) e como verbo no particípio no papel de predicador (P).

No caso de ambiguidade, o Identificador Morfológico assume a classificação mais frequente como inicial e verifica se a classificação mais frequente do token seguinte é consistente com o que indica a etiqueta do token analisado. Se for, vale a classificação mais frequente, senão o Identificador analisa a próxima classificação, repetindo o algoritmo. Caso o algoritmo não retorne uma classificação única, o Identificador passa todas as classificações encontradas para os módulos seguintes, para que ambiguidade seja resolvida pelas regras gramaticais do reconhecedor.

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 (Fig.4). 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}

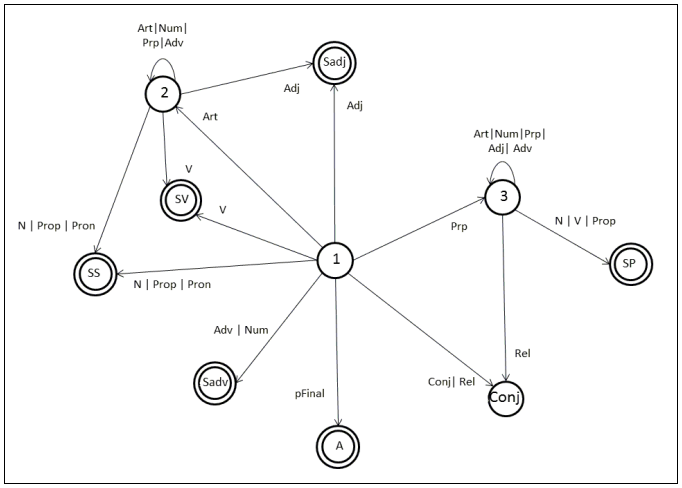


Figura 4. Configuração Completa do Autômato Construtor de Sintagmas.

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]}

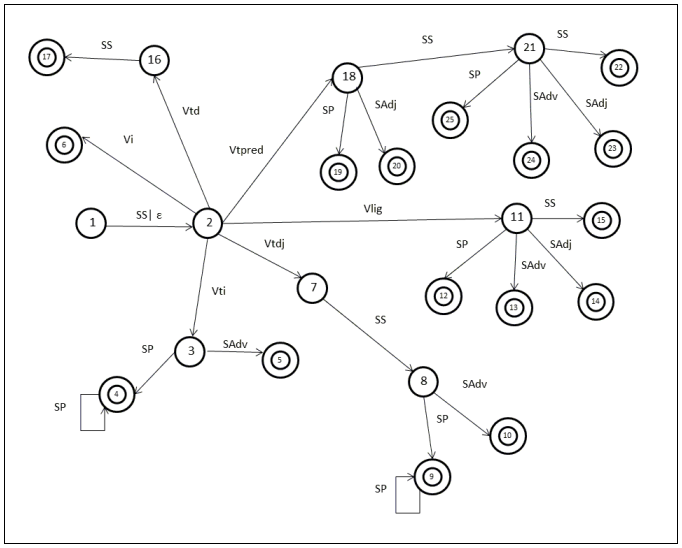


Figura 5.1. Configuração Completa do Reconhecedor Sintático.

No entanto, para que a análise sintática seja feita, não são necessárias todas as ramificações da configuração completa do autômato (Fig. 5.1). 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.

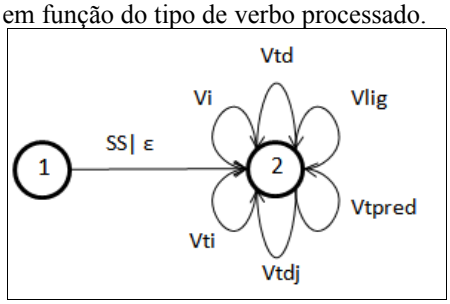


Figura 5.2. Configuração Inicial do Reconhecedor Gramatical.

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.

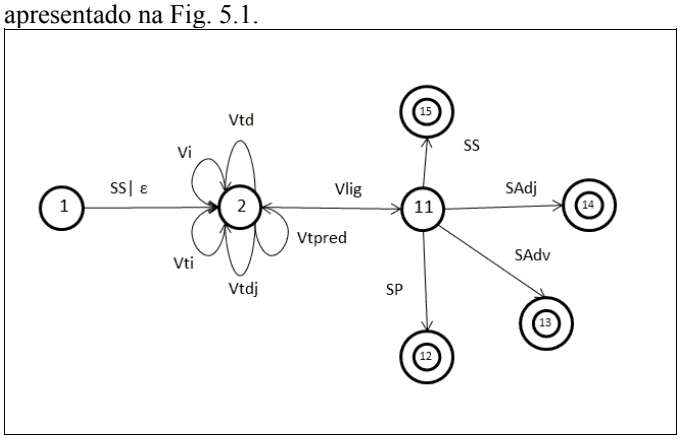


Figura 5.3. Configuração do autômato após o processamento do verbo de ligação.

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.

Desenvolvimento do Linguistico – Preparação do Texto

O Línguístico está sendo desenvolvido na linguagem de programação Python[18], escolhida devido aos recursos nativos de processamento de expressões regulares, programação dinâmica e por oferecer flexibilidade de implementação tanto no paradigma procedural quanto na orientação a objetos. O primeiro componente desenvolvido, chamado Texto, cuida da preparação do texto que vai ser analisado. Ele é formado por uma classe chamada Texto que incorpora o Sentenciador e o Tokenizador do Linguístico. A classe Texto possui 3 métodos que são acionados em sequência, sempre que encontra um novo texto para ser analisado. O método Prepara_Texto se encarrega de eliminar aspas simples e duplas, e identificar abreviaturas e palavras compostas na base de dados formada pelos léxicos Bosque e TeP2.0. O método Divide_Texto cria uma coleção de sentenças através de expressões regulares que interpretam como caracteres limitadores de cada sentença o ponto final, de interrogação e de exclamação, seguidos de um espaço em branco. Ao final, o método Tokeniza_Sentenca cria uma coleção de tokens a partir das sentenças, usando como critérios de divisão, regras para identificação de tokens alfanumericos, valores monetários, horas e minutos, numerais arábicos e romanos, percentuais, além das abreviaturas e palavras compostas identificadas na etapa de preparação.

Desenvolvimento do Linguistico - O Identificador Morfológico

O Identificador Morfológico encontra-se em fase de projeto e foi concebido para utilizar tecnologias adaptativas (Fig.6.1). O Identificador Morfológico é composto por um Autômato Mestre

e um conjunto de submáquinas especialistas. O Autômato Mestre é responsável pelo sequenciamento chamadas às submáquinas, de acordo com um conjunto de regras cadastradas em base de dados.

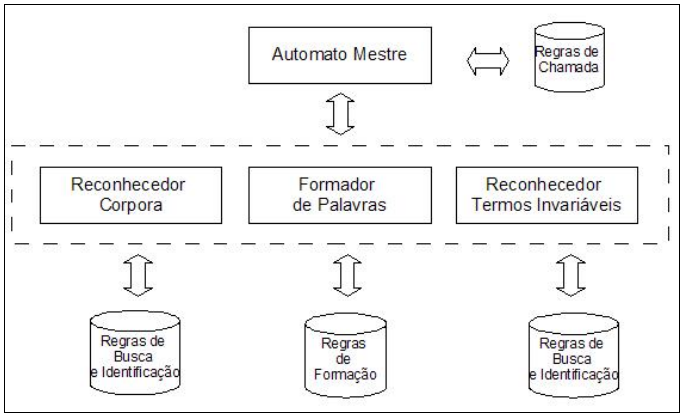


Figura 6.1. Arquitetura do Identificador Morfológico.

A prioridade é obter as classificações morfológicas do corpora (no caso, o Bosque, mas poderia ser outro, se houvesse necessidade); caso o termo procurado não exista no corpora, o Autômato Mestre procura por substantivos, adjetivos e verbos, no formato finito e infinito (flexionados e não flexionados), através de uma submáquina de formação e identificação de palavras, que usa, como base, o vocabulário do TeP2.0 (aqui também existe a possibilidade de usar outra base de dados, substituindo ou complementando o TeP2.0); por fim, o Autômato Mestre procura por termos invariáveis, ou seja, termos cuja classificação morfológica é considerada estável pelos linguistas, tais como, conjunções, preposições e pronomes, no caso, extraídos no léxico do Portal São Francisco. A Fig. 6.2 apresenta a estrutura adaptativa do Autômato Mestre.

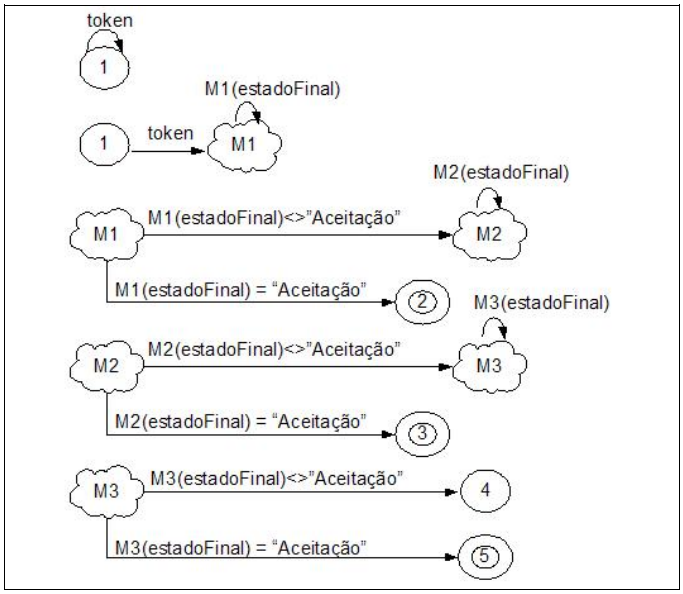


Figura 6.2. Estrutura Adaptativa do Autômato Mestre.

O Autômato Mestre inicia seu processamento recebendo o token da coleção de tokens criada pela classe Texto. Em seguida, antes de processá-lo, o autômato se modifica através de uma função adaptativa, criando uma submáquina de processamento (M1), uma transição entre o estado 1 e M1, e uma transição que aguarda o estado final da submáquina M1. O token é passado para M1 e armazenado em uma pilha. Quando M1 chega ao estado final, o autômato se modifica novamente, criando uma nova submáquina M2, o estado 2 e as transições correspondentes. Caso o estado final de M1 seja de aceitação, o processo é finalizado no estado 2, caso contrário a máquina M2 é chamada, passando o token armazenado na pilha. O processo se repete quando M2 chega ao final do processamento, com a criação da submáquina M3, do estado 3 e das transições correpondentes. Um novo ciclo se repete e se o estado final de M3 é de aceitação, o autômato tranciona para o estado 5, de aceitação, caso contrário, ele vai para o estado 4 de não aceitação. M1, M2 e M3 representam, respectivamente, as submáquinas do Reconhecedor de Corpora, do Formador de Palavras e do Reconhecedor de Termos Invariáveis. Portanto, caso o Autômato Mestre encontre a classificação morfológica ao final de M1, ele não chama M2; caso encontre em M2, não chama M3 e, caso também não encontre em M3, ele informa aos demais módulos do Linguístico que não há classificação morfológica para o termo analisado.

As submáquinas M1, M2 e M3 também foram projetadas de acordo com a tecnologia adaptativa. A Máquina M1 usa um automato adaptativo que se automodifica de acordo com o tipo de token que está sendo analisado: palavras simples, palavras compostas, números, valores e símbolos. A Fig. 6.3 apresenta a estrutura adaptativa de M1.

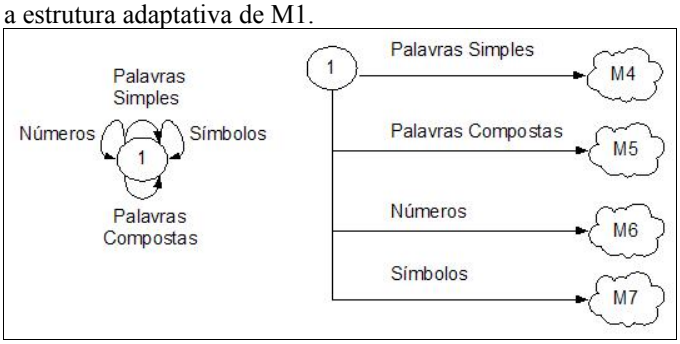


Figura 6.3. Estrutura Adaptativa do Reconhecedor de Corpora M1.

Inicialmente o autômato é composto por um único estado e por transições para ele mesmo (Fig.6.3, à esquerda). Ao identificar o tipo de token que será analisado (obtido no processo de tokenização), o autômato cria submáquinas e as transições correspondentes. As alternativas de configuração são apresentadas à direita, na Fig.6.3. As submáquinas M4, M5, M6 e M7 reconhecem os tokens através de outro tipo de autômato que processa os tokens byte a byte (Fig. 6.4).

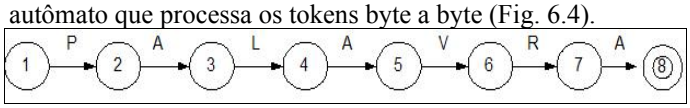


Figura 6.4. Reconhecedor de Palavras Simples.

No exemplo apresentado na Fig.6.4, o token "Palavra" é processado pela máquina M4; se o processamento terminar em um estado de aceitação, o token é reconhecido. As submáquinas M4, M5, M6 e M7 são criadas previamente por um programa que lê o corpora e o converte em autômatos finitos determinísticos. Junto com os estados de aceitação, são armazenadas as classificações morfológicas.

O Formador de Palavras, submáquina M2, também é montado previamente por um programa construtor, que o faz levando em consideração regras de formação de substantivos, adjetivos e verbos. No caso de substantivos e adjetivos, utilizam-se o radical e as terminações de gênero, número e grau para obter

as respectivas derivações. No caso de verbos, utilizam-se o radical e as terminações de tempo, modo, voz e pessoa para obter as derivações. Já a submáquina M3 é um autômato que varia em função do tipo de termo (conjunções, preposições e pronomes) e utiliza uma estrutura arborea similar a M4.

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.

O Linguístico encontra-se em fase de contrução, dividida em etapas em função da estrutura do reconhecedor. A primeira versão do sentenciador e do tokenizador foram finalizadas. O trabalho encontra-se na fase de projeto do Identificador Morfológico, que foi totalmente concebido para utilizar tecnologia adaptativa.

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. 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] Tep2. Thesauro Eletrônico para o Português do Brasil. Disponível em: <http://www.nilc.icmc.usp.br/tep2/>

[15] Portal São Francisco. Materiais de Língua Portuguesa. Disponível em: <http://www.portalsaofrancisco.com.br/alfa/materias/index-lingua-portuguesa.php/>

[16] Wikipedia. Disponível em: <http://pt.wikipedia.org/wiki/P%C3%A1gina_principal/>

[17] Karlsson, F. Constraint Grammar as a Framework for Parsing Running Text. 13o. International Conference on Computational Linguistics, Helsinki (Vol.3, pp.168-173).

[18] Python. Python Programming Language Official Web Site. 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.

Adaptability in Recommendation Systems: perspectives

F.G. Cozman, M. R. Pereira-Barretto, W. J. Fuks

***Abstract*— Recommendation Systems and Navigation Systems are more and more helping in the search for the need thing on the Internet. This paper discusses possible applications of adaptability to them.**

***Keywords*— Recommendation systems, Navigation Systems, Semantic Web.**

I. INTRODUÇÃO

Sistemas de Recomendação têm sido utilizados principalmente em e-commerce, particularmente em lojas na Internet, sugerindo produtos relacionados aos que se está adquirindo ou com base no perfil do consumidor. Claramente, o interesse em seu desenvolvimento é o aumento das vendas. Outras aplicações envolvem a busca dirigida de conteúdo na Internet, podendo ser considerada como tecnologia associada à Web Semântica [3],[4]: indicações de páginas que contenham assuntos de interesse de uma pessoa, sem que esta pessoa as tenha buscado de forma ativa, considerando como ativa a busca que acontece quando se realiza buscas em um *engine* como Google. Estas ideias podem ser estendidas a outros domínios de recomendação, como a indicação de restaurantes, roupas, filmes, assuntos históricos ou outros.

De uma forma geral, os Sistemas de Recomendação visam a determinação de indicações que sejam de interesse da pessoa que navega. Uma visão mais recente incorpora a ideia de serendipidade (neologismo provindo do Inglês *serendipity*) [5]. Na Internet, o conceito está relacionado com o fato de que, durante a navegação, frequentemente encontra-se um link que leva a um assunto relacionado ou mesmo a um outro assunto fracamente associado ao assunto inicial, que atrai a atenção da pessoa que navega, fazendo-a passar a uma outra trajetória de navegação. Ao analisar-se a navegação, parece que foi realizada de modo fortuito, quase aleatório. Por este motivo, os Sistemas de Recomendação tem sido complementados ou substituídos por Sistemas de Navegação, que buscam exibir várias opções de continuação na busca. Assim, Sistemas de Recomendação não são idênticos a Sistemas de Navegação, embora o objetivo final seja semelhante: sugerir algo que seja do interesse daquele que navega na Internet.

M. R. Pereira-Barretto, Escola Politécnica da Universidade de São Paulo (EP-USP), marcos.barretto@poli.usp.br

F. G. Cozman, Escola Politécnica da Universidade de São Paulo (EP-USP), fgcozman@usp.br

W. J. Fuks, Escola Politécnica da Universidade de São Paulo (EP-USP), william.fuks@gmail.com

Sistemas de Recomendação ou de Navegação contribuem para a construção de uma *user experience* que seja única, adaptada a cada pessoa, em cada momento. Sheryl Sandberg, COO do Facebook, escreveu que "sites que não forem customizados tornar-se-ão relíquia histórica". Gigantes como Microsoft, Apple, Facebook e Amazon são conhecidos por serem "máquinas de predição de gostos" [6]. Cerca de 30% do faturamento da Amazon e 60% da Netflix provêem de Sistemas de Recomendação[6].

Entretanto, mesmo com este sucesso, as técnicas utilizadas, normalmente, não levam em conta a trajetória pessoal em cada momento; ao contrário, buscam a criação de agrupamentos (frequentemente estatísticos) ou formas de classificação de perfis de consumo que são gerais e não individuais. Este artigo propõe a utilização de técnicas adaptativas para a construção de uma experiência única e pessoal, ajustada ao momento de cada pessoa.

As próximas seções deste artigo estarão organizadas da seguinte maneira: na seção 2, faz-se uma breve descrição das estratégias correntes para a construção de Sistemas de Recomendação. Na seção 3, o conceito de adaptabilidade, como utilizado neste trabalho, é apresentado. Já na seção 4, formas de utilização do conceito de adaptabilidade são discutidas, bem como sua aplicação a Sistemas de Recomendação. Por fim, na seção 5 estarão as considerações finais.

II. SISTEMAS DE RECOMENDAÇÃO

A Fig.1 ilustra os principais tipos de Sistemas de Recomendação, utilizando-se da classificação feita por Ricci [7].

Figura 1. Principais tipos de Sistemas de Recomendação.

Tem-se:

- sistemas baseados em filtragem colaborativa, que buscam realizar uma clusterização, agrupando usuários de comportamento semelhante, derivado a partir dos dados (de navegação na Internet, por exemplo).
- sistemas baseados em conteúdo, que buscam realizar uma indicação com base no histórico (de navegação, por exemplo).
- sistemas baseados em comunidades, que buscam realizar indicações a partir do comportamento dos "amigos" (em sentido amplo: pessoas que fazem parte da mesma comunidade).
- sistemas baseados em informações socio-demográficas, que são a tradição em estudos do comportamento do consumidor, determinando indicações a partir de comportamentos típicos de acordo com idade, classe social, estado civil, etc.
- sistemas baseados em expertise, que realizam indicações para problemas específicos como "ferramentas para conserto de encanamento".
- sistemas híbridos, que utilizam duas ou mais abordagens, simultaneamente.

Os Sistemas de Recomendação baseados em filtragem colaborativa são os mais comuns [8] e podem ser utilizados como caracterização da generalização que, em geral, é buscada com as técnicas atuais. Em [8], os sistemas baseados em filtragem colaborativas são divididos em três categorias:

- baseados em memória (*memory-based*), que utilizam-se exclusivamente dos dados (de navegação, por exemplo)
- baseados em modelos (*model-based*), que utilizam-se de modelos de comportamento
- híbridos, que associam as técnicas acima ou ainda outras.

Estes sistemas utilizam-se de técnicas como as relacionadas na Fig. 1.

CF categories	Representative techniques
Memory-based CF	*Neighbor-based CF (item-based/user-based CF algorithms with Pearson/vector cosine correlation) *Item-based/user-based top-*N* recommendations
Model-based CF	*Bayesian belief nets CF *clustering CF *MDP-based CF *latent semantic CF *sparse factor analysis *CF using dimensionality reduction techniques, for example, *SVD, PCA*
Hybrid recommenders	*content-based CF recommender, for example, *Fab* *content-boosted CF *hybrid CF combining memory-based and model-based *CF* algorithms, for example, Personality Diagnosis

Figura 1. Técnicas de filtragem colaborativa [8].

Entre os modelos discutidos em [8], tem particular relação com o presente trabalho aqueles que se utilizam de modelos MDP (Markov Decision Process)[9], que vêem o problema de recomendação não como de predição mas como de otimização sequencial ("o que o usuário vai escolher em seguida?").

Um processo de decisão de Markov é uma tupla <***S, A, T, R***> onde:

- ***S*** é um conjunto de estados em que o processo pode estar;
- ***A*** é um conjunto de ações que podem ser executadas em diferentes épocas de decisão
- ***T*** é uma função que dá a probabilidade de o sistema passar para um estado s_1, dado que o processo estava no estado s_0 e o agente decidiu executar uma determinada ação ***a***;
- ***R*** é uma função que dá o custo (ou recompensa) por tomar a decisão de executar a ação a quando o processo está em um estado s_0.

Uma aplicação de MDP a Sistemas de Recomendação encontra-se em [10], em que relata-se que o uso de um sistema baseado em MDP à loja de livros *online* Mitos, de Israel, produziu um resultado bastante superior que uma simples cadeia de Markov.

III. Adaptabilidade

Neste trabalho, o conceito de adaptatividade está associado a Máquinas de Markov Adaptativas, utilizando-se a formulação em [11], baseada em [1],[2].

Uma máquina de Markov adaptativa *M* é definida por uma quíntupla

$$M = (Q, S, T, q_0, F) \quad (1)$$

onde Q é um conjunto finito com n estados, S é o alfabeto de saída, q_0 é o estado inicial da rede e T é um conjunto de transições entre estados.
Tem-se F como um conjunto de funções adaptativas. Toda função adaptativa pertencente a F pode ser definida como uma quádrupla

$$f = (\Psi, V, G, C) \quad (2)$$

onde:

- Ψ é um conjunto de parâmetros formais (ψ_1, ψ_2, ψ_3,..., ψ_m);
- V é um conjunto de identificadores de variáveis (v_1, v_2, ..., v_n), cujos valores são desconhecidos no instante de chamada de f mas que uma vez preenchidos terão seus valores preservados durante toda a execução da função;
- G é um conjunto de identificadores de geradores (g^*_1, g^*_2,..., g^*_n), variáveis especiais que são preenchidas com novos valores, ainda não utilizados pelo autômato, a cada vez que a função é chamada;
- C é sequência de ações adaptativas elementares executadas em f.

Cada transação γ_{ij} é caracterizada por:

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}, p_{ij}, a_{ij}). \quad (3)$$

onde ρ é a probabilidade do estado q_j ser atingido estando em q_i.

Observe que cada transição γ_{ij} é única em T, isto é, não há duas transições diferentes partindo de um mesmo estado q_i e chegando a q_j. Além disso, seja Γ_q como o conjunto de todas as transições que se originam em um estado q. Então

$$\forall q \in Q, \sum_{\gamma \in \Gamma_q} \rho_\gamma = 1 \quad (4)$$

Isto é, a soma das probabilidades de todas as transições iniciando em q é exatamente um, para todo q pertencente a Q.
Tem-se a_{ij} como uma ação adaptativa da forma

$$a_{ij} = f(\omega_1, \omega_2, ..., \omega_n) \cup \{\varepsilon\} \quad (5)$$

sendo f uma função adaptativa pertencente a F e (ω_1, ω_2, ..., ω_n) uma lista de argumentos que correspondem posicionalmente à lista de parâmetros Ψ declaradas para f. Finalmente, quando γ_{ij} é acionada, o símbolo $p_{ij} \in S \cup \{\varepsilon\}$ é inserido na cadeia de saída de $\mathcal{M}$.

Define-se um SMA (Sistema de Markov Adaptativo) como um conjunto de máquinas adaptativas de Markov que podem se relacionar através de novas ações adaptativas, introduzidas a seguir. Assim, pode-se estabelecer uma relação de escopo sobre as ações adaptativas definidas para cada máquina, classificando-as em: ações que modifiquem apenas a topologia local da máquina e ações que podem interferir no comportamento de outras máquinas pertencentes ao sistema.

Desta forma, cada máquina M^k pertencente a um sistema de Markov adaptativo é definida como uma quíntupla:

$$M^k = (Q^k, \Sigma, T^k, q_0{}^k, F^k) \quad (10)$$

onde Σ é o alfabeto de saída, comum a todas as máquinas do sistema, e Q^k, T^k, $q_0{}^k$ e F^k são como na definição de Q, T, q_0 e F, respectivamente.

IV. Adaptabilidade em Sistemas de Recomendação

A aplicação de adaptabilidade em Sistemas de Recomendação será aqui discutida apenas no contexto de e-commerce, para que exemplos significativos e de fácil compreensão possam ser apresentados. Outras aplicações, como as indicadas na Introdução deste trabalho, podem ser extrapoladas a partir das discussões que se seguem.

Na busca por uma experiência pessoal e única, enfrenta-se de saída a questão da identificação do usuário na Internet: para oferecer uma experiência pessoal de navegação, é necessário saber-se quem está navegando. A tecnologia mais comumente utilizada para esta finalidade é o HTTP *cookie*, que fica armazenado no computador daquele que navega na Internet. Esta tecnologia é insuficiente para a determinação de quem está navegando, já que normalmente as pessoas navegam na Internet utilizando-se de dispositivos diversos como computadores pessoais e no trabalho, telefones móveis, tablets, etc. Além disso, substituem seus dispositivos a intervalos inferiores a 2 anos, fazendo com que, no médio prazo, perca-se o *cookie*. É raro fazer-se autenticação (*login*) para navegação. O fenômeno das redes sociais pode alterar esta situação, já que uma crescente quantidade de usuários da Internet mantém páginas pessoais em redes sociais, que exigem autenticação para acesso. Assim, a partir de serviços fornecidos pelas redes sociais, é possível ter-se a identificação única daquele que realiza a navegação. No restante deste trabalho, admite-se que seja possível identificar-se unicamente o usuário por intervalos longos de tempo, embora não sejam utilizadas informações pessoais como idade ou sexo.

As abordagens comumente empregadas em Sistemas de Recomendação, como exposto, determinam uma categoria, que é fixa no tempo. Assim, por exemplo, se um indivíduo é classificado como "comprador de livros de ficção científica", o sistema tenderá a oferecer-lhe este tipo de produto ou, talvez, produtos correlatos como filmes de ficção científica. Entretanto, a vida de um indivíduo altera-se com o passar do tempo: livros de ficção científica podem ser acompanhados (ou substituídos) por livros sobre computação, significando talvez que um jovem ingressou no curso superior em Computação e alterou seus hábitos de consumo. Mais tarde (ou ao mesmo tempo), livros sobre culinária podem acompanhar (ou substituir) os anteriores, significando talvez que o universitário iniciou sua vida profissional e está morando sozinho, tendo agora a necessidade de cozinhar. Assim, há alterações no comportamento de consumo no longo prazo, causado pelos eventos que ocorrem na vida das pessoas, de forma geral: o ingresso na Universidade, o início da vida profissional, o (possível) casamento, a (eventual) chegada dos filhos, a mudança de casa, etc. Este aspecto pode ser chamado de "dinâmica de longa duração": cada fase persiste durante meses ou até anos. Estas alterações de comportamento, entretanto, nem sempre são definitivas: possivelmente, um

interesse anterior pode ser ressucitado: em um dado momento, um livro de ficção científica pode novamente atrair a atenção, dado que no passado o fez.

Por outro lado, tem-se "dinâmicas de curta duração iniciadas por data", que duram dias e que aparecem na vida das pessoas em datas específicas como Dia dos Namorados, Dia das Mães, aniversário do(a) companheiro(a), aniversários de pais e amigos, etc. O conceito também se aplica ao consumo sazonal: busca-se por pacotes de férias mais provavelmente nos meses de janeiro, fevereiro e julho, por exemplo. Nestas datas, o comportamento altera-se: se normalmente há o interesse por livros de ficção científica, passa-se a buscar por presentes para a mãe, por exemplo. Note-se que há dinâmicas em que se conhece o instante em que podem ocorrer (por exemplo, proximamente ao Dia das Mães) e outras em que não há como sabê-lo, a priori (por exemplo, o aniversário da mãe), mas que, possivelmente, repetem-se a cada ano.

Pode-se também considerar, particularmente em casos como e-commerce, "dinâmicas de curta duração iniciadas pelo produto", que também duram dias e que aparecem na vida das pessoas quando um determinado produto, adquirido (ou buscado) no passado, está próximo da data de sua substituição. Os telefones celulares, por exemplo, são substituídos a cada ano, por conta dos planos das operadoras de telefonia, que favorecem esta substituição para fidelização de clientes. Assim, para maior probabilidade de aceitação, pode-se sugeri-los depois deste tempo. Tal dinâmica também pode ser importante no lançamento de um determinado produto: o lançamento de um novo modelo de telefone celular fortemente divulgado pela mídia pode também sugerir sua inclusão como recomendação, por certo tempo.

Deve-se ainda destacar que, tipicamente, uma pessoa apresenta perfis de consumo distintos, simultaneamente: o "estudante universitário de Computação" pode, ao mesmo tempo, ser um "*biker* ativo", ter grande interesse por filmes e música, etc.

Esta discussão, embora simplista, permite demonstrar os principais aspectos de um dos possíveis modos de incluir-se adaptabilidade (no conceito anteriormente definido) no contexto dos Sistemas de Recomendação e Navegação: um modelo em que os diferentes perfis de consumidor são simultaneamente considerados, ao mesmo tempo, utilizando-se da adaptabilidade para examinar o passado (ou parte deste) para a determinação de uma recomendação.

A Fig.2 mostra uma proposta de aplicação do conceito de adaptabilidade a Sistemas de Recomendação e Navegação para e-commerce, ilustrando a MMA (Máquina de Markov Adaptativa) para recomendação.

Na Fig.2, tem-se cada estado contendo um produto (ou categoria), além de outros, correlacionados. O produto principal está indicado pelo retângulo escuro e corresponde a algum produto já buscado (ou comprado) anteriormente pelo indivíduo. Os produtos correlacionados podem ser determinados por análise funcional (como na figura, em que tem-se produtos correlacionados por este critério) ou por análise estatística como a correlação de Pearson, como realizado em [12].

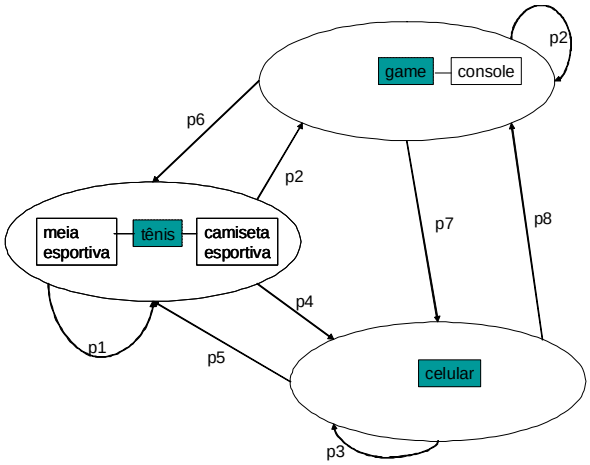


Figura 2. MMA para recomendação

A recomendação é realizada pela transição de estados; pode-se ter a indicação do produto principal ou de um (ou mais) dos produtos correlacionados, a cada recomendação.

A dinâmica de adaptabilidade pode seguir o algoritmo da Fig.3, executado antes da transição da MMA.

```
Se houve a navegação para outro produto
{
   1.Incluir estado relativo ao produto na MMA
   2.Recalcular as probabilidades
}
Se é uma época notável geral (como Dia das Mães) E
   ainda não foram incluídos os produtos adequados
{
    1.Incluir estado relativo aos produtos
        adequados à época notável
    2.Recalcular as probabilidades
}
Se é uma época notável individual
  (como aniversário da mãe) E
  ainda não foram incluídos os produtos adequados
{
    1.Incluir estado relativo aos produtos
        adequados à época notável
    2.Recalcular as probabilidades
}
Se passou-se mais de uma semana desde
   o último cálculo de probabilidades
{
   Recalcular as probabilidades
}
```

Figura 3. Adaptação da MMA

Possivelmente, a deteção de datas notáveis pode ser realizada através de técnicas de identificação de sazonalidade, como ARIMA [14] ou outras, como [13], caracterizando-se alterações fortes de comportamento em determinadas épocas do ano, analisadas em horizontes relativamente longos como 10 anos.

O algoritmo descrito pode ser totalmente executado *client-side*, a partir de informações trazidas do servidor que incluem, basicamente, o registro dos acessos do indivíduo ao sistema, bem com da MMA. Como este conjunto de informações é relativamente reduzido, não há sobrecarga na rede.

V. CONCLUSÃO

A adaptabilidade não vem sendo propriamente considerada no contexto de Sistemas de Recomendação e de Navegação e sua inclusão pode trazer resultados positivos. O artigo apresentou uma das várias possíveis maneiras de considerá-la. Os modelos apresentados, embora parciais, sugerem que a abordagem proposta possa complementar as técnicas normalmente utilizadas nesta categoria de sistemas. Entretanto, a comprovação prática da melhoria que podem introduzir não foi realizada, devendo-se fazê-la em futuro próximo.

AGRADECIMENTOS

Os autores agradecem à Buscapé pelo patrocínio e apoio à realização deste trabalho, através do projeto FDTE-1176.

REFERÊNCIAS

[1] Neto, J. J. "Contribuições à Metodologia de Construção de Compiladores", Tese de Livre Docência, São Paulo, Escola Politécnica da USP, 1993.

[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 Universidade de São Paulo, 2000.

[3] Shadbolt, N.; Hall, W.; Berners-Lee, T. "The Semantic Web Revisited". IEEE Inteligent Systems, 2004.

[4] Ziegler, C. "Semantic web recommender systems". Lecture Note in Computer Science vol.3268 pp.78-89, 2004.

[5] Herlocker, J.L.; Kostan, J.A.; Terveen, L.G.; Riedl, J.T. "Evaluating collaborative filtering recommender systems". ACM Transactions on Information Systems, vol.22 no.1, 2004.

[6] Exame. "Como Google e Facebook filtram a Internet". http://exame.abril.com.br/tecnologia/noticias/como-google-e-facebook-filtram-a-internet?page=1&slug_name=como-google-e-facebook-filtram-a-internet. Acessado em 10 de outubro de 2011.

[7] Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. "Recommender Systems Handbook". Springer, 2011.

[8] Xiaoyuan, S.; Khoshgoftaar, T. M. "A survey of collaborative filtering techniques". Advances in Artificial Intelligence, vol.2009, 2009.

[9] Pellegrini, W.; Wainer, J. "Processos de decisão de Markov". RITA vol.XIV no.2, 2007.

[10] Shani, G.;Heckerman, D.;Brafman, R.I. "AnMDP-based recommender system," Journal of Machine Learning Research, vol. 6, pp. 1265–1295, 2005.

[11] Alfenas, D.A.; Shibata, D.P.; Neto, J.J.; Pereira-Barretto, M.R. "Sistemas de Markov Adaptativos: formulação e plataforma de desenvolvimento". Submetido ao WTA2012.

[12] Segaran, T. "Programming collective intelligence". O´Reilly, 2007.

[13] Thio, N.; Karunasekera, S. "Medium-term client-perceived performance prediction". Journal of Emerging Technologies in Web Intelligence, vol.2 no.1, 2010.

[14] Brockwell, P.J.; Davis, A.D. "Introduction to time series and forecasting". Springer, 1996.

Marcos Ribeiro Pereira-Barretto é graduado em Engenharia Elétrica (1983), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Mecânica (1993) pela Escola Politécnica da Universidade de São Paulo. É professor da Escola Politécnica da USP desde 1986. Atua nas seguintes áreas de pesquisa: robôs sociáveis, computação afetiva e arquitetura de sistemas para aplicações críticas como Automação Industrial.

Fabio Gaglardi Cozman é é Professor Titular da Escola Politécnica da Universidade de São Paulo (Dept. Engenharia Mecatrônica) desde 2007, tendo ingressado na Escola Politécnica em 1990. Possui graduação em Engenharia Elétrica Modalidade Eletrônica pela Universidade de São Paulo (1989), mestrado em Engenharia pela Universidade de São Paulo (1991), Phd pela Carnegie Mellon University (1997), Livre-docência pela Universidade de São Paulo (2003). Atualmente é membro da comissão de inteligência artificial da SBC, membro do comitê editorial do Int. Journal on Approximate Reasoning. Pesquisas focam na automação de processos de decisão sob incerteza, incluindo representação de conhecimento e aprendizado (tópicos: inteligência artificial, redes bayesianas, conjuntos de probabilidade, modelos estatísticos gráficos)..

Willian Jean Fuks é graduado pela escola Politécnica da USP em Engenharia Elétrica com ênfase em Controle e Automação. Atualmente cursa o programa de Mestrado pelo departamento de Engenharia Mecânica da USP realizando pesquisas na área de aprendizagem de máquina e inteligência artificial para processos de decisão em lojas e-commerce.

Uso da Tecnologia Adaptativa em um SPLN para criação automática de atividades de leitura

J. L. Moreira Filho and Z. M. Zapparoli

***Abstract*— This paper presents the initial steps of a research on the use of the Adaptative Technology to develop a natural language processing system for creating reading activities in English. More specifically, it proposes the application of adaptative devices in pronominal anaphora resolution for textual cohesion exercises.**

***Keywords*— Natural Language Processing, Corpus Linguistics, Adaptative Technology, Anaphor Resolution.**

I. INTRODUÇÃO

ATUALMENTE, é crescente o número de pesquisas sobre o emprego de materiais de ensino baseados em *corpora*, uma vez que privilegiam a [1]língua em uso.

Embora seja desejável o uso de materiais baseados em *corpus* e haja uma série de ferramentas computacionais que auxiliam na análise de *corpora* para a sua criação, seu processo de preparação ainda é muito demorado e, geralmente, realizado apenas por pesquisadores; muitas vezes, requer a análise prévia de grandes quantidades de dados, como concordâncias, listas de frequência, listas de palavras-chave, anotação de *corpus*, entre outros tipos. Podemos citar, como exemplo, a pesquisa de [1], que descreveu todo o percurso do uso de dois *corpora* na elaboração de uma tarefa para ensino de inglês por meio de análises propiciadas por essas ferramentas.

Devido a esses motivos, professores podem ter dificuldades na preparação de tais materiais e, em consequência, não utilizá-los com certa frequência e/ou fazer uso de materiais tradicionais não significativos para a aprendizagem dos alunos.

Dentre as ferramentas computacionais que podem auxiliar na análise de *corpora*, citamos dois programas de computador desenvolvidos especialmente para o trabalho com textos e que reúnem recursos computacionais, matemáticos e estatísticos – *WordSmith Tools* e *Stablex PC*.

De autoria de Mike Scott, Universidade de Liverpool, o programa *WordSmith Tools* (1998) é publicado pela Oxford University Press e distribuído via *World Wide Web* (http://www.lexically.net/wordsmith). Em sua quinta versão, disponível para PC/Windows 2000 ou superior, incluindo Windows 7, disponibiliza diversos recursos – *Wordlist*, *Concord*, *Keywords*, *Splitter*, *Text Converter*, *Dual Text Aligner*, *Viewer* − para tarefas específicas de análises de textos.

O programa *Stablex* (STA – de statistique, TAB – de tableaux, LEX – de lexique e T...EX – de texte), de autoria de André Camlong e Thierry Beltran, Universidade de Toulouse II, inicialmente desenvolvido para *Macintosh* (Toulouse, Teknea, 1991), conta, atualmente, com a sua versão PC (São Paulo, Pirus Tecnologia, 2004). Os seus recursos − geração de léxicos, indexação, extração de sequências e concordâncias, lematização, tratamento estatístico − foram desenvolvidos em função de um modelo de análise lexical, textual e discursiva – *método matemático-estatístico-computacional de análise de textos* de André Camlong. Trata-se, por conseguinte, da aplicação de um programa que serve de ferramenta para um método de análise de textos.

O método é fundado na matemática e na estatística paramétrica (estatística descritiva); possibilita o estudo descritivo, objetivo e indutivo do texto; permite a análise quantiqualitativa do léxico, que indica apontamentos para a análise textual e discursiva. Nele, o texto é o ponto de referência: as operações estatísticas partem do texto e, por sua vez, refletem o texto.

Destacamos o fato de as listas originárias da análise de textos pelo *Stablex* exibirem o léxico do *corpus* e dos textos que o integram – e não apenas do *corpus* como um todo, caso do *WordSmith* −, o que facilita uma visão contrastiva do todo – *corpus* – em relação às partes – textos que integram o *corpus* – e das partes em relação ao todo, bem como das partes entre si. Já o *WordSmith* assume importância fundamental no tratamento de *corpora* extensos, voltados à construção de dicionários e de glossários, aos exames dos padrões linguísticos, aos estudos ligados à tradução e ao gênero.

A partir do desenvolvimento, aplicação e análise de um sistema de montagem automática de atividades *online* de leitura em língua inglesa com *corpora*, por meio do uso de técnicas adaptativas, a pesquisa em andamento tem o objetivo de suprir a necessidade de professores que desejam utilizar materiais baseados em *corpora* em suas aulas, mas que não estão familiarizados com o uso de ferramentas de processamento e exploração de *corpora* e/ou que não possuem muito tempo para preparar atividades.

A investigação está baseada em um estudo realizado em uma pesquisa de mestrado [2], que teve como produto final um *software desktop* para preparação semiautomática de atividades de leitura em inglês, tomando como entrada um texto selecionado pelo usuário com fins pedagógicos e conduzindo-o, através de etapas, como um assistente eletrônico, até a publicação de uma unidade didática. A versão mais recente do *software*, *Reading Class Builder*, está disponível no sítio: http://www.fflch.usp.br/dl/li/x/?p=409.

Nos primeiros protótipos, diferentes exercícios são preparados automaticamente, incluindo atividades baseadas em concordâncias (*data-driven learning*), predição, léxico-gramática e questões para leitura crítica. Para tanto, o programa faz várias análises automáticas do texto selecionado por meio de fórmulas estatísticas: lista de frequência, palavras-

[1] Textos autênticos, não inventados para ensinar língua.

chave, possíveis palavras cognatas, etiquetagem morfológica, possíveis padrões (n-gramas) e densidade lexical do texto.

Embora os resultados obtidos tenham demonstrado a viabilidade e o potencial de, por meio do computador, analisar textos e gerar automaticamente determinados tipos de exercícios para ensino de estratégias de leitura, há ainda a necessidade de muita pesquisa e desenvolvimento de melhorias para que a ferramenta possa ser usada pelo usuário final.

Há situações em que o programa gera exercícios com erros, tal como incluir na lista de palavras gramaticais uma palavra de conteúdo. Nesses casos, o usuário pode fazer a correção manualmente na atividade gerada. Porém, se a quantidade de erros for numerosa e exigir constantemente esse tipo de intervenção do usuário/professor, o sistema pode perder sua utilidade, impossibilitando seu uso pedagógico.

Outro ponto importante a ser foco de melhorias é a pouca variedade de exercícios disponíveis e a limitação do usuário a um modelo fixo. O programa poderia gerar exercícios em relação a outros itens de ensino comuns no ensino de estratégias de leitura, como grupos nominais, marcadores discursivos, grau de adjetivos e advérbios, questões de *skimming* e *scanning*, formas verbais e referência pronominal. Seria desejável também que a interface com o usuário permitisse a seleção de determinados pontos a serem ensinados e que o sistema auxiliasse na escolha de texto e exercícios adequados tais fins.

Neste artigo, descrevemos as possibilidades de uso e aplicação de dispositivos adaptativos em um dos módulos do sistema, cuja função específica é a criação de exercícios baseados em coesão textual. Tal uso será considerado como um estudo piloto em relação à aplicação da Tecnologia Adaptativa.

II. OBJETIVOS

Os principais objetivos do estudo são: i. Apresentar, de maneira sucinta, alguns fundamentos linguísticos em relação à coesão textual; ii. Fazer uma revisão dos principais métodos tradicionais de resolução de anáfora; iii. Propor uma abordagem alternativa pelo uso de dispositivos adaptativos.

A intenção não é realizar uma pesquisa completa em relação à resolução de anáforas pronominais, mas fazer um ensaio de possíveis modelos de pesquisa para adaptação de métodos e abordagens existentes, utilizando conceitos e formalismos da Tecnologia Adaptativa.

Buscando estender os objetivos do mestrado [2], pretendemos empregar a Tecnologia Adaptativa em todos os níveis aplicáveis de um sistema de processamento de língua natural a ser construído, o qual, a partir do fornecimento de um texto e/ou algum tipo de entrada preestabelecida, deverá gerar atividades didáticas de leitura em língua inglesa.

No processo, estudaremos a possibilidade do uso da Tecnologia Adaptativa, a princípio, em quatro níveis/camadas do sistema: i. Opções do usuário; ii. Análise linguística da entrada; iii. Análise pedagógica; iv. Montagem de exercícios.

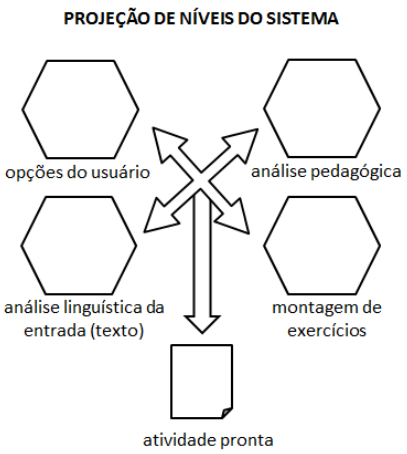


Figura 1. Níveis/camadas do sistema a ser construído

A atividade de ensino a ser gerada deverá levar em consideração as informações de todos os níveis. Desse modo, o produto final estará condicionado às opções do usuário (seleção de itens de ensino, tipos de exercício e itens léxico-gramaticais incluídos), análise linguística de textos (informações de frequência, palavras-chave, anotação morfossintática, entre outras), análise pedagógica da entrada (a partir da análise linguística, quais tipos de exercícios são possíveis e adequados), montagem de exercícios (como extrair e organizar os itens linguísticos).

III. PRESSUPOSTOS TEÓRICO-METODOLÓGICOS

O projeto encontra suporte teórico na Linguística de Corpus, Linguística Textual, Linguística Aplicada e Tecnologia Adaptativa.

A Linguística de Corpus, área que estuda os fatos da língua em uso e utiliza o computador no armazenamento e no tratamento e análise de dados empíricos, preocupa-se com a construção de *corpora* eletrônicos a partir de textos e discursos reais.

A Linguística Textual é a ciência que se dedica a investigar a estrutura e o funcionamento dos textos, sendo uma de suas tônicas os estudos sobre coesão e coerência textuais.

A Linguística Aplicada, em sua concepção atual, que vai além da aplicação da Linguística ao ensino/aprendizagem de línguas, dedica-se às situações de uso da língua e, pois, ao desenvolvimento de pesquisas a partir da análise de *corpora*.

A Tecnologia Adaptativa é o principal aporte teórico e metodológico para o desenvolvimento do sistema proposto.

Conforme [3], a Tecnologia Adaptativa está relacionada a técnicas, métodos e disciplinas que estudam as aplicações da adaptatividade, que pode ser entendida como uma propriedade que um determinado modelo tem de modificar espontaneamente seu próprio comportamento em resposta direta a uma entrada, sem auxilio externo.

Um sistema adaptativo é aquele que possui a propriedade de se automodificar a partir de determinada entrada, sem a necessidade de um agente externo.

Dentro da Tecnologia Adaptativa, há a noção de dispositivo, uma abstração formal. O dispositivo pode ser adaptativo ou não adaptativo. O dispositivo não adaptativo pode ser formado por um conjunto finito de regras estáticas que, em linguagem de programação, pode ser representado na forma de cláusulas IF-THEN. A operação do dispositivo se dá pela aplicação das regras, tendo como retorno determinados estados. Quando o dispositivo não aplica nenhuma regra, a operação é terminada, gerando um erro. As ações de dispositivos adaptativos podem ser chamadas quando ocorre algum erro (quando nenhuma regra é aplicável), ou quando a operação do dispositivo não adaptativo está em um determinado estado. Basicamente, os dispositivos adaptativos são formados por três ações adaptativas elementares [4]: i. Consulta de regras/estados; ii. Exclusão de regras; iii. Inclusão de regras. Seu uso está ligado a situações complexas em que há a necessidade de tomadas de decisões não triviais, por exemplo, na área de estudos da linguagem, resolução de ambiguidades em programas de anotação (morfológica, sintática, etc.).

Tendo em vista a complexidade do estudo pretendido, uma das áreas de interesse para a pesquisa, em relação à Tecnologia Adaptativa, é o processamento de linguagens naturais. A aplicação da Tecnologia Adaptativa ao processamento de línguas naturais é um campo de extrema importância.

IV. Fundamentos linguísticos da coesão textual

Para Halliday & Hasan [5], a coesão textual é definida como um conceito semântico que se refere às relações de sentido existentes no interior do texto e que o definem como um texto, ocorrendo quando a interpretação de algum elemento no discurso é dependente de outra.

Conforme os autores em [5], cinco principais mecanismos ou fatores de coesão podem ser identificados: referência, substituição, elipse, conjunção e coesão lexical. As relações de referência se desdobram em dois tipos: situacional (exófora) e textual (endófora). Chamamos de referência exofórica aquela em que a referência é feita a algum elemento da situação de comunicação, estando fora dos domínios do texto, sendo a interpretação apenas possível pela análise de tal situação. Na referência endofórica, ao contrário, é possível encontrar a interpretação dentro do próprio texto.

Em Processamento de Língua Natural, é comum a manipulação computacional de elementos de referência endofóricos. As referências endofóricas são divididas em dois tipos de acordo com a posição do referente: anafórica e catafórica. Quando a entidade referenciada antecede o elemento coesivo, temos uma relação anafórica. Quando a entidade referenciada está depois do item coesivo, temos uma relação catafórica.

No exemplo a seguir, destacamos a relação anafórica em que o pronome de terceira pessoa '*ela*' faz referência à entidade '*presidente Dilma Rousseff*'.

"*Os acordos anunciados durante a visita da presidente Dilma Rousseff à China significaram um começo razoável no que ela propôs como um novo capítulo no relacionamento bilateral*" [6].

Neste trabalho, tratamos apenas de métodos e abordagens computacionais de resolução de anáforas.

Existem várias definições de anáfora. A palavra anáfora possui origem no grego antigo, composta por ana-'para trás' e –phora 'ação de levar/transportar'.

Conforme [7], é comum o uso da definição clássica de Halliday e Hasan baseada na noção de coesão. Segundo esses estudiosos, a anáfora é coesão (pressuposição) que aponta de volta para algum item anterior. De acordo com tal definição, a entidade/item no fenômeno de referência que aponta para trás, para algum antecedente, é chamada de anáfora.

Existem muitos tipos de anáfora. Contudo, na literatura de Linguística Computacional, encontramos uma grande quantidade de trabalhos que privilegiam anáforas pronominais (*pronominal anaphoras*), que são realizadas por pronomes anafóricos, como os de terceira pessoa.

Há também uma distinção entre anáforas intrassentenciais e interssentenciais. Anáforas intrassentenciais referem-se aos antecedentes na mesma sentença. Anáforas interssentenciais referem-se aos antecedentes em uma sentença diferente.

V. Métodos tradicionais e conceitos básicos de resolução de anáfora

Remetemos, aqui, à categorização das estratégias computacionais de resolução de anáfora feita por [7], que divide as abordagens em tradicionais ou alternativas. Os modelos tradicionais eliminam possíveis candidatos até que os mais prováveis restem. Os modelos alternativos fazem uso de técnicas da Inteligência Artificial para ranquear os candidatos mais prováveis.

O processo de determinar o antecedente de uma anáfora é chamado de resolução de anáfora.

O processo de resolução de anáfora geralmente envolve a definição de variáveis, como as características dos antecedentes e o escopo de busca. Os sistemas de resolução de anáfora, em sua maioria, têm como antecedentes os sintagmas nominais (SNs) − tarefa menos complexa em comparação a sintagmas verbais (SVs) −, e o escopo de busca é delimitado à sentença atual ou precedente.

Com base nessa definição, os SNs que precedem a anáfora são identificados como candidatos a antecedentes. A escolha do antecedente correto é feita a partir de fatores de resolução.

Os fatores tipicamente empregados são: concordância de gênero e número; restrições; consistência semântica; paralelismo sintático; paralelismo semântico; saliência; proximidade etc. Tais fatores podem ser de dois tipos: eliminação ou preferência.

Em Linguística Computacional, há divergências em relação aos termos empregados. Alguns autores defendem o emprego apenas do termo preferência.

De qualquer forma, o uso desses termos leva à distinção entre duas principais arquiteturas em resolução de anáfora: arquiteturas baseadas em restrições e arquiteturas baseadas em

preferências. A restrição é o conceito mais poderoso no processo de resolução de anáfora [8].

As restrições podem ser definidas como um conjunto de propriedades que um candidato a antecedente deve possuir. O candidato poderá ser descartado da lista de possíveis candidatos caso não possua as propriedades desejáveis.

As restrições podem estar atreladas a determinados tipos de conhecimento: sintático, semântico, pragmático etc. Um exemplo de restrição sintática é a exigência de que a anáfora e seu antecedente devem concordar em gênero e número.

A título de exemplificação, fazemos referência à teoria de [2]*Centering* de Grosz e Sidner [9], um sistema de regras e restrições que define o discurso como uma estrutura hierárquica de intenções e estados de atenção constituída por segmentos. A estrutura intencional é composta por componentes globais (propósito do discurso) e componentes locais (propósito dos segmentos). O estado de atenção é composto por conjuntos de objetos, relações e propriedades no discurso.

Nos segmentos, há focos de atenção constituídos por entidades salientes. As entidades mais salientes são chamadas de centros. Os centros são objetos semânticos que podem conectar um enunciado a outros enunciados.

De acordo com [10], o conceito de centro da teoria de *Centering* é uma ferramenta importante para os fenômenos de correferência. Em resolução de anáforas interssentenciais, seu uso pode limitar grandes quantidades de candidatos. Pode ser útil também em casos ambíguos, em que o candidato mais provável pode ser o centro do segmento.

Embora a aplicação prática da teoria ainda não seja algo totalmente resolvido, tendo em vista que há vários métodos de determinar o centro de segmentos, o uso do conceito é uma constante.

A diferença entre preferências e restrições está no fato de que as preferências não são condições obrigatórias. Algumas preferências são: paralelismo sintático, paralelismo semântico e centro de atenção. O último é bastante útil em casos ambíguos, em que a decisão de escolha é baseada no candidato mais saliente, foco de atenção.

VI. PROPOSTA DE ABORDAGEM ALTERNATIVA PARA RESOLUÇÃO DE ANÁFORA

A abordagem alternativa visa à criação de dispositivos adaptativos para resolução de anáfora a partir de fatores e estratégias de métodos tradicionais e análise de *corpus* anotado.

A ideia central é a de que estudos específicos de fatores de restrição ou preferência, baseados em *corpus*, podem indicar dados interessantes, servindo para a determinação de peso, por exemplo, quando utilizados em conjunto com outros fatores, além de servirem, também, para a criação de regras para os dispositivos adaptativos e não adaptativos. É possível, ainda, que a análise dos dados sirva para o estabelecimento de fatores ainda não empregados nos métodos tradicionais.

[2] Centralização.

Os principais procedimentos metodológicos a serem executados para a pesquisa são:

1. Coleta de um *corpus* na Internet de [3]textos em inglês de determinados gêneros;
2. Etiquetagem automática pelo etiquetador disponível no sítio http://beta.visl.sdu.dk/visl/en/ via programação, enviando variáveis para o formulário *online* do etiquetador e baixando as páginas de resultado;
3. Etiquetagem manual de resolução de anáforas pronominais utilizando uma ferramenta criada especificamente para a tarefa;
4. Identificação de fatores específicos para pesquisa no *corpus* processado;
5. Geração/extração automática de todos os possíveis dados interessantes em relação aos fatores identificados via programação;
6. Análise e avaliação dos dados a fim de estabelecer relações entre métodos e abordagens existentes e novos critérios de restrição ou preferência para resolução de anáforas nos textos;
7. Desenvolvimento de dispositivos adaptativos para a resolução de anáforas e montagem de exercícios;
8. Avaliação dos resultados obtidos por meio do uso de dispositivos adaptativos.

Em relação ao desenvolvimento de dispositivos adaptativos e uso da Tecnologia Adaptativa, é possível estabelecer um conjunto de regras inicial a partir de métodos existentes e análise de fatores no *corpus* anotado. Em seguida, ampliar o número de regras em uma abordagem de aprendizado automático, usando regras obtidas a partir do *corpus* anotado, sendo aprendidas em treinamento. Uma abordagem de aprendizado automático, conforme [9], evita a necessidade de realizar inúmeros testes no estabelecimento de regras.

As informações linguísticas a serem inferidas serão provenientes dos fatores obtidos pela anotação, tal como distância, concordância de gênero e número e outras a serem verificadas.

A possibilidade de uso de dispositivos adaptativos nas tarefas propostas anteriormente está baseada no trabalho de [11], que ressalta a capacidade de autômatos adaptativos serem utilizados em sistemas de aprendizado automático, especificamente no processamento de língua natural.

VII. CONSIDERAÇÕES FINAIS

A partir do estudo de fatores de restrição e preferência específicos para a resolução de anáforas com o emprego de técnicas adaptativas, esperamos apontar as possibilidades de uma abordagem alternativa que conjugue fatores de métodos tradicionais, análise de *corpus* e modelos formais. O emprego de modelos formais no desenvolvimento do SPLN para a criação automática de atividades de leitura deve minimizar inconsistências na execução de tarefas que foram observadas na investigação para o mestrado. Assim sendo, a investigação

[3] Textos autênticos que possam ser utilizados com o objetivo de criação de atividades de compreensão de leitura.

pode oferecer contribuições e benefícios: no âmbito da Linguística, em especial da Linguística Aplicada, para a disseminação do uso de materiais de ensino baseados em *corpora*; na interface entre a Linguística e a Computação, para o enriquecimento da interação entre a ciência da linguagem e as ciências exatas.

REFERÊNCIAS

[1] CONDI, R. Dois corpora, uma tarefa. O percurso de coleta, análise e utilização de corpora eletrônicos na elaboração de uma tarefa para ensino de inglês como Língua Estrangeira. Dissertação de Mestrado Inédita, LAEL, PUC-SP, 2005.

[2] MOREIRA FILHO, P. Desenvolvimento de um software para preparação semiautomática de atividades de leitura em inglês. Dissertação de Mestrado Inédita, LAEL, PUC-SP, 2007.

[3] DIZERÓ, W. Formalismos Adaptativos Aplicados na Modelagem de Softwares Educacionais. Tese de Doutorado, EPUSP, São Paulo, 2010.

[4] PADOVANI, D.; CONTIER, A.; JOSÉ NETO, J. J.; Tecnologia Adaptativa Aplicada ao Processamento da Linguagem Natural. In: WTA 2010; Quarto Workshop de Tecnologia Adaptativa, 2010, São Paulo. Memórias do WTA 2010: Quarto Workshop de Tecnologia Adaptativa. São Paulo: Laboratório de Linguagens e Técnicas Adaptativas, 2010. p. 35-42.

[5] KOCH, I.G.V. *A coesão textual.* 22ª ed. São Paulo: Contexto, 2010. 84 p.

[6] Disponível em: <http://www2.jornaldacidade.net/artigos_ver.php?id=5247>. Acesso em 25/10/2011.

[7] MITKOV, R. Anaphora resolution: the state of the art, Working paper, (Based on the COLING'98/ACL'98 tutorial on anaphora resolution), University of Wolverhampton, Wolverhampton, 1999.

[8] LEFFA, V. J. A resolução de anáfora no processamento da língua natural. Relatório Final de Pesquisa da Universidade Católica de Pelotas, 2001.

[9] NUGUES, P.M. An introduction to language processing with Perl and Prolog. Berlin: Springer-Verlag, 2006.

[10] PARABONI, I. Uma arquitetura para a resolução de referências pronominais possessivas no processamento de textos em língua portuguesa. Dissertação de Mestrado Inédita. PUCRS, Porto Alegre, 1997.

[11] MENEZES, C. E. D.; JOSÉ NETO, J. Um método híbrido para a construção de etiquetadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos. Anais da Conferencia Iberoamericana en Sistemas, Cibernética e Informática, 19-21 de Julio, 2002, Orlando, Florida

José Lopes Moreira Filho é Doutorando em Semiótica e Linguística Geral (USP). Possui Mestrado em Linguística Aplicada e Estudos da Linguagem pela Pontifícia Universidade Católica de São Paulo (PUCSP). Possui graduação em Letras – Português e Inglês (Bacharelado Tradução) pela Universidade de Mogi das Cruzes (UMC). Atualmente, é Professor Coordenador de Língua Inglesa de Oficina Pedagógica em Diretoria Regional de Ensino da SEE-SP, mantendo interesses na área de Linguística, Linguística Aplicada, Linguística Informática, Linguística de Corpus, Processamento de Linguagem Natural, atuando principalmente no desenvolvimento de ferramentas computacionais para exploração de *corpora*, ensino de línguas, entre outras aplicações que envolvem linguagem e tecnologia.

Zilda Maria Zapparoli é professora associada aposentada junto ao Departamento de Linguística da Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo, 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, linha de pesquisa Informática no Tratamento de *Corpora* e na Prática da Tradução. Desde 1972, atua em Linguística Informática, com tese de doutorado, tese de livre-docência, pós-doutorado na Université de Toulouse II e trabalhos publicados na área. É líder do Grupo Interdisciplinar de Pesquisas em Linguística Informática, certificado pela USP e cadastrado no Diretório de Grupos de Pesquisa no Brasil do CNPq, em 2002. 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.

Adaptatividade em Robôs Sociáveis: uma Proposta de um Gerenciador de Diálogos

D. A. Alfenas, M. R. Pereira-Barretto

***Abstract*— Within sociable robots, a Dialog Manager is responsible for conducting a conversation. This paper proposes architecture to convey natural conversation about things in general, lying on the use of Adaptivity combined with Episodic and Semantic memories to produce human-like behavior.**

***Keywords*— Sociable Robots, Dialog Manager, adaptive technology, episodic memory, semantic memory.**

I. INTRODUÇÃO

Robôs sociáveis são robôs autônomos que interagem e se comunicam com humanos e outros agentes físicos autônomos seguindo as regras de comportamento social e as regras associadas ao seu papel na sociedade. Parte essencial de sua construção, portanto, refere-se à sua capacidade de comunicação.

A comunicação entre humanos dá-se de forma verbal e não verbal. Assim, deve-se aliar a capacidade de expressão em linguagem natural à habilidade de compreensão e expressão não-verbal, com especial destaque às emoções transportadas na voz, na face e no gestual.

Avanços na compreensão e expressão verbal têm sido demonstrados; de fato, interfaces por comando de voz (SR, *speech recognition*) estão disponíveis até em telefones celulares e bibliotecas para conversão de texto-para-fala (TTS, *text-to-speech*) de boa qualidade são encontráveis. Entretanto, embora expressivos, tais bibliotecas permitem apenas a execução da entrada-saída: reconhecimento e fonação de palavras. Aspectos prosódicos, como frases interrogativas, ainda não são tratados de forma ampla pelas soluções disponíveis. Os aspectos gramaticais, ligados à compreensão do contexto, ainda são objeto de pesquisas.

No que se relaciona à compreensão e expressão de aspectos não-verbais, particularmente de emoções, os avanços tem ocorrido principalmente nos últimos anos. A deteção de emoções na face e na voz já encontra algoritmos com grau de acerto que se aproxima, em situações controladas, do obtido por seres humanos.

Este artigo propõe uma arquitetura de referência para a construção de um sistema de conversação para robôs sociáveis com a utilização de adaptatividade no gerenciamento de diálogo. A conversação será voltada a "coisas em geral", como aquelas que uma pessoa gosta (ou desgosta), que possui (ou não possui), que aconteceram recentemente: o que se chama, em forma coloquial, como "papo furado" ou "bater papo". Robôs sociáveis com tal habilidade poderão ser utilizados, por exemplo, como acompanhantes para doentes, idosos ou pessoas solitárias em geral. Tal habilidade, entretanto, poderá estar disponível em robôs sociáveis em outros tipos de situação, já que conhecer o interlocutor, no que se relaciona ao que este faz, gosta ou possui, é uma das demonstrações mais fortes de sociabilidade.

As próximas seções deste artigo estarão organizadas da seguinte maneira: na seção 2, faz-se uma breve descrição de trabalhos mais diretamente correlacionados à presente proposta. Na seção 3, o conceito de adaptatividade é sucintamente apresentado. Já na seção 4, apresenta-se a arquitetura proposta. Na seção 5, detalha-se o Gerenciador de Diálogo (*Dialog Manager*), elemento da arquitetura em que a adaptatividade é diretamente empregada. Por fim, na seção 6 estão as considerações finais.

II. TRABALHOS CORRELACIONADOS

Foram considerados como trabalhos correlacionados aqueles que (i)abordam o tema de geração de diálogos em linguagem natural e (ii)baseiam-se em modelos com memória episódica e semântica para a geração do diálogo. Assim, trabalhos baseados em AIML (*Artificial Intelligence Markup Language*), como A.L.I.C.E [3], não foram incluídos, principalmente por não se utilizarem de memória na forma de memória episódica ou semântica, em que pese seu resultado impressionante.

Desde o trabalho pioneiro de Tulving [4], a função de memória nos seres humanos foi entendida na forma mostrada na Fig. 1 [5].

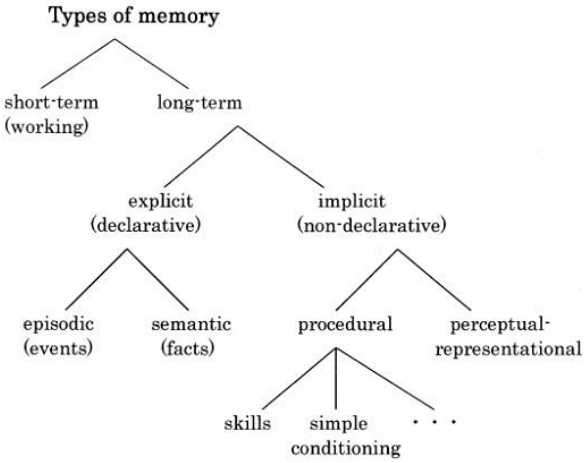


Figura 1. Tipos de memória [5]

M. R. Pereira-Barretto, Escola Politécnica da Universidade de São Paulo (EP-USP), marcos.barretto@poli.usp.br

D. A. Alfenas, Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE), d.alfenas@fdte.org.br

A classificação diferencia os seguintes tipos de memória:

- Memória de curto prazo (*short-term*, *working memory*), responsável pela memória nos acontecimentos das últimas horas;
- Memória explícita, que se divide em memória episódica, responsável pela memorização dos eventos (autobiográfica), e memória semântica, responsável pela memorização de fatos;
- Memória implícita, dividindo-se em memória procedural (responsável por procedimentos) e perceptual-representacional (relacionada a conceitos e abstrações).

Para os propósitos do presente trabalho, têm relevância apenas as memórias episódica e semântica.

Em [6], um agente conversational (CA, *conversational agent*) foi proposto na forma da Fig. 2, voltado a aconselhar estudantes em débito com as mensalidades da Universidade.

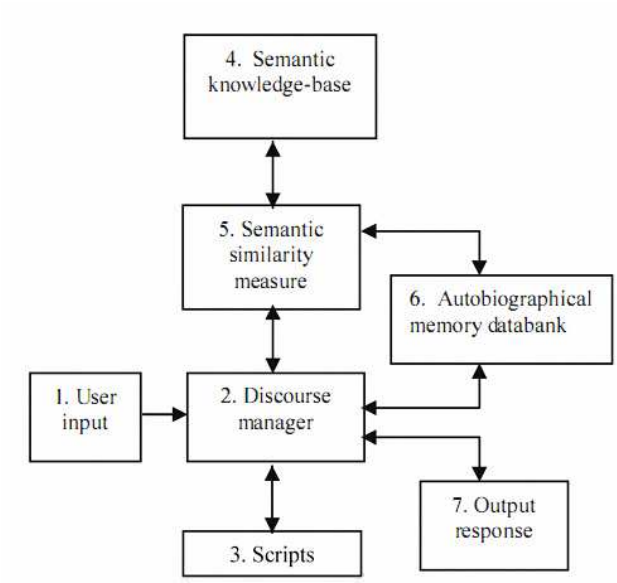


Figura 2. Arquitetura proposta em [6].

O Gerenciador de Diálogo (*Discourse Manager* na fig. 2) está claramente caracterizado, utilizando-se da memória episódica restrita à conversação corrente (*Autobiographical memory databank*) e utilizando-se de um processo de identificação de similaridade de conceitos para determinação do contexto da conversação (*semantic similarity measure*), com base na distância entre conceitos contidos em sua memória semântica (*semantic knowledge base*), que é formada pelos conceitos obtidos da WordNet [7]. O Gerenciador de Diálogo, neste caso, assemelha-se ao utilizado com AIML: identifica-se uma situação a partir do conceito mais similar encontrado e da estrutura da frase, gerando-se a saída a partir de um *template* de resposta.

Em [8] é proposta a arquitetura mostrada na Fig. 3, construída para auxiliar estudantes no aprendizado de redes de computadores. O Gerenciador de Diálogo (*Dialogue Manager* na figura 3) também está claramente identificado, bem como os módulos de memória. Neste trabalho, o Gerenciador de Diálogo é baseado em HTN (*Hierarquical Task Network*), uma técnica comumente utilizada como *task planner* em robôs, que escolhe a situação de diálogo a aplicar. Em seguida, máquinas de estados finitos (FSM, *finite state machine*) são utilizadas para a condução do diálogo propriamente dito.

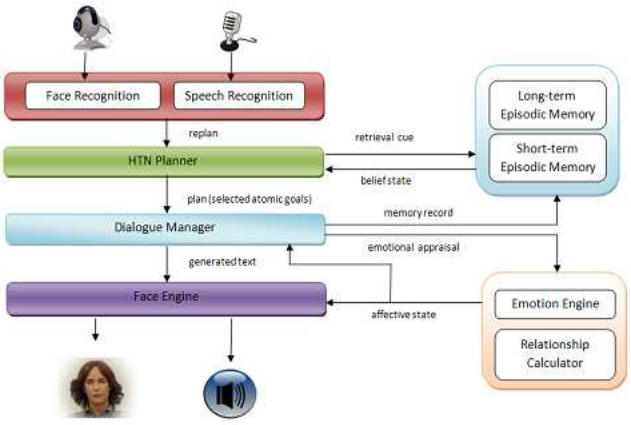


Figura 3. Arquitetura proposta em [8].

A utilização de HTN leva a uma estrutura relativamente fixa da conversação, mostrada na Fig. 4.

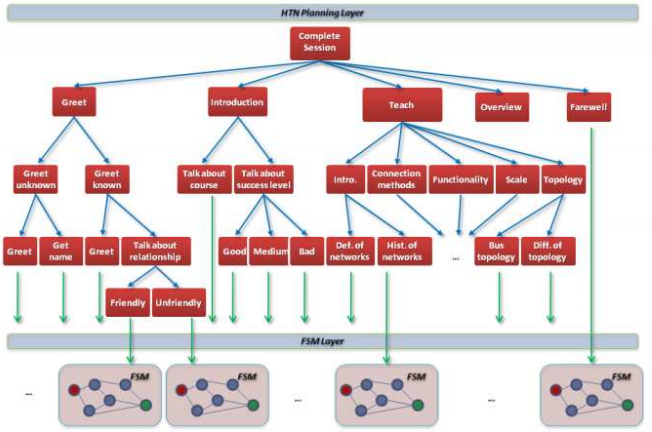


Figura 4. Estrutura da conversação em [8].

Uma estrutura também relativamente fixa, por conta do uso de FSM, é observada nos diálogos: por exemplo, uma pergunta é feita e, dependendo da resposta certa ou errada, passa-se à nova pergunta ou à uma explicação do assunto correlacionado.

Destaca-se que este trabalho leva em consideração, na condução do diálogo, o sentimento do robô em relação ao interlocutor (por exemplo, ressentimento por um grande número de respostas erradas) bem como o histórico de sua inter-relação (por exemplo, sessões anteriores em que o robô ressentiu-se do tratamento recebido), o que está representado na Fig. 3 (*emotion engine*, *relationship calculator*).

III. Máquinas de Markov Adaptativas

Neste trabalho, o conceito de adaptatividade está associado a Máquinas de Markov Adaptativas, utilizando-se a formulação em [9], baseada em [1], [2].

Uma máquina de Markov adaptativa M é definida por uma quíntupla

$$M = (Q, S, T, q_0, F) \quad (1)$$

onde Q é um conjunto finito com n estados, S é o alfabeto de saída, q_0 é o estado inicial da rede, T é um conjunto de transições entre esses estados e F é um conjunto de funções adaptativas.

Toda função adaptativa pertencente a F pode ser definida como uma quádrupla

$$f = (\Psi, V, G, C) \quad (2)$$

onde:

- Ψ é um conjunto de parâmetros formais (ψ_1, ψ_2, ψ_3,..., ψ_m);
- V é um conjunto de identificadores de variáveis (v_1, v_2, ..., v_n), cujos valores são desconhecidos no instante de chamada de f mas que uma vez preenchidos terão seus valores preservados durante toda a execução da função;
- G é um conjunto de identificadores de geradores (g^*_1, g^*_2,..., g^*_n), variáveis especiais que são preenchidas com novos valores, ainda não utilizados pelo autômato, a cada vez que a função é chamada;
- C é sequência de ações adaptativas elementares executadas em f.

Cada transição γ_{ij} é definida como uma quíntupla

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}, p_{ij}, a_{ij}). \quad (3)$$

onde ρ_{ij} é a probabilidade do estado q_j ser atingido estando em q_i, $p_{ij} \in S \cup \{\varepsilon\}$ é um símbolo inserido na cadeia de saída de M e a_{ij} é uma ação adaptativa associada a transição.

Observe que cada transição γ_{ij} é única em T, isto é, não há duas transições diferentes partindo de um mesmo estado q_i e chegando a q_j. Além disso, seja Γ_q como o conjunto de todas as transições que se originam em um estado q. Então

$$\forall q \in Q, \sum_{\gamma \in \Gamma_q} \rho_\gamma = 1 \quad (4)$$

Isto é, a soma das probabilidades de todas as transições iniciando em q é exatamente um, para todo q pertencente a Q.

A ação adaptativa a_{ij} tem a forma

$$a_{ij} = f(\omega_1, \omega_2, ..., \omega_n) \cup \{\varepsilon\} \quad (5)$$

sendo f uma função adaptativa pertencente a F e (ω_1, ω_2, ..., ω_n) uma lista de argumentos que correspondem posicionalmente à lista de parâmetros Ψ declaradas para f.

Define-se um SMA (Sistema de Markov Adaptativo) como um conjunto de máquinas adaptativas de Markov que podem se relacionar através de novas ações adaptativas, introduzidas a seguir. Assim, pode-se estabelecer uma relação de escopo sobre as ações adaptativas definidas para cada máquina, classificando-as em: ações que modifiquem apenas a topologia local da máquina e ações que podem interferir no comportamento de outras máquinas pertencentes ao sistema. Desta forma, cada máquina M^k pertencente a um sistema de Markov adaptativo é definida como uma quíntupla:

$$M^k = (Q^k, \Sigma, T^k, q_0{}^k, F^k) \quad (10)$$

onde Σ é o alfabeto de saída, comum a todas as máquinas do sistema, e Q^k, T^k, $q_0{}^k$ e F^k são como na definição de Q, T, q_0 e F, respectivamente.

IV. Arquitetura proposta

A arquitetura proposta está mostrada na Fig. 5. Os elementos arquiteturais representados têm as seguintes funções:

- ASR (*automatic speech recognition*): transformação da fala em texto. Uma solução existente, comercial ou não, será utilizada na implementação da arquitetura proposta.
- Deteção de silêncio: módulo auxiliar, voltado a auxiliar na identificação de frases completas bem como na atitude pro-ativa do robô em provocar uma conversa se não houver intervenção do interlocutor.
- Deteção de prosódia: determinar o tipo de frase (afirmativa, interrogativa, exclamativa) sendo pronunciada pelo interlocutor.
- Deteção de frase: transformação das saídas dos módulos anteriores em uma forma normalizada, contendo pontuação completa.
- POS (*part-of-speech*) *tagger*: deteção da classe de cada palavra, bem como sua função gramatical, sua flexão e forma primitiva.
- Resolvedor sintático: componente voltado a resolver as relações sintáticas (por exemplo, o pronome "que" em "meu amigo que é um aluno da Escola Politécnica ...")
- Resolvedor de referência: componente voltado a resolver referências contextuais do diálogo, utilizando-se principalmente a memória episódica. Notar que, na arquitetura proposta, a memória episódica assume também as funções de memória de curto prazo.
- Construtor de Diálogo ou Gerenciador de Diálogo: responsável pela condução da conversação.
- Construtor de frases: responsável pela transformação da saída do Gerenciador de Diálogo em uma frase completa.
- Construtor de prosódia: responsável por adicionar elementos prosódicos à saída.
- TTS (text-to-speech): responsável pela geração da fala propriamente dita.
- Memória episódica: responsável pela memória episódica, como conceituada anteriormente
- Memória ontológica: trata-se da memória semântica, mas assim denominada porque, na proposta, este módulo será construído a partir de ontologias.

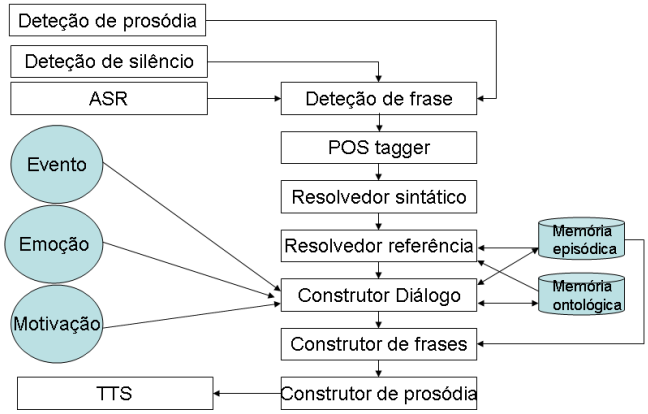


Figura 5. Arquitetura proposta.

A Fig. 5 ainda destaca os módulos:

- Emoção, responsável pela deteção da emoção demonstrada pelo interlocutor. Este módulo já conta com implementações descritas em [10], [11].
- Evento, responsável pela deteção de eventos, como a aproximação ou chegada de um interlocutor ou mesmo extra-conversação, como um ruído muito forte;
- Motivação, responsável pela determinação da tarefa a ser realizada pelo robô. Uma implementação, similar à descrita em [8], está sendo realizada.

O foco deste trabalho é no módulo Gerenciador de Diálogo, cuja construção em curso baseia-se na adaptatividade, como apresentada anteriormente. De fato, propõe-se a utilização de um SMA (Sistema de Markov Adaptativo) em que uma máquina de nivel superior, Condutor, será responsável por definir a condução do diálogo, seguindo idéias de estruturação como realizadas em [2]. Um possível estágio de evolução para esta máquina está mostrado na Fig. 6.

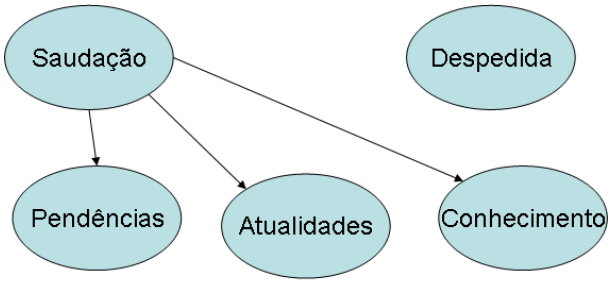


Figura 6. Gerenciador de Diálogo – nivel Condutor.

Como ilustra a Fig. 6, a partir da Saudação inicial, pode-se dirigir o diálogo para Pendências (“Como você foi na prova de Cálculo?”), Atualidades (“Você soube da morte de Dennis Ritchie?”) ou Conhecimento (“Quantos irmãos você tem?”). Em função das regras de adaptação, a conversa eventualmente evoluirá até a Despedida.

Antes de caracterizar o nivel Diálogo, que está subordinado ao nivel Condutor, torna-se importante apresentar, de forma sucinta, a organização proposta para a memória ontológica. Sua estrutura está baseada nos blocos *vwCRK* (*Concept-relation-keyword*) propostos em [12]. Um bloco vwCRK pode ser representado como mostra a Fig. 7.

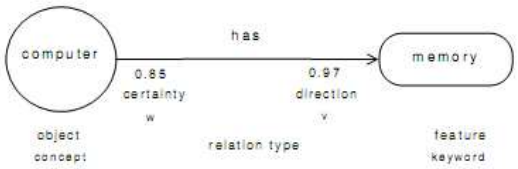


Figura 7. Bloco vwCRK [12].

Os parâmetros **w** e **v** representam, respectivamente, a certeza e a força da relação. Por exemplo, na frase “o cisne é branco”, tem-se **w** próximo de +1, enquanto em “o cisne é amarelo” tem-se **w** perto de -1. Entrentanto, ambas as frases têm **v** próximo de 1, significando que tem-se certeza deste conhecimento. Entretanto, a memória ontológica não é simplesmente textual; na realidade, as memórias episódica e semântica são, na realidade, multimidia. A conceituação completa da memória ontológica pertencente à arquitetura não será aqui realizada; para os propósitos deste artigo, pode-se considerá-la como textual e semelhante à proposta em [7].

A memória semântica proposta está organizada nas seguintes ontologias:

- *Web of everything* (WoE), ontologia de referência, contendo todo o conhecimento do robô sociável sobre a estrutura do mundo;
- *Web of myself* (WoMe), conhecimento do robô sobre si mesmo
- *Web of others* (WoO), conhecimento do robô sobre cada conceito (pessoa, animal, lugar, livro, etc) com quem ele teve contato, seja virtual ou físico.

Assim, durante um diálogo, o robô utiliza o conhecimento que tem do mundo (WoE) para determinar se deve continuar a explorar aspectos de um mesmo conceito ou se deve alternar para outro conceito, anotando seu conhecimento na WoO específica do interlocutor e sua relação com o conceito em WoMe. Como exemplo, considere o conteúdo da WoE como mostrado na Fig. 8.

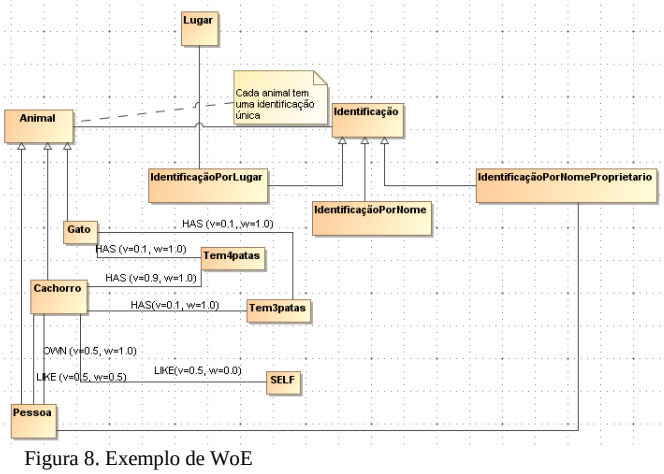


Figura 8. Exemplo de WoE

Supondo-se que em nível Diálogo esteja-se no estado Conhecimento. Uma possível realização de máquina para este estado, visando explorar a WoE, está mostrada na Fig. 8.

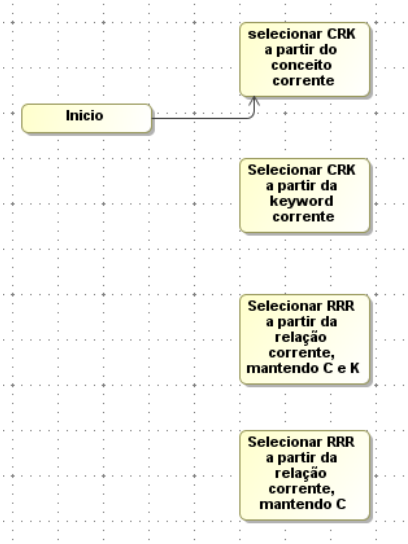


Figura 9. Um possível estágio de evolução da MMA.

Na Fig. 9, mostra-se o estágio inicial, que a única transição possível é a mostrada na figura. A partir desta transição, as funções de adaptação farão com que sejam explorados os caminhos possíveis na WoE. As funções de adaptação são condicionadas aos módulos constantes da Figura 5:

- O módulo Evento pode causar o fim da execução da submáquina corrente: por exemplo, a chegada de uma pessoa pode levar à interrupção da execução da submáquina Conhecimento, causando a adaptação de aumentar a probabilidade de acionar-se a máquina Despedida.
- O módulo Emoção pode detectar o desagrado do interlocutor, causando uma adaptação que eleve a probabilidade de alteração do conceito sendo explorado.
- O módulo Motivação pode causar uma ação de adaptação que leve ao fim da conversa, por cansaço.

Então, um possível diálogo que explore a WoE poderia ser aquele mostrado na Fig. 10, onde R representa o robô e J representa uma pessoa, chamada J.

R	1. Decisão MMA: Selecionar CRK (conceito-relação-keyword) a partir do conceito corrente (inicialmente, igual a Pessoa) 2. Selecionado: Pessoa-LIKE-Cachorro 3. Perguntar: Você gosta de cachorro?
J	Sim! (adicionar a WoO::J:: J-LIKE(v=1.0,w=1.0)-Cachorro
R	1. Adicionar a WoO::J:: J-LIKE(v=1.0,w=1.0)-Cachorro 2. Decisão MMA: Selecionar RRR (relação-relação-relação) a partir da relação corrente (LIKE) 3. Selecionado LIKE-OWN 4. Perguntar: Você possui um cachorro?
J	Sim!
R	1. Adicionar a WoE::J::J-OWN(v=1.0,w=1.0)-Cachorro 2. Erro: não pode criar identificador para Cachorro porque não conhece o nome do cachorro. 3. Perguntar: Qual o nome do cachorro?
J	Gaia.
R	1. Adicionar a WoE::J::J-OWN(v=1.0,w=1.0)-Cachorro 2. Decisão MMA:Selecionar CRK a partir do conceito corrente 3. Selecionado Pessoa-Like-Gato. 4. Perguntar: Você gosta de gato?

Figura 10. Um possível diálogo.

V. CONCLUSÃO

O artigo propôs uma arquitetura para a construção de diálogo humano-robô sociável, baseado em conceitos de adaptatividade. Assim como em composição musical, como demonstrado em [2], a adaptatividade apresenta-se como uma técnica viável para a construção de comportamentos que possam ser considerados como criativos (ou humanos), de forma aceitável. O modelo proposto está sendo implementado, para a comprovação desta tese.

REFERÊNCIAS

[1] J. J. Neto, "Contribuições à metodologia de construção de compiladores", *Post-Doctoral Tese de Livre Docência*, EPUSP, São Paulo, 1993.

[2] B. A. Basseto, "Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos", *Dissertação de Mestrado*, EPUSP, São Paulo, 2000.

[3] Alicebot Org. Disponível em www.alicbot.org. Acesso em 10 de outubro de 2011.

[4] E. Tulving, "Episodic Memory: From Mind to Brain". *Annual Review of Psychology 53*, pp. 1–25, 2002.

[5] Y. Miyashita, "Cognitive memory: cellular and network machineries and their top-down control". *Science Magazine*, vol.306, pp.435-440, 2004.

[6] K. O´Shea, K. Crockett e Z. Bandar, "A proposed mechanism for memory simulation within a semantic-based conversational agent framework". *IEEE International Conference on Systems, Man and Cybernetics*, Istanbul, Turkey, 2010.

[7] G. A. Miller, "WordNet: A Lexical Database for English", *Comm. ACM, Vol. 38, no. II*, 1995, pp. 39-41.

[8] Z. Kasap e N. Magnenat-Thalmann, "Towards episodic memory-based long-term affective interaction with a human-like robot". *19th International Symposium on Robot and Human Interactive Communication*, Viareggio, Italy, 2010.

[9] D. A. Alfenas, D. P. Shibata, D.P, J. J. Neto e M. R. Pereira-Barretto, "Sistemas de Markov Adaptativos: formulação e plataforma de desenvolvimento", *submetido ao WTA2012*.

[10] D. R. Cueva, R. A. M. Gonçalves, F. G. Cozman e M. R. Pereira-Barretto, "Fusão de observações afetivas em cenários realistas". *Anais do ENIA (Encontro Nacional de Inteligência Artificial)*, Natal, RN, Brasil, 2011.

[11] Gonçalves, R.A.M.; Cueva, D.R.; Pereira-Barretto, M.R.; Cozman, F.G. "Determinação da emoção demonstrada pelo interlocutor". Anais do ENIA (Encontro Nacional de Inteligência Artificial), Natal, RN, Brasil, 2011.

[12] J. Szymanski e W. Duch, "Knowledge Representation and Acquisition for Large-Scale Semantic Memory", *International Joint Conference on Neural Networks*, 2008.

Daniel Assis Alfenas é formado em Engenharia da Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 2004. Trabalhou, nos últimos dez anos, nas diversas áreas do desenvolvimento de sistemas, desde a definição da demanda até a implantação do sistema. Recentemente tem trabalhado como coordenador técnico no desenvolvimento de sistemas complexos que envolvem simulação, otimização e interfaces ricas. Atualmente, é mestrando no Laboratório de Técnicas Adaptativas da EPUSP e a área de pesquisa é a aplicação da tecnologia adaptativa no diálogo em linguagem natural entre usuários e sistemas.

Marcos Ribeiro Pereira-Barretto é graduado em Engenharia Elétrica (1983), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Mecânica (1993) pela Escola Politécnica da Universidade de São Paulo. É professor da Escola Politécnica da USP desde 1986. Atua nas seguintes áreas de pesquisa: robôs sociáveis, computação afetiva e arquitetura de sistemas para aplicações críticas como Automação Industrial.

Uso de Programação Orientada a Aspecto no Desenvolvimento de Aplicações que utilizam conceitos de Tecnologia Adaptativa

R. A. Casachi, A. R. Camolesi

Rafael.Casachi@raizen.com, camolesi@femanet.com.br
Coordenadoria de Informática, Fundação Educacional do Município de Assis, Brasil

***Resumo* — Este trabalho tem por objetivo apresentar os conceitos de Tecnologia Adaptativa, de Programação Orientada a Aspecto e o uso da Programação Orientada a Aspectos no desenvolvimento de aplicações que utilizam-se dos conceitos de Tecnologia Adaptativa.**

***Palavras Chaves*— Tecnologia Adaptativa, Programação Orientada a Aspectos, Sistemas de Informação.**

I. INTRODUÇÃO

SIMPLICIDADE e eficiência sempre foram necessidades primordiais no desenvolvimento de software. As linguagens estão se desenvolvendo através destes fatores para facilitar as tarefas de implementação e otimizar os projetos.

As evoluções nas linguagens de programação chegaram até ao Paradigma de Programação Orientada a Objetos (POO) que é o método mais utilizado em desenvolvimento atualmente. A POO possui a capacidade de abstrair os objetos reais e abstratos em classes garantindo melhor visibilidade ao código. Porém, mesmo com seus recursos, a POO não consegue tratar algumas características dos sistemas corretamente, como por exemplo, o controle de acesso (logging).

Em 1997, Gregor Kiczales e alguns cientistas do PARC (Xerox Palo Alto Research Center) desenvolveram o Paradigma de Programação Orientada a Aspecto (POA) [1]. Tal paradigma tem por objetivo solucionar as falhas da Programação Orientada a Objeto com relação à unificação dos códigos secundários (crosscutting concerns) ao código principal. Segundo [2] o mecanismo da Programação Orientada a Aspecto busca separar os interesses transversais das classes em módulos bem definidos e centralizados, nos quais um analista pode dedicar-se a um interesse de forma independente.

Mesmo com a POA, a necessidade de uma linguagem de componentes eficiente é fundamental. Esta linguagem de componentes pode ser um paradigma orientado a objetos ou pode ser uma linguagem estruturada, conforme descrito em [3]. Porém a POO, possui um melhor nível de abstração e possui um melhor suporte ao paradigma de Kiczales [1].

Uma outra característica importante nas aplicações complexas é a possibilidade das mesmas automodificar seu comportamento. Neste contexto o uso da tecnologia adaptativa tem apresentado bons resultados e permite aos desenvolvedores criar aplicações que se automodificam durante a sua execução. Neste trabalho será apresentado um estudo que visa a aplicação da Programação Orientada a Aspecto na implementação de aplicações adaptativas.

Este trabalho está organizado da seguinte forma, a seção II traz os principais conceitos da Programação Orientada a Aspectos e a sua forma de utilização. Na sequencia, na seção III são apresentados os conceitos de Tecnologia Adaptativa e uma comparação entre os seus conceitos e a orientação a aspectos será descrito. Um estudo de caso, com o objetivo de ilustrar a utilização da Programação Orientada a Aspectos no desenvolvimento de aplicações que utilizam os conceitos de Tecnologia Adaptativa será realizado na seção IV. Por fim, na seção V serão tecidas algumas conclusões e trabalhos futuros.

II. SEPARAÇÃO DE INTERESSES

Um sistema possui características, princípios, requisitos, algoritmos ou idéias similares. Este conjunto de similaridades chama-se interesses.

A separação destes interesses constitui-se de um método para dividir estes requisitos a fim de melhor solucionar o problema. "A estratégia de dividir para conquistar é bastante comum para a solução de problemas de qualquer natureza" [4].

A proposta de dividir os problemas de um sistema para melhorar a solução tem sido utilizada desde a programação estruturada. Existiam diversos algoritmos conhecidos como *Divide-and-conquer algorithms* (do inglês, Algoritmos Dividir-e-Conquistar) que dividiam o problema em subproblemas para tornar mais fácil a sua resolução. Estes subproblemas eram solucionados independentemente e combinados para solucionar o problema original [5].

Em um sistema comum, os interesses estão espalhados em todo o código (*scattering code*). Isto dificulta a manutenção do sistema, além de deixar o código ilegível. Na Figura 1, podemos verificar como um interesse se espalha em diversas classes:

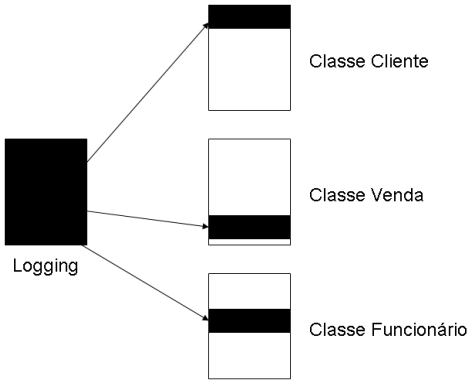


Figura 1. Código Espalhado por Diversas Classes

Além do código espalhado, pode-se verificar que uma classe acumula vários interesses, ocorrendo outro fenômeno chamado código emaranhado (*tangled code*), conforme exemplo da Figura 2.

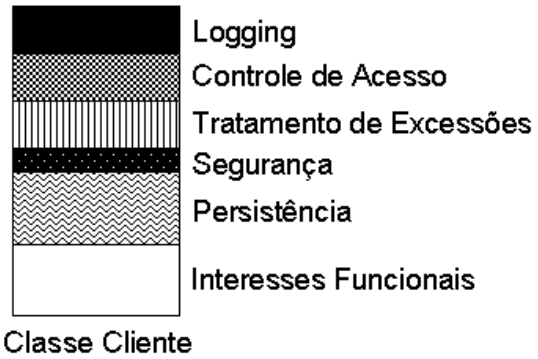


Figura 2. Código com interesses emaranhados no interesse funcional

A fim de evitar que ocorrências como estas aconteçam, separamos os nossos interesses em dois: Interesses Funcionais e Interesses Sistêmicos.

A. *INTERESSES FUNCIONAIS*

Os interesses funcionais (ou *core concerns*) são as características do domínio da aplicação, ou seja, são requisitos para que a aplicação atenda aos requisitos de funcionamento do sistema.

Quando um interesse funcional está desorganizado e ilegível, toda a aplicação pode ser comprometida e a probabilidade de erro pode ser muito grande.

B. *INTERESSES SISTÊMICOS*

Os interesses sistêmicos são os requisitos que dão suporte aos interesses funcionais. Estes interesses sistêmicos são à base do paradigma de programação orientado a aspecto, assim, necessitam ser implementados separadamente para serem controlados e organizados [6].

Para identificar os interesses sistêmicos é necessário analisar o esqueleto do programa (em um sistema que não foi implementado em POA) ou localizar as características que não fazem parte do escopo da funcionalidade.

C. *ASPECTJ E O PROCESSO DE COMBINAÇÃO*

Os códigos funcionais, implementados em uma linguagem de componentes, e os códigos sistêmicos, implementados em uma linguagem aspectual, são unidos através de um processo chamado combinação (ou *Weaving*, em inglês).

O processo de combinação utiliza de um combinador, ou Weaver, para unir os interesses funcionais e sistêmicos em um código intermediário. Após gerar este código intermediário, o arquivo é compilado por um compilador, conforme mostra o Diagrama de Petri da Figura 3.

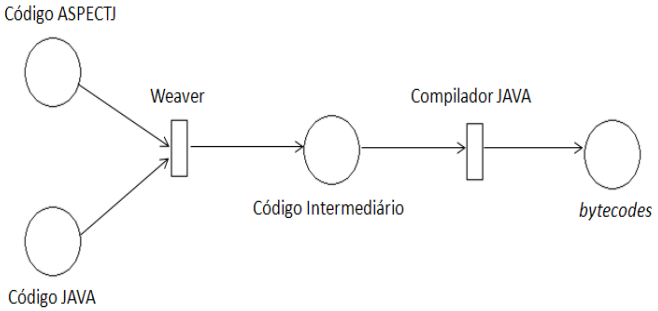


Figura 3. Representação do processo de combinação com uma Rede de Petri

O combinador pioneiro é o AspectJ, criado em 1997 junto com o paradigma POA. O AspectJ é utilizado para ser o combinador aspectual da linguagem de componentes Java e a linguagem de aspectos AspectJ. Em 2002, a IBM agregou o projeto AspectJ à IDE Eclipse, melhorando o suporte ao combinador.

D. *PONTOS DE JUNÇÃO (JOIN POINTS)*

Os interesses separados devem ser unidos no processo de combinação através de um ponto de junção.

Conforme descrito em [6] *"O ponto de junção é um ponto bem definido na execução de um sistema"*, ou seja, eles são pontos selecionados para controlar o fluxo do programa.

Para que esta união seja bem-sucedida a definição de dois elementos no aspecto é essencial: o ponto de atuação e o adendo.

E. *PONTOS DE ATUAÇÃO (POINTCUTS)*

O ponto de atuação é o componente da POA que identifica em quais pontos de junção do código principal o aspecto será combinado.

A estrutura de um ponto de atuação é composta pela palavra-chave "*pointcut*" seguido pelo tipo de acesso que o ponto de atuação irá possuir (privado ou público), o nome do ponto de junção com seus possíveis parâmetros e as assinaturas dos pontos de junções capturados com os seus designadores. No exemplo abaixo, o ponto de atuação chamado registra será executado quando o método *public String getNome()*, da classe Cliente, for chamado (*call*).

```
pointcut registra() : call(public String
        Cliente.getNome());
```

Um designador é um elemento encontrado no ponto de atuação que identifica o tipo de comportamento que um ponto de junção deve ter para que possa ser capturado. A Figura 4 apresenta a lista de todos os designadores de um ponto de atuação.

Designador	Descrição
Execution	Corresponde à execução de um método ou de um construtor
Call	Corresponde à chamada para um método ou para um construtor
Initialization	Corresponde à inicialização de um objeto, representada pela execução do primeiro construtor para uma classe
Handler	Corresponde à manipulação das excessões
Get	Corresponde à referência para um atributo de uma classe
Set	Corresponde à definição de um atributo de um classe
This	Retorna o objeto associado com o ponto de junção em particular ou limita o escopo de um ponto de junção utilizando um tipo de classe
Target	Retorna o objeto alvo de um ponto de junção ou limita o escopo do mesmo
Args	Expõe os argumentos para o ponto de junção ou limita o escopo de um ponto de atuação
Cflow	Retorna os pontos de junção na execução do fluxo de outro ponto de atuação
Cflowbelow	Retorna os pontos de junção na execução do fluxo de outro ponto de atuação, exceto o ponto corrente
Staticinitialization	Corresponde à inicialização dos elementos estáticos de um objeto
Withincode	Corresponde aos pontos contidos em um método ou construtor
Within	Corresponde aos pontos de junção contidos em um tipo específico
If	Permite que uma condição dinâmica faça parte de um ponto de atuação
Adviceexecution	Corresponde aoo adendo (advice) do ponto de junção
Preinitialization	Corresponde à pré-inicialização de um ponto de junção

Figura 4. Lista de designadores de um ponto de atuação (In: GOETTEN; WINCK, 2006, P87)

F. ASSINATURA DE UM PONTO DE JUNÇÃO

A função da assinatura é descrever em qual ponto de junção, o aspecto deve ser combinado. Esta assinatura pode variar conforme o tipo de designador utilizado. A Figura 5 descreve a lista das assinaturas por designador.

Designador	Assinatura
Execution	execution(<tipo_acesso> <valor_retorno> <classe>.<nome_metodo>({<lista_parametros>}))
Call	call(<tipo_acesso> <valor_retorno> <classe>.<nome_metodo>({<lista_parametros>}))
Initialization	initialization(<tipo_acesso> <classe>.new({<lista_parametros>}))
Handler	handler(<tipo_excessao>)
Set	get(<tipo_campo> <classe>.<nome_campo>)
Set	set(<tipo_campo> <classe>.<nome_campo>)
This	this(<tipo ou identificador>)
Target	target(<tipo ou identificador>)
Args	args(<tipo ou identificador>,..)
Cflow	cflow(<pointcut>)
Cflowbelow	cflowbelow(<pointcut>)
staticinitialization	staticinitialization(<typePattern>)***
Withincode	withincode(<tipo_acesso> <valor_retorno> <classe>.<nome_metodo>({<lista_parametros>}))
Within	within (<typePattern>)***
If	if(<Expressão Booleana>)
Adviceexecution	adviceexecution()
preinitialization	preinitializantion(<tipo_acesso> <classe>.new({<lista_parametros>}))
Métodos que geram exceções	<designador>(<tipo_acesso> <classe>.<nome_metodo>({<lista_parametros>}) throws <excessao_a_ser_tratada>)

Figura 5. Lista de Assinaturas por designador

G. ADENDO (ADVICE)

Quando o ponto de atuação une o aspecto ao ponto de junção um adendo entra em operação. Um adendo é um bloco de código similar ao método que é executado num momento auto-configurado.

O adendo possui todo o código que será combinado à aplicação. Um aspecto pode ter vários adendos de diversos tipos, porém eles não possuem nome de referência, sua única referência é o ponto de atuação.

Um adendo varia conforme o momento de sua execução. Este pode ter um modificador *before*, *after* ou *around* [7].

O adendo before é mais simples. Este é executado antes da ocorrência do ponto de junção e não possui nenhum critério (como no adendo *after*) ou possibilidade de seu contexto ser alterado (como no adendo *around*) [4].

O adendo *after* é o bloco de código executado no fim da computação do ponto de junção e é divido em três subtipos de acordo com o critério da execução. Se o objetivo é utilizar o adendo apenas após a computação correta do ponto de junção, então se deve utilizar o adendo after returning. Se há a necessidade de execução do adendo depois da computação sem sucesso de um ponto de junção, então se deve utilizar o adendo after throwing. Porém se o resultado da computação não influenciar, pode-se utilizar apenas *after* [4].

O último tipo de adendo é o around que é executado durante a execução do ponto de junção ou bloco de código, ou seja, o adendo around cerca toda a execução do ponto de junção. Para executar o ponto de junção é necessário discriminar a palavra-chave *proceed()* com os mesmos argumentos coletados no corpo do adendo, caso este comando não for encontrado no adendo, o ponto de junção é contornado [4].

A sequência de execução de um adendo é descrita no autômato da Figura 6, o estado inicial *q0* é o momento em que o ponto de atuação encontra um ponto de junção. A partir deste momento, os adendos do aspecto seguem a estrutura descrita. A sequência é simples, qualquer adendo do tipo *before* no aspecto é executado primeiro e antes do ponto de junção. Um adendo *around* é executado após de um adendo *before* e quando houver um comando *proceed* o ponto de junção é executado, e depois, o adendo *around* é finalizado. Após a execução do ponto de junção ou do adendo *around* pode haver um adendo *after* que pode ser de três tipos: quando o ponto de junção foi executado com sucesso (*after returning()*), quando o ponto de junção retornou erro (*after throwing()*) ou quando o estado final do ponto de junção não importar (*after()*).

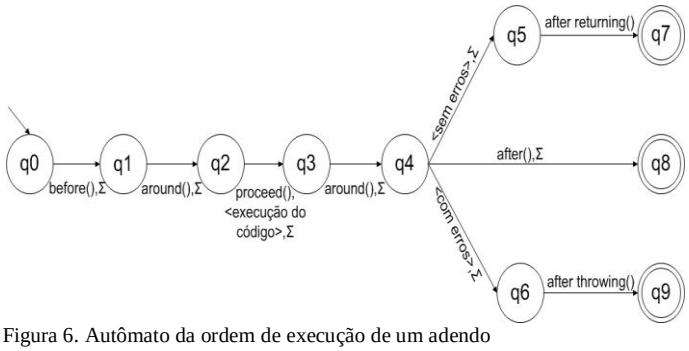


Figura 6. Autômato da ordem de execução de um adendo

H. ASPECTO

Um aspecto é a unidade básica do POA, assim como o objeto é a unidade básica do POO. *"Aspecto é uma parte de um programa POA que separa os interesses ortogonais dos interesses principais"* [6].

Um aspecto é como uma classe. Ela possui os atributos modificadores de acessos (*private*, *protected* e *public*), a palavra-chave *aspect* e o nome de referência. E seu bloco de código abriga todos os elementos da programação orientada a aspecto.

III. TECNOLOGIA ADAPTATIVA

Um comportamento é denominado adaptativo sempre que este modifica-se dinamicamente, em resposta aos estímulos de entrada, sem interferência de agentes externos e de seus usuarios. Historicamente, o estudo da tecnologia adaptativa teve início com trabalhos realizados na busca de uma forma eficaz e de baixa complexidade para a resolução do problema da análise sintática de Linguagens Livres de Contexto [8]. Diversos trabalhos foram realizados nesta linha e em [9] foi apresentado o autômato e o transdutor adaptativo como dispositivos de reconhecimento e transdução sintática.

Com base no trabalho desenvolvido em [9], foram realizados outros trabalhos aplicando o uso da tecnologia adaptativa no projeto de sistemas reativos. Em [10] é apresentada uma evolução da notação de *Statecharts* [11], na qual foram acrescentadas algumas características provenientes da teoria de Autômatos Adaptativos.

Dispositivos adaptativos devem ser auto-modificáveis e as possíveis mudanças no comportamento do dispositivo devem ser conhecidas por todas as atividades desempenhadas pelo mesmo, em qualquer passo de operação na qual as mudanças ocorrem. Entretanto, dispositivos adaptativos devem ser capazes de eliminar todas as situações que causam possíveis modificações não desejadas e para adequar reações há imposições que correspondem a mudanças no comportamento do dispositivo. Os dispositivos adaptativos considerados têm seu comportamento baseado na operação de dispositivos subjacentes não adaptativos que são descritos por um conjunto finito de regras. Um dispositivo adaptativo dirigido por regras pode ser obtido anexando ações adaptativas as regras do formalismo subjacente.

Em [12] foi proposta uma representação genérica para formalismos adaptativos que permite representar e manipular dispositivos adaptativos dirigidos por regras e estados. A principal característica desta formulação é que preserva totalmente a natureza do formalismo adaptativo, desde que o dispositivo adaptativo resultante pode ser facilmente entendido por pessoas que possuem conhecimento em relação ao dispositivo subjacente.

A. Dispositivos Adaptativos Dirigidos por Regras

A formalização apresentada em [12] fundamenta-se em um mecanismo adaptativo (AM) que envolve o núcleo de um dispositivo subjacente, não adaptativo (ND). Desta forma, um dispositivo adaptativo (AD) é definido formalmente por um óctuplo AD= (C, AR, S, c_0, A, NA, BA, AA).

Nesta formulação C é o conjunto de todas as possíveis configurações de ND e $c_0 \in C$ é a sua configuração inicial. S é o conjunto de todos os possíveis eventos de que se compõem a cadeia de entrada de AD e o conjunto A representa as configurações de aceitação para ND.

Os conjuntos BA e AA são conjuntos de ações adaptativas. NA é um conjunto de todos os símbolos que podem ser gerados com saídas por AD, em resposta à aplicação de regras adaptativas. AR é o conjunto das regras adaptativas que definem o comportamento adaptativo de AD e é dado por uma

relação Ar ⊆ BA×C×S×C×NA×AA, na qual, ações adaptativas modificam o conjunto corrente de regras adaptativas AR de AD para um novo conjunto AR adicionando e/ou eliminando regras adaptativas em AR.

Com base na formulação geral para dispositivos adaptativos foi apresentado em [13] a formulação geral para o Autômato de Estados Finitos Adaptativo. Um Autômato Finito (dispositivo subjacente) é um dispositivo simples e está intimamente relacionado com a classe das Linguagens Regulares, conforme denominado na hierarquia de Chomsky. Em [14] um Autômato de Estados Finitos (AEF) é constituído por um conjunto de estados e por uma função de transição que informa quais transições o mecanismo deve executar, a partir de um estímulo obtido na sua cadeia de entrada.

Um dispositivo torna-se adaptativo ao agregar uma camada adaptativa envolvendo o seu núcleo subjacente. Nesse contexto define-se um Autômato de Estados Finito Adaptativo (AEF_{Adp}) por meio do acréscimo de uma camada adaptativa envolvendo o seu núcleo subjacente. A Figura 7 ilustra a estrutura de um AEF_{Adp} constituída por uma camada adaptativa contendo funções anteriores e posteriores e suas respectivas ações.

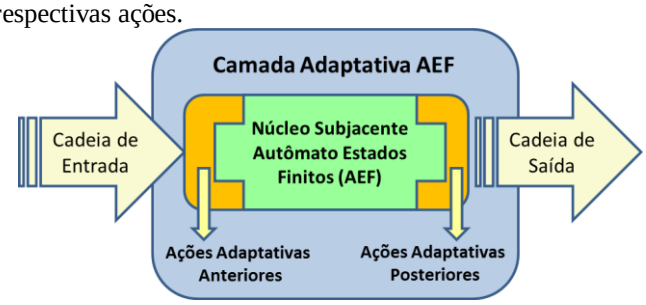


Figura 7. Dispositivo Autômato de Estados Finitos Adaptativo

B. Tecnologia Adaptativa e Programação Orientada a Aspectos.

Ao fazer uma relação dos conceitos de Tecnologia Adaptativa aos conceitos de Orientação a Aspectos encontra-se uma forte relação entre estas duas tecnologias. Como dito anteriormente a orientação a aspecto se fundamenta em aspectos e na combinação destes. Ao projetar uma aplicação orientada a aspectos devemos representar os seus pontos de junção (*join points*) e os pontos de atuação (*pointcuts*). Um ponto de atuação une o aspecto ao ponto de junção um adendo entra em operação. Um adendo é um bloco de código similar ao método que é executado num momento auto-configurado. Os adendos podem ser de três tipos; *before*, *after* ou *around*. Ao realizar o mapeamento dos conceitos de Tecnologia Adaptativa aos conceitos de Programação Orientada a Aspecto realiza-se o mapeamento das ações adaptativas anteriores (*before*) aos adendos *before* de programação orientada a aspecto. As funções adaptativas posteriores (*after*) devem ser mapeadas aos adendos *after* , e por fim, os adendos *around* mapeiam os trechos de código de execução normal (tecnologia adaptativa) que são envolvidos pelas funções adaptativas conforme ilustrado na Figura 8.

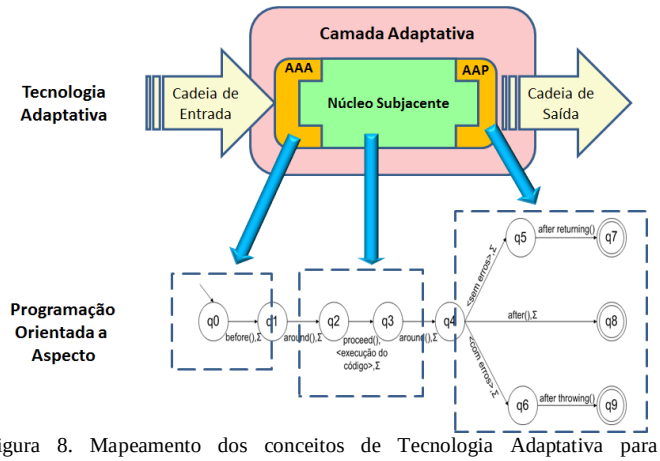


Figura 8. Mapeamento dos conceitos de Tecnologia Adaptativa para Programação Orientada a Aspectos

IV. ESTUDO DE CASO

Com o intuito de aplicar os conceitos do Paradigma de Programação Orientado a Aspecto associado aos conceitos de Tecnologia Adaptativa para o desenvolvimento de aplicações foi realizado um estudo de caso com o foco no desenvolvimento de um Sistema para Gestão de Bibliotecas. Em tal sistema os aspectos atuam no código principal para modificar seu comportamento.

A aplicação desenvolvida consiste em um software para gestão que tem como principais funcionalidades manter os cadastros de alunos, de funcionários, de livros e exemplares. Esta aplicação permite o empréstimo de exemplares e sua futura devolução, além de proibir o empréstimo aos alunos inadimplentes. Para manter a integridade, o software deve restringir o acesso de alguns usuários a certas funcionalidades do sistema de acordo com seu cargo. A Figura 9 apresenta o caso de uso geral do referido sistema.

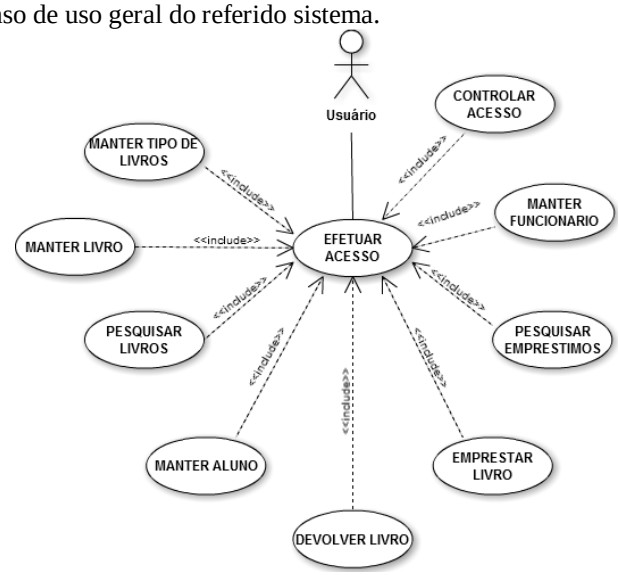


Figura 9. Caso de uso geral Sistema Gestão Biblioteca

O sistema foi projetado para ser implementado na Linguagem Java, para o armazenamento dos dados foi utilizado o banco de dados HSQLDB e o framework de persistência a dados

Hibernate[1] foi utilizado para auxiliar nas operações de manipulação dos dados necessárias ao sistema. Na Figura 10 é apresentado um Diagrama de Classes resumido que ilustra as principais classes que compõe o modelo objeto-relacional da aplicação desenvolvida.

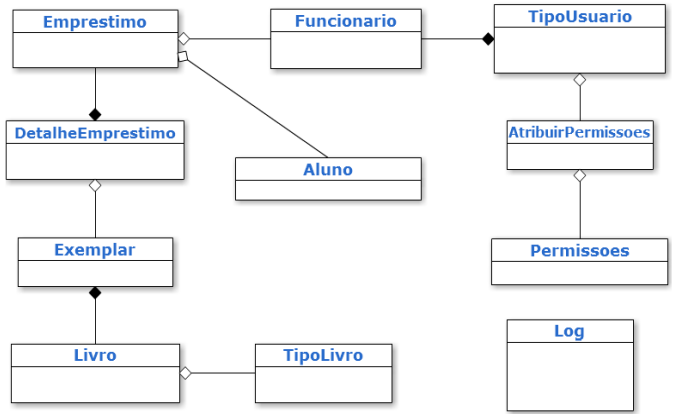


Figura 10. Diagrama de classes resumido

Ao organizar a arquitetura do sistema utilizou-se como base o conceito de separação de interesses, neste contexto o software foi dividido em duas partes: o pacote *Componentes* e o pacote *Funcionalidades*. Na Figura 11, pode-se verificar a estrutura do sistema, onde todos os aspectos estão atuando diretamente nos objetos instanciados nas classes do pacote ***Interfaces***. Este pacote contém os componentes do sistema que interagem com o usuário. As classes do pacote ***Interfaces*** comunicam-se com o banco de dados por meio do framework de persistência *Hibernate* que dá suporte a manipulação de dados no sistema.

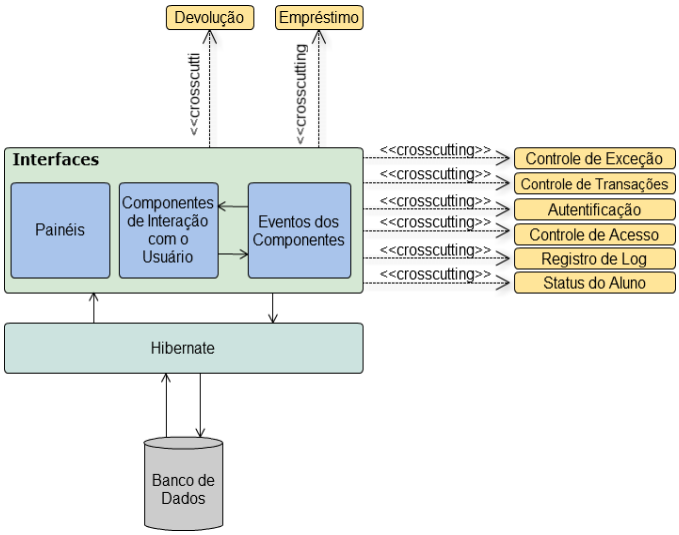


Figura 11. Arquitetura do Sistema de Gestão Bibliotecas

Devido a não existência de métodos e técnicas específicos para a modelagem de sistema que utilizam-se dos conceitos de Orientação a Aspectos, foi utilizado o Diagrama de Casos de Uso (UML) para descrever o comportamento do sistema. Na Figura 12 é apresentado um caso de caso de uso do sistema no qual são destacados os interesses ortogonais (elipses em laranja). Por exemplo, os interesses *Alterar Status do Exemplar*, *Consultar Multa* e *Impressão da Fatura* são interesses que possuem o mesmo objetivo, o de manter a integridade na regra de negócio "devolver livros". Desta forma tais interesses foram agrupados em um único aspecto. Este mesmo critério foi utilizado para agrupar os interesses *Determinar Prazo de Devolução*, *Alterar Status do Exemplar* e *Checar Disponibilidade do Exemplar* no aspecto *Empréstimo*. O aspecto *Status do Aluno* agrupa os interesses *Alterar Status do Aluno* e *Checar Disponibilidade do Aluno*.

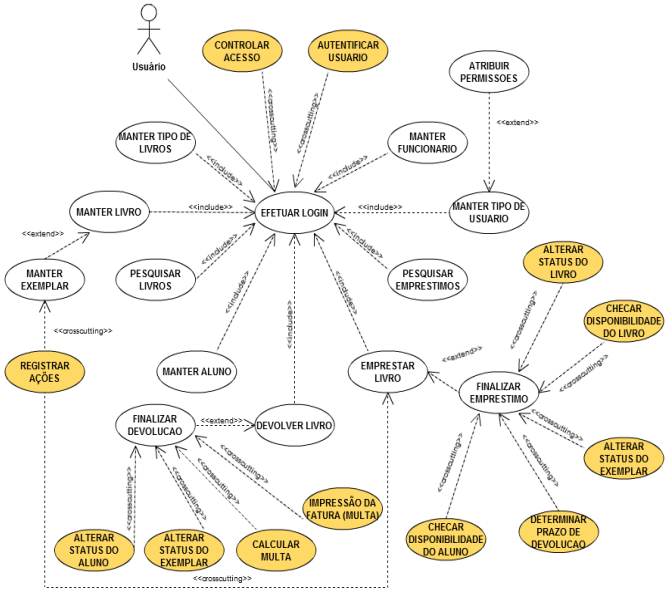


Figura 12 - Diagrama de Casos de Uso

Existem aspectos que não são possíveis de ser representados no caso de uso, como os aspectos que controlam exceções e transações. Os aspectos de controle de exceção capturam os pontos de junção que necessitam de um bloco de tratamento de exceções (*try/catch)* e inserem os mesmos no código no momento da combinação. O mesmo acontece com o controle de transações, o aspecto insere um código de iniciação da transação antes da sua utilização, e conforme o resultado da transação insere um comando de *commit* do banco ou um *rollback*, conforme código abaixo:

```
before() : transaction(){
  HibernateUtil.begin();
}

after() returning() : transaction(){
  HibernateUtil.commit();
}

after() throwing() : transaction(){
  HibernateUtil.rollback();
}
```

Figura 13. Trecho de Código Orientado a Ascpecto

Durante a implementação, os componentes foram programados primeiro utilizando o paradigma de programação orientado a objetos, mais especificamente a linguagem de programação Java. Os aspectos programados por último e

[1] O **Hibernate** é um framework para o mapeamento objeto-relacional escrito na linguagem Java disponível em http://www.hibernate.org/

divididos em dois tipos de acordo com suas funcionalidades: suporte ao sistema e suporte às regras de negócio.

A. Implementação Funcionalidade de Acesso ao Sistema

O código abaixo descreve o aspecto de controle de acesso onde o ponto de junção é a chamada do método *checarPermissao* na classe *Principal*. Este método atribui falso para todos os acessos no sistema. Porém quando este método é chamado, o aspecto altera os valores que serão atribuidos ao acesso.

```
public void buscarPermissoes(Funcionario user){
AtribuirPermissaoDao dao = new
               AtribuirPermissaoDao();
listPermissoes = (List<AtribuirPermissoes>)
               dao.pesquisar(user.getTipoUsuario());

String perm[] = new String[9];
perm[0] = "CADASTRO DE LIVROS";
perm[1] = "PESQUISA DE LIVROS";
perm[2] = "CADASTRO TIPO DE LIVRO";
perm[3] = "CADASTRO DE ALUNOS";
perm[4] = "CADASTRO DE FUNCIONÁRIOS";
perm[5] = "CADASTRO DE TIPO DE USUARIO";
perm[6] = "RELATÓRIO DE LOGS";
perm[7] = "CADASTRO DE EMPRÉSTIMOS";
perm[8] = "PESQUISA DE EMPRÉSTIMOS";

boolean autPerm[] = new boolean[9];
for(int i = 0; i < 9; i++)
               autPerm[i] = false;

checarPermissao(perm,autPerm,listPermissoes);
}

public void checarPermissao(String permissao[],
               boolean aut[],List
               <AtribuirPermissoes> atPerm){

  mntmCadastrarLivros.setEnabled(aut[0]);
  mntmPesquisarLivros.setEnabled(aut[1]);
  mntmTipoDeLivros.setEnabled(aut[2]);
  cadastrarAlunoBtn.setEnabled(aut[3]);
  cadastrarFuncionarioBtn.setEnabled(aut[4]);
  mntmTipoDeUsuario.setEnabled(aut[5]);
  mntmLog.setEnabled(aut[6]);
  mntmNovoEmprestimo.setEnabled(aut[7]);
  mntmPesquisarEmprestimos.setEnabled(aut[8]);
}

public aspect AspectoControleAcesso {

  pointcut controle(String perm[], boolean at[],
List <AtribuirPermissoes> atPerm) :
    call(void
telas.Principal.checarPermissao(String[],boolean[],
List <AtribuirPermissoes>))
      && args(perm[],at[],atPerm);

  void around(String perm[],boolean at[], List
<AtribuirPermissoes> atPerm) :
controle(perm[],at[], atPerm){
    boolean aut = false;
    for (int cont = 0; cont < 9; cont++){
      aut = false;
      for (int x = 0; x < atPerm.size() && aut ==
false; x++){
  if(atPerm.get(x).getPermissao().getPermissao().com
pareTo(perm[cont]) == 0){
          aut = true;
        }
      }
      if(aut){
        at[cont] = true;
      } else {
        at[cont] = false;
      }
    }
      proceed(perm,at,atPerm);
  }
}
```

Figura 12. Trecho de Código Orientado a Ascpecto

O aspecto captura os três argumentos do método *checarPermissao*: *perm*[], *at*[] e atPerm[]. O vetor de String *perm*[] possui todas as permissões de acesso, cada posição do vetor booleano *at*[] é uma acesso que foi permitido ao usuário e a lista *atPerm* possui a relação de permissões que foi atribuida ao usuário que obteve acesso. Deste modo o aspecto *AspectoControleAcesso* combina no ponto de junção deste método e executa o adendo *around*(). A adendo *around* checa se usuário possui o acesso, se possuir este acesso ele seta *true* na posição correspondente no vetor *at*[]. O comando proceed(perm,at,atPerm) encerra a primeira fase do adendo *around* enviando os valores modificados ao método *checarPermissao*, para que o mesmo possa alterar o acesso conforme aos novos valores atribuidos. Como após o comando *proceed(perm,at,atPerm)* não existe mais nenhum comando, a instância do aspecto é encerrado.

B. Controle de transação

Considerando o código abaixo que é a forma de implementar um controle de transação sem a utilização da linguagem de programação orientada a aspecto, o código necessitaria ser estruturado individualmente para cada abertura da transação que o sistema possuir. Além de aumentar o número de linhas do seu código, a visibilidade, a complexidade do código e o tempo irão ser muito grandes.

```
btnSalvar.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent arg0) {
      try {
           HibernateUtil.begin();
           botaoSalvar();
           HibernateUtil.commit();
           } catch (Exception e) {
           HibernateUtil.rollback();
      }
   }
});
btnSalvar.setFont(new Font("Segoe UI", Font.PLAIN,
11));
btnSalvar.setBounds(698, 462, 89, 23);
getContentPane().add(btnSalvar);
```

Figura 13. Trecho de Código Orientado a Ascpecto Botão *Salvar*

Se utilizar o conceito de aspectos, pode-se criar um único aspecto que controle a transação do *hibernate* para todos os outros pontos que necessitaram deste comando. Assim, a implementação abaixo demonstra o uso de um aspecto para o código do botão *salvar* apresentado na Figura 13.

```
public aspect AspectoTransactionHibernate {
   pointcut    transaction()    :        call    (void
telas.CadastrarAluno.botaoSalvar());
   before() : transaction(){
      HibernateUtil.begin();
   }
   after() returning() : transaction(){
      HibernateUtil.commit();
   }
   after() throwing() : transaction(){
      HibernateUtil.rollback();
   }
}
```

Figura 14. Trecho de Código *AspectoTransactionHibernate*.

Os métodos *HibernateUtil.begin(), HibernateUtil.commit()* e *HibernateUtil.rollback()* estão encapsulados dentro do aspecto *AspectoTransactionHibernate* (Figura 14), podendo ser utilizados em qualquer outro ponto de junção descrito dentro do corpo do aspecto, não necessitando de nenhuma chamada externa ao aspecto. Deste modo, o código principal fica limpo deste interesses, como observa-se no código do botão salvar após a aplicação do conceito da POA ilustrado na Figura 15.

```
btnSalvar.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent arg0) {
            botaoSalvar();
   }
});
btnSalvar.setFont(new  Font("Segoe  UI",  Font.PLAIN,
11));
btnSalvar.setBounds(698, 462, 89, 23);
getContentPane().add(btnSalvar);
```

Figura 15. Trecho de Código Orientado a Ascpecto Botão *Salvar* com o uso de POA.

V. CONCLUSÃO

A forma de implementação dos componentes, inicialmente ter sido realizada Orientado a Objetos, e depois numa segunda fase o projeto foi convertido para POA permitiu a visualização de como seria a transformação de um projeto Orientado a Objetos em um projeto com o *AspectJ*, além de mostrar as dificuldades da implantação de uma nova funcionalidade ao programa e a versatilidade na manutenção dos aspectos.

O uso da POA permitiu a implantação da Tecnologia Adaptativa de uma forma fácil e segura, também permite que novas funcionalidades (aspectos) sejam adicionados ao software sem a necessidade de modificar o código fonte já produzido. O programador deve apenas acrescentar novos aspectos ao software e o mesmo se adapta ao código já existente. A POA foi interessante em relação a implementação de Tecnologia Adaptativa, porém comprova-se que os aspectos de TA não são implementados completamente, uma vez que os programas em POA são combinados antes da compilação e os programas produzidos não são adaptativos em tempo de execução.

REFERÊNCIAS

[1] G. KICZALES, J. LAMPING, A. MENDHEKAR, C. MAEDA, C.V. LOPES, J. LOINGTIER. "Aspect-Oriented Programming. In: *European Conference on Object-Oriented Programming (ECOOP")*, 06,1997. Finlândia. Anais Springer-Verlag LNCS 1241, 06, 1997

[2] J. M. LOUREIRO, J. P. C. S. G. COSTA, R. M. S. B. FONSECA, V. A. N. LOUREIRO, "Programação Orientada a Aspecto". *FEUP Universidade do Porto*, Porto, Portugal.

[3] KRUPA, Artur. Analyse "Aspect-Oriented Software Approach and Its Application". Athabaska, Alberta, Canadá: Athabaska University, 2010.

[4] V. J. GOETTEN, D. WINCK. "AspectJ – Programação Orientada a Aspectos com Java", *Novatec Editora*, São Paulo, 2006.

[5] G. L. HEILEMAN, Data Structures, Algorithms, and Object-Oriented Programming *McGraw-Hill*, Singapore, 1996.V.

[6] R. SAFONOV, "Using Aspect-Oriented Programming for Trustwothy Software Development". *Wiley-Interscience*, New Jersey, 2008.

[7] R. BODKIN, R. LADDAD, Zen and the art of Aspect-Oriented Programming. *Linux Magazine*, April, 2004.

[8] J. J. NETO e M. E. S. MAGALHÃES, "Um Gerador Automático de Reconhecedores Sintáticos para o SPD". *VIII SEMISH - Seminário de Software e Hardware*, pp. 213-228, Florianópolis, 1981.

[9] J. J. NETO. "Contribuições à metodologia de construção de compiladores". Tese de Livre Docência, USP, São Paulo, 1993.

[10] J. R. ALMEIDA "STAD - Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos". Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo, 1995.

[11] D. HAREL et al. "On the formal semantics of statecharts". In: Symposyum on logic in Computer Science, 2°, Ithaca, Proceedings, IEEE Press, pp. 54-64, New York, 1987.

[12] 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, Springer-Verlag, Vol.2494, pp. 234-250, Pretoria, South Africa, July 23-25, 2001.

[13] A. R. CAMOLESI. "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

[14] H. LEWIS; C. H. PAPADIMITRIOUS. "Elements of the theory or computation". Prentice Hall, editor, 1998.

Rafael Casachi nasceu na cidade de Maracaí, São Paulo, Brasil, em 18 de Novembro de 1990. Estudante de Bacharel em Ciência da Computação na Fundação Educacional do Município de Assis (FEMA) em Assis, São Paulo, Brasil. Entre os anos de 2008 e 2011. Suas principais áreas de pesquisas são: Programação Orientada a Aspectos e desenvolvimento de Aplicações Java.

Almir Rogério Camolesi possui graduação em Processamento de Dados pela Fundação Educacional do Município de Assis (1992), mestrado em Ciência da Computação pela Universidade Federal de São Carlos (2000) e doutorado em Engenharia de Computação e Sistemas Digitais pela Universidade de São Paulo (2007). Atualmente é professor titular da Fundação Educacional do Município de Assis. Tem experiência na área de Ciência da Computação, com ênfase em Tecnologias Adaptativas, atuando principalmente nos seguintes temas:. tecnologia adaptativa, computação distribuída, teoria da computação, modelagem abstrata, ensino a distância,, algorítmos e programação para web.

Sistemas de Markov Adaptativos: Formulação e Plataforma de Desenvolvimento

D.A. Alfenas, D.P. Shibata, J.J. Neto, M. R. Pereira-Barretto

***Abstract*— This paper extends Adaptative Markov Machines formalism and presents a framework allowing to design, implement, test and embed Markov adaptive systems into Java applications.**

***Keywords*— Markov machines framework, adaptivity, adaptive Markov machines.**

I. INTRODUÇÃO

O desenvolvimento de aplicações baseadas em tecnologias adaptativas ainda é incipiente. Uma das razões é que as ferramentas de suporte ao desenvolvimento destes softwares não alcançaram plena maturidade, mesmo que diversas pesquisas recentes tenham demonstrado como as técnicas adaptativas podem ser aplicadas nas mais diversas áreas. Foi exibida a aplicação no processamento digital de imagens e aprendizado de máquina [1], compiladores [1] [2], aprendizado à distância [1], tradução texto-fala [3], segurança da informação [4] e geração de música [5], entre muitas outras áreas.

Especialmente relevante para este artigo é o trabalho apresentado em [5]. Nele foi desenvolvido o LASSUS, um sistema capaz de gerar e executar música em tempo real. O autor utilizou geradores de sequências pseudo-aleatórias para abastecer um núcleo de execução baseado em máquinas de Markov adaptativas. A aleatoriedade faz com que cada execução do LASSUS produza uma composição diferente e imprevisível. Embora as qualidades musicais nos trechos compostos pelo sistema sejam reconhecidas por qualquer um que escute as saídas produzidas pelo sistema, não houve continuidade do trabalho nos dez anos seguintes.

As máquinas de Markov adaptativas utilizadas na geração de música podem ser generalizadas e aplicadas em outras áreas, pois viabilizam simular ações criativas através de decisões aleatórias, isto é, sistemas cuja evolução não depende, obrigatoriamente, da interação entre o usuário do sistema e a máquina.

Este artigo apresenta o *MMAdapt (Máquinas de Markov Adaptativas)*, uma plataforma de desenvolvimento de sistemas baseados em máquinas de Markov adaptativas. Descreve-se a estrutura interna e a interface de programação (*API*), assim como sugere possíveis aplicações. Descreve também a formulação matemática necessária para mostrar como é a teoria que originou toda a plataforma.

D. A. Alfenas, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, d.alfenas@fdte.org.br.

D. P. Shibata, Fundação para o Desenvolvimento da Engenharia (FDTE), São Paulo, Brasil, d.shibata@fdte.org.br

J. J. Neto, Escola Politécnica da USP (EPUSP), São Paulo, Brasil, joao.jose@poli.usp.br

M. R. Pereira-Barretto, Escola Politécnica da USP (EPUSP), São Paulo, Brasil, marcos.barretto@poli.usp.br.

II. MÁQUINAS DE MARKOV ADAPTATIVAS

A formalização das máquinas de Markov adaptativas utilizada neste trabalho se baseia, ao mesmo tempo em que a estende, naquela introduzida originalmente em [5], como parte do desenvolvimento do LASSUS. Neste item, faz-se uma introdução sucinta acerca de adaptatividade e cadeias de Markov para, em seguida, apresentar o formalismo em que a construção da plataforma é baseada.

A. Adaptatividade

Uma das idéias centrais no formalismo adaptativo é que dispositivos mais poderosos, em relação à capacidade de expressão e de computação, podem ser obtidos gradualmente a partir dos recursos oferecidos por um dispositivo mais simples.

Pode-se generalizar o conceito, aqui aplicado nas redes de Markov, para qualquer dispositivo guiado por regras. As regras mapeiam cada configuração do dispositivo em uma nova configuração, eventualmente estimulado por alguma entrada ou gerando alguma saída. Ele inicia em uma determinada configuração e segue aplicando regras até que não haja mais estímulos de entrada ou se atinja uma configuração à qual nenhuma regra possa ser aplicada.

Em um dispositivo não-adaptativo, o conjunto de regras é o mesmo durante toda a sua execução. Um dispositivo adaptativo adiciona um conjunto especial de regras, chamadas de ações adaptativas, que agem modificando o conjunto original através de adições, alterações e remoções. A camada composta desse conjunto de ações adaptativas é chamada de *camada adaptativa*; aquela composta das regras originais tem o nome de *camada subjacente*. Além das redes de Markov, a adaptatividade já foi incorporada a outros dispositivos como: autômatos [6], gramáticas [9], árvores e tabelas de decisão [8] e statecharts [7], entre outros. Uma visão ampla acerca da adaptatividade encontra-se em [6], [1].

B. Cadeias de Markov

Define-se uma cadeia de Markov como uma tripla

$$K = (Q, q_0, T) \quad (1)$$

em que Q é um conjunto finito com n estados, q_0 é o estado inicial da rede e T é um conjunto de transições entre estados. Cada transação γ_{ij} é caracterizada como uma tripla

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}) \quad (2)$$

em que ρ é a probabilidade do estado q_j ser atingido estando em q_i. Observe que cada transição γ_{ij} é única em T, isto é, não há duas transições diferentes partindo de um mesmo estado q_i e chegando a q_j. Além disso, seja Γ_q como o conjunto de todas as transições que se originam em um estado q. Então

$$\forall q \in Q, \sum_{\gamma \in \Gamma_q} \rho_\gamma = 1 \quad (3)$$

Isto é, a soma das probabilidades de todas as transições iniciando em q é exatamente um, para todo q pertencente a Q.

A definição dada por (1), (2) e (3) garante que a probabilidade de um dado estado ser atingido depende exclusivamente do estado atual da rede, isto é, a cadeia de Markov aqui definida é de primeira ordem. Pode-se provar também que, partindo-se do estado inicial q_0 e após um número suficientemente grande de iterações da rede, a probabilidade da rede encontrar-se em um de seus n estados é constante [10].

Cabe ressaltar uma diferença em relação à representação utilizada na definição original [10]: enquanto aqui T é um conjunto de transições, como na definição de autômatos, o texto original o apresenta como uma matriz n x n, com as mesmas propriedades. Desta forma, uma transição com ρ igual a zero é interpretada diferente de uma transição inexistente, o que será importante para a definição das máquinas de Markov adaptativas.

C. Máquinas de Markov

Uma máquina de Markov $\mathcal{M}$ é definida pela quádrupla

$$\mathcal{M} = (Q, S, T, q_0) \quad (4)$$

em que Q, T e q_0 são como definidos anteriormente para cadeias e S é o alfabeto de saída. A equação (2) é substituída por

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}, p_{ij}) \quad (5)$$

como definição de transição. Quando γ_{ij} é acionada, o símbolo $p_{ij} \in S \cup \{\varepsilon\}$ é inserido na cadeia de saída de $\mathcal{M}$.

Desta forma, é possível utilizar $\mathcal{M}$ como um dispositivo gerador de linguagem; a linguagem formada pelas possíveis cadeias de saída de $\mathcal{M}$ é linear. É possível mostrar, também, que o método de produção de linguagens por meio da máquina de Markov é formalmente equivalente ao das gramáticas lineares [5].

D. Máquina de Markov Adaptativa

Uma máquina de Markov adaptativa M é definida por uma quíntupla

$$M = (Q, S, T, q_0, F) \quad (6)$$

em que Q, S, T, e q_0 são como definidos anteriormente para $\mathcal{M}$ e F é um conjunto de funções adaptativas. Toda função adaptativa pertencente a F pode ser definida como uma quádrupla

$$f = (\Psi, V, G, C) \quad (7)$$

em que:

- Ψ é um conjunto de parâmetros formais (ψ_1, ψ_2, ψ_3,..., ψ_m);
- V é um conjunto de identificadores de variáveis (v_1, v_2, ..., v_n), cujos valores são desconhecidos no instante de chamada de f mas que uma vez preenchidos terão seus valores preservados durante toda a execução da função;
- G é um conjunto de identificadores de geradores (g^*_1, g^*_2,..., g^*_n), variáveis especiais que são preenchidas com novos valores, ainda não utilizados pelo autômato, a cada vez que a função é chamada;
- C é uma sequência de ações adaptativas elementares executadas em f. Ressalta-se que a imposição de uma ordem nas ações não faz parte da definição original do formalismo adaptativo [6].

A equação (5) é substituída por sua forma final

$$\gamma_{ij} = (q_i, q_j, \rho_{ij}, p_{ij}, a_{ij}). \quad (8)$$

como definição de transição, em que q_i, q_j, ρ_{ij} e p_{ij} são como definidos anteriormente e a_{ij} é uma ação adaptativa da forma

$$a_{ij} = f(\omega_1, \omega_2, ..., \omega_n) \cup \{\varepsilon\} \quad (9)$$

sendo f uma função adaptativa pertencente a F e (ω_1, ω_2, ..., ω_n) uma lista de argumentos que correspondem posicionalmente à lista de parâmetros Ψ declaradas para f. $\{\varepsilon\}$ representa o conjunto vazio, isto é, é possível que uma transição não tenha ação adaptativa associada.

Definidos desta forma, o procedimento de mudança de estado de uma máquina de Markov adaptativa M é:

1. Obtém-se a lista de transições possíveis Γ_{qi} a partir do estado ativo q_i em M;
2. Escolhe-se uma transição γ_{ij} a disparar utilizando-se as probabilidades ρ_{ik} das transições $\gamma_{ij} \in \Gamma_{qi}$;
3. A máquina muda para o estado q_j;
4. O símbolo p_{ij} é inserido na cadeia de saída de M;
5. Se $a_{ij} \neq \varepsilon$, então a função adaptativa f é chamada com os argumentos (ω_1, ω_2, ..., ω_n), realizando a ação adaptativa sobre M.

1) Ações adaptativas elementares

As ações adaptativas elementares utilizadas em C podem ser dos seguintes tipos:

- *Consulta de transição*: busca transições conforme um filtro recebido por parâmetro e as retorna;
- *Inserção de transição*: cria uma nova transição entre dois estados existentes. Se ela já existir, a substitui;
- *Remoção de transições*: remove uma ou mais transições recebidas por parâmetro.

Além destas, foram introduzidas três novas ações adaptativas elementares para aumentar o poder de expressão, mas que, entretanto, não aumentam o poder de computação:

- *Alteração de transições*: altera ρ, p_{ij} ou a_{ij} em uma ou mais transições recebidas como parâmetro. É equivalente a uma ação de remoção de transição entre os estados q_i e q_j seguida por outra de inserção de transição entre os mesmos estados q_i e q_j;
- *Chamada de ação adaptativa não-elementar*: recebe como parâmetro uma função adaptativa f e uma lista de argumentos (ω_1, ω_2, ..., ω_n) que correspondem posicionalmente à lista de parâmetros Ψ declaradas para

f. Introduzida aqui para melhorar a reutilização das funções adaptativas;

- *Compara duas variáveis*: esta ação recebe quatro parâmetros (ω_1, ω_2, $a_=$, $a_\neq$). Se ω_1 e ω_2 forem iguais, executa-se $a_=$; caso contrário, executa-se $a_\neq$.

Para demonstrar que não há aumento do poder computacional com a definição de ação elementar de comparação de duas variáveis, seja uma função adaptativa *x* que recebe os mesmos quatro argumentos (ω_1, ω_2, $a_=$, $a_\neq$) mas que é construída somente com ações elementares dos cinco tipos anteriores e é capaz de:

- executar $a_=$ apenas se ω_1 e ω_2 são iguais;
- executar $a_\neq$ apenas se ω_1 e ω_2 são diferentes

Para isso, constrói-se a máquina de Markov de forma que, entre os vários estados $q \in Q$, existam dois estados q_1 e q_2, chamados de *marcadores*. Eles precisam ser inatingíveis, isto é, não existe transição da forma γ_{ik} em *T*, $k \in \{1,2\}, i \notin \{1,2\}$. Além disso, q_1 não possui nenhuma transição partindo de si próprio e q_2 possui transições para todos $q \in Q$. Então, constrói-se *x* contendo as seguintes ações:

1. Insere-se uma transição γ_{1i};
2. Executa-se uma ação elementar de consulta de transição começando em q_1 e terminando em *v*. Se *v* e q_i forem iguais, retorna $r_1 = \gamma_{1i}$, caso contrário, retorna $r_1 = \varepsilon$;
3. Executa-se uma ação adaptativa $b_=$ com os mesmos argumentos passados para *x* e mais um argumento contendo r_1. A única ação dentro da função de $b_=$ é chamar $a_=$. Se r_1 for ε, a ação adaptativa com $b_=$ não é executada e a primeira condição (executar $a_=$ apenas se iguais) é satisfeita;
4. Remove-se a transição γ_{2i};
5. Executa-se uma ação elementar de consulta de transição começando em q_2 e terminando em *v*. Se *v* e q_i forem diferentes, retorna $r_2 = \gamma_{2v}$, caso contrário, $r_2 = \varepsilon$;
6. Executa-se a chamada de ação adaptativa $b_\neq$ com os mesmos argumentos passados para *x* e mais um argumento contendo r_2. A única ação dentro da função de $b_\neq$ é chamar $a_\neq$. Se r_2 for ε, a ação adaptativa com $b_\neq$ não é executada e a segunda condição (executar $a_\neq$ apenas se diferentes) é satisfeita;
7. Remove γ_{1i} e insere γ_{2i}, para manter a condição inicial de q_1 e q_2.

Com estes sete passos, a demonstração está feita.

Quando se executa uma ação de alteração de probabilidade ou remoção ou inserção com $\rho > 0$ para uma transição γ_{ij}, é necessário reajustar as probabilidades para as demais transições pertencentes a Γ_q de tal forma que (3) continue válida. Seja ρ' a nova probabilidade (considere ρ' igual a zero no caso de remoção) e ρ_{ij} a probabilidade antiga (considere ρ_{ij} igual a zero no caso de inserção). Então

$$\forall \gamma_{ik} \in \Gamma_q, k \neq j, \rho_{ik} = \rho_{ik} \bullet \frac{(1-\rho')}{1-\rho_{ij}} \qquad (10)$$

A linguagem *L*(*M*) gerada por uma máquina de Markov adaptativa é sensível ao contexto [5]. É importante ressaltar que a definição acima estende o formalismo original em três pontos: (i) formaliza o conceito de funções adaptativas; (ii) inclui novas funções adaptativas e (iii) introduz a idéia de *adaptatividade de segunda ordem*, ou seja, a possibilidade da alteração da função adaptativa [14]. Aplicações destas extensões são demonstradas adiante, neste trabalho.

E. Sistema de Markov Adaptativo

Define-se um SMA (Sistema de Markov Adaptativo) como um conjunto de máquinas adaptativas de Markov que podem se relacionar através de novas ações adaptativas, introduzidas a seguir. Assim, pode-se estabelecer uma relação de escopo sobre as ações adaptativas definidas para cada máquina, classificando-as em: ações que modifiquem apenas a topologia local da máquina e ações que podem interferir no comportamento de outras máquinas pertencentes ao sistema. Desta forma, cada máquina M^k pertencente a um sistema de Markov adaptativo é definida como uma quíntupla:

$$M^k = (Q^k, \Sigma, T^k, q_0^k, F^k) \qquad (11)$$

em que Σ é o alfabeto de saída, comum a todas as máquinas do sistema, e Q^k, T^k, q_0^k e F^k são como na definição de *Q*, *T*, q_0 e *F*, respectivamente.

Além dos tipos de ações elementares definidos para uma máquina de Markov adaptativa, as máquinas pertencentes a um SMA podem executar novos tipos, que operam sobre as outras máquinas do sistema. São eles:

- Dos mesmos tipos definidos para *M*, porém operando sobre outras máquinas do sistema;
- *Desativa* M^w, $w \neq k$. Desabilita a máquina M^w.Uma máquina desabilitada permanece em seu estado mais recente, não acionando qualquer transição ou executando qualquer ação adaptativa, até que seja novamente habilitada;
- *Ativa* M^w, $w \neq k$. Habilita a máquina M^w;
- *Consulta estado de* M^w, $w \neq k$. Consulta o estado atual (mais recente) de M^w e o retorna.

A definição de sistemas de Markov adaptativos serve apenas a fins práticos, uma vez que a distribuição do controle em várias máquinas não aumenta o potencial de computação do sistema [5].

III. A PLATAFORMA

O MMAdapt foi feito para acelerar o desenvolvimento de aplicações baseadas em sistemas de Markov adaptativos. Ao desenhá-lo, algumas diretrizes foram adotadas:

- Controle de execução: necessidade de criar um mecanismo que permita ao desenvolvedor ter o controle do instante em que as transições são acionadas;
- Opção de representação do sistema através de XML. Isso adiciona independência de linguagem (supondo que, eventualmente, haverá implementações do MMAdapt para outras linguagens), maior independência do resto do sistema e maior facilidade de compartilhamento;
- Futura criação de uma ferramenta gráfica, baseada no MMAdapt, para desenhar os sistemas de Markov

adaptativos, semelhante ao que o Adaptools faz para autômatos adaptativos [11];
- Permitir a customização do alfabeto de saída, através de implementação de uma interface disponibilizada pela plataforma.

Para viabilizar essas diretrizes, foi necessário fazer novas extensões ao formalismo, adaptando-o ao ambiente de execução. O texto abaixo, sempre que introduzir uma alteração, a ressaltará, de forma a não passar despercebida.

A. *Arquitetura Geral*

A arquitetura geral está mostrada na Fig. 1.

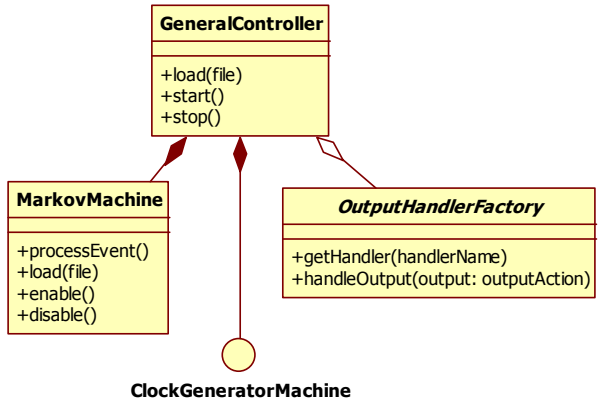


Figura 1. Diagrama com as principais classes do MMAdapt.

No MMAdapt, existem três classes principais que definem o comportamento do SMA:
- *MarkovMachine* (MM): implementa uma MMA. Ao menos uma precisa ser definida, adaptativa ou não;
- *ClockGeneratorMachine* (Máquina geradora de clock, ou CGM): determina o momento em que as transições das máquinas de Markov são acionadas, sem interferir nas probabilidades de escolha de transição. Ao menos uma ClockGeneratorMachine precisa ser definida;
- *OutputHandlerFactory* (OHF): permite estender o modo como o alfabeto de saída é interpretado. Sua existência é opcional.

A plataforma permite que a configuração do SMA seja definida através de um arquivo XML, como o mostrado na Fig. 2. Alternativamente, as classes podem ser criadas a partir de chamadas à API.

```
<?xml version="1.0" encoding="UTF-8"?>
- <root>
    <outputHandlerFactory name="composer.Band"/>
    <clock name="clock1" file="clock1.xml"/>
    <clock name="clock2" file="clock2.xml"/>
    <clock name="clock3" file="clock3.xml"/>
    <markov name="pri" file="primeira.xml"/>
    <markov name="ritmo" file="ritmo.xml"/>
    <markov name="regente" file="wta_regente.xml"/>
    <markov name="beat" file="rockbeat2.xml"/>
    <markov name="tempo" file="tempo.xml" enabled="false"/>
  </root>
```

Figura 2. Exemplo de arquivo de configuração de um SMA

Na Fig. 2 é possível observar as definições de várias MM e CGM, além de definir uma classe customizada para manipulação das saídas (*composer.Band*). A configuração de cada componente do sistema fica em arquivo próprio.

Para iniciar uma SMA, basta instanciar a classe *GeneralController*, chamar o método *load* passando como parâmetro o arquivo *.sma* e, finalmente, chamar o método *start*. Para parar a execução, basta chamar *stop* na mesma instância do *GeneralController*.

B. *Máquinas de Markov*

As máquinas de Markov implementadas no MMAdapt tem poucas mas importantes diferenças em relação ao formalismo apresentado anteriormente, introduzidas com base em sua utilização, visando aumentar a praticidade na construção de aplicações.

A primeira é relativa aos símbolos de saída, que são substituídos por *ações de saída* (*OutputAction*).

A segunda é que o conceito de função adaptativa é generalizado, passando a conter ações não só adaptativas, mas também ações de saída, entre outras. O nome do conceito generalizado é *função de Markov* e a transição passa a ser uma quádrupla,

$$\gamma_{ij} = (q_i, q_j, \rho, a_{ij}). \quad (12)$$

Em (12), a_{ij} é uma *ação de Markov* da forma

$$a_{ij} = f(\omega_1, \omega_2, ..., \omega_n) \cup \{\varepsilon\} \quad (13)$$

sendo f uma função de Markov declarada para a mesma máquina que contém a transição e $(\omega_1, \omega_2, ..., \omega_n)$ uma lista de argumentos que correspondem posicionalmente à lista de parâmetros de f.

Um exemplo de um arquivo simples de configuração de MM pode ser visto na Figura 3.

```
- <markov>
    <general name="beat" stepControl="clock1" initial="s1"/>
  - <outputHandler name="perscussion">
      <property name="texture" value="0"/>
      <property name="tempo" value="120"/>
    </outputHandler>
    <state name="s1"/>
    <state name="s5"/>
    <trans name="t1" src="s1" dest="s1" p="0.1" a="cym60"/>
    <trans name="t2" src="s1" dest="s5" p="0.9" a="lft"/>
    <trans name="t3" src="s5" dest="s1" p="0.9" a="cym60"/>
    <trans name="t4" src="s5" dest="s5" p="0.1" a="lft"/>
  - <function name="cym60">
    - <output>
        <property name="note" value="cymbal_1"/>
        <property name="volume" value="60"/>
      </output>
    </function>
  - <function name="lft">
    - <output>
        <property name="note" value="low_floor_tom"/>
      </output>
    </function>
  </markov>
```

Figura 3. Exemplo de MM sem funções adaptativas.

A primeira tag, chamada de *general*, serve para configurar propriedades gerais da máquina, como nome da máquina, nome do CGM que controla os passos e estado inicial.

A segunda tag, *outputHandler*, traz configurações adicionais sobre como manipular as saídas desta máquina específica. Ela só pode ser definida se o arquivo principal do SMA define um OHF. Porém, a definição do *outputHandler* na MM é opcional. Neste caso, as saídas serão direcionadas diretamente para o OHF.

Em seguida, observam-se três seções do XML. A primeira contém a definição de dois estados. A segunda contém a definição de quatro transições, cada uma contendo estado de origem, estado de destino, probabilidade e ação de Markov (veja que não há parâmetros para as funções utilizadas). A definição deste último é opcional, no caso de não haver qualquer ação a executar no disparo da transição.

Na última seção, estão as próprias funções de Markov. Há um tópico dedicado a elas no final deste capítulo.

C. *Máquinas Geradoras de Clock (CGM)*

Uma CGM é definida pela interface *CockGeneratorMachine*, que possui três métodos:

- *setAsControllerOf*: adiciona uma MM na lista de máquinas controladas. É chamada pelo *framework* durante a leitura do arquivo de máquina de Markov;
- *start*: inicia a execução da CGM. É chamada pelo framework de dentro do método *start* do *GeneralController*;
- *stop*: interrompe a execução. É chamada pelo *framework* de dentro de método *stop* do *GeneralController*.

A Figura 4 mostra os três tipos de CGM existentes.

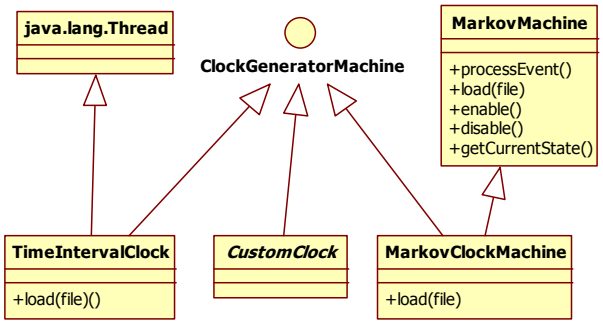


Figura 4. Diagrama de classe contendo os três tipos diferentes de clock.

São eles:

- *TimeIntervalClock*. Gera os clocks em tempos constantes. A Figura 5 contém um exemplo de aquivo de configuração desse gerador. O tamanho dos intervalos é definido em milissegundos. É declarada no arquivo *.sma* através da tag *clock*;

```
<?xml version="1.0" encoding="UTF-8"?>
<clock>
    <general timeInterval="200" name="clock1"/>
</clock>
```

Figura 5. Exemplo de máquina de clock gerado por intervalo de tempo

- *MarkovClockMachine*: É, ao mesmo tempo, uma MM e uma CGM. É declarada no arquivo *.sma* através da tag *markovClock*. Aciona as transições nas máquinas de Markov controladas através de uma ação especial, chamada de *ChangeStateAction*;
- *CustomClock*: É um gerador de clocks adaptado pelo desenvolvedor. Permite controle máximo da execução e é útil quando se deseja processar os eventos de acordo com um estímulo externo, ou processar a máquina de Markov através de um loop no código Java. Basta implementar a interface da CGM e declará-lo no arquivo *.sma* com a tag *customclock*. O nome da classe customizada deve ser passada no atributo *name*.

D. *OutputHandlerFactory e OutputHandlers*

Sempre que uma instância de *OutputAction* é encontrada durante o processamento de uma função de Markov, o *framework* obtém a instância do *OutputHandler* associada à máquina de Markov e chama o método *handleOutput*, cuja obrigação é, finalmente, processar a saída de fato.

O *OutputHandler* de cada máquina é obtido durante a leitura do arquivo da MM. Quando a tag *outputHandler* for encontrada, o *framework* obtém a instância do *OutputHandlerFactory*, declarado no arquivo principal do SMA, e chama o método *getHandler*, cuja obrigação é retornar um manipulador adequado ao nome passado como parâmetro.

A Fig.6 mostra as classes envolvidas.

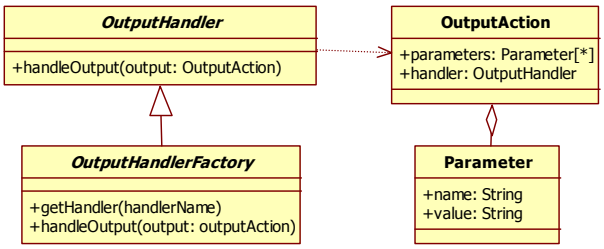


Figura 6. Diagrama com as classes para manipulação da saída de um MM.

O próprio OHF é, ele mesmo, um *OutputHandler*, utilizado sempre que o arquivo MM não declara a necessidade de uma instância específica. Outro fator importante é que todo SMA possui um OHF, mesmo que ele não tenha sido declarado. Essa instância implícita aceita como saída apenas caracteres, que serão direcionados para o *system.out* do Java. Nesta situação, qualquer propriedade do *OutputAction* é ignorada, e apenas o nome é impresso.

E. *Funções de Markov*

A função de Markov implementada pela plataforma é muito similar à do formalismo, tendo pequenas diferenças para aumentar a reutilização. Existem três grupos de dados que podem ser configurados em cada função:

- *Parâmetros*: Agrupados dentro da tag *parameters*. Opcional;
- *Variáveis*: Agrupados dentro da tag *vars*. Opcional;
- *Ações*: As ações devem vir ordenadas logo após o grupo das variáveis. A existência de pelo menos uma ação é obrigatória.

Algumas palavras são reservadas pelo MMAdapt e não podem ser utilizadas como nomes de variáveis, funções ou parâmetros. São elas:

- *this*: Palavra reservada para representar a transição que chamou a função. Por exemplo, é possível passar *this* como parâmetro ou usar *this.src* para obter o estado origem da transição atual;
- *each*: Palavra reservada para representar o valor de qualquer item em uma coleção. Pode ser utilizado em ações que recebem coleções como parâmetros: *UpdateProbabilitiesAction* e *UpdateFunctionsAction*;
- *null*: Palavra reservada para representar o valor nulo, que pode ser retornado pelas ações de consulta quando nenhuma transição atende ao filtro de pesquisa. Pode ser utilizada como parâmetro da ação *ChangeStateAction*.

F. Ações Elementares (Adaptativas e não adaptativas)

Neste tópico, são sucintamente apresentadas as ações atualmente suportadas pela plataforma de desenvolvimento. O objetivo é demonstrar as potencialidades atuais da plataforma.

Sempre que uma ação aceitar uma MM como parâmetro, ele é opcional, e quando não informado a ação é executada na máquina local.

Um parâmetro do tipo *limites de probabilidade* pode ter as seguintes formas:

- *Variável*: uma variável que contém o valor de probabilidade a atribuir;
- *Número*: valor exato de probabilidade;
- *Variável* "+" *Número*: combinação variável mais valor;
- *"[Combinação1,Combinação2]"*: combinação1 e combinação2 podem ser substituídos por qualquer uma das opções anteriores. Será gerado um valor aleatório de probabilidade, em tempo de execução, entre os valores calculados para as combinações.

1) FindUniqueAction

Descrição: Procura uma transição iniciando em q_i e terminando em q_j.
Parâmetros: Estado de origem, estado de destino, MM, parâmetro de retorno com a transição encontrada.
Retorno: Transição se encontrada, *null* em caso contrário.

2) SearchAction

Descrição: Procura por transições que obedeçam aos critérios de pesquisa.
Parâmetros: Estado de origem (opcional), estado de destino (opcional), MM, parâmetro de retorno com uma lista de transições.
Retorno: Lista de transições encontradas, *null* em caso contrário.

3) SetTransitionAction

Descrição: Configura uma transição conforme os parâmetros. Se a transição não existir, ela é criada.
Parâmetros: Estado de origem, estado de destino, função de Markov (pode ser nula), limites de probabilidade, MM.

4) RemoveTransitionAction

Descrição: Remove uma transição, que não será mais retornada pelas consultas.
Parâmetros: Lista de transições, MM.

5) UpdateProbabilitiesAction

Descrição: Atualiza as probabilidades das transições recebidas como parâmetro. Aceita a palavra reservada *each* como parte do parâmetro de limites de probabilidade.
Parâmetros: Lista de transições, MM, limites de probabilidade.

6) UpdateFunctionsAction

Descrição: Altera as funções de Markov das transições recebidas como parâmetro.
Parâmetros: Lista de transições, MM, função de Markov.

7) CompareVarsAction

Descrição: Compara duas variáveis e, dependendo do resultado, executa a função para igualdade ou para diferença. Pelo menos uma das duas funções precisa ser fornecida.
Parâmetros: variável 1, variável 2, função para igualdade (opcional), função para diferença (opcional).

8) EnableMachineAction

Descrição: Ativa uma máquina de Markov.
Parâmetros: MM.

9) DisableMachineAction

Descrição: Desativa uma máquina de Markov.
Parâmetros: MM.

10) ChangeStateAction

Descrição: Chama o método *processEvent* em uma MM, que procederá com a mudança de estado
Parâmetros: MM.

11) GetCurrentStateAction

Descrição: Consulta uma máquina de Markov para saber qual é o estado ativo.
Parâmetros: MM.
Retorno: Estado ativo em MM no momento do processamento da ação. Se ela estiver ocupada, no momento da chamada, com uma mudança de transição, o último estado ativo será retornado.

12) OutputAction

Descrição: Aciona o *OutputHandler* da máquina para receber a saída atual.
Parâmetros: nome da saída, lista de parâmetros.

G. Observações sobre a boa utilização da Plataforma

Algumas considerações devem ser feitas quanto a possíveis problemas relativos à concorrência. Cada máquina pode executar apenas uma tarefa simultaneamente, para evitar problemas de sincronismo. Por exemplo, uma máquina M^l não pode mudar de estado enquanto outra M^k, $l \neq k$, executa uma ação em M^l. Deste modo, a mudança de estado de M^l permanece em espera até que M^k termine.

Também é importante notar que o MMAdapt não impõe uma hierarquia de chamada às maquinas. Qualquer uma delas pode executar ações adaptativas em qualquer outra máquina do sistema. Isso pode fazer com que, eventualmente, ocorram *deadlocks* se o SMA não for bem desenhado, pelo menos na versão atual, devido ao modelo de sincronismo adotado.

Algumas situações, na prática, podem fazer com que a equação (3) seja totalmente incompatível com a ação adaptativa em execução. Isto acontece sempre que uma transição γ_{ij}, começando em q_i e tendo 100% de probabilidade de acionamento, e uma ação tenta diminuir essa probabilidade ou apagar a transição. Então (10) gera erro de divisão por zero e a execução do SMA falha – não há uma forma correta de

resolver este problema automaticamente. O melhor que o desenvolvedor pode fazer é alterar a função de Markov para, antes de diminuir a probabilidade de γ_{ij}, aumentar a probabilidade de pelo menos uma transição γ_{ik}, $k \neq j$.

Finalmente, a configuração inicial do SMA pode conter um estado q que não tem transições disponíveis para acionar. Nestes casos, a plataforma de execução será capaz de detectar o erro apenas quando, estando em q, um evento for acionado, mas, mesmo assim, não será capaz de corrigi-lo, e a execução do SMA é interrompida.

IV. CONCLUSÃO

O MMAdapt, como aqui descrito, já está implementado e funciona como base de uma aplicação para síntese automática de música, cujo nome é *Markovianas*. Os arquivos exibidos nas figuras 2, 3 e 5 fazem parte desta aplicação. O SMA gera as notas musicais para diversos instrumentos (guitarra, bateria e baixo) e as envia para uma classe que armazena pelo menos trinta segundos de música (isto é, é mantém um *buffer* musical) antes de começar a reproduzi-la.

Planos futuros relacionados ao MMAdapt incluem o desenvolvimento de uma ferramenta gráfica, como mencionado anteriormente, para permitir desenhar, executar e depurar um sistema de Markov adaptativo e exportar o XML deste SMA.

Além da geração de música, outras aplicações poderiam ser desenvolvidas na plataforma:

- Geração automática de texto (poesias, artigos, diálogos): uma abordagem utilizando SMA pode ser interessante se comparada às abordagens hoje utilizadas, como aquelas baseadas em templates, em técnicas evolutivas ou em gramáticas. Uma visão geral das técnicas para geração de poesias pode ser encontrada em [12];
- Geração automática de letras para música: uma abordagem que combine técnicas de geração automática de poesia àquelas utilizadas para o desenvolvimento das *Markovianas* pode ser interessante. Já existem trabalhos que demonstram a possibilidade de gerar letras a partir da melodia [13];
- Jogos: modelagem de decisões baseadas em probabilidade, como, por exemplo, quando se deseja que NPC's (*non-player character*, personagens que não são controlados pelo jogador) tomem decisões que o jogador não espera;
- Geração de casos de teste: a partir da modelagem do conjunto de dados de entrada e saída, documentos (telas, mensagens, etc.) e relacionamentos que caracterizam um sistema, e consequente classificação desses conjuntos em casos de teste positivos e negativos, pode-se desenvolver um SMA que gere milhares de casos aleatórios automaticamente;
- Geração de exercícios de aritmética (e de muitas outras disciplinas baseadas em regras matemáticas): pode-se seguir uma linha similar à da geração automática de casos de teste, porém voltada a aplicações pedagógicas, explorando a cobertura das regras em que se deseja efetuar o treinamento, bem como a variação do formato e da complexidade dos exercícios, automaticamente controlada por uma métrica voltada à avaliação do aproveitamento do aluno.

Pode-se concluir que o MMAdapt é uma ferramenta adequada para o desenvolvimento, pois tem uma API bem definida e reutilizável, pois delega toda a interpretação do alfabeto de saída para outras partes de um sistema mais abrangente. Mas é possível que, ao lidar com novas aplicações, alterações no código-fonte e na API sejam necessárias para o funcionamento desejado.

REFERÊNCIAS

[1] H. Pistori, "Tecnologia Adaptativa em Engenharia de Computação: Estado da arte e aplicações", *Tese de Doutorado*, EPUSP, São Paulo, 2003.

[2] J. C. Luz e J. J. Neto, "Tecnologa Adaptativa Aplicada à Otimização de Código em Compiladores", *IX Congreso Argentino de Ciencias de la Computación*, La Plata, Argentina, 6-10 de Outubro, 2003.

[3] D. P. Shibata, "Tradução Grafema-Fonema para a Língua Portugues Baseada em Autômatos Adaptativos", *Dissertação de Mestrado*, EPUSP, São Paulo, 2008.

[4] E. J. Pelegrini e J. J. Neto, "Applying Adaptive Technology in Data Security", *Proceedings of the 6th Peruvian Computer Week - JPC 2007*, Trjillo, Peru, pp. 31-40, Novembro 5-10, 2007.

[5] B. A. Basseto, "Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos", *Dissertação de Mestrado*, EPUSP, São Paulo, 2000.

[6] J. J. Neto, "Contribuições à metodologia de construção de compiladores", *Post-Doctoral Tese de Livre Docência*, EPUSP, São Paulo, 1993.

[7] J. J. Neto, J. R. A. Junior e J. M. M. dos Santos, "Synchronized statecharts for reactive systems", *In: Proceedings of the IASTED International Conference on Applied Modelling and Simulation*. Honolulu, Hawaii, pp. 246-251, 1998.

[8] A. H. Tchemra, "Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério", *Tese de Doutorado*, EPUSP, São Paulo, 2009.

[9] C. A. Bravo Pariente, "Gramáticas Livres de Contexto Adaptativas com Verificação de Aparênci", *Tese de Doutorado*, EPUSP, São Paulo, 2004.

[10] R. A. Howard, "Dynamic Probabilistic Systems", v. 1 e 2, 1a ed., *John Wiley & Sons, Inc.*, New York, 1971;

[11] L. Jesus, D. G. Santos, A. A. Castro Jr. e H. Pistori, "AdapTools 2.0: Aspectos de Implementação e Utilização", *Revista IEEE América Latina. Vol. 5, Num. 7*, ISSN: 1548-0992, pp. 527-532, novembro 2007.

[12] H. R. G. Oliveira, "Automatic generation of poetry: an overview", *Universidade de Coimbra*, Portugal, 2009.

[13] H. R. G. Oliveira, F. A. Cardoso e F. C. Perreira, "Exploring difference strategies for the automatic generation of song lyrics with TraLa Lyrics", *Proceedings of the Portuguese Conference on Artificial Intelligence*, Guimarães, Portugal, 2007.

[14] J. J. Neto, "Adaptive Technology and Its Applications", *Machine Learning: Concepts, Methodologies, Tools and Applications. IGI Global, 201*, doi:10.4018/978-1-60960-818-7.ch108, pp. 87-96, outubro 2011.

Daniel Assis Alfenas é formado em Engenharia da Computação pela Escola Politécnica da Universidade de São Paulo (EPUSP) em 2004. Trabalhou, nos últimos dez anos, nas diversas áreas do desenvolvimento de sistemas, desde a definição da demanda até a implantação do sistema. Recentemente tem trabalhado como coordenador técnico no desenvolvimento de sistemas complexos que envolvem simulação, otimização e interfaces ricas. Atualmente, é mestrando

no Laboratório de Técnicas Adaptativas da EPUSP e a área de pesquisa é a aplicação da tecnologia adaptativa no diálogo em linguagem natural entre usuários e sistemas.

Danilo Picagli Shibata é mestre em Engenharia da Computação pela Escola Politécnica da USP (EPUSP), com título obtido em 2008 pelo estudo da aplicação de autômatos adaptativos na tradução texto-fala para textos escritos na língua portuguesa. Trabalhou com Análise de segurança e confiabilidade de sistemas computacionais aplicados a sistemas críticos e desenvolvimento de software na área de segurança da informação. Trabalha atualmente no desenvolvimento de aplicação para simulação de transporte de fluídos.

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.

Marcos Ribeiro Pereira-Barretto é graduado em Engenharia Elétrica (1983), mestre em Engenharia Elétrica (1988) e doutor em Engenharia Mecânica (1993) pela Escola Politécnica da Universidade de São Paulo. É professor da Escola Politécnica da USP desde 1986. Atua nas seguintes áreas de pesquisa: robôs sociáveis, computação afetiva e arquitetura de sistemas para aplicações críticas como Automação Industrial.

Autômato Adaptativo para definição autônoma do intervalo de amostragem de dados em Rede de Sensores Sem Fio

I. M. Santos e C. E. Cugnasca

Resumo — Rede de Sensores Sem Fio é um tipo especial de rede ad hoc, formada por vários sensores capazes de coletar e processar informações do ambiente em que estão distribuídos. Essa tecnologia tem sido empregada em diversos tipos de problemas, com destaque para aplicações de monitoramento. Um dos principais desafios na utilização das redes de sensores está relacionado com o consumo de energia dos sensores. Este trabalho apresenta um Autômato Adaptativo a ser empregado nas Redes de Sensores Sem Fio para determinar dinamicamente o intervalo de amostragem dos nós sensores durante a coleta de dados. Quando as informações coletadas pelo sensor seguir um comportamento com baixa variação nos dados, o autômato irá determinar intervalos de amostragem mais longos, enquanto que a ocorrência de eventos com alta variação nos dados determina intervalos de amostragem mais curtos. Dessa forma, há economia de energia, pois o nó sensor irá coletar e transmitir menor número de dados enquanto o ambiente monitorado apresenta um comportamento esperado, sem que isso represente perda de eficiência, pois o sistema se adapta a situações que demandam maior atenção, reduzindo o intervalo entre as amostras de dados. A formalização dessa estratégia é feita por meio de um Autômato Adaptativo e sua Tabela de Decisão Adaptativa.

Palavras-chave – redes de sensores sem fio, conservação de energia, autômato adaptativo.

I. Introdução

As Redes de Sensores Sem Fio (RSSF) são um dos primeiros exemplos de computação pervasiva, é uma tecnologia emergente que promete revolucionar e ampliar o potencial de aplicações de monitoramento, instrumentação e controle. Uma RSSF é um tipo especial de rede ad hoc, formada por centenas ou milhares de sensores, densamente distribuídos, com comunicação sem fio e capazes de coletar e processar informações do ambiente em que estão distribuídos [1, 2, 3].

Uma típica RSSF possui pouca ou nenhuma infra-estrutura, consiste de um grande número de sensores que trabalham em conjunto monitorando um ambiente ou região para obter informações. As RSSF têm características e propriedades específicas, que as difere das redes de computadores tradicionais, principalmente quanto aos recursos disponíveis [4]. São aspectos das RSSF a limitação quanto a fonte de energia, processamento, alcance de comunicação, banda de transferência e armazenamento de dados. Um dos principais desafios na aplicação das RSSF é a limitação de energia dos nós sensores, de modo que o uso racional e otimizado desse recurso é fundamental, pois determina o tempo de vida útil da rede. Na rede de sensores, cada nó é autônomo e responsável por utilizar de maneira eficiente sua carga de energia disponível.

As RSSF podem ser aplicadas em diversas áreas, como por exemplo em sistemas de monitoramento ambiental, militar, na agropecuária, no monitoramento médico, em sistemas de suporte pessoal, mobilidade, segurança, controle logístico, processos industriais, sistemas embarcados, entre outros [1, 5, 6, 7, 8, 9, 10, 11, 12].

Em razão da autonomia dos sensores, do dinamismo da topologia das RSSF e do vasto conjunto de aplicações, com diferentes contextos e objetivos, não há um conjunto de protocolos e algoritmos computacionais amplamente eficiente para qualquer contexto de aplicação das RSSF. Dessa forma, diversos sistemas têm sido desenvolvidos com foco em demandas específicas de certas aplicações. Nesse sentido, pode-se destacar a utilização das RSSF no monitoramento agrícola, que consiste em observar continuamente uma área de plantio, com o objetivo de avaliar as mudanças ocorridas nesse ambiente. Geralmente, o foco desse monitoramento são as condições climáticas (temperatura, umidade, outros). Esse monitoramento é importante, pois fornece informações determinantes no processo de tomada de decisão do agricultor, sendo fundamental em procedimentos de prevenção de ataque de pragas e doenças e em procedimentos de manejo, como correção do solo e aplicação de insumos.

Em aplicações nas quais os dados se modificam lentamente, como no contexto da aplicação em agricultura de precisão, pode-se esperar que os nós sensores utilizem intervalos de amostragem longos em situações de normalidade (com baixa variação) e adotem intervalos de amostragem curtos, maior frequência, caso sejam observados dados que representam fenômenos que merecem maior atenção (situações especiais, como frio ou calor excessivo). A utilização dessa estratégia pode proporcionar maior economia de energia na RSSF, já que o número de medições de dados e o envio de pacotes pelos nós sensores da rede é reduzido. Esse procedimento não deve, no entanto, significar perda de eficiência da RSSF no processo de monitoramento do ambiente.

Em sua pesquisa, [13] empregou uma estratégia que considerava dois intervalos de amostragem distintos, um curto

e outro longo. Em sua proposta foi determinada uma faixa limite de valor para a propriedade monitorada. Sempre que o dado registrado pela rede de sensores ultrapassasse esse limite o sistema passava a considerar o segundo valor de intervalo disponível. Já [14] apresentou um Autômato Adaptativo (AA) [15] que permite a RSSF definir dinamicamente o intervalo de amostragem a ser adotado. Em sua proposta foi considerado um intervalo de normalidade para a propriedade monitorada, sempre que o sensor registrava um dado com valor fora do intervalo de normalidade, o sistema passava a reduzir gradativamente o intervalo de amostragem por meio do AA. Entretanto, nessa proposta ocorrem algumas limitações, o sistema não é totalmente autônomo, pois depende da definição prévia do conceito de "intervalo de normalidade", do número de intervalos de amostragem, bem como seus respectivos valores de tempo de espera. Em razão dessa rigidez, caso o sistema seja mal configurado, o objetivo de economizar energia pode ser atingido parcialmente ou de maneira insatisfatória.

Este trabalho apresenta um AA para o ajuste dinâmico dos intervalos de amostragem dos nós sensores da RSSF, independente de definições prévias de intervalos ou faixas de normalidades.

Este trabalho apresenta na Seção II uma breve contextualização do monitoramento agrícola, as potencialidades de se empregar RSSF nesse contexto e descreve a estratégia adaptativa que será utilizada pelo AA. A Seção III discute e descreve o AA para o problema de definição autônoma do intervalo de amostragem em RSSF, além de apresentar a Tabela de Decisão Adaptativa e sua verificação. Finalmente na Seção IV são feitas as considerações finais do trabalho.

II. RSSF NO MONITORAMENTO AGRÍCOLA.

A agricultura de precisão consiste em dividir o terreno da área cultivada em parcelas e tratá-lo de modo diferenciado (específico), buscando atender as necessidades de cada parcela. Dessa forma, espera-se o aumento da produção, menor custo e menor impacto no meio ambiente [16]. A aplicação das RSSF na agricultura de precisão é uma alternativa promissora, possibilitando um melhor monitoramento da cultura e das propriedades do ambiente de cultivo [12].

No monitoramento agrícola, geralmente os dados (fenômenos monitorados) variam lentamente. Assim, uma RSSF não precisa monitorar o ambiente com uma alta frequência de amostragem de dados, pois isso representa desperdício de energia.

Uma possível estratégia é considerar um intervalo de valores como normais (faixa de normalidade) e então fazer com que a rede de sensores adote intervalos com tempos longos enquanto registrar dados pertencentes a este intervalo, conforme propõe [14]. Entretanto, nem sempre é possível definir um intervalo adequado, além do fato que este intervalo pode ser dinâmico durante o monitoramento. Como por exemplo, diferentes momentos em um dia ou estações climáticas no ano. Dessa forma, é coerente considerar o parâmetro do intervalo de amostragem mínimo e máximo, e permitir que cada nó sensor da rede seja autônomo em decidir quando e quanto alterar o seu intervalo de amostragem. Assim, espera-se que o sensor seja capaz de aumentar a frequência das amostras quando ocorrerem variações consistentes nos dados que estão sendo monitorados e reduzir essa frequência caso contrário, economizando energia.

A estratégia descrita é representada na Figura 1 por meio de um gráfico. O eixo *x* corresponde ao tempo, enquanto que o eixo *y* ao valor do dado registrado. Os pontos marcados no gráfico correspondem ao instante e valor registrado por um sensor da rede. A linha representa o comportamento da grandeza monitorada pela RSSF. Nota-se que enquanto ocorre uma baixa variação entre os dados registrados pelo sensor há um intervalo maior entre uma amostra e outra. Enquanto que, quando ocorre uma alta variação nos dados obtidos, aumenta-se a frequência das amostras.

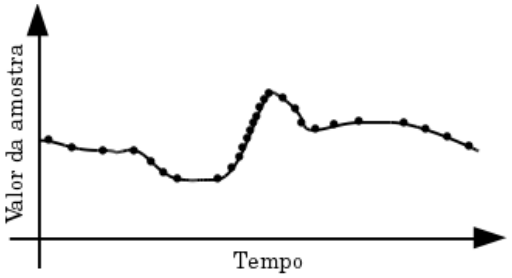


Fig. 1. Exemplo de uma sequência de medidas realizadas por um nó sensor ao longo do tempo. Quando há variação acentuada nos dados registrados o intervalo entre uma amostra e outra é reduzido.

Com essa estratégia, espera-se economizar energia enquanto o ambiente apresenta uma situação de baixa variação nas propriedades monitoradas e garantir a eficiência do monitoramento em situações de variação acentuada.

III. DESCRIÇÃO DO MODELO ADAPTATIVO

Um AA é uma máquina de estados a qual são impostas sucessivas alterações, resultantes da aplicação de ações adaptativas associadas às regras de transições executadas pelo autômato [15]. Estados e transições podem ser eliminados ou incorporados ao autômato em decorrência de cada um dos passos executados durante a análise da entrada. De modo geral, pode-se dizer que um AA é constituído por um dispositivo convencional (não-adaptativo) e um conjunto de mecanismos adaptativos responsáveis pela automodificação do sistema.

Para determinar a mudança nos intervalos de amostragem nos nós sensores da RSSF, é proposto um AA que assume as seguintes considerações:

- Cada estado do AA corresponde a um intervalo de amostragem para o nó sensor.
- O estado inicial corresponde ao intervalo de amostragem mais curto.
- Se o valor da amostra coletada pelo nó sensor

mantém-se estável, ou com baixa variação, então o AA altera seu estado por meio de uma função adaptativa, transitando para um estado que corresponde a um intervalo de amostragem maior.

- Se o valor da amostra coletada pelo nó sensor varia acentuadamente (acima de um limite), então o AA retorna para o estado inicial, para o menor intervalo de amostragem disponível.

Será considerado um parâmetro que corresponde à variação máxima que a grandeza monitorada pelo sensor pode sofrer. Neste trabalho ela será representada por *w*. Se a variação entre as amostras for superior a *w*, então o estado corrente do AA volta a ser o estado inicial. Caso contrário o AA transita para outro estado com intervalo de amostragem maior, que será definido dinamicamente. É importante destacar que um limite máximo para o valor do intervalo de amostragem deve ser determinado, para evitar um possível intervalo excessivamente longo, causando inconsistência no procedimento de registro dos dados pelo sensor.

O AA será executado em cada nó sensor da rede, de modo que cada nó é autônomo na decisão do seu intervalo de amostragem, em função dos fenômenos que estão sendo monitorados por ele.

A Figura 2 apresenta o AA proposto. No autômato, o símbolo = corresponde ao conjunto de amostras com baixa variação (inferior ao limite *w*), enquanto que o símbolo + corresponde ao conjunto de amostras com alta variação (superior ao limite *w*). Nota-se que quando ocorre uma leitura de um dado com baixa variação é executada uma ação adaptativa "pré". Essa ação irá criar um novo estado no autômato, que irá corresponder a um intervalo de amostragem maior que o estado atual e então o autômato irá transitar para esse novo estado.

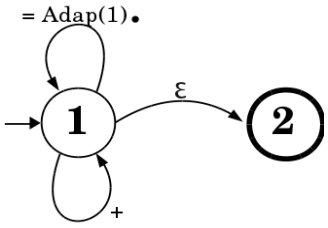


Fig. 2. Autômato Adaptativo para determinação autônoma do intervalo de amostragem em RSSF.

O novo valor para o intervalo de amostragem, a ser definido para o novo estado do AA, será inversamente proporcional à variação do dado. Portanto, quanto menor a variação, maior será o incremento no intervalo. Diferentes funções podem ser adotadas nesse processo, aqui será proposto a Equação 1 para determinação do valor do intervalo de amostragem:

$$t' = t + s \times (1 - k/w) \times z \quad (1)$$

onde:

t é o valor do intervalo de amostragem do estado atual em minutos;

s corresponde ao valor do intervalo de amostragem mínimo (em minutos);

k é a variação entre os dados registrados pelo sensor;

w corresponde à variação máxima permitida;

z é um fator multiplicador adimensional, que permite aumentar ou reduzir a grandeza da variação do intervalo.

A Figura 3 ilustra um exemplo de execução do AA proposto. Para melhor exemplificar o funcionamento do autômato, considere uma suposta aplicação em que a RSSF está monitorando a temperatura ambiente de uma cultura. Considere os seguintes parâmetros para o exemplo: o intervalo de amostragem inicial e mínimo (*s*) é de 5 minutos, a variação máxima (*w*) é de 0,5 ºC e o fator multiplicador (*z*) é 1. Na Figura 3(a) o AA se encontra em seu estado inicial, portanto o estado 1 do autômato corresponde ao intervalo de amostragem de 5 minutos. Supondo um novo registro do sensor, onde a variação entre os dados tenha sido de 0,1ºC, portanto menor que *w*, deverá ser executada uma ação adaptativa. Nesta ação será criado um novo estado no autômato (#1) que poderá ser alcançado com a variação entre os dados registrados (*k*), conforme demonstra a Figura 3(b). Considerando os dados do exemplo e a Equação 1, o valor do intervalo de amostragem do estado #1, será de 9 minutos. O novo estado criado também possui uma ação adaptativa. Ao imaginar um novo registro do sensor, supondo agora uma variação de 0,4 ºC, uma nova ação adaptativa é executada, criando um novo estado no autômato (##1), conforme ilustra a Figura 3(c). Considerando a variação de 0,4 ºC, o valor do intervalo do estado #1 (9 minutos) e os parâmetros iniciais, o novo estado do autômato (##1) terá o intervalo de amostragem igual a 10 minutos (incremento em 1 minuto).

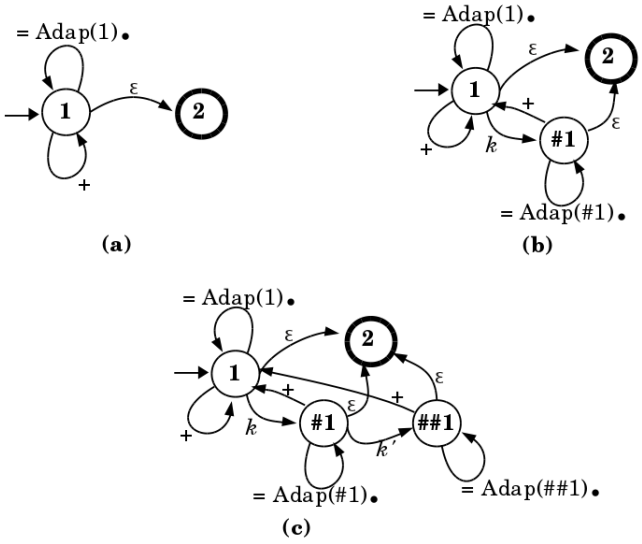


Fig. 3. Exemplo de execução do Autômato Adaptativo. Em (a) o autômato está em seu estado inicial. Em (b) ocorreu uma ação adaptativa no estado 1. Em (c) ocorreu uma nova ação adaptativa, agora no estado #1.

Em qualquer estado do AA se for registrado um novo dado cujo a variação seja superior a *w*, o autômato retorna para seu estado inicial, que corresponde ao menor intervalo de amostragem.

A. *Tabela de Decisão Adaptativa*

Durante a execução de uma ação adaptativa, o AA sofre alguma mudança em sua configuração (eliminando, acrescentando, ou simplesmente modificando estados e transições). Isto faz com que uma nova máquina de estados ocupe o lugar da anterior, caracterizando a execução de um passo adicional [15].

Uma função adaptativa é formada por um conjunto de ações adaptativas, que podem ser executadas em sequência na ocasião da aplicação da função. Há três ações elementares que podem ser usadas para compor as ações adaptativas, que permitem: consultar (representado por "?"), eliminar (representado por "-") e adicionar (representado por "+") transições no autômato. Uma Tabela de Decisão Adaptativa (TDA) permite representar as regras de uma função adaptativa e consequentemente representar toda a lógica do AA.

Considerando o AA apresentado no diagrama da Figura 2, para o problema de determinação autônoma do intervalo de amostragem em uma RSSF, a Tabela I apresenta a respectiva TDA, que descreve as ações adaptativas. Nesta TDA ocorrem somente ações adaptativas de adição (+). Após a aplicação da função adaptativa, é importante lembrar que a variação *k* obtida pelo nó sensor deve ser retirada do conjunto de entradas do estado atual, conjunto representado por "=". De modo a evitar ambiguidade no autômato.

TABELA I
Tabela de Decisão Adaptativa referente ao AA para determinação autônoma do intervalo de amostragem em RSSF

Label	Tag	Condições		Ações			Funções Adaptativas	Parâmetros	Geradores
		Estado =	Entrada =	Estado ↓	Consome	Aceita ↓	R1	P1	G1
1	S			1	√				
2	R	1	+	1	√				
3	R	1				x			
4	R	1	ε	2					
5	R	1	+	1			√	1	
6	R								
7	R	2				√			
8	E								
	H						√	√	√
	+	P1	=	G1	√				
	+	G1	+	P1	√				
	+	G1	=	G1			√	G1	
	+	G1	ε	2					

B. *Validação da TDA*

Visando verificar a TDA apresentada na Tabela I, foi utilizado o *software* AdapTools [17], que permite representar, implementar e simular o comportamento de um AA a partir da sua TDA. O sistema inclui ainda recursos de depuração, visualização de variáveis e representação gráfica.

As simulações executadas demonstraram o comportamento esperado do AA, portanto, a correta construção da TDA. A Figura 4 corresponde à representação da TDA no *software* AdapTools e a Figura 5 demonstra a representação gráfica do AA em um determinado instante da simulação.

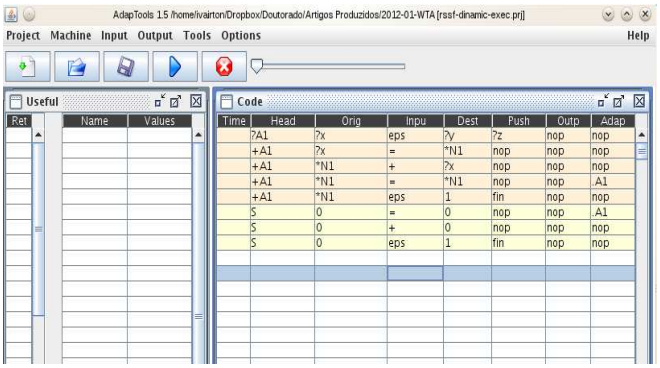


Fig. 4. Parte da tela principal do *software* AdapTools com o AA proposto representado.

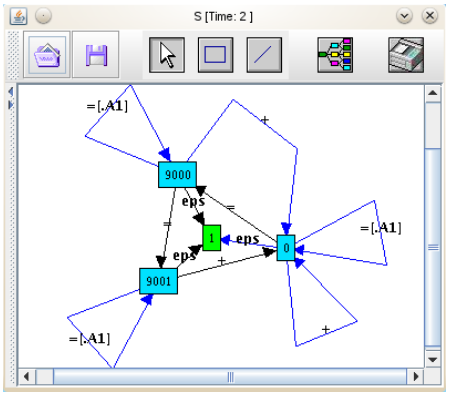


Fig. 5. Tela com a representação gráfica do AA durante a simulação no *software* AdapTools.

IV. Conclusões

As RSSF têm apresentado potencial de aplicação em diversas áreas. Um dos seus principais desafios é a economia de energia, sem que isso signifique perda de eficiência na utilização da rede de sensores. Para aplicações que não exigem um monitoramento com alta frequência de coleta de dados, como na agricultura de precisão, pode-se adotar uma estratégia em que o intervalo entre as amostras dos dados seja longo, enquanto as informações obtidas pelos sensores apresentarem baixa variação. Ao ocorrer variações acima de um limite pré definido, a rede de sensores se ajusta, de maneira autônoma, buscando monitorar o evento com maior frequência, reduzindo o intervalo entre as amostras de dados. Essa estratégia representa economia de energia nos nós sensores e significa maior tempo de vida útil da RSSF.

Este trabalho apresentou um AA capaz de modificar, de maneira autônoma, a configuração dos intervalos de amostragem dos nós sensores em uma RSSF. Esse processo

ocorre sem depender de definições prévias de intervalos de normalidade, ou quantidade de intervalos de amostragem e seus valores. O autômato proposto é capaz de definir dinamicamente a quantidade de intervalos de amostragem a serem utilizados pela rede de sensores e os seus respectivos valores. Essa autonomia da rede pode representar melhor desempenho da RSSF em diferentes condições, ampliando o contexto de aplicação. Sendo desnecessária a reconfiguração da RSSF.

Nos trabalhos futuros, a eficiência e a economia de energia promovida pelo AA devem ser exploradas, verificadas e mensuradas em um ambiente de simulação computacional. Pretende-se verificar também o comportamento do AA em situações onde os nós sensores registram baixas variações por longos períodos, seguido de variações bruscas. Nesse contexto o registro de dados pode ficar comprometido. É importante que o AA seja capaz de prever tal situação e evitá-la, determinando um limite máximo adequado para o intervalo de amostragem. Além disso, espera-se elaborar uma métrica que qualifique os valores gerados para os intervalos de amostragem, de modo que o AA possa escolher por intervalos mais apropriados e "aprender" durante sua execução, segundo os padrões encontrados.

AGRADECIMENTOS

Os autores agradecem a Fundação de Amparo à Pesquisa do Estado de Mato Grosso – FAPEMAT – pelo apoio a este trabalho via projeto de pesquisa Nº 465450/2009 e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES.

REFERÊNCIAS

[1] I. F. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, E. CAYIRCI. "Wireless sensor networks: a survey", Computer networks. 38, p.393-422, 2002.

[2] M. TUBAISHAT, S. MADRIA. "Sensor networks: an overview". IEEE Potentials. 22 (2), 2003.

[3] P. GAJBHIYE, A. MAHAJAN. "A survey of architecture and node deployment in Wireless Sensor Network." In: Applications of Digital Information and Web Technologies, ICADIWT, p.426-430, 2008.

[4] T. CAMP, J. BOLENG, V. DAVIES. A Survey of Mobility Models for Ad Hoc Network Research. Wireless Communications e Mobile Computing (WCMC), v. 2, p. 483-502, 2002.

[5] D. ESTRIN, L. GIROD, G. POTTIE, M. SRIVASTAVA. "Instrumenting the world with wireless sensor networks." In: International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, USA, 2001.

[6] G. J. POTTIE, W. J. KAISER. "Wireless integrated network sensors." Communications of the ACM 43, p.51–58, 2000.

[7] D. ESTRIN, R. GOVINDAN, J. HEIDEMANN, S. KUMAR. "Next century challenges: Scalable coordination in sensor networks." In: Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCom'99), Seattle, Washington, USA, ACM Press, 1999.

[8] J. YICK, B. MUKHERJEE, D. GHOSAL. Wireless sensor network survey. Computer Networks, v. 52, n. 12, p. 2292-2330, 2008.

[9] K. SOHRABY, D. MINOLI, T. ZNATI. Wireless Sensor Networks – Technology, Protocols, and Applications. Wiley, 2007.

[10] F. ZHAO, L. GUIBAS, Wireless Sensor Networks – An information processing approach. Elsevier, 2004.

[11] P. SIKKA, Wireless adhoc sensor and actuator networks on the farm. Nashville-USA: ACM. 2006. p. 492-499.

[12] I. M. SANTOS, M. A. DOTA, C. E. CUGNASCA. "Visão Geral da Aplicabilidade de Redes de Sensores Sem Fio no Monitoramento Agrícola no estado de Mato Grosso". Anais do Congresso Brasileiro de Agricultura de Precisão – ConBAP 2010. Ribeirão Preto/SP, Setembro de 2010.

[13] J. C. C. BENAVENTE, C. E. CUGNASCA, H. P. SANTOS. "Um Estudo da Variabilidade Microclimática em um Vinhedo Cultivado sob Cobertura Plástica Mediante o uso de uma Rede de Sensores Sem Fio". Anais do Congresso Brasileiro de Agricultura de Precisão- ConBAP 2010. Ribeirão Preto – SP. Setembro de 2010.

[14] I. M. SANTOS, M. A. DOTA, C. E. CUGNASCA. Modelagem de Autômato Adaptativo para a definição dinâmica do intervalo de amostragem em Rede de Sensores Sem Fio. In: V Workshop de Tecnologia Adaptativa, 2011, São Paulo. 2011.

[15] 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.

[16] J. P. MOLIN, "Tendências da agricultura de precisão no Brasil". In: Congresso Brasileiro de Agricultura de Precisão, 2004. Anais do Congresso Brasileiro de Agricultura de Precisão - ConBAP 2004. Piracicaba, p. 1-10, 2004.

[17] L. D. JESUS, D. G. D. SANTOS, A. A. D. CASTRO, H. PISTORI. WTA 2007 - II.2 - "AdapTools 2.0: Implementation and Utilization Aspects." IEEE Latin Am. Trans. 5, 527-532 (2007).

Ivairton Monteiro Santos é graduado em Ciência da Computação pela Universidade Federal de Mato Grosso (2002) e mestre em Ciência da Computação pela Universidade Federal Fluminense (2005). Atualmente é aluno de doutorado da Escola Politécnica da USP, sob a orientação do Prof. Dr. Carlos Eduardo Cugnasca. É membro do Laboratório de Automação Agrícola da Escola Politécnica da USP e professor da Universidade Federal de Mato Grosso, no Campus Universitário do Araguaia. Tem experiência na área de Ciência da Computação e seus interesses em pesquisa concentram-se em Redes de Sensores Sem Fio e otimização combinatória.

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: instrumentação 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).

Proposta de Modelo Adaptativo para Geração de Contextos na Recomendação de Locais

C. V. Crivelaro[1], F. J. Barth[2], R. L. A. Rocha[3]

***Resumo*— Sistemas de Recomendação dominaram o mercado, especialmente na Internet para a recomendação de itens para o usuário. No entanto, muitos Sistemas de Recomendação de Locais tratam apenas de um possível contexto do usuário, mesmo este podendo estar em vários contextos ao mesmo tempo, como trabalho e viagens de férias. Esta proposta tem como objetivo propor um modelo adaptativo para a criação de múltiplos contextos para o mesmo usuário e, assim, fazer recomendações mais aderentes.**

***Palavras-Chave*— Sistemas de Recomendação, Adaptatividade, Geolocalização**

I. Introdução

Os Sistemas de Recomendação ganharam grande apelo em aplicações comerciais, por recomendar novos itens aos usuários como locais para ir, produtos como livros [1], músicas e até amigos nas redes sociais. Em muitos casos, os Sistemas de Recomendação aumentaram os acessos aos itens dos sites ou mesmo ao número de vendas, como no site de venda de livros Amazon [1].

As recomendações são feitas de acordo com o histórico de navegação do usuário e pela comunidade em que o usuário está inserido. Assim, ele se sente familiarizado com o item oferecido. Neste caso de estudo, serão utilizados o locais como itens para recomendação do site brasileiro Apontador.

No site Apontador, o usuário pode encontrar informações sobre locais em todo o Brasil e fazer avaliações para compartilhar a sua experiência do local com a comunidade. Um dos desafios é apresentar aos usuários os locais mais aderentes ao seu interesse de visitação e que sejam mais próximos do seu cotidiano, como um restaurante diferente para a hora do almoço, um restaurante para o final de semana com os amigos, ou até mesmo locais para visitação durante as férias.

Ao fazer a recomendação de locais ao usuário, infere-se que o usuário vai gostar do local e atribui-se um valor para ordenar os *Top N* locais que lhe serão oferecidos. No entanto, o usuário pode modificar o seu contexto temporariamente como a verificação de locais para uma viagem ou ter dois contextos diferentes, como a passagem de alguns dias durante o meio da semana em uma cidade e o final de semana em outra cidade. Esse fato diminui a qualidade das recomendações, pois, se a distância entre os locais sugeridos e os avaliados for levada em consideração, pode haver recomendações fora do contexto ou fora do foco das sugestões em que o usuário não está mais presente.

Com este problema, esta proposta de trabalho propõe uma solução adaptativa para que o usuário possa ter vários contextos em paralelo à medida que ele visita ou avalia os locais no site e fornece mais informações sobre o seu contexto. Com mais de um contexto do usuário, é possível focar a recomendação no momento em que o usuário estiver dentro de um contexto específico, evitando a mistura de recomendações de locais diferentes.

Esta proposta está estruturada da seguinte forma: na Seção 2 é apresentada uma breve descrição sobre Sistemas de Recomendação; na Seção 3 é fornecida uma definição de Sistema de Recomendação com contexto de locais; na Seção 4 é apresentado um breve resumo sobre Tecnologia Adaptativa; na Seção 5 é apresentada a proposta deste trabalho; e na Seção 6 são apresentadas as medidas para avaliar se essa proposta terá um ganho sobre outras propostas já conhecidas.

II. Sistemas De Recomendação

Os Sistemas de Recomendação [2][3] surgiram no contexto de Sistemas de Recuperação de Informações, que consistiam na busca dos itens mais interessantes ao usuário baseado apenas em sua navegação e em seus termos de busca. Esses Sistemas de Recomendação são conhecidos como Sistema de Recomendação Baseados em Conteúdo.

Nessa abordagem, são comparados os itens que o usuário interagiu com os outros itens da base. Quanto mais próximos forem os itens sugeridos dos itens interagidos, melhores pontuações terão para a recomendação, sendo uma abordagem interessante para recomendação de locais parecidos com o que o usuário já visitou. Para fazer essa recomendação, basta tratar os itens como documentos e fazer a similaridade entre eles por meio do modelo de espaço vetorial com pesos TF–IDF (frequência de termos – inverso da frequência do documento). [4]. Entretanto, essa abordagem tem vários problemas devido à recomendação ser feita apenas com os dados do passado do usuário, e não ligar outros locais que não são próximos semanticamente.

No contexto de locais, as pessoas não estão interessadas apenas em lugares iguais aos anteriores que elas visitaram, mas também em lugares em que pessoas que têm mesmo interesse vão. Para isso, há os Sistemas de Recomendação por Filtragem Colaborativa.

Com o surgimento do *Tapestry* [5], surgiu o termo Filtragem Colaborativa, cuja ideia é, se os usuários tiveram os mesmo interesses no passado, terão os mesmos interesses no futuro. Isso implica não só em ligar os itens pelo seu conteúdo, mas ligar os itens ou os usuários pelo interesse. Com esse conceito, podemos ter recomendações de acordo com a comunidade de pessoas que vão aos mesmos lugares e oferecer os lugares que estão em maior destaque no momento

[1] Celso V. Crivelaro – Apontador e Escola Politécnica da USP – celso.crivelaro@apontador.com
[2] Fabrício J. Barth – Apontador – fabricio.barth@apontador.com
[3] Ricardo L. A. Rocha – Escola Politécnica da USP – luis.rocha@poli.usp.br

ao usuário que demonstra afinidade com outros usuários. Formalizando o problema, o Sistema Recomendação faz a sugestão de lugares baseado na matriz Usuários ($\mathcal{U}$) e Itens ($\mathcal{I}$) para gerar uma lista de Recomendações (R), sendo essas recomendações os itens que o usuário ainda não avaliou:

$$\mathcal{U} \text{ x } \mathcal{I} \rightarrow \text{R} \qquad (1)$$

Um exemplo de uma matriz $\mathcal{U}$ x $\mathcal{I}$ pode ser mostrado abaixo:

	i_1	i_2	i_3	i_4	i_5
u_1	3	1	3	5	
u_2	4		5	4	4
u_3		3		4	3
u_4	5		4		5
u_5		2		4	

Tabela 1: Matriz Usuário X Item de avaliações

Pela Tabela *1*, podemos ver as avaliações dos usuários *u* para os itens *i*. Nos espaços em branco, não houve avaliação. O objetivo dos Sistemas de Recomendação é inferir qual nota o usuário daria para o item não avaliado, para que ele seja recomendado ao usuário segundo o seu interesse.

Uma das formas de fazer essa inferência é usando técnicas Baseadas em Memória, como o Baseado em Vizinhança. Nessa técnica, é calculado o peso ou a similaridade entre os usuários ou itens e, para produzir a recomendação, é usada a média ponderada dos itens avaliados pela similaridade dos itens.

De acordo com Sarwar [6], para sistemas em que existem muito mais itens do que usuários, é recomendado basear o Sistema de Recomendação em Itens, pois achar usuários correlacionados tornaria a tarefa muito complexa pela esparsidade da matriz $\mathcal{U}$ x $\mathcal{I}$.

Dessa forma, para obter os locais relacionados, pode-se usar cálculos de similaridade, sendo o mais comum a Correlação de *Pearson*:

$$w_{i,j} = \frac{\sum_{u \in \mathcal{U}}(r_{u,i}-\bar{r}_i)\cdot(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}}(r_{u,i}-\bar{r}_i)}\sqrt{\sum_{u \in \mathcal{U}}(r_{u,j}-\bar{r}_j)}}, \qquad (2)$$

sendo que $\mathcal{U}$ é o conjunto de todos os usuários u que coavaliaram os itens *i* e *j*, $r_{u,i}$ a avaliação que o usuário *u* deu ao item *i* (valor numérico de 1 a 5) e $\bar{r}_i$ a média das avaliações para o item *i*.

Com um histórico de itens que o usuário já avaliou, é feita uma soma ponderada entre as similaridades do local sugerido *k* com as avaliações de outros locais que o usuário já avaliou:

$$p_{colaborativo}(u,k) = \bar{r}_k + \frac{\sum_{i \in \mathcal{I}}(r_{u,i} - \bar{r}_i)\cdot w_{i,k}}{\sum_{i \in \mathcal{I}}|w_{i,k}|}, \qquad (3)$$

sendo que $p_{colaborativo}(u,k)$ é a avaliação estimada do usuário *u* para *i* item i. Fazendo isso para os locais relevantes, basta ordenar para que se tenha a lista dos *Top N* locais para serem sugeridos ao usuário.

Pelos itens da recomendação serem locais geográficos, o sistema de recomendação sofre o problema do contexto do usuário; ou seja, as recomendações só terão sentido se conhecido onde o usuário está geograficamente. Dessa forma é necessário um Sistema de Recomendação Híbrido, que mescla abordagem colaborativa com abordagem baseada em contexto

Segundo Burke [10], o Sistema de Recomendação com o contexto combinado com o resultado do algoritmo colaborativo forma um Sistema de Recomendação Híbrido com característica de cascata; ou seja, o resultado do algoritmo colaborativo é alterado pelo fator de contexto para se obter o resultado final. Assim, a recomendação final é feita multiplicando o resultado do algoritmo colaborativo com o fator de contexto do usuário *u* e o local *k*:

$$p_{final}(u,k) = p_{colaborativo}(u,k) \times p_{contexto}(u,k) \qquad (4)$$

Vários trabalhos fazem sugestões de locais [7][8] em que o usuário tem um dispositivo móvel que fornece a captura da sua coordenada GPS no momento de gerar a sugestão. No caso do site Apontador e de outros sites similares, muitos usuários apenas visitam as páginas dos locais, não sendo possível descobrir a sua localização exata.

Dessa maneira, espera-se que os locais sugeridos para o usuário pelo algoritmo colaborativo se tornem mais relevantes de acordo com o resultado do fator de contexto.

III. Sistema de Recomendação com Contexto de Locais

Ao navegar pela busca de locais, o usuário está dando algumas dicas de como é o seu contexto ou região onde ele está, pois irá buscar por locais onde ele deseja frequentar e irá avaliar locais onde ele frequentou. Além disso, ele poderá também fazer *checkin* com a aplicação mobile do Apontador, indicando, no momento, que estava naquele local.

Diferente do contexto de aplicações como *Foursquare*, em que sabemos onde o usuário está no momento por meio dos *checkins*, a maioria dos usuários do site Apontador apenas visita as páginas dos locais nos seu computador pessoal, o que permite visitar as páginas dos locais das mais variadas cidades e distâncias do seu local de residência. Com esse problema, é necessário inferir o contexto do usuário para que a recomendação seja correta.

Com o contexto do usuário definido, deseja-se inferir o quanto o local sugerido *i* pelo algoritmo colaborativo está próximo do contexto do usuário. Como locais têm uma coordenada geográfica *(X, Y)* para identificação espacial, os modelos de predição podem ser usados para avaliar se o local está inserido no contexto do usuário. Desta forma, seria possível inferir se o usuário iria ao local sugerido *i* se ele visitou recentemente uma lista de locais $L = \{j_1, j_2, \ldots, j_k\}$.

O modelo proposto por Ye [9] se encaixa nesta necessidade. Para isso, foi definida uma função PL de probabilidade do usuário ir ao local *i* pela lista de lugares L que ele já visitou ou avaliou:

$$p_{contexto} = PL(i,L) = \prod_{j \in L} P(i,j), \qquad (5)$$

sendo que *P* calcula a probabilidade de um usuário visitar os dois locais apenas pela distância entre eles:

$$P(i,j) = a \times b^{d(i,j)} \quad (6)$$

Ye [9] propõe que a probabilidade de um usuário visitar um local *i* e um local *j* segue uma distribuição de probabilidade na forma de potência, sendo que a e b são os parâmetros da distribuição. Os parâmetros a e b são calculados para cada sistema usando método do Gradiente Descendente e conjunto de dados é cada distância entre lugares *i* e *j* que um mesmo usuário avaliou. A distância usada neste modelo é a distância euclidiana, já que temos as coordenadas *(X, Y)* dos locais *i* e *j*:

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (7)$$

Este modelo atende apenas quando analisamos o contexto cotidiano do usuário, que seriam os locais que ele vai durante o dia e os lugares próximos à residência do usuário. Por isso, este modelo não sabe distinguir se um novo local muito distante dos outros que o usuário visitou é uma busca fora do contexto do usuário ou se ele está mudando o seu contexto, como uma mudança de cidade ou interesse em viagem de férias.

IV. Tecnologia Adaptativa

A área de Tecnologia Adaptativa [11] tem mostrado um forte crescimento com aplicação de técnicas de aprendizado a partir de experiências anteriores e com aplicações na área de Data Mining [12].

O dispositivo adaptativo é composto de uma camada subjacente; que contém um dispositivo não adaptativo com um formalismo já conhecido, que forma o núcleo do sistema; e uma camada adaptativa. Suas ações modificam o núcleo apenas com as funções pré definidas e com os dados de entrada, tornando o dispositivo automodificável.

Assim, um dispositivo adaptativo pode ser formalizado como $DA = (ND_0, AM)$, com:

- DA: dispositivo adaptativo;
- ND_0: dispositivo não adaptativo subjacente;
- AM: Mecanismo Adaptativo, tal que MA $\subset AA\ x\ RDN\ x\ AP$;
- RDN: regras do dispositivo não adaptativo;
- AA: conjunto das funções adaptativas anteriores; e
- AP: conjunto das funções adaptativas posteriores.

As funções adaptativas contêm três regras básicas sobre o dispositivo subjacente:

1. Consulta de regra;
2. Adição de regra; e
3. Remoção de regra.

Dispositivos adaptativos são interessantes no ponto que o dispositivo subjacente pode ser usado tanto para processamento quanto para memória. Isso permite fazer tratamentos mais elaborados e se automodificar de acordo com o seu contexto. Podemos ver a adaptatividade como transições no espaço de todos os possíveis dispositivos não adaptativos, em que as funções adaptativas Posterior e Anterior escolhem qual dispositivo não adaptativo será executado.

V. Modelo Adaptativo Proposto

Para melhorar o fator de contexto na recomendação de locais é proposto um modelo adaptativo no dispositivo do contexto dos usuários.

A melhoria se daria em que o fator de contexto não trabalhe apenas com um contexto único como é apresentado na seção III, mas sim com vários contextos ao mesmo tempo, sendo-os inseridos durante a captura de locais que o usuário visitou ou avaliou. A adaptatividade é um conceito que se encaixa bem nessa necessidade, pois a atualização do contexto seria feita online de acordo com a navegação do usuário pelo site e pela busca de locais. Destas formas, poderíamos instantaneamente fazer recomendações com os novos contextos.

Cada contexto é uma lista $L_n = \{j_1, j_2, \ldots, j_k\}$ com os locais que o usuário visitou, tendo tamanho máximo $\| L \| = K$, sendo *K* um valor inteiro arbitrário.

Um contexto pode ser definido como um local *i* dentro de L_n, cuja distância euclidiana entre *i* e os itens j_n é menor do que um valor θ arbitrário para pelo menos $\frac{\|L\|}{2}$ locais do mesmo contexto. Ou seja, todos os locais estarão dentro de uma região física, sendo que a distância euclidiana é a distância definida entre os locais *i* e *j* pelas coordenadas *(X,Y)* de cada um.

Ao respeitar essa regra, não há nenhuma ação adaptativa, apenas o local mais antigo que o usuário interagiu será retirado na lista se $\| L \| = K$, e o novo local *i* será adicionado. Nesse momento, o dispositivo adaptativo apenas faz as ações de consulta em vez de executar alguma ação sobre o dispositivo subjacente, que apenas inclui o novo item na sua lista, conforme apresentado na Figura 1 e Figura 2.

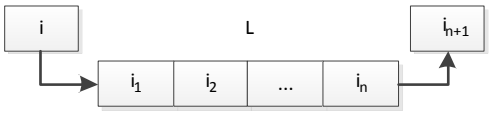


Figura 1 - Inserção de item na lista

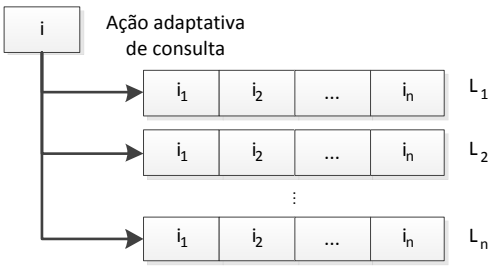


Figura 2 - Ação adaptativa de consulta

Se novo local *i* tiver mais do que $\frac{\|L\|}{2}$ locais distantes pelo limite θ em um contexto, será comparado com outro contexto. Se em nenhum contexto o local *i* puder ser inserido, então um novo contexto L_{n+1} será criado. Além disso, os locais dos outros contextos (tal que $D(i,j) < \theta$) serão inseridos no contexto recém criado, respeitando o tamanho $\| L_{n+1} \| \leq K$. Para isso serão inseridos os lugares mais próximos de *i*.

Nesse momento, a ação adaptativa cria uma nova regra que será o contexto L_{n+1}, fazendo a inserção do local i e dos outros locais, tal que $D(i,j) < \theta$.

Para a recomendação de um local com múltiplas listas, o valor do algoritmo colaborativo multiplica-se pelo fator de contexto. Havendo múltiplos contextos, o resultado não será apenas uma lista *TOP N* de recomendações, mas sim *n* listas de recomendações. Para o caso de um sistema que tiver apenas uma lista *Top N*, é necessário que o contexto tenha apenas mais importância para aquele local *i*:

$$Fator\ de\ Contexto = arg\max_{n} P(L_n, i) \quad (8)$$

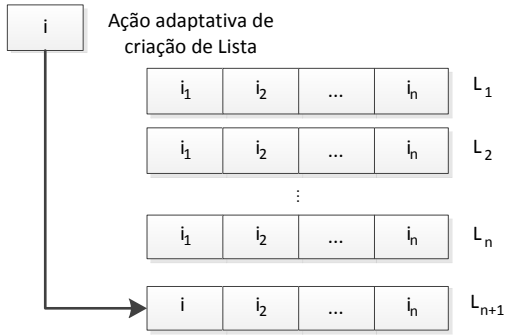


Figura 3 - Ação adaptativa de inclusão

Para fazer a recomendação final ao usuário, basta multiplicar o Fator de Contexto pelo resultado do algoritmo colaborativo. Assim, os locais mais distantes do contexto perderão importância na sugestão e os mais próximos ficarão em destaque.

Como os contextos podem ser temporários, como por exemplo uma viagem de férias, eles podem ser apagados depois de um tempo *T*. Dessa forma, se o último local inserido no contexto for inserido a mais tempo do que *T*, ele será eliminado, evitando a poluição de dados antigos.

VI. Resultados Esperados

Para avaliar a melhoria da inclusão da técnica adaptativa no Sistema de Recomendação baseado em Contexto, este será comparado com o mesmo Sistema de Recomendação sem o contexto.

A forma mais tradicional de medir a qualidade de um Sistema de Recomendação de acordo com [13], é pela acurácia da predição. Uma das medidas que será usada é a Raiz do Erro Quadrático Médio (*REQM*), que foi usada na avaliação do prêmio oferecido pela Netflix [14]:

$$REQM = \sqrt{\frac{1}{n}\sum_{u \in \mathcal{U}, i \in \mathcal{I}} (p_{u,i} - r_{u,i})^2} \quad (9)$$

Em que *n* é o total de avaliações sobre todos os usuários, $p_{u,i}$ é a avaliação inferida do item *i* ao usuário *u*, e $r_{u,i}$ é a avaliação real.

Outras medidas usadas são mais conhecidas na área de Recuperação de Informação, como a Precisão e o *Recall* [15]:

- **Precisão**: fração dos itens retornados que são relevantes;
- ***Recall***: fração dos itens relevantes que são retornados.

VII. Conclusões

Esse trabalho propõe a inclusão de tecnologia adaptativa para a recomendação de locais. A técnica estende o conceito de contextos para que seja possível detectar mais de um contexto ao mesmo tempo em que o usuário tenha interesse.

A aplicação desse trabalho não serve apenas para a recomendação de locais, mas também pode ser aplicada para recomendação de outros itens, como venda de produtos no *e-commerce*, em que o usuário pode ter vários interesses distintos na compra ou pode estar comprando para outras pessoas.

Referências

[1] Linden, G., Smith, B., and York, J. Amazon.com recommendations:Item-to-item collaborative filtering. IEEE Internet Computing 4, 1 (January 2003).

[2] X. Su and T. M. Khoshgoftaar, A Survey of Collaborative Filtering Techniques, Advances in Artificial Intelligence (2009).

[3] Adomavicius, G., Tuzhilin A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, (2005) 17(6): p. 734-749

[4] Baeza-Yates, R. AND Ribiero-Neto, B. 1999. Modern Information Retrieval. Addison-Wesley Longman, Boston, Mass.

[5] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (December 1992), 61-70.

[6] Sarwar, B. M., Karypis, G., Konstan, J. A., AND RIEDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th InternationalWorldWideWeb Conference (WWW10).

[7] C. Stiller, F. Roß, and C. Ament. Towards Spatial Awareness in Recommender Systems. In Proceedings of the 4th International Conference for Internet Technology and Secured Transactions – ICITST'09, Nov. 2009.

[8] M. Brunato and R. Battiti, "Pilgrim: A Location Broker andMobility-Aware Recommendation System", Technical Report DIT-02-092, Informatica e Telecomunicazioni, University of Trento.

[9] M. Ye, P. Yin, W.-C. Lee, and D. L. Lee. Exploiting Geographical Influence for Collaborative Point-of-Interest Recommendation. In Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2011.

[10] Burke, R., Hybrid Recommender Systems: Survey and Experiments, User Modeling and User-Adapted Interaction, v.12 n.4, p.331-370, November 2002

[11] 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,

[12] Camargo, R., Raunheitte, L., Convolução Adaptativa. Workshop de Tecnologia Adaptativa (5 : 2011 : São Paulo) Memórias do WTA´2011. – São Paulo ; EPUSP, 2011. 111 p

[13] Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 22 (1). 5-53

[14] Netflix prize, http://www.netflixprize.com/

[15] Ricci, F., Rokach, L., and Shapira, B., editors (2010). Recommender Systems Handbook. Springer.

Celso Vital Crivelaro é nascido em São Paulo - SP, em 06 de Janeiro de 1985 e graduou-se em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (2008). Desde 2009 é mestrando em Engenharia de Computação pela EPUSP. Atualmente é responsável pela área de Recomendações na empresa Apontador (www.apontador.com.br) e é membro do IEEE (Institute of Electrical and Electronics Engineers).

Fabrício J. Barth é nascido em Ibirama – SC, em 21 de junho de 1978. Graduou-se em Ciências da Computação pela Universidade Regional de Blumenau. É Mestre e Doutor em Engenharia de Computação pela EPUSP (2003 e 2009, respectivamente). Suas áreas de atuação incluem Recuperação de Informação, Aprendizagem de Máquina e Engenharia de Conhecimento. É membro da ACM (Association for Computing Machinery) e da SBC (Sociedade Brasileira de Computação).

Ricardo Luis de Azevedo da Rocha é natural do Rio de Janeiro-RJ e nasceu em 29/05/1960. Graduou-se em Engenharia Elétrica na modalidade Eletrônica na PUC-RJ, em 1982. É Mestre e Doutor em Engenharia de Computação pela EPUSP (1995 e 2000, respectivamente). Suas áreas de atuação incluem Tecnologias Adaptativas e Fundamentos de Computação e Modelos Computacionais. Dr. Rocha é membro da ACM (Association for Computing Machinery), do IEEE (Institute of Electrical and Electronics Engineers) e da SBC (Sociedade Brasileira de Computação).

Especificações adaptativas e objetos: Uma técnica de *design* de software a partir de *statecharts* com métodos adaptativos

Ítalo S. Vega

***Abstract*— The use of flowcharts for the modularization of software applications is a common practice in structured development. However, there is a recommendation to avoid the use of this tool for decomposing an application into modules. This paper presents a proposal to modularization in terms of objects from adaptive specifications. Over the course of designing the software architecture, various artifacts are produced, with emphasis on the state transition tables extended with adaptive methods. The resulting architecture defines static dependencies that facilitate the understanding of the implementation model and management of a software development project.**

Keywords—adaptive specifications, adaptive methods, static dependencies, software architecture, design patterns, principles of object modeling.

I. Introdução

O DESENVOLVIMENTO de aplicações de software deve ser conduzido de tal forma que o funcionamento de um computador automático programado ou, simplesmente, computador, atenda às necessidades e metas (NMs) do seu usuário. Nesse sentido, a disciplina de análise de um processo de desenvolvimento de software [1], [2] assume o propósito fundamental de especificar os requisitos de tal funcionamento:quais condições ou capacidades o computador deverá exibir para que o usuário seja atendido nas suas NMs [3]. Atuando como engenheiro de software, um programador assumirá a responsabilidade de elaborar um programa contendo instruções executáveis pelo computador e que o façam operar conforme a especificação declarada de requisitos.

A. Organização de instruções

Mas como elaborar tal programa? Uma forma é modelar o funcionamento do computador (percebido como uma máquina de estados) por descrição dos passos computacionais a serem por ele realizados. Este modelo da computação (sequência de passos computacionais) torna-se a base para que o programador elabore as instruções do programa, o que é feito pelo emprego de técnicas de *design* (projeto) e de implementação [2]. Neste sentido, Parnas descreve duas estratégias para a programação de um computador por modularização de instruções: (i) decomposição funcional e (ii) decomposição por tipos de dados abstratos. Muito embora as instruções estejam diretamente relacionadas aos passos computacionais, também torna-se importante, durante, a programação, a particular maneira como tais instruções são organizadas na estrutura do programa. Parnas [4], p. 1058, afirma que não se deve organizar as instruções utilizando a estratégia de decomposição funcional considerando apenas a sequência dos passos computacionais ao longo do tempo[1]:

> "We have tried to demonstrate by these examples that it is almost always incorrect to begin the decomposition of a system into modules on the basis of a flowchart ... Since, in most cases, design decisions transcend time of execution, modules will not correspond to steps in the processing."

Mas, se a recomendação é não derivar a organização das instruções de um programa a partir de um fluxograma, que alternativas devem ser consideradas? Seguindo a linha de Parnas, as instruções devem ser organizadas por tipos de dados abstratos cujas execuções são coordenadas por um módulo de controle [4], p. 1054.

B. Máquinas de estados

Uma outra possibilidade é expressar o funcionamento desejado por meio de uma máquina de estados que modela o comportamento do computador. Douglass [5] define uma máquina de estados da seguinte maneira[2]:

> "A finite state machine is a machine specified by a finite set of conditions of existence (called states) and a likewise finite set of transitions among states triggered by events".

Este artigo assume tal definição de máquina de estados, de modo que os comportamentos de um computador sejam essen-

Ítalo S. Vega trabalha no Departamento de Computação da Faculdade de Ciências Exatas e Tecnologia da Pontifícia Universidade Católica de São Paulo, São Paulo, capital, Brasil e-mail: italo@pucsp.br.

[1] Tradução livre: "Tentamos demonstrar, por esses exemplos, que é quase sempre incorreto começar a decomposição de um sistema em módulos com base em um fluxograma ... Assim, como na maioria dos casos as decisões de projeto transcendem o tempo de execução, módulos não corresponderão a passos do processamento."

[2] Tradução livre: "Uma máquina de estados finita é uma máquina especificada por um conjunto finito de condições de existência (chamados estados) e um igualmente conjunto finito de transições entre os estados, disparadas por eventos."

cialmente descritos, em modelos, por meio de estados e transições. É comum a representação de máquinas de estados por meio de tabelas de transição. Elas suportam a exploração sistemática do modelo de uma máquina que exibe comportamentos dependentes da sua história. Outra representação de máquinas de estados é proposta por Harel. Trata-se de um formalismo visual para representar modelos de máquinas cujo comportamento pode ser expresso por um conjunto de estados discretos conhecido por diagramas (de transição) de estados. O formalismo oferece suporte à especificação hierárquica de estados e à evolução concorrente de eventos [6].

A partir de uma especificação comportamental formulada por uma máquina de estados, diversas estratégias de implementação podem ser aplicadas para a obtenção das instruções e da sua organização:

- sequências de instruções de decisão a partir de uma condição expressa pelo valor de uma variável de estado [7];
- uma estrutura de dados matricial utilizada por uma lógica de controle [8].

Entretanto, tais estratégias conduzem a uma organização na qual as instruções encontram-se fortemente acopladas [1], [9]. A consequência primária deste acoplamento é que eventuais edições de instruções relacionadas com uma parte do modelo comportamental desencadeiam reedições (ao menos recompilações) de outras instruções que se encontram em módulos distintos. Dificuldades decorrentes de alterações em módulos que não se referem à implementação da funcionalidade revisada são resultado de uma arquitetura ruim [10], [11].

O paradigma de orientação a objetos introduz técnicas mais apropriadas para lidar com o problema de organização das instruções de um programa.

C. Paradigma de objetos

Na visão de Rumbaugh et al., um diagrama de estados deve ser utilizado para descrever os comportamentos de uma única classe de objetos [12], p. 90. Ou seja, uma classe, no paradigma de objetos, atua como elemento de organização de comportamentos no modelo de implementação. Por conseguinte, o programa será constituído por um conjunto de classes de objetos cujos comportamentos podem ser descritos por diagramas de estados.

Agora, seguindo Parnas, as classes não devem ser originárias dos passos computacionais que especificam o funcionamento de um computador. À medida que o funcionamento da máquina passa a ser descrito por um comportamento mais complexo, é de se esperar que sejam necessárias cada vez mais instruções para a realização dos passos computacionais. Lakos investiga a influência das dependências físicas (dependências entre instruções) em projetos de software de larga escala [11]. Ele reforça a importância da modularização e do desacoplamento entre grupos de instruções. Garlan e Shaw, por sua vez, apresentam diversos estilos arquiteturais para a organização das instruções de um programa. Um deles é o orientado a objetos [9]. Isso leva ao uso do paradigma de objetos como um modelo com foco primário na organização das instruções que constituem um programa de computador.

D. Programas e máquinas

Jackson também se preocupa com a estrutura de sistemas de software de grande porte [13]. Mas ele investiga uma outra abordagem para o problema. De acordo com Jackson, um programa é uma "descrição de uma máquina" [14]. Neste sentido, ele propõe que a máquina deve ser construída por programação; ela resulta da execução das instruções por um computador de propósito geral [15].

A Fig. 1 ilustra a ideia de Jackson, enfatizando que uma máquina concreta, ao executar as instruções de um programa, origina uma nova máquina em tempo de execução. Sob a perspectiva funcional, um programador deve elaborar um programa que descreva as funcionalidades da máquina concreta. Sob a perspectiva de Jackson, no entanto, um programador deve se preocupar com a descrição de uma nova máquina — programas como descrições de máquinas.

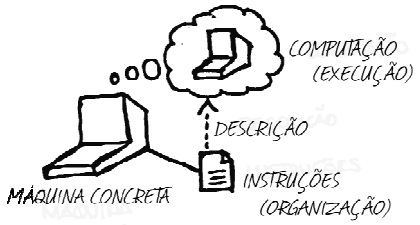


Figura 1. Programas e máquinas

E. Organização do artigo

Portanto pode-se elaborar um programa considerando que ele irá descrever uma nova máquina (virtual) cujo comportamento pode ser especificado por diagramas de estados. Este artigo apresenta uma proposta para a elaboração de programas que descrevem máquinas partindo de especificações comportamentais adaptativas. Na Seção II, aplicam-se as técnicas tabelas-verdade, tabelas de transição e tabelas de decisão na especificação funcional e comportamental de uma aplicação de software. Na Seção III, estendem-se as tabelas de decisão com métodos adaptativos, visando o suporte à reconfiguração de estados de uma máquina. Na Seção IV, emprega-se um padrão de projeto como base para originar uma arquitetura organizada por estados, mas com propriedades ruins de dependências Na Seção V, aplica-se um princípio de modelagem para revisar a arquitetura, com preservação de funcionalidade. Finalmente, na Seção VI conduz-se uma nova

revisão arquietural com suporte a edições sistemáticas das instruções do modelo de implementação.

II. Requisitos Funcionais e Comportamentais

O problema de avaliação de um aluno ao longo de um período letivo servirá como contexto de discussão neste trabalho. A avaliação envolverá a realização de até três provas, na seguinte ordem de ocorrência: *P1*, *P2* e *PS*. É em função das notas de tais provas que se calculará a média final do aluno para determinar a sua situação de aprovação.

A. Requisitos Funcionais

Um requisito funcional pode ser entendido como uma espécie de promessa de funcionamento da máquina. Parte do problema de programação se refere ao projeto de uma máquina cujo comportamento atenda à declarada promessa de funcionamento. A técnica de tabelas-verdade ajuda na identificação das condições que devem ser verdadeiras para acionar uma funcionalidade da máquina.

1) Condições de cálculo: As condições de cálculo da média final de um aluno serão expressas na forma de uma tabela-verdade. A Tab. I especifica sob quais condições as funcionalidades de cálculo poderão ser solicitadas. Por exemplo, na condição 1 (primeira linha da tabela), é verdade que o aluno realizou (fez) as três provas e que o período letivo foi encerrado. É possível, nesta condição, aplicar a fórmula de cálculo da média do aluno e determinar se foi ou não aprovado. Nesta situação, diz-se que o aluno foi avaliado. Um par de colchetes será utilizado para indicar uma condição. Um aluno foi avaliado se a condição [Avaliado] for verdadeira.

TABELA I

CONDIÇÕES DE AVALIAÇÃO

	[fpl]	[fezP1]	[fezP2]	[fezPS]	[Avaliado]
1	T	T	T	T	T
2	T	T	T	F	T
3	T	T	F	T	T
4	T	T	F	F	T
5	T	F	T	T	T
6	T	F	T	F	T
7	T	F	F	T	F
8	T	F	F	F	T
9	F	—	—	—	F

As condições 7, 9 e seguintes referem-se àquelas nas quais as funcionalidades de cálculo para a avaliação do aluno no período letivo não podem ser acionadas. Na condição 7, não se deve considerar o caso de alunos que fizeram apenas a prova PS, por se tratar de uma situação que não pode ocorrer neste domínio. Nos outros casos, não se pode calcular a situação do aluno sem o término do período letivo.

2) Ações de cálculo: As ações de cálculo nomeiam as funcionalidades da máquina em desenvolvimento. Elas serão apresentadas na forma de uma tabela de decisão, conforme descrito por Rowlett [16]. A Tab. II especifica os possíveis cenários de avaliação de um aluno. As ações (funcionalidades) e a ordem na qual deverão ser realizadas para o correto cálculo da situação no aluno são indicadas para cada um dos cenários de funcionamento.

No cenário 1, é verdade que o aluno realizou as três avaliações e houve encerramento do período letivo. As seguintes ações deverão ser realizadas. Primeiro, será feito o cálculo da sua média final pela aplicação da regra associada à ação p1p2psmf. Isto resultará no valor *MF*. Em seguida, aplicarse-á a regra da ação situaçãoMF, que resultará no valor *SIT*, considerando *MF*. Este valor indica aprovação ou reprovação do aluno. É verdade que um aluno estará [Avaliado] quando ele souber o valor de *MF* e o valor de *SIT* do seu caso.

TABELA II

FUNCIONALIDADES PARA A AVALIAÇÃO DE UM ALUNO

	Cenário						
	1	2	3	4	5	6	7
Condição							
[fpl]	T	T	T	T	T	T	T
[fezP1]	T	T	T	T	F	F	F
[fezP2]	T	T	F	F	T	T	F
[fezPS]	T	F	T	F	T	F	F
Ação							
p1p2psmf	1						
p1p2mf		1					
p1psmf			1				
p2psmf					1		
situaçãoMF	2	2	2		2		
situação				1		1	1

3) Fórmulas de cálculo: As especificações das fórmulas associadas ao cálculo da média final de um aluno, correspondente ao valor *MF*, encontram-se na Tab. III. A fórmula da ação p1p2psmf faz uso da função max(lista). Ao ser avaliada, tal função retornará a soma das duas maiores notas fornecidas no argumento lista. As demais fórmulas são de interpretação imediata.

TABELA III

FÓRMULAS PARA O CÁLCULO DA MÉDIA FINAL

Ação	Fórmula de *MF*
p1p2psmf	$\frac{\max(P1,P2,PS)}{2}$
p1p2mf	$\frac{P1+P2}{2}$
p1psmf	$\frac{P1+PS}{2}$
p2psmf	$\frac{P2+PS}{2}$

As regras que se referem ao cálculo da situação de um aluno encontram-se especificadas na Tab. IV. Estas regras encontram-se associadas às ações que determinam a situação de um aluno no final do período letivo, denotado por *SIT*. Na ação situaçãoMF, a aplicação da fórmula depende do valor *MF* calculado em um passo anterior. A fórmula situação, por outro lado, retorna um valor constante *RP*.

TABELA IV

FÓRMULAS PARA O CÁLCULO DA SITUAÇÃO

Ação	Fórmula de *SIT*
situaçãoMF	*AP*, se $MF \geq 5$
	RP, se $MF < 5$
situação	*RP*

A ordem na qual as ações deverão ocorrer ao longo do tempo é parte da especificação comportamental, tema da Subseção II-B.

B. ESPECIFICAÇÃO COMPORTAMENTAL

Uma categoria de modelos para lidar com problemas de sequenciamento de eventos no tempo baseia-se em autômatos de estados finitos [6], [17], [18]. Tais modelos podem ser utilizados para especificar particulares sequências de eventos considerando-se o estado corrente e o histórico de ocorrências.

Autômatos de estados finitos podem ser representados por tabelas de transição de estados. Em tais tabelas, cada coluna denota um tipo de evento e cada linha se refere a um estado do autômato. O preenchimento da tabela é feito de tal forma a indicar o estado seguinte quando da ocorrência de um evento do tipo nomeado pela respectiva coluna. Utiliza-se o símbolo — para indicar que o evento não provoca transição de estado e o símbolo → para mostrar qual o estado inicial do autômato.

TABELA V

TABELA DE TRANSIÇÕES DE ESTADOS DO PROBLEMA

	Estado	Evento			
		fezP1	fezP2	fezPS	fpl
→	**NãoAvaliado**	**2**	**3**	—	**4**
2	**FaltaP2PS**	—	**5**	**6**	**4**
3	**FaltaPS**	—	—	**7**	**4**
4	**Avaliado**	—	—	—	—
5	**P1P2MF**	—	—	**8**	**4**
6	**P1PSMF**	—	—	—	**4**
7	**P2PSMF**	—	—	—	**4**
8	**P1P2PSMF**	—	—	—	**4**

A especificação das sequências de eventos para o caso em estudo encontra-se na Tab. V. Partindo-se do estado inicial de um aluno que ainda não realizou provas no período letivo, indicado por **NãoAvaliado**, três situações podem ocorrer: realização da prova *P1* ou da prova *P2* ou o encerramento do período letivo sem a realização de provas. Ou seja, partindo-se do estado **NãoAvaliado**, apenas três transições originadas neste estado deverão ser permitidas. No texto, tipos de eventos serão indicados por uma sublinha ondulada. Assim, partindo-se do estado **NãoAvaliado** da Tab. V, na ocorrência de um evento do tipo

- fezP1, o autômato passa para o estado **FaltaP2PS** (representado por **2**);
- fezP2, autômato passa para o estado **FaltaPS** (representado por **3**);
- fpl, autômato passa para o estado Avaliado (representado por **4**).

1) Visualização por diagramas de transição de estados: O modelo comportamental pode ainda ser descrito por um Diagrama de Máquina de Estados UML (DME) [19] com a semântica estabelecida por Harel. A Fig. 2 mostra o diagrama deste modelo cujo propósito é especificar a semântica da ordem de ocorrência das notas das provas realizadas ao longo do tempo. No diagrama, o estado inicial é **NãoAvaliado**. Três transições partem deste estado, nomeadas de acordo com o tipo do evento de disparo: fezP1, fezP2 e fpl. Na ocorrência de um evento do tipo fezP1, passa-se para o estado **FaltaP2PS** e assim por diante.

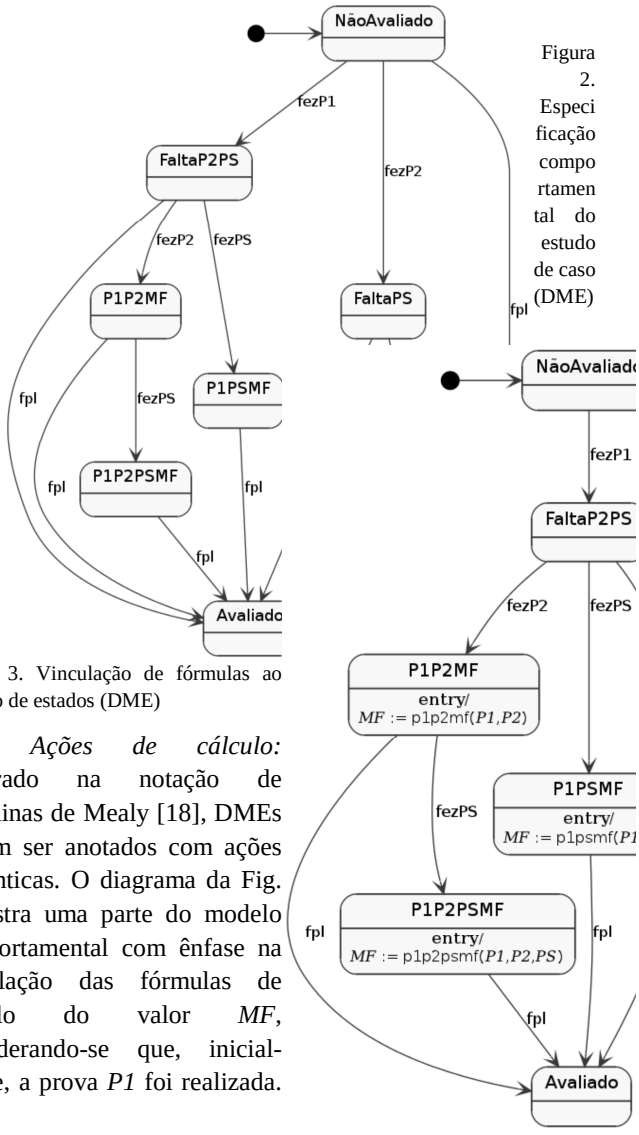


Figura 2. Especificação comportamental do estudo de caso (DME)

Figura 3. Vinculação de fórmulas ao modelo de estados (DME)

2) Ações de cálculo: Inspirado na notação de Máquinas de Mealy [18], DMEs podem ser anotados com ações semânticas. O diagrama da Fig. 3 ilustra uma parte do modelo comportamental com ênfase na vinculação das fórmulas de cálculo do valor *MF*, considerando-se que, inicialmente, a prova *P1* foi realizada.

No estado **FaltaP2PS**, o passo computacional *MF*:=p1p2mf(*P1*,*P2*) será realizado quando da ocorrência de um evento do tipo fezP2. A sua realização calculará a média final do aluno a partir das notas *P1* e *P2*. Este comportamento ocorrerá na entrada do estado **P1P2MF**. Tal tipo especial de evento é indicado por entry.

Deve-se interpretar os demais casos envolvendo as ações associadas aos eventos do tipo entry desta mesma forma. Neste diagrama, observa-se que a organização das ações de cálculo do valor *MF* é determinada pelos diferentes estados da máquina. Na transição para o estado **P1P2MF**, aplica-se a fórmula p1p2mf e assim por diante.

III. Especificações Adaptativas

Na Seção I foram apresentadas diversas estratégias para a obtenção e organização das instruções que compõem um programa. Neste trabalho, entretanto, será proposta uma estratégia alternativa para a elaboração do modelo de implementação. Especificações utilizando autômatos adaptativos dirigidos por regras serão consideradas como ponto de partida para a elaboração de um modelo de implementação baseado no paradigma de objetos.

TABELA VI

Tabela de transições adaptativas de estados do problema

	Estado	Evento			
		fezP1	fezP2	fezPS	fpl
→	**NãoAvaliado**	ma1P1	ma1P2	—	fa1fpl
		2	**3**	—	**4**
2	**FaltaP2PS**	—	ma2P2	ma2PS	ma2fpl
		—	**5**	**6**	**4**
3	**FaltaPS**	—	—	ma3PS	ma3fpl
		—	—	**7**	**4**
4	**Avaliado**	—	—	—	—
		—	—	—	—
5	**P1P2MF**	—	—	ma5PS	ma5fpl
		—	—	**8**	**4**
6	**P1PSMF**	—	—	—	ma6fpl
		—	—	—	**4**
7	**P2PSMF**	—	—	—	ma7fpl
		—	—	—	**4**
8	**P1P2PSMF**	—	—	—	ma8fpl
		—	—	—	**4**

Um autômato adaptativo é um autômato que possui a capacidade de alterar a sua própria estrutura de estados e transições (topologia) por meio da realização de ações adaptativas de inserção e de remoção, conforme definido por Neto [20]. No caso de uma representação na forma de diagrama de transições de estado adaptativas, Vega propõe uma semântica operacional implementada por um modelo de objetos [21]. O trabalho de Neto et al. [22] a respeito de *statecharts adaptativos*, servirá como base para a elaboração de um modelo comportamental para o caso da avaliação de um aluno ao longo de um período letivo.

A. Especificações comportamentais com funções adaptativas

Para especificar a ordem de ocorrência de tipos de eventos com autômatos adaptativos, será feito uso de uma abstração da semântica das funções adaptativas proposta por Neto [20] denominada métodos adaptativos.

1) Métodos Adaptativos: A cada transição do autômato finito subjacente, será vinculada uma abstração do efeito conjunto das funções adaptativas *before* e *after* sobre a configuração da topologia de estados e transições do autômato. Deste modo, na ocorrência de um evento, altera-se o escopo da semântica de transição do autômato subjacente de acordo com o resultado da execução do método adaptativo a ela associada. Isso poderá provocar uma reconfiguração na topologia do autômato subjacente, em termos similares aos quais um autômato adaptativo se altera por força da aplicação de alguma função adaptativa.

Uma apropriada combinação de tais efeitos dos métodos adaptativos deverá ser o objetivo durante a elaboração de uma especificação de requisitos comportamentais. Associada a cada transição, dever-se-á descrever quais mudanças a configuração do autômato deverá sofrer para que se expresse o comportamento desejado.

2) Especificação Adaptativa: A Tab. VI apresenta o resultado da vinculação de abstrações de funções adaptativas (métodos adaptativos) no estudo de caso deste trabalho.

Na Fig. 4 encontra-se um diagrama de transições de estado estendido para expressar os métodos adaptativos (DMA) da configuração inicial da topologia do autômato adaptativo que modela a especificação comportamental do estudo em andamento. Os nomes das transições correspondem aos nomes sublinhados dos tipos de eventos vinculados. Precedendo o nome do tipo do evento, encontra-se o nome do método adaptativo associado à transição.

Uma vez nomeados e vinculados os métodos adaptativos a cada uma das transições, parte-se para a modelagem das alterações topológicas que caracterizam o comportamento da máquina sendo projetada.

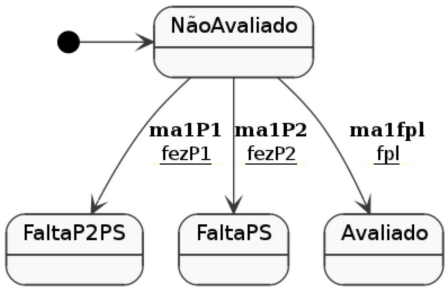


Figura 4. Configuração inicial da máquina de estados adaptativa (DMA)

3) Efeito de ma1P1 : O efeito do método adaptativo ma1P1 deve ser tal que a configuração da topologia do autômato seja alterada para aquela apresentada na Fig. 5. Ocorrendo um evento do tipo que dispare a transição fezP1 no modelo da configuração ilustrada na Fig. 4, a topologia deverá ser alterada de acordo com o modelo representado pelo diagrama da Fig. 5. Observe-se que o estado **FaltaP2PS** tornou-se inicial na nova configuração da topologia do autômato.

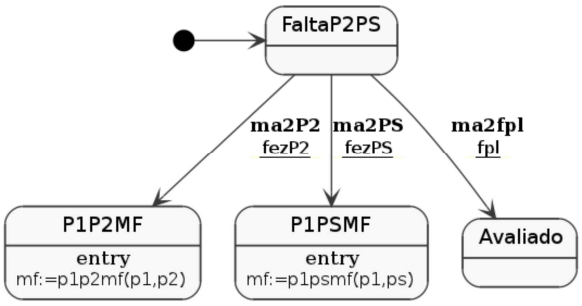


Figura 5. Efeito do método adaptativo ma1P1 (DMA)

4) Efeito de ma1P2 : A ocorrência de eventos que disparem transições do tipo fezP2 durante o estado inicial **NãoAvaliado** deverá provocar uma nova transformação na configuração da topologia do autômato (Fig. 6). A execução do método adaptativo ma1P2 tornará inicial o estado **FaltaPS** e novas transições com métodos adaptativos farão parte da reconfiguração topológica. Apenas dois outros estados podem ser atingidos nesta nova configuração.

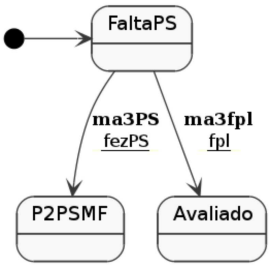


Figura 6. Efeito do método adaptativo ma1P2 (DMA)

5) Efeito de ma1fpl : A ocorrência de eventos que disparam a transição fpl na configuração ilustrada no diagrama da Fig. 4 alterará a configuração da topologia conforme o modelo da Fig. 7. Nesta configuração existe apenas o estado inicial **Avaliado**: houve encerramento do período letivo e o aluno não realizou prova alguma.

Figura 7. Efeito do método adaptativo ma1fpl (DMA)

6) Efeitos dos demais métodos adaptativos: A diagramação dos modelos referentes às demais mudanças de configurações segue uma linha de raciocínio similar. A título de ilustração, o diagrama da Fig. 8 ilustra a configuração de estados que resulta da execução dos métodos adaptativos referentes à sequência de transições fezP1, fezP2 e aluno avaliado devido ao encerramento do período letivo (fpl).

Isto conclui a elaboração da especificação adaptativa do estudo de caso. Com base em tal especificação, dever-se-á elaborar o modelo de uma máquina que se comporte conforme tal especificação. O projeto do modelo seguirá o paradigma de objetos para a concepção da arquitetura de implementação.

IV. ORGANIZAÇÃO COM *STATE*

Um possível modelo orientado a objetos para a implementação de máquinas de estados segue a proposta do padrão de projeto *State* de Gamma et al. [23]. A Fig. 9 ilustra a estrutura do padrão por meio de um diagrama de classes UML (DCL). A variabilidade comportamental da classe-papel Contexto é encapsulada na hierarquia de classes que se origina no ancestral cujo tipo-papel é *Estado*. Este tipo define uma interface de operações que devem ser implementadas de forma apropriada, em função do estado que se encontram os objetos da classe-papel Contexto. Cada descendente implementa a operação que irá determinar o comportamento do Contexto em um dos seus particulares estados.

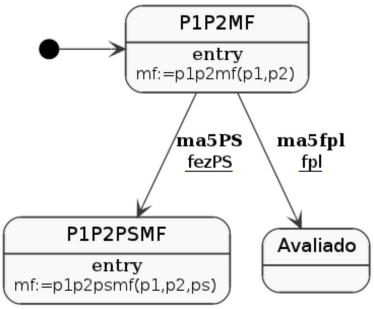


Figura 8. Efeito do método adaptativo ma1P2 (DMA)

No estudo de caso, a topologia adaptativa do autômato assume o papel de Contexto e as diferentes configurações topológicas serão vistas como sendo do tipo *Estado*, ancestral da hierarquia de classes do padrão *State*. Desta forma, tem-se a organização inicial da implementação do modelo comportamental representado pelos diagramas de efeitos das funções adaptativas da Seção III. As próximas subseções discutem os refinamentos a partir deste ponto.

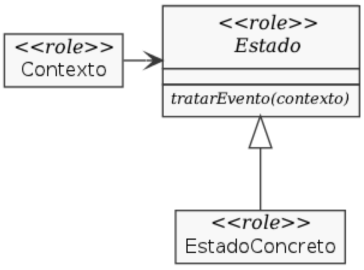


Figura 9. Estrutura do padrão de projeto *State* (DCL)

A. Estratégia inicial para o projeto de classes

O foco primário do paradigma de objetos é na organização das instruções do código-fonte [11], [24]. Seguindo a proposta de Gamma et al., pode-se modelar a topologia adaptativa do autômato por meio da classe TopAdapt. Em relação ao padrão *State*, esta classe corresponde ao papel Contexto. As diferentes configurações desta topologia serão modeladas como elementos do tipo *Config*, conforme ilustrado na Fig. 10. O modelo de implementação será derivado a partir de cada uma das configurações adaptativas especificadas na Tab. VI. Cada estado da especificação de requisitos comportamentais originará uma classe de implementação. Assim, para o modelo representado na Fig. 4 ter-se-á uma classe do tipo *Config* no modelo de objetos, oriunda dos estados de chegada das transições do estado **NãoAvaliado**.

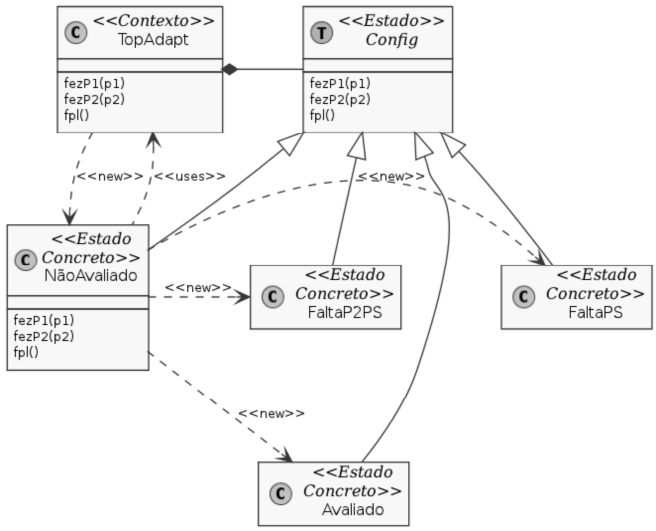


Figura 10. Organização inicial das classes do estudo de caso (DCL)

O tratamento da ocorrência de um evento do tipo fezP1 se inicia em um objeto da classe TopAdapt. A reação deste objeto é tal que ele encaminha uma mensagem envolvendo a operação fezP1(*P1*) para um objeto do tipo *Config*, mas da classe NãoAvaliado. Esta parte do modelo de objetos implementa a transição do estado **NãoAvaliado** para o estado **FaltaP2PS** da especificação adaptativa.

A execução do método adaptativo ma1P1 envolve uma transformação na estrutura do modelo de objetos: torna-se necessária a presença de uma variável para armazenar o valor da nota *P1* informada na ocorrência do evento fezP1. O objeto da classe NãoAvaliado instancia um FaltaP2PS e, em seguida, passa-lhe o valor de *P1* para armazenagem. Isso introduz uma dependência estática (de compilação de código-fonte) entre estas duas classes. Da mesma forma refina-se o modelo na ocorrência do evento fezP2.

A transição adaptativa fpl da configuração inicial apenas desencadeia a instanciação de um objeto da classe Avaliado , substituindo o objeto NãoAvaliado na ligação com o TopAdapt. Ou seja, realizada esta transição, apenas dois objetos passarão a existir na computação em andamento: um da classe TopAdapt e outro da classe Avaliado.

Por conseguinte, o diagrama de estados adaptativo da Fig. 5 é implementado por um objeto da classe TopAdapt, inicialmente conectado a um objeto da classe NãoAvaliado, mas do tipo *Config* . Na ocorrência de uma das transições fezP1 , fezP2 ou fpl, este objeto instancia um outro da classe FaltaP2PS, FaltaPS ou Avaliado, respectivamente, comandando a sua substituição por aquele recém instanciado. A substituição será realizada pelo objeto da classe TopAdapt. Como resultado, este objeto ficará conectado ao objeto que representa a nova configuração topológica.

Uma importante propriedade deste modelo se refere à encapsulação do efeito da transição e da adaptação nas classes descendentes. Por outro lado, as setas (de dependências de uso e de herança) revelam preocupações de acoplamento de instruções durante o refinamento do modelo. Em particular, a dependência cíclica entre TopAdapt e NãoAvaliado significa que o entendimento de uma classe está vinculado ao entendimento da outra (e vice-versa). O refinamento do modelo de implementação referente às classes FaltaP2PS e FaltaPS introduzirá novas dependências na direção de TopAdapt.

B. Arquitetura inicial que satisfaz à especificação adaptativa

O diagrama de classes UML apresentado na Fig. 11 foi produzido com a ajuda da ferramenta BlueJ em um projeto liderado por Kölling [25]. Há uma ênfase apenas nos relacionamentos de herança e de dependências entre as classes de implementação. Isso é suficiente para que sejam investigados os efeitos de mudança de instruções no código-fonte. A direção das setas indica a direção de dependência; a seta aponta para a classe que, se alterada, afetará a classe que se encontra na base da seta.

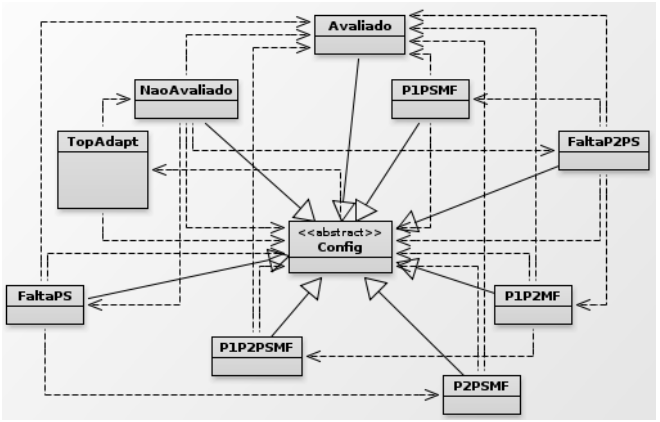


Figura 11. Arquitetura do modelo de implementação com *State* (BlueJ)

Observa-se que todas as classes descendentes de *Config* dependem de TopAdapt. A origem de tais dependências é que a configuração corrente deverá instanciar a configuração seguinte e passá-la para a topologia adaptativa (como um

objeto do tipo *Config*) de modo que o modelo de implementação reflita a topologia originalmente especificada.

Por outro lado, a dependência da classe TopAdapt para a NãoAvaliado se refere à instanciação do objeto que implementa a configuração inicial da topologia. TopAdapt instancia o objeto da classe NãoAvaliado, conectando-se a ele. Tal estrutura de objetos implementa uma configuração adaptativa inicial, conforme especificado no diagrama da Fig. 4.

Uma eventual edição de instruções da classe TopAdapt afetaria quais outras classes desta arquitetura? A ferramenta BlueJ usa um padrão de hachuramento para ajudar na visualização do alcance das alterações. Alterando-se a classe TopAdapt ela fica hachurada. Mas também ficarão hachuradas todas as demais classes que dela dependem (seja por uso, seja por herança). Na arquitetura até o momento elaborada, as notícias não são boas: todas as demais classes serão também hachuradas. A imagem da Fig. 12 mostra o padrão de hachuramento quando se altera alguma instrução da classe TopAdapt.

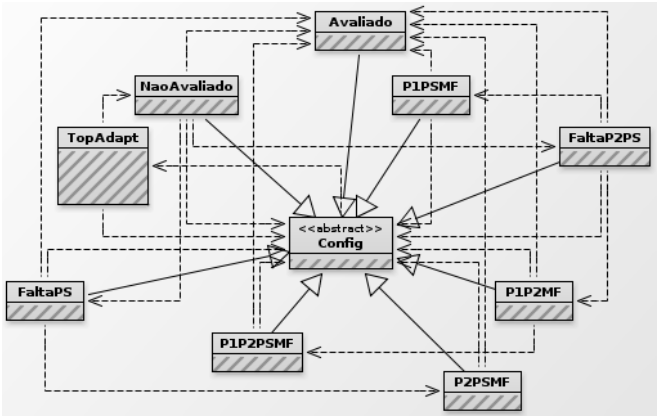


Figura 12. Classes afetadas por alterações em TopAdapt (BlueJ)

Devido a esta arquitetura de dependências, a edição de qualquer outra classe também provocará hachuramento generalizado. Tal propriedade resulta da dependência cíclica entre as classes de instruções deste modelo de implementação.

Sob o ponto de vista de controle de edições, tal arquitetura não é recomendável. Martin [10] propõe uma série de métricas para quantificação e crítica das propriedades de acoplamento e coesão. Neste trabalho, utilizar-se-á uma avaliação mais simples, baseada no padrão de hachuramento de classes do BlueJ: quanto maior a quantidade de classes hachuradas, pior é arquitetura de classes de implementação.

Como reorganizar as instruções desta implementação, ainda que satisfazendo à especificação adaptativa discutida na Seção III? O Princípio da Inversão de Dependências (DIP[3]) pode ajudar neste ponto.

[3] Dependency Inversion Principle

V. Desacoplamento por DIP

Martin afirma que os ciclos de dependências podem ser interrompidos pela introdução de abstrações de desacoplamento [24]. O princípio DIP tem o seguinte enunciado:

> "(A) High level modules should not depend upon low level modules. Both should depend upon abstractions.
>
> (B) Abstractions should not depend upon details. Details should depend upon abstractions."[4]

A abstração *Topologia* resulta do emprego do DIP no modelo de classes até agora elaborado. Para ela são redirecionadas as dependências das classes que descendem de *Config*.

A. Refatoração

Nesta refatoração (no sentido de Fowler [26]), preserva-se a equivalência comportamental, embora a organização das instruções que compõem o programa tenha sido alterada. Em termos de manutenção de código, agora, alterações em TopAdapt não se propagam para as classes descendentes de *Config*. A abstração *Topologia* atua como uma barreira de alterações, impedindo que elas alcancem a classe NãoAvaliado. Por outro lado, alterações em NãoAvaliado propagam-se para TopAdapt. Tal dependência resulta da necessidade de instanciação do objeto que denota o estado inicial da configuração representado pelo diagrama da Fig. 13.

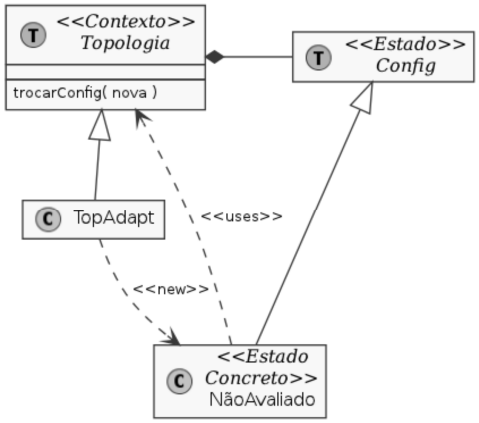


Figura 13. Desacoplamento de TopAdapt (DCL)

Com esta alteração arquitetural (introdução do tipo *Topologia*), o modelo de implementação assume a forma mostrada no diagrama da Fig. 14.

[4] Tradução livre: "(A) Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações. (B) Abstrações não devem depender de detalhes. Detalhes devem depender de abstrações."

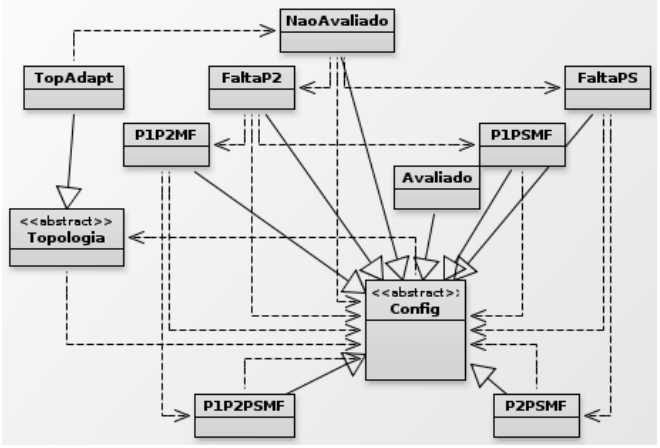


Figura 14. Revisão da arquitetura de classes com DIP (BlueJ)

B. Análise de alterações

O que acontece quando se altera alguma instrução da classe TopAdapt? Como nenhuma outra classe desta arquitetura depende de TopAdapt, apenas ela ficará hachurada. A Fig. 15 ilustra esta afirmação. Por outro lado, dependendo da classe descendente de *Config* que se altere, diversas outras ficarão hachuradas. Alguma mudança em instruções da classe NãoAvaliado, desencadearão alterações em TopAdapt: serão hachuradas. Entretanto, alterações na classe Avaliado, não provocarão hachuramento apenas nas classes *Topologia*, *Config* e Aprovado. Todas as demais dependem, direta ou indiretamente, da classe Avaliado.

Uma técnica complementar de desacoplamento pode ser empregada no sentido de manter constante a família de classes hachuradas por decorrência de edições: desacoplamento por *Factory Method*.

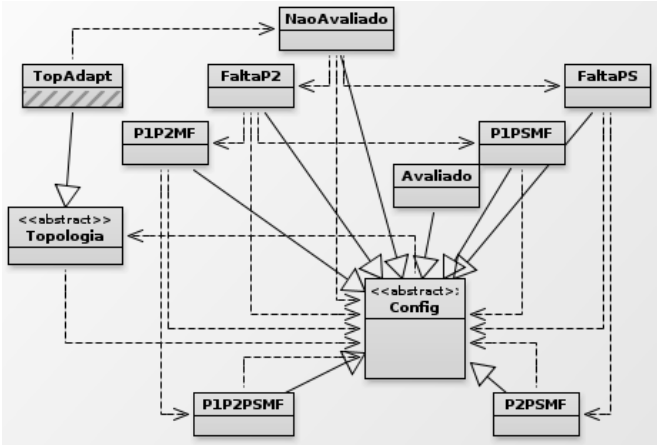


Figura 15. Alteração em TopAdapt (BlueJ)

VI. Desacoplamento por Factory Method

Gamma et al. [23] propuseram três grupos de padrões de projeto: criação, estrutura e comportamento. Aqueles de criação procuram encapsular a lógica de instanciação de objetos em tempo de execução de modo que haja um isolamento entre a natureza e o uso objeto. Ou seja, a interação entre dois objetos exige que eles conheçam as suas naturezas (classes) ou é suficiente o conhecimento dos seus usos (tipos) para a realização de um passo computacional?

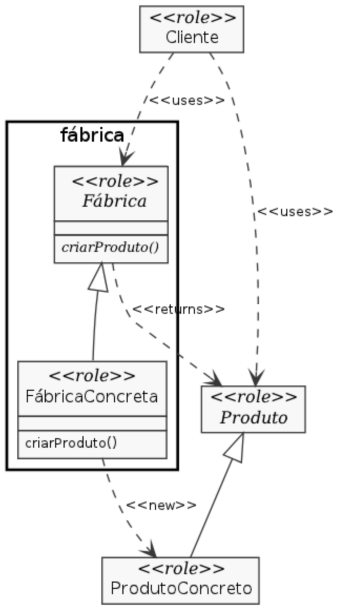


Figura 16. Estrutura de classes do Padrão *Factory Method* (DCL)

No problema em estudo, as classes que representam configurações de estados da topologia são acopladas em decorrência de precisarem instanciar as suas configurações sucessoras. Consequentemente, refinamentos em uma classe que se propõe a implementar uma particular configuração desencadeia revisões em classes que representam configurações anteriores. O padrão de projeto *Factory Method* procura resolver este problema por meio de uma interface de instanciação que redireciona a dependência para uma interface de uso, ao invés de uma interface de natureza (Fig. 16). Objetos da classe Cliente solicitam à *Fábrica* objetos que serão utilizados como *Produto* , não importando se eles terão a natureza de um ProdutoA ou ProdutoB .

A. Desacoplamento do espaço de configurações

O padrão de projeto *Factory Method*, aplicado ao problema tratado neste trabalho, resulta no modelo de classes destacadas no diagrama da Fig. 17.

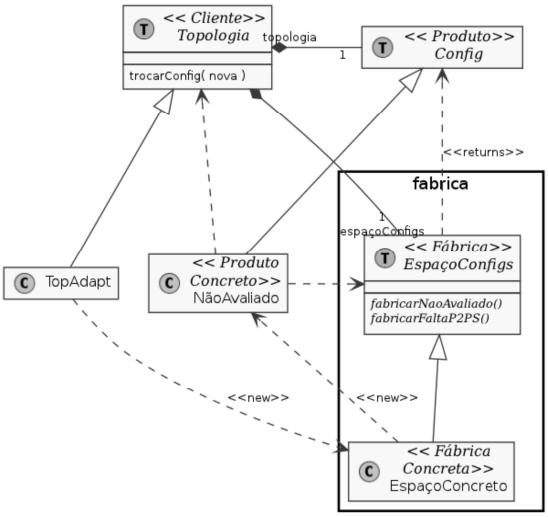


Figura 17. EspaçoConfigs como método de fabricação (DCL)

Em primeiro lugar, TopAdapt instancia um objeto de natureza EspaçoConcreto de configurações do tipo *EspaçoConfigs*. Deste modo, descendentes de *Config* (como é o caso de NãoAvaliado, por exemplo) poderão solicitar ao objeto do tipo *EspaçoConfigs* que instancie a próxima configuração da *Topologia*, sem ter uma dependência estática para isso.

Uma vez instanciado um novo objeto do tipo *Config* (por outro do mesmo tipo), ele é enviado para que a *Topologia* faça a troca da sua configuração. Por exemplo, considerando-se a configuração inicial do autômato, na ocorrência do evento fezP1, um objeto TopAdapt encaminha uma mensagem solicitando a realização da operação fezP1() para um objeto do tipo *Config*. Neste momento, o objeto que irá atender à mensagem é da classe NãoAvaliado. Ele solicita a um objeto do tipo *EspaçoConfigs* que instancie um objeto FaltaP2PS do tipo *Config*. Ele passa o valor da nota *P1* para este objeto e o encaminha para que a *Topologia* conduza a troca de estado da configuração.

O comportamento deste modelo de objetos é similar para o caso dos dois outros tipos de eventos detectáveis na configuração implementada pelos objetos da classe NãoAvaliado: fezP2 e fpl. Mas, supondo que a configuração implementada pela classe FaltaP2PS tenha sido atingida, como se refina este modelo de objetos? Quais transformações sofrerá a máquina de estados no caso das transições adaptativas? Como desenhar o comportamento adaptativo?

B. Evolução da arquitetura

No modelo de classes, a arquitetura até agora proposta indica como deverá ser feito o refinamento. Para detalhar o modelo de implementação da configuração referente ao estado **FaltaP2PS**, por exemplo, deve-se (Fig. 18):

1. adicionar as classes de implementação P1P2MF e P1PSMF;
2. indicá-las como sendo do tipo *Config*;
3. adicionar os métodos fabricarP1P2MF e fabricarP1PSMF na fábrica EspaçoConfigs.
4. implementar os métodos adaptativos ma2P2, ma2PS e ma2fpl na classe FaltaP2PS.

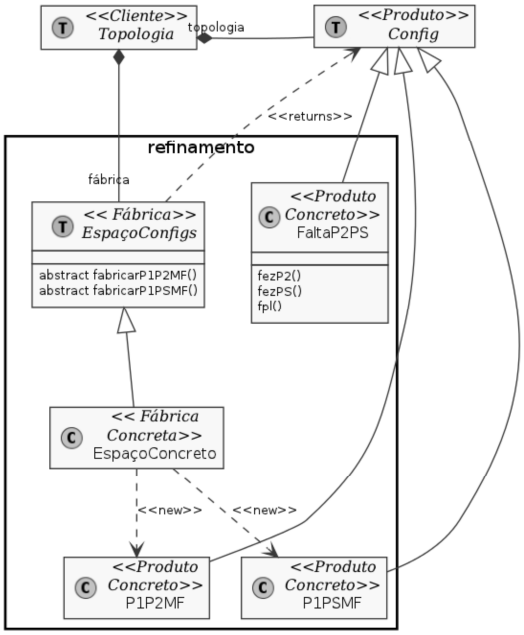


Figura 18. Refinamento do estado **FaltaP2PS** (DCL)

Um processo similar de refinamento ocorrerá para as demais classes que representam os estados da configuração adaptativa.

C. Modelo de Implementação com Factory Method

A versão da arquitetura contemplando o isolamento do mecanismo de instanciação é ilustrada no diagrama da Fig. 19. Observa-se que a arquitetura está se tornando gradualmente mais complexa no sentido de envolver cada vez mais elementos de implementação. Entretanto, o propósito fundamental das decisões de projeto é gerenciar a propagação das ondas de edição das instruções pelas classes de implementação.

A arquitetura ilustrada na Fig. 19 foi concebida de modo a organizar dois grupos de alterações: aquelas envolvidas nas instruções da classe TopAdapt e aquelas que se referem às descendentes de *Config*. As abstrações *Topologia* e *EspaçoConfigs* foram propostas exatamente para obter-se o seguinte efeito: (i) edições em um descendente de *Config* desencadeiam recompilações de TopAdapt e EspaçoConcreto; (ii) edições na classe TopAdapt não desencadeiam edições complementares de classes.

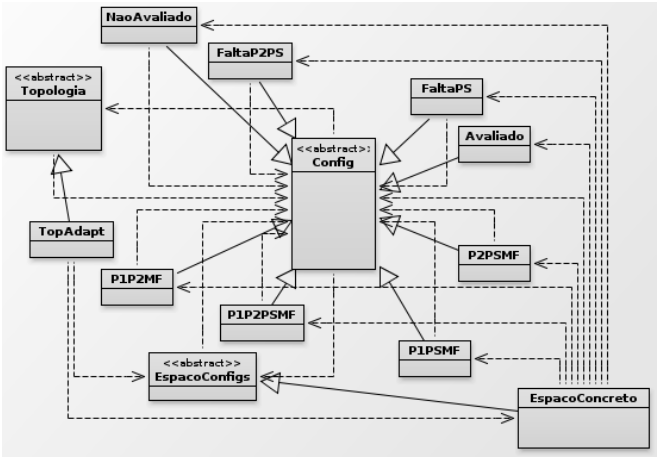


Figura 19. Isolamento de instanciação e desacoplamento (BlueJ)

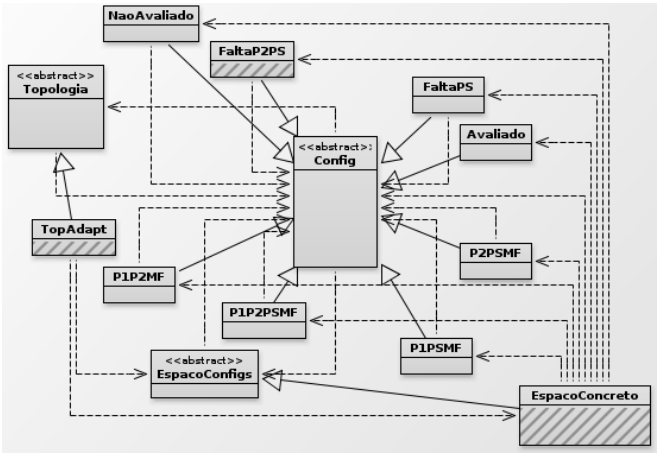


Figura 20. Alteração em FaltaP2PS (BlueJ)

O diagrama da Fig. 20 ilustra o padrão de hachuramento quando se edita a classe FaltaP2PS. Há revisão (ao menos de recompilação) apenas em outras duas classes arquiteturais.

O que acontece quando ocorre uma edição nas instruções da classe TopAdapt? Como as refatorações não afetaram as dependências em direção a esta classe, apenas ela será hachurada.

VII. CONCLUSÃO

Este trabalho de pesquisa teve foco no projeto de uma aplicação orientada a objetos, partindo de uma especificação adaptativa. A preocupação central lidou com a afirmação de Parnas a respeito da modularização e a sua relação com modelos de tempos de execução (modelos de computação). A questão básica da modularização considera fundamental o efeito dos critérios adotados para a decomposição em módulos.

Uma especificação adaptativa, proposta como resultado deste trabalho, suporta a organização do modelo comportamental de modo a indicar uma estratégia de modularização que enfatize as consequências de edição de código ao longo de uma arquitetura de software. Ou seja, classes de implementação são utilizadas para encapsular abstrações de funções adaptativas, originando métodos adaptativos. Desta forma, encapsulam-se as instruções que desencadeiam reconfigurações de topologias em classes de objetos, facilitando o entendimento do efeito da aplicação das funções adaptativas.

Especificações adaptativas, como introduzidas neste trabalho, podem ser sistematicamente produzidas a partir de tabelsaverdade, tabelas de decisão e tabelas de transição de estados. Tais técnicas consolidadas de programação, quando complementadas com métodos adaptativos (no sentido de representarem o efeito da aplicação de funções adaptativas), levam à elaboração de uma arquitetura de software que facilita a gerência das dependências estáticas.

O emprego de princípios de modelagem e de padrões de projeto também resultou em uma constatação: auxiliam na elaboração de uma arquitetura com dependências estáticas gerenciáveis. Ou seja, os princípios e padrões indicam pontos do modelo que são problemáticos quando se consideram as dependências estáticas.

O prosseguimento deste trabalho pode considerar as consequências de se desacoplar (no modelo de classses) a fábrica de métodos do espaço de configurações das edições da topologia de um autômato. Também pode-se investigar a geração automática da arquitetura de implementação referente aos métodos adaptativos. Isso automatizaria a geração de uma parte do código oriunda diretamentamente das tabelas de transição estendidas com abstrações de funções adaptativas.

APÊNDICE

Algumas das instruções do modelo de implementação do ensaio estão aqui apresentados. Ênfase foi dada para os serviços de instanciação dos objetos que representam as configurações do autômato e um detalhe de codificação do método adaptativo ma1P2.

EspacoConfigs — serviços de instanciação oferecidos pelos métodos de fabricação:

```
public abstract class EspacoConfigs {
  abstract Config fabricarNaoAvaliado();
  abstract Config fabricarFaltaP2PS();
  abstract Config fabricarFaltaPS();
  abstract Config fabricarAvaliado();
  abstract Config fabricarP1P2MF();
  abstract Config fabricarP1PSMF();
  abstract Config fabricarP2PSMF();
  abstract Config fabricarP1P2PSMF();
}
```

FaltaP2PS — implementação do método adaptativo ma1P2 e da ação semântica associados à transição de saída fezP2 do estado **FaltaP2PS**:

```
public class FaltaP2PS extends Config {
```

```
@Override
public void fezP2(Double p2) {
  // ma1P2
  Config proxima = fabrica.fabricarP1P2MF();
  configAdapt.trocarConfig(proxima);
  // semantic action
  proxima.definirP1(p1);
  proxima.definirP2(p2);
}
…
```


AGRADECIMENTOS

O autor agradece ao amigo e colega João J. Neto pelo apoio e incentivo para a realização deste trabalho. Suas ideias fundamentaram a proposta dos métodos adaptativos como uma técnica para refinar o módulo de controle de uma arquitetura de software. O autor também agradece aos dedicados membros do grupo de pesquisa GEMS-TIDD-PUCSP pelas valiosas discussões a respeito de técnicas e princípios para a modelagem com objetos. Finalmente, pelos importantes comentários e correções, o autor agradece aos anônimos revisores.

REFERÊNCIAS

[1] R. Pressman, *Software Engineering: A Practitioner's Approach*. McGraw-Hill Science/Engineering/Math, 7 ed., 2009. ISBN-13: 978-0073375977.

[2] P. Kruchten, "Architectural blueprints — the "4+1" view model of software architecture," *IEEE Software*, vol. 12, no. 6, pp. 42 – 50, 1995.

[3] "IEEE Standard glossary of software engineering termnology, std 610.12," 1990.

[4] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.

[5] B. P. Douglass, *Real-Time Design Patterns*. Object Technology Series, Reading, Massachusetts: Addison-Wesley, 2003.

[6] D. Harel, "Statecharts: A visual formalism for complex systems," Sci. Comput. Programming, pp. 231–274, 1987.

[7] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1988.

[8] V. Setzer, *A construção de um compilador*. Campus, 1983.

[9] D. Garlan and M. Shaw, "An introduction to software architecture," in *Advances in Software Engineering and Knowledge Engineering*, pp. 1–39, Publishing Company, 1993.

[10] R. C. Martin, *Designing Object-Oriented C++ Applications Using The Booch Method*. Prentice Hall, 1995. ISBN 0-13-203837-4.

[11] J. Lakos, *Large-Scale C++ Software Design*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1996. ISBN 0-201-63362-0.

[12] J. R. Rumbaugh, M. R. Blaha, W. Lorensen, F. Eddy, and W. Premerlani, *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ.: Prentice-Hall, Inc., 1991. ISBN-13 978-0136298410.

[13] M. Jackson, "Representing structure in a software system design," *The Journal of Design Studies*, special issue on Studying Professional Software Designers, vol. 31, November 2010.

[14] M. Jackson, "Aspects of system description," in *Programming Methodology* (A. McIver and C. Morgan, eds.), p. 137160, Springer Verlag, 2003.

[15] M. Jackson, "Where, exactly, is software development?," in *Formal Methods at the Crossroads: from Panacea to Foundational Support; 10th Anniversary Colloquium of UNU/IIST* (B. K. Aichernig and T. Maibaum, eds.), (Lisbon), International Institute for Software Technology of The United Nations University, Springer-Verlag, March 18-21, 2002 2003. LNCS 2757.

[16] T. Rowlett, *The object-oriented development process: developing and managing a robust process for object-oriented development*. Prentice Hall PTR, 2001.

[17] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Boston, MA, USA:

[18] Addison-Wesley Longman Publishing Co., Inc., 2006. [18] M. V. M. Ramos, J. J. Neto, and Í. S. Vega, *Linguagens Formais: Teoria, Modelagem e Implementação*. Porto Alegre: Bookman, 1 ed., 2009.

[19] M. Fowler and K. Scott, *UML Distilled*. Addison Wesley, 2 ed., 2000.

[20] J. J. Neto, "Adaptive rule-driven devices — general formulation and case study," in *International Conference on Implementation and Application of Automata* (Springer-Verlag, ed.), pp. 234–250, 2001.

[21] Í. S. Vega, "An adaptive automata operational semantic," *Latin America Transactions, IEEE*, vol. 6, no. 5, pp. 461 – 470, 2009. ISSN 1548-0992.

[22] J. J. Neto, J. R. de Almeida Jr., and J. M. N. dos Santos, "Synchronized statecharts for reactive systems." Technical Report, 1998.

[23] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. ISBN 0201633612.

[24] R. C. Martin, *UML for Java Programmers*. Prentice Hall, 2003. ISBN 978-0131428485.

[25] D. J. Barnes and M. Kölling, *Objects First with Java: A Practical Introduction Using BlueJ*. Prentice Hall / Pearson Education, 4th ed., 2008. ISBN 0-13-606086-2.

[26] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley, 1999. ISBN 0-201-48567-2.

Ítalo S. Vega recebeu os títulos de Mestre e Doutor em Engenharia de Software na Escola Politécnica da Universidade de São Paulo, Brasil, em 1993 e 1998, respectivamente. Professor Associado do Departamento de Computação da Pontifícia Universidade Católica de São Paulo e líder do Grupo de Estudos em Modelagem de Software (GEMS) do programa de pós-graduação em Tecnologia da Inteligência e Design Digital (TIDD).

Mineração Adaptativa de Dados: Aplicação à Identificação de Indivíduos

P. R. M. Cereda e J. J. Neto

***Resumo*— Este artigo apresenta uma proposta de modelo adaptativo para identificação de indivíduos voltada a grandes volumes de dados. Adicionalmente, apresenta-se também uma proposta inicial do conceito de mineração adaptativa de dados como possível solução computacional aplicável aos problemas oriundos de grandes volumes de dados.**

***Palavras-chave:*— Dispositivos Adaptativos, Identificação de Indivíduos, Mineração de Dados.**

I. Introdução

Com o advento e popularização dos serviços web e das redes sociais, o volume de dados disponíveis aumentou exponencialmente nos últimos anos. Estudos recentes estimam os totais de usuários ativos cadastrados, e os números impressionam: a rede de relacionamento *Facebook* lidera o *ranking* de cadastros, possuindo atualmente 500 milhões de usuários, seguida pelo serviço de e-mail *GMail*, com 190 milhões, e pela rede *MySpace*, com 126 milhões [1]. As informações de tais redes e serviços, quando analisadas e interpretadas, podem oferecer subsídios para um mapeamento mais preciso sobre perfis de consumo e padrões comportamentais de seus usuários. Entretanto, o maior desafio é como atuar sobre grandes volumes de dados em tempo hábil.

Técnicas tradicionais de mineração de dados para extração de conhecimento são amplamente utilizadas em grandes volumes de dados, porém muitas delas apresentam tempo de execução relativamente lento ou ainda, no pior caso, de ordem exponencial [2]. Bancos de dados de serviços web e de redes sociais já ultrapassam milhares de *terabytes*, e a utilização de técnicas tradicionais pode exaurir facilmente recursos computacionais, demandar tempo impraticável e inviabilizar o processo de extração de conhecimento.

A *adaptatividade* é o termo utilizado para denotar a capacidade de um dispositivo em modificar seu próprio comportamento, sem a interferência de agentes externos. Existem inúmeras aplicações para a adaptatividade em sistemas computacionais, incluindo robótica [3], tomadas de decisão e aprendizado de máquina [4], [5], processamento de linguagem natural [6], otimização de código em compiladores [7], controle de acesso [8], servidores web [9], linguagens de programação [10], [11], meta-modelagem [12], [13] e computação evolutiva [14], [15]. De acordo com Neto [16], a generalidade resultante do modelo, a capacidade de aprendizado devida à característica de auto-modificação, bem como o poder de expressão faz dos dispositivos adaptativos uma alternativa muito atraente para expressar fatos complexos e manipular situações difíceis que surgem durante uma busca de soluções computacionais para problemas complexos.

Este artigo apresenta uma proposta de modelo adaptativo para identificação de indivíduos em meio a grandes populações. O autômato adaptativo foi escolhido como dispositivo do modelo devido à sua simplicidade de abstração e poder computacional. É importante destacar que outros dispositivos adaptativos poderiam ser utilizados neste mesmo contexto. Além do modelo, apresenta-se também uma proposta inicial do conceito de mineração adaptativa de dados como possível solução computacional aplicável aos problemas oriundos de grandes volumes de dados.

A organização deste artigo é a seguinte: a Seção II contextualiza a identificação de indivíduos, cálculo e aplicações. A mineração adaptativa de dados é apresentada na Seção III, juntamente com a mineração de dados tradicional, exemplos e aplicações. A Seção IV apresenta um experimento realizado para analisar o impacto do modelo adaptativo e algumas possíveis vantagens da proposta deste artigo. As discussões sobre o modelo e a mineração adaptativa de dados são apresentadas na Seção V. A Seção VI apresenta as conclusões.

II. Identificação de Indivíduos

A *identificação de indivíduos*, no contexto deste artigo, é definida como um método aplicado a um conjunto de indivíduos, reduzindo-o a um subconjunto com características específicas comuns. Se o subconjunto resultante possuir apenas um elemento, a identificação é dita *única*. No caso de um subconjunto vazio, a identificação sob as características comuns não é aplicável.

Do ponto de vista comercial, a identificação de indivíduos oferece subsídios para a criação de segmentos de mercado. Através de tais segmentos, é possível estabelecer, heuristicamente, perfis de consumo e padrões comportamentais. No caso de uma identificação única, o grau de corretude das informações relacionadas disponíveis é muito maior, mas pode incorrer em eventuais violações de privacidade [17], [18].

A aquisição das características dos indivíduos de um conjunto pode ocorrer de dois modos, independentes ou não

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: cereda@users.sf.net e joao.jose@poli.usp.br.

entre si: o primeiro, *explícito*, é oriundo de interações diretas de um indivíduo com os mais diversos tipos de coletores de dados, incluindo formulários de empresas e bancos, currículos, cadastros de prestações de serviços, entrevistas, concursos públicos. Espera-se que, neste caso, o grau de corretude das informações coletadas seja muito elevado, pois a manipulação e alteração propositais de tais informações por parte do indivíduo pode constituir crime de falsidade ideológica [19].

O segundo modo trata de uma aquisição *implícita*, sem necessariamente o conhecimento e consentimento do indivíduo. A aquisição ocorre em diversos níveis de interação indireta e resulta em um fluxo enorme de informações [20], [9]. Os coletores de dados mais comuns na aquisição implícita atuam no escopo da internet e incluem diversos mecanismos e técnicas disponíveis, tais como *crawlers*, *spiders* e *web bugs* [21], [22], [23], [24], [25]. Com o crescimento e difusão do comércio eletrônico e da popularização de redes sociais, a aquisição implícita tornou-se fundamental na tentativa de identificação de indivíduos com o máximo de características disponíveis.

Além das interações em serviços web, os possíveis padrões comportamentais podem ser também monitorados ou rastreados através de serviços baseados em localização. Tais serviços utilizam milhares de informações de campo, como coordenadas geográficas, temperatura externa, sons e ruídos, para definir o conjunto de dados a ser processado [26] [27] [28] [29] [30]. A computação móvel ainda permite que dispositivos em sistemas ubíquos e pervasivos se comuniquem e troquem dados entre si, o que aumenta expressivamente o conjunto de dados disponível acerca de um indivíduo e de seu comportamento [31].

Outra forma de rastreamento ocorre através de *impressões digitais*, em hardware e software. Na área de hardware, existem diversos registros na literatura sobre dispositivos que possuem características e variações únicas [32], [33], [34], [35]. Essas informações favorecem uma eventual identificação do dispositivo e consequente rastreamento de seu possuidor.

Na área de software, o rastreamento pode ou não ocorrer de modo deliberado. Alguns aplicativos utilizam métodos para geração e envio de *hashes* ou qualquer outra informação desejável a um determinado servidor. Esse procedimento é amplamente utilizado para verificação e validação de chaves de registros. Em outros casos, um aplicativo pode fornecer informações de acordo com uma determinada solicitação. Por exemplo, um navegador web envia sua própria identificação e versão juntamente com dados do sistema operacional no cabeçalho HTTP [36]. Existem estudos que analisam a impressão digital de navegadores web através das informações contidas no cabeçalho HTTP juntamente com a lista de *plugins* e complementos disponibilizados [37], e seus resultados são impressionantes: um em aproximadamente 287 mil navegadores compartilhará a mesma impressão digital que outro navegador, e em 99,1% dos casos é possível identificar um navegador reincidente, mesmo que este tenha sido atualizado.

Técnicas de proteção de identidade e melhoria de privacidade foram propostas para tentar atenuar a exposição de dados sensíveis [38], [39], [40], [41]. Algumas delas utilizam-se do conceito de *anonimato de grupo*, que consiste na situação em que um indivíduo se torna anônimo através do grupo em que ele está inserido. Um dos exemplos mais intuitivos do anonimato de grupo é o batedor de carteiras, que após praticar o delito, corre em direção à multidão ao invés de lugares mais abertos; quanto maior a multidão, mais difícil será encontrá-lo. Seguindo esta lógica, seria razoável imaginar que, dada a população mundial – aproximadamente 7 bilhões de indivíduos em 2011 [42] – o anonimato de um indivíduo, no âmbito global, estaria preservado. Entretanto, estudos indicam que cada indivíduo é facilmente rastreável através de análises das informações coletadas [37], [43].

Uma das métricas utilizadas para a análise de identificação de um indivíduo é o *cálculo dos bits de informação*, obtido através da Fórmula 1 [43].

$$\Gamma = -\log_2 P(A) \quad (1)$$

De acordo com a Fórmula 1, o valor obtido representa a quantidade de informação necessária para se identificar um indivíduo. É calculado utilizando-se o logaritmo binário de uma probabilidade *P(A)* dada [43]. Como exemplo, considere a probabilidade de se encontrar um indivíduo em 7 bilhões – a população mundial – como sendo $P(A) = 1/7000000000$. Assim, o cálculo de Γ_1 é demonstrado na Fórmula 2 [43].

$$\Gamma_1 = -\log_2 \frac{1}{7000000000} \quad (2)$$

O resultado obtido na Fórmula 2 é de 32,7047 indicando que apenas 32,7 bits de informação são suficientes para se identificar univocamente um indivíduo entre a população mundial. É importante observar que 32,7 bits de informação representa o valor obtido no pior caso, pois quanto menor o grupo, menor é o número de bits de informação necessário para se identificar univocamente um indivíduo nesse grupo.

O cálculo do número de bits de informação permite não somente a identificação única de um indivíduo, mas pode indicar uma classificação em subconjuntos contendo indivíduos com características semelhantes [43]. A partir desses grupos, perfis de consumo e padrões comportamentais são extraídos através das mais diversas técnicas de mineração de dados.

III. Mineração Adaptativa de Dados

Mineração de dados é o nome associado ao conjunto de técnicas para aquisição de conhecimento relevante em grandes volumes de dados. Também conhecida como *data mining*, a mineração de dados constitui a segunda fase em um processo de *descoberta de conhecimento em bases de dados*, um ramo da computação que utiliza ferramentas e técnicas computacionais para sistematizar o processo de extração de conhecimento [44].

A primeira fase do processo de descoberta de conhecimento é chamada de *preparação de dados*, na qual os dados são pré-processados para a posterior submissão ao

processo de mineração. É composta das seguintes etapas, a saber:

a) *definição dos objetivos do problema*, que constitui o entendimento do domínio da aplicação e quais os objetivos a serem alcançados,
b) *criação do conjunto de dados*, na qual o conteúdo bruto de dados a serem analisados é agrupado e organizado,

filtragem e pré-processamento de dados, tendo como objetivo assegurar a qualidade dos dados selecionados; esta etapa é uma das mais onerosas de todo o processo – pode atingir até 80% do tempo estimado – principalmente devido às dificuldades de integração de conjuntos de dados heterogêneos [45],

c) *redução e projeção de dados*, que constitui a alocação e armazenamento adequados em bancos de dados para facilitar a aplicação das técnicas de mineração de dados.

A segunda fase consiste na *mineração de dados* propriamente dita sobre os dados preparados na fase anterior. É composta das seguintes etapas, a saber:

a) *escolha das técnicas de mineração*, na qual selecionam-se uma ou mais técnicas para atuar sobre o conjunto de dados de acordo com o problema definido; não existe uma técnica universal, portanto cada problema exige uma escolha de quais técnicas são mais adequadas,
b) *mineração*, que efetivamente aplica as técnicas no conjunto de dados,
c) *interpretação dos padrões de exploração*, que faz a análise da informação extraída em relação aos objetivos definidos,
d) *consolidação do conhecimento descoberto*, que consiste na filtragem das informações, eliminando possíveis padrões redundantes ou irrelevantes, gerando assim o conhecimento a partir da fase de mineração de dados.

As técnicas de mineração de dados apresentam características e objetivos distintos, de acordo com a descrição de cada problema em particular. As mais tradicionais incluem classificação, relacionamento entre variáveis, agrupamento, sumarização, modelo de dependência, regras de associação e análise de séries temporais [46], todas elas utilizando conceitos das áreas de aprendizado de máquina, reconhecimento de padrões e estatística. Propõe-se neste artigo a inserção e utilização dos conceitos da área de tecnologia adaptativa no processo de descoberta de conhecimento em bancos de dados, possibilitando que a mineração de dados se torne adaptativa.

A *mineração adaptativa de dados* é definida como sendo um conjunto de técnicas adaptativas para aquisição de conhecimento. Através da utilização de dispositivos adaptativos, espera-se reduzir o número de etapas do processo de descoberta de conhecimento em bancos de dados e o número de passos computacionais. Além disso, a mineração adaptativa de dados, através de modelos adaptativos, pode permitir a análise, mineração e interpretação incremental dos dados, independentes do término das etapas de coleta, pré-processamento e verificação.

A tecnologia adaptativa tem obtido resultados expressivos na resolução de problemas referentes à extração de perfis de consumo e padrões comportamentais. Estudos recentes demonstraram que um dispositivo adaptativo pode, por exemplo, substituir com sucesso uma técnica de extração baseada em regras de associação muito utilizada no processo de mineração de dados [9]. As vantagens da utilização de um dispositivo adaptativo sobre as técnicas tradicionais incluem:

a) *tempo de execução linear*, mesmo para grandes conjuntos de dados,
b) *simplicidade formal e computacional do modelo*,
c) *capacidade de auto-modificação*, sem a necessidade de interferência externa de um especialista de domínio,
d) *modelo de dados disponível de modo incremental*,
e) *consistência contínua*, não havendo a necessidade de filtragem do conhecimento gerado [9].

O autômato adaptativo [47], [48] foi escolhido como dispositivo adaptativo a ser utilizado para a definição de um modelo de extração de conhecimento e um estudo de caso. Suas características permitem um modelo consistente, simplificado e poderoso computacionalmente [16], além de oferecer subsídios para uma implementação direta a partir da definição proposta.

O modelo apresentado a seguir tem como objetivo a extração de conhecimento a partir de um conjunto de dados disponível, na tentativa de identificação de indivíduos a partir de suas características. Inicialmente, o autômato adaptativo *M* é definido com uma configuração que reflete o conjunto de dados disponível (Figura 1). Durante seu tempo de vida, *M* sofre alterações em sua topologia para extrair conhecimento e identificar indivíduos. O caso aqui apresentado é um exemplo ilustrativo e pode ser adequado para ser utilizado em outras aplicações.

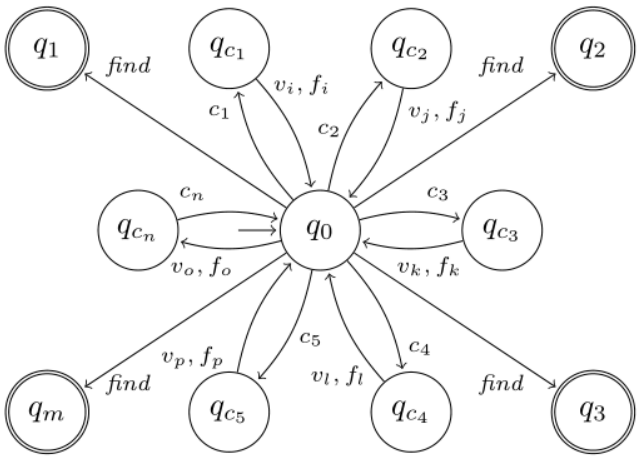


Figura 1. Autômato adaptativo *M* do modelo.

O autômato adaptativo *M* recebe cadeias de entrada que representam as consultas sobre o conjunto de dados. O tipo de consulta a ser submetida ao modelo parte dos objetivos da extração e da definição do problema.

Considere os conjuntos $C = \{c_1, c_2, \ldots, c_n\}$ de características e $V = \{v_1, v_2, \ldots, v_i\}$ de valores associados às características. O autômato adaptativo *M* da Figura 1 possui o estado inicial q_0, *n* estados intermediários, onde $n \in \mathbb{N}$ é o total de características existentes no conjunto de dados, e *m* estados finais, $m \in \mathbb{N}$ é o total de indivíduos disponíveis. Os estados intermediários q_i são alcançáveis a partir de q_0 através de uma transição consumindo a característica $c_1 \in C$, além de haver uma transição de volta consumindo um determinado valor $v_j \in V$ associado à característica c_i e executando uma determinada função adaptativa f_k correspondente. Finalmente, o autômato possui *m* transições consumindo o símbolo especial <*find*> partindo de q_0 até os estados finais. O alfabeto utilizado é definido como $\Sigma = C \cup V \cup \{<find>\}$. A cadeia *w* submetida ao autômato deve seguir o formato $w = (cv) * <find>$, $c \in C$, $v \in V$, ou seja, procuram-se indivíduos que contêm os pares <*característica*>=<*valor*> seguido do símbolo especial <*find*>, que efetivamente realiza a busca no autômato.

O autômato adaptativo *M* do modelo é definido de forma não-determinística, a princípio. Semanticamente, o indeterminismo denota os indivíduos candidatos à escolha. No início do processo de reconhecimento da cadeia *w*, todos os indivíduos têm a mesma probabilidade de escolha, portanto todos os estados finais que os representam são alcançáveis a partir do estado inicial q_0 consumindo o símbolo especial <*find*>.

Do ponto de vista de extração de conhecimento a partir do conjunto de dados disponível, o quão indeterminístico o autômato adaptativo for após a submissão e reconhecimento da cadeia de entrada tem relação direta com a entropia do conjunto de indivíduos disponível. Ao longo do processo de reconhecimento da cadeia, espera-se que o indeterminismo seja gradativamente reduzido, através da filtragem dos indivíduos candidatos à escolha. Através do modelo adaptativo, a busca por características e a eventual identificação de indivíduos ocorre em tempo real. O modelo é compacto e simplificado, o que favorece a visualização e interpretação dos resultados obtidos.

A topologia do autômato adaptativo *M* ao final do reconhecimento da cadeia de entrada indica semanticamente qual foi o conhecimento extraído. Se a cadeia não foi aceita – não há estados finais alcançáveis, independente da situação de indeterminismo ou não do autômato – não foi possível identificar indivíduos de acordo com os parâmetros definidos na cadeia. Caso *M* seja determinístico e a cadeia foi aceita, o estado final $q_i \in F$ alcançável indica que o indivíduo *i* foi univocamente identificado. Quando *M* é indeterminístico e a cadeia foi aceita, todos os estados $\{q_i, q_j, \ldots, q_k\} \subseteq F$ alcançáveis representam um grupo de indivíduos com as características procuradas.

O indeterminismo permite a visualização e definição de grupos de interesse na extração de conhecimento de modo incremental. Não é necessário filtragem e pós-processamento dos resultados, pois a topologia resultante do autômato garante a consistência relacional dos dados obtidos. As funções adaptativas do modelo realizam "cortes" no autômato, removendo estados e transições que não aderem aos objetivos definidos na cadeia de entrada para extração de conhecimento. A adaptatividade permite a redução e simplificação do conjunto de dados até o nível desejado.

Como exemplo, considere o conjunto de dados da Tabela 1. Cada um dos cinco indivíduos tem suas quatro características mapeadas. O autômato adaptativo correspondente é ilustrado na Figura 2. Observe que a coluna *nome*, por tratar-se da identidade de cada indivíduo, é associada aos estados finais.

Tabela I. Exemplo de conjunto de dados para ilustração do modelo adaptativo.

id	nome	sexo	olhos	cabelo	estatura
1	Ana	feminino	azuis	castanho	alta
2	José	masculino	azuis	preto	alta
3	João	masculino	castanhos	loiro	mediana
4	Maria	feminino	verdes	loiro	baixa
5	Pedro	masculino	castanhos	ruivo	baixa

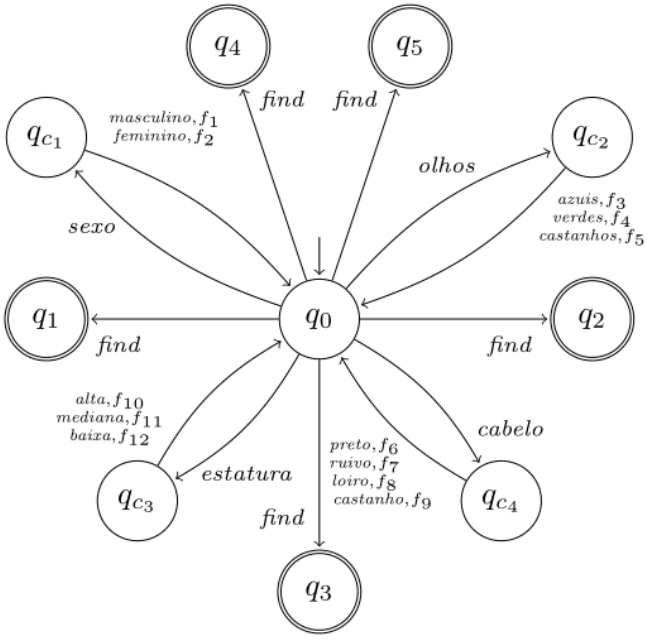


Figura 2. Autômato *M* representando os dados da Tabela I.

Deseja-se, por exemplo, identificar todos os indivíduos que possuem olhos verdes. Para tal, a cadeia de entrada $w_1 = <olhos><verdes><find>$ é submetida ao autômato adaptativo *M*. Como w_1 foi aceita por *M*, o conjunto de dados contém indivíduos que possuem as características informadas. O autômato resultante é ilustrado na Figura Figura 3.

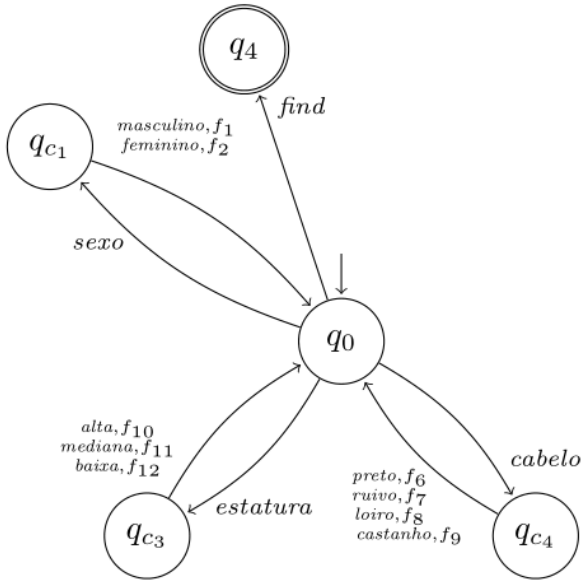


Figura 3. Autômato adaptativo *M* resultante após a submissão da cadeia $w_1 =$ <olhos><verdes><find>.

De acordo com a Figura 3, o autômato adaptativo *M* resultante é determinístico, significando que um único indivíduo foi encontrado. O estado final alcançável é q_4, associado a *Maria*, o único indivíduo da Tabela I que possui olhos verdes.

Em outro exemplo, deseja-se identificar todos os indivíduos que possuem cabelo loiro. Para tal, a cadeia de entrada $w_2 =$ <cabelo><loiro><find> é submetida ao autômato adaptativo *M*. O autômato resultante é ilustrado na Figura 4.

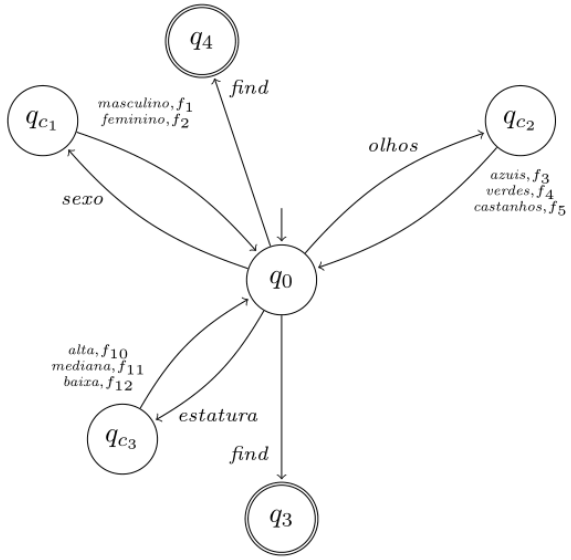


Figura 4. Autômato adaptativo *M* resultante após a submissão da cadeia $w_2 =$ <cabelo><loiro><find>.

De acordo com a Figura 4, o autômato adaptativo *M* resultante é indeterminístico, significando que um grupo de indivíduos com as mesmas características foi encontrado. Os estados finais alcançáveis são q_3 e q_4, associados a *João* e *Maria*. Caso a intenção seja obter apenas as mulheres com cabelo loiro, acrescenta-se um par <*característica*>=<*valor*> à cadeia, de modo que $w_3 =$ <cabelo><loiro><sexo><feminino><find>. Neste caso, somente o indivíduo *Maria* é encontrado.

A mineração adaptativa de dados pode favorecer o processo de descoberta de conhecimento em bancos de dados através de modelos adaptativos simplificados e consistentes. Como se pode intuir a partir da ilustração apresentada, a utilização da área de tecnologia adaptativa permite a extração incremental de conhecimento, de forma eficiente.

IV. Experimento e Análise

A mineração adaptativa de dados apresenta-se como uma alternativa atraente à mineração de dados tradicional. É importante destacar que mineração adaptativa e a tradicional não são mutuamente exclusivas, ao contrário, são completamente compatíveis uma com a outra, podendo por isso ser usadas simultaneamente, de acordo com a conveniência em cada caso. Esta seção descreve um experimento proposto para coletar dados estatísticos acerca do desempenho do modelo adaptativo e da consistência desses dados.

A primeira parte do experimento consistiu na definição e implementação de um simulador para a identificação de indivíduos através do modelo adaptativo ilustrado anteriormente. O simulador possui as seguintes funcionalidades:

a) geração, leitura e mapeamento de dados, em memória ou disco rígido, para automatizar o gerenciamento e processamento do conhecimento a ser extraído,
b) definição do modelo de autômato adaptativo correspondente ao problema, de acordo com o conjunto de dados disponível ao simulador,
c) terminal de consulta na forma de cadeias de entrada de tamanho arbitrário submetidas ao autômato adaptativo, no formato requerido apresentado na Seção III,
d) geração de *snapshots* da configuração do autômato para cada símbolo consumido, representados através de arquivos no formatos `dot`[1] e `png`[2],
e) geração de gráficos de análise dos dados, baseado no reconhecimento das cadeias de entrada, no tempo de execução e no cálculo dos bits de informação.

O simulador foi escrito utilizando-se a linguagem Python e executado através de linha de comando em um ambiente Linux de 64 bits.

As amostras utilizadas neste experimento foram geradas através do simulador. Inicialmente, foram gerados milhares de conjuntos de dados aleatórios, de tamanhos arbitrários, com registros no formato apresentado na Tabela II. Todas as amostras geradas possuem características estatísticas controladas e bem definidas.

[1] Linguagem de descrição de grafos em texto puro.
[2] Formato de dados utilizado para imagens.

Tabela II. Formato de registro para geração dos conjuntos de dados.

id	sexo	olhos	cabelos	estatura	pele	estado civil	idade
-	masc	verdes	cast.	baixa	clara	solt.	criança
-	fem	azuis	preto	med.	morena	cas.	jovem
-	-	cast.	ruivo	alta	negra	-	adulto
-	-	-	loiro	-	-	-	idoso

A partir dos conjuntos gerados, foram escolhidos três deles, com os seguintes tamanhos: $n(C_1) = 10$, $n(C_2) = 100$ e $n(C_3) = 1000$ indivíduos. Cada conjunto escolhido representou uma determinada população a ser analisada. Para cada indivíduo, foram geradas sete características e um identificador único para verificação e validação dos dados, de acordo com a Tabela II.

Na segunda parte do experimento, após a definição dos três conjuntos de dados, foram realizadas medições de tempo de execução e consistência dos dados durante as consultas aos autômatos, de acordo com as tarefas definidas na Tabela III.

Tabela III. Tarefas principais do experimento.

ID	Descrição
t_1	Tempo de execução das consultas sob os conjuntos C_1, C_2 e C_3
t_2	Consultas para identificação única de indivíduos
t_3	Consultas para identificação de grupos de indivíduos

De modo complementar, outras medições relacionadas a alguns métodos tradicionais de extração de conhecimento foram realizadas, de acordo com as tarefas definidas na Tabela IV.

Tabela IV. Tarefas complementares do experimento.

ID	Descrição
t_4	Produto cartesiano das características, calculada a taxa dos bits de informação comparativamente ao valor limite dos bits de informação do conjunto de dados
t_5	Maior valor para cada combinação das características, de modo comparativo ao valor limite dos bits de informação
t_6	Cálculo dos bits de informação dos conjuntos C_1, C_2 e C_3

A Figura 5 ilustra o tempo de execução das consultas sob os conjuntos C_1, C_2 e C_3. O simulador gerou mil consultas de tamanho arbitrário, variando entre um e quinze símbolos, que foram submetidas na forma de cadeias de entrada, de acordo com o formato da cadeia de entrada apresentado na Seção III, aos três autômatos correspondentes. O eixos *x* e *y* representam o número de pares <*característica*>=<*valor*> consultados na cadeia e o tempo de execução em segundos, respectivamente.

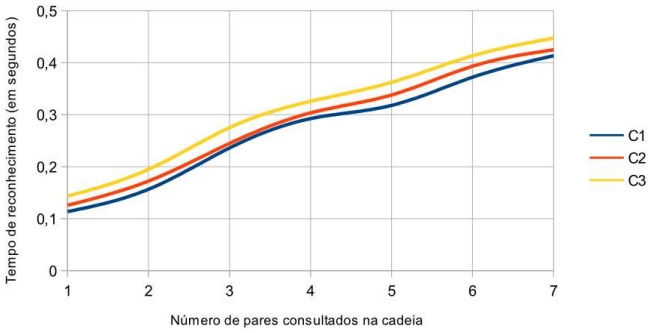


Figura 5. Tempo de execução das consultas sob os conjuntos C_1, C_2 e C_3.

De acordo com a Figura 5, o tempo de reconhecimento das cadeias de entrada para os três conjuntos analisados mostrou-se linear, proporcional ao comprimento da cadeia. É possível observar que não houve uma variação significativa entre os tempos dos três conjuntos, apesar da diferença entre o total de elementos de cada conjunto.

Utilizando as consultas geradas no teste anterior, foram verificados ainda os tempos de execução de consultas que resultavam em identificação única – apenas um estado final alcançável, gerando portanto um autômato determinístico – e grupo de indivíduos – dois ou mais estados finais alcançáveis, preservando o indeterminismo do autômato. Os resultados são ilustrados na Figura 6. É importante destacar que, para este teste, os resultados foram normalizados para que não houvesse discrepância entre os tamanhos das consultas e os tempos de execução.

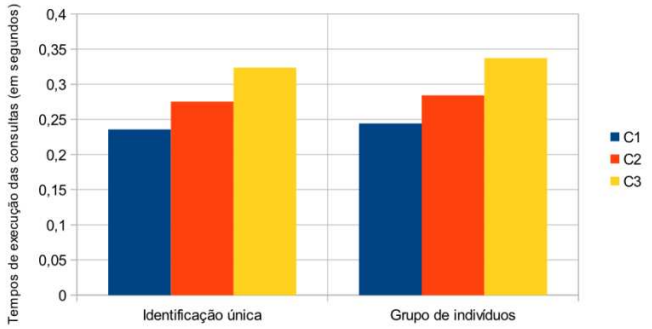


Figura 6. Tempos de execução das consultas para identificação única e de grupo.

De acordo com a Figura 6, é possível observar que os autômatos apresentaram um tempo semelhante para o reconhecimento das cadeias de entrada, não fazendo distinção do tipo de identificação retornada – única ou grupo. Nota-se que o modelo proposto aparentemente não gera sobrecarga de recursos computacionais ao retornar um grupo arbitrário de indivíduos – o tempo é praticamente o mesmo, independente do resultado retornado.

A Figura 7 ilustra o produto cartesiano das características presentes em cada um dos conjuntos e seus respectivos bits de informação. Alguns dos valores nulos foram omitidos para facilitar a visualização, uma vez que não interferem na interpretação dos resultados.

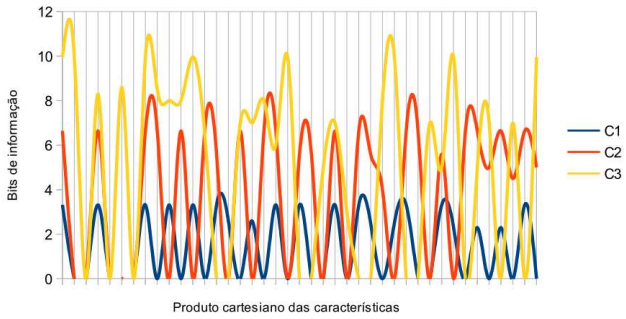


Figura 7. Produto cartesiano das características presentes em cada conjunto e seus respectivos bits de informação.

De acordo com a Figura 7, os conjuntos C_1 e C_2 apresentam mais picos do que o conjunto C_3. Dado o tamanho do conjunto C_3 e os registros no formato definido na Tabela II, este possui maior grau de entropia do que os outros conjuntos, portanto torna-se mais difícil a identificação única de um indivíduo. O grau de entropia aumenta ou diminui de acordo com o tamanho da amostra. Pode-se inclusive falar em *taxa de redução de indeterminismo*, inversamente proporcional ao grau de entropia do conjunto. É importante observar que o valor máximo dos bits de informação é diferente para cada conjunto.

A Figura 8 ilustra os maiores valores de bits de informação encontrados para cada combinação de pares <*característica*>=<*valor*> de cada conjunto, comparados ao valor limite dos bits de informação.

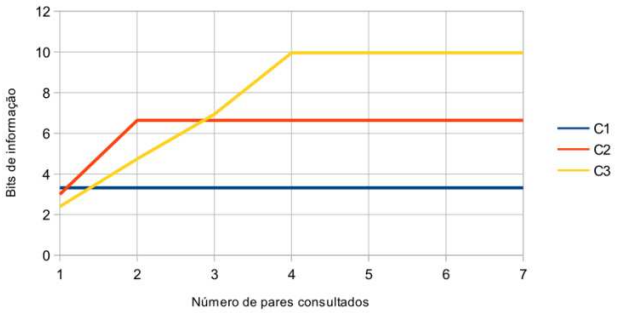


Figura 8. Maiores valores para cada combinação das características de cada conjunto comparados ao valor limite dos bits de informação.

De acordo com a Figura 8, é possível verificar a influência do tamanho da amostra para a identificação única de indivíduos. O conjunto C_1, por possuir apenas 10 indivíduos, permite a identificação única de um indivíduo com apenas um par <*característica*>=<*valor*>; o conjunto C_3 contendo 1000 indivíduos, no entanto, requer um refinamento maior na cadeia de entrada para a identificação única. No modelo adaptativo proposto, a relação de entropia e indeterminismo está associada ao tamanho do volume de dados disponível e a quantidade de características.

Como teste adicional, o simulador gerou um conjunto especial contendo 1000 indivíduos com 32 características cada. Durante a reprodução dos testes anteriores, o autômato adaptativo do modelo proposto ainda apresentou sua média de execução abaixo de 0,5 segundos, mesmo para cadeias de entrada com comprimento expressivo – por exemplo, 65 símbolos. Não foi possível reproduzir o mesmo teste adicional utilizando uma técnica tradicional de extração de conhecimento devido à explosão combinatória do algoritmo, gerando aproximadamente $1{,}83 \times 10^{19}$ combinações, exaurindo os recursos computacionais, demandando tempo impraticável e inviabilizando o seu uso. Outras técnicas tradicionais não foram aplicadas por serem genéricas demais para o problema em questão.

V. Discussões

Esta seção trata das discussões acerca do modelo adaptativo para identificação de indivíduos e dos conceitos de mineração adaptativa de dados. A seguir, são apresentados alguns questionamentos.

Os resultados obtidos sobre o tempo de execução do modelo adaptativo para extração de conhecimento, proposto neste artigo, apresentam-se favoráveis, dado que uma parcela significativa das técnicas tradicionais de mineração de dados apresenta tempo exponencial para extração e análise dos dados, o que onera o processo de mineração e inviabiliza sua utilização para volumes de dados com tamanho considerável. Entretanto, é importante notar que seriam necessárias mais amostras de conjuntos de dados de tamanho arbitrário e um comparativo entre as técnicas de mineração de dados existentes para atestar de modo definitivo a eficiência do modelo adaptativo.

Foi possível verificar, através do experimento descrito na Seção IV, que o reconhecimento em tempo quase linear da cadeia de entrada submetida ao autômato adaptativo do modelo permite uma visualização incremental contínua do processo de extração. Além disso, dado o rigor da definição formal do autômato adaptativo, os dados e a topologia que os representa apresentam uma consistência contínua, mantida durante todo o processo até o consumo do último símbolo da cadeia. A consistência de tais dados permite uma homogeneidade do modelo. Na mineração de dados tradicional, é necessário aguardar o término das fases descritas na Seção III para efetivamente analisar quaisquer dados obtidos e extrair conhecimento consistente, eventualmente através de um especialista de domínio.

Embora os testes de tempo de execução demonstrem uma eficiência considerável do modelo adaptativo, é importante destacar que este foi definido para solucionar um problema específico – a identificação de indivíduos. A mineração adaptativa de dados pode requerer um modelo para cada problema, ao contrário da mineração de dados tradicional, na qual seus modelos e técnicas são mais abrangentes. Além disso, a modelagem adaptativa pode tornar-se complexa comparativamente aos parâmetros e variáveis das técnicas tradicionais.

O modelo adaptativo para identificação de indivíduos é genérico o suficiente para ser utilizado em diversas aplicações, tais como motor de estratégias para jogos de tabuleiro, seleção de candidatos, busca de currículos, formação de equipes esportivas, otimização de problemas de logística, escolha de estratégias comerciais, *marketing* direcionado, análise de perfil populacional, otimização de campanhas de vacinação,

distribuição de recursos, identificação de criminosos, mapas territoriais, identificação de plágios em trabalhos acadêmicos e análises ambientais. Além das aplicações diretas, o modelo adaptativo oferece subsídios para a definição de um banco de dados adaptativo, otimizado para grandes consultas e para a crescente área de computação em nuvem.

VI. Conclusões

Este artigo apresentou uma proposta de modelo adaptativo para identificação de indivíduos, juntamente com o conceito de mineração adaptativa de dados. O modelo adaptativo mostrou-se viável para a utilização em grandes volumes de dados, permitindo a identificação única de indivíduos ou retornando grupos de interesse, sendo sua aplicação extensível para as mais diversas áreas. O tipo de tratamento utilizado para a identificação de indivíduos pode servir como elo de aproximação entre métodos estatísticos e métodos discretos, e que o seu estudo poderá trazer luzes sobre a utilização de métodos adaptativos – que são discretos – no estudo de fenômenos contínuos. Isso também ajudaria a facilitar o uso de tratamentos discretos para casos usualmente tratados estatisticamente, como acontece em algumas situações no caso do processamento de linguagem natural.

O conceito de mineração adaptativa de dados, introduzido neste artigo, pode tornar-se uma alternativa viável aos métodos existentes para extração de conhecimento. A crescente área de computação em nuvem pode beneficiar-se de técnicas e modelos adaptativos para oferecer soluções de baixo custo, alto poder computacional e suporte a grandes volumes de dados. Além disso, a idéia de um banco de dados adaptativo é desafiadora, podendo gerar contribuições para áreas correlatas, como otimização, recuperação da informação e mineração de dados.

A área de tecnologia adaptativa pode proporcionar soluções consistentes que atendam às necessidades inerentes a cada aplicação. A utilização de dispositivos adaptativos para a definição de técnicas de extração de conhecimento pode beneficiar não somente o processo de extração em si, mas permitir a obtenção de outros dados que não estão disponíveis através dos métodos tradicionais.

Referências

[1] Jess3, "The social universe," Jess3 Data Visualization Agency, Tech. Rep., 2010.

[2] H. Blockeel and M. Sebag, "Scalability and efficiency in multi-relational data mining," *SIGKDD Explorations*, vol. 5, pp. 17–30, 2003.

[3] M. A. A. Sousa and A. H. Hirakawa, "Robotic mapping and navigation in unknown environments using adaptive automata," in *Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005*, Coimbra, Portugal, March 21–23 2005.

[4] H. Pistori and J. J. Neto, "Adaptree – proposta de um algoritmo para indução de Árvores de decisão baseado em técnicas adaptativas," in *Anais da Conferência Latinoamericana de Informática – CLEI 2002*, Montevideo, Uruguai, Novembro 2002.

[5] T. Pedrazzi, A. H. Tchemra, and R. L. A. Rocha, "Adaptive decision tables – a case study of their application to decision-taking problems," in *Proceedings of International Conference on Adaptive and Natural Computer Algorithms – ICANNGA 2005*, 2005.

[6] C. Menezes and J. J. Neto, "Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos," in *V PROPOR – Encontro para o processamento computacional de Português falado e escrito*, 2000.

[7] J. Luz and J. J. Neto, "Tecnologia adaptativa aplicada à otimização de código em compiladores," in *IX Congreso Argentino de Ciencias de La Computación – CACIC 2003*, 2003.

[8] P. R. M. Cereda, "Modelo de controle de acesso adaptativo," Dissertação de Mestrado, Departamento de Computação, Universidade Federal de São Carlos, 2008.

[9] ____, "Servidor web adaptativo," in *WTA 2010: Workshop de Tecnologia Adaptativa*, Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2010.

[10] R. L. A. Rocha and J. J. Neto, "Uma proposta de linguagem de programação funcional com características adaptativas," in *IX Congreso Argentino de Ciencias de la Computación*, 2003.

[11] A. V. Freitas and J. J. Neto, "Adaptive languages and a new programming style," in *6th WSEAS International Conference on Applied Computer Science*, 2006.

[12] A. R. Camolesi and J. J. Neto, "Modelagem adaptativa de aplicações complexas," in *Conferência Latinoamericana de Informática – CLEI*, 2004.

[13] A. R. Camolesi, "Proposta de um gerador de ambientes para modelagem de aplicações usando tecnologia adaptativa," Ph.D. dissertation, Escola Politécnica da USP, 2007.

[14] H. Pistori, J. J. Neto, and M. C. Pereira, "Adaptive non-deterministic decision trees: general formulation and case study," *INFOCOMP – Journal of Computer Science*, 2006.

[15] C. Bravo, J. J. Neto, F. S. Santana, and A. M. Saraiva, "Towards an adaptive implementation of genetic algorithms," in *Conferência Latinoamericana de Informática – CLEI*, 2007.

[16] J. J. Neto, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa," *IEEE Latin America Transactions*, vol. 5, no. 7, pp. 496–505, 2007.

[17] W. Aiello and P. McDaniel, *Lecture 1, Intro: Privacy*, Stern School of Business, NYU, 2004.

[18] Electronic Frontier Foundation, "Annual report 2009–2010," Electronic Frontier Foundation, Tech. Rep., 2010.

[19] Brasil, *Código Penal*, Brasília, Distrito Federal, 1940.

[20] M. Teltzrow and A. Kobsa, "Communication of privacy and personalization in e-business," in *Workshop Wholes: A multiple view of individual privacy in a networked world*, Stockholm, Sweden, 2004.

[21] S. Azambuja, "Estudo e implementação da análise de agrupamento em ambientes virtuais de aprendizagem," Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 2005.

[22] M. Koch and K. Moeslein, "User representation in ecommerce and collaboration applications," in *BLED 2003 Proceedings*, 2003.

[23] D. Kristol and L. Montulli, "HTTP state management mechanism," Bell Laboratories, Lucent Technologies, RFC 2965, 2000.

[24] A. L. Montgomery, "Using clickstream data to predict www usage," University of Maryland, Tech. Rep., 2003.

[25] S. Sae-Tang and V. Esichaikul, "Web personalization techniques for e-commerce," in *Active Media Technology: 6th International Computer Science Conference*, 2005.

[26] D. L. Lee, M. Zhu, and H. Hu, "When location-based services meet databases," *Mob. Inf. Syst.*, vol. 1, pp. 81–90, 2005.

[27] J. Raper, G. Gartner, H. Karimi, and C. Rizos, "Applications of location-based services: a selected review," *J. Locat. Based Serv.*, vol. 1, pp. 89–111, 2007.

[28] J. Paay and J. Kjeldskov, "Understanding the user experience of location-based services: five principles of perceptual organisation applied," *J. Locat. Based Serv.*, vol. 2, pp. 267–286, 2008.

[29] K. Wac and L. Ragia, "Lspenv: location-based service provider for environmental data," *J. Locat. Based Serv.*, vol. 2, pp. 287–302, 2008.

[30] J. Raper, G. Gartner, H. Karimi, and C. Rizos, "A critical evaluation of location based services and their potential," *J. Locat. Based Serv.*, vol. 1, pp. 5–45, 2007.

[31] G. Ghinita, "Private queries and trajectory anonymization: a dual perspective on location privacy," *Trans. Data Privacy*, vol. 2, pp. 3–19, 2009.

[32] J. Lukás, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, pp. 205–214, 2006.

[33] E. Y. L. Kai San Choi and K. K. Wong, "Source camera identification using footprints from lens aberration," *SPIE-IS&T Electronic Imaging*, vol. 6069, 2006.

[34] O. Hilton, "The complexities of identifying the modern typewriter," *Journal of Foresinc Sciences*, vol. 2, 1972.

[35] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, 2005.
[36] IETF, "Hypertext Transfer Protocol – HTTP/1.0," Internet Engineering Task Force, Tech. Rep., 1996.
[37] P. Eckersley, "How unique is your web browser?" Electronic Frontier Foundation, Tech. Rep., 2009.
[38] M. Ackerman and L. F. Cranor, "Privacy critics – safeguarding users' personal data," 1999.
[39] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.
[40] I. Goldberg, D. Wagner, and E. Brewer, "Privacy-enhancing technologies for the internet," in *IEEE Spring Computer Conference*. San Jose, CA, USA: IEEE Computer Society Press, February 1997, pp. 103–110.
[41] D. Goldschlag, M. Reed, and P. Syverson, *Anonymous Connections and Onion Routing*, Assurance Computer Systems, Naval Research Laboratory, Washington, DC, 1996.
[42] U.S. Census Bureau, "Total population of the world," U.S. Census Bureau, Tech. Rep., 2011.
[43] P. Eckersley, "A primer on information theory and privacy," Electronic Frontier Foundation, Tech. Rep., 2010.
[44] H. Jiawei and M. Kamber, *Data Mining: Concept and Techniques*. Morgan Kaufmann, 2006.
[45] H. Mannila, "Data mining: machine learning, statistics and databases," *International Conference on Statistics and Scientific Database Management*, 1996.
[46] U. Fayyad, G. Piatetsky-shapiro, P. Smyth, and T. Widener, "The KDD process for extracting useful knowledge from volumes of data," *Communications of the ACM*, vol. 39, pp. 27–34, 1996.
[47] J. J. Neto, "Contribuições à metodologia de construção de compiladores," Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.
[48] ____, "Adaptive automata for context-dependent languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, 1994.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005) e mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008). Atualmente, é desenvolvedor de software, atuando em diversos segmentos de mercado, com enfoque principal em software livre. É membro ativo do repositório de código fonte *SourceForge* desde 2005, contribuindo com bibliotecas e softwares de propósito geral. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: inteligência artificial, linguagens de programação, teoria da computação e tecnologia 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.

Confidence in Decision-Making Through Adaptive Devices

R. S. Okada and J. J. Neto

***Abstract*— This paper presents a confidence-oriented model for decision-making devices, capable of acquiring new information at runtime. Typically, machine-learning strategies consist of building an abstraction – such as decision trees - out of a training data before putting it into use, assuming that the data is accurate. Learning at runtime, however, not only requires handling uncertain data, but it also needs a flexible device to accept new information dynamically. In this paper, we introduce an environment for decision-making, which, along with adaptive methods, allows the handling of dynamic learning under uncertainty. It also introduces some of the state-of-art methods for dynamic learning.**

***Keywords*— Decision-Making, Adaptive Device, Case-Based Reasoning, Confidence Interval, k-Nearest Neighbor, Instance-Based Learning.**

I. INTRODUÇÃO

Tomada de decisão pode ser compreendido como um processo cognitivo em que uma ação deve ser escolhida e tomada como resposta a um estímulo externo – este, formado por um conjunto de pares atributo-valor que representam um evento [1]. Escolher uma ação, então, torna-se uma questão de comparar as alternativas possíveis e escolher aquela que melhor se adequa ao problema.

Tipicamente, modelos computacionais de tomada de decisão tentam imitar a forma com que humanos realizam esta mesma tarefa. Contudo, sabe-se que este processo não é perfeitamente racional [2], sofrendo influência de fatores emocionais [3] e experiências passadas [4], comumente ignoradas pelos modelos de decisão ou de aprendizado de máquina.

Fatores temporais também se mostram influentes, uma vez que as pessoas, de modo geral, direcionam sua atenção para eventos mais recentes, frequentemente ignorando ou esquecendo aprendizados obtidos em um tempo distante. O foco em eventos recentes nos permite uma rápida adaptação às mudanças comportamentais no processo a ser prognosticado [5] – a este comportamento, dá-se o nome *concept drift*.

Dispositivos de estrutura rígida, contudo, só conseguem alterar sua estrutura reconstruindo-a novamente, ação de difícil realização em tempo de execução. Por não se adaptarem a mudanças externas, suas taxas de acertos se degradam ao longo do tempo, fazendo-se necessário reconstruí-los para evitar perda de desempenho.

Este trabalho visa à criação de um ambiente para tomada de decisão adaptativa, capaz de adquirir novos conhecimentos ao longo de sua execução, sendo fundamentado no raciocínio baseado em casos e em um sistema de realimentações de resultados. Este ambiente deve ser reutilizável, permitindo o uso de diferentes algoritmos de tomada de decisão – desde que tenham comportamento adaptativo.

R. S. Okada, Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, rsuzuki.okada@gmail.com

J. J. Neto, Escola Politécnica, Universidade de São Paulo, São Paulo, Brasil, jjneto@gmail.com

II. CONCEITOS

A. Tecnologia Adaptativa:

O termo *adaptatividade* é empregado para descrever dispositivos capazes de modificar seu próprio comportamento como resposta aos dados de entrada, sem a interferência de agentes externos [6]. De forma geral, um dispositivo adaptativo é composto por um dispositivo convencional – como exemplo, autômatos de estados finitos, gramáticas e tabelas de decisão – e uma camada adaptativa, que contém a lógica necessária para alterar a estrutura do dispositivo subjacente [7], conforme ilustra a Figura 1.

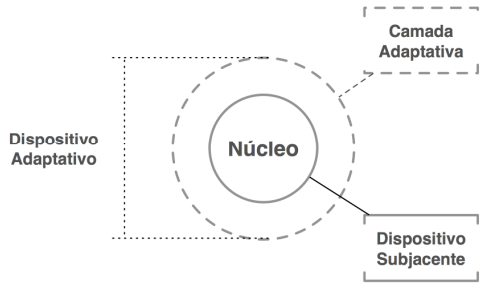


Figura 1. Estrutura de um dispositivo adaptativo.

O primeiro formalismo geral empregando o conceito de adaptatividade para um dispositivo guiado a regras foi introduzido por Neto [8], apesar de que o conceito de estrutura automodificável já houvesse sido explorado em formalismos individuais [9], [10], [11], [12]. Um dos primeiros exemplos notáveis deste tipo de mecanismo pode ser encontrado nos autômatos de estados finitos, em que o uso de funções adaptativas permite a criação e remoção de transições, bem como utilizar novos estados, conforme necessário – fato que faz com que este autômato seja Turing-equivalente [12]. Deve-se ressaltar que esta estrutura flexível permite reutilizar formalismos consolidados e torná-los adaptativos, ao custo de um pequeno acréscimo em sua complexidade [7].

Como exemplo de reuso, Tchemra fez uso desta tecnologia para permitir alterações no conjunto de regras de uma tabela de decisão, permitindo que novas regras fossem criadas tanto ao executar uma regra quanto ao verificar que nenhuma regra pode ser executada – neste caso, invoca um segundo método de decisão para que tome a decisão, cuja solução será

incorporada à tabela através de funções adaptativas [13]. A esta tabela, foi dado o nome de Tabela de Decisão Adaptativa Estendida (*TDAE*). A Figura 2 ilustra seu funcionamento, em que uma tabela de decisão convencional utiliza uma camada adaptativa para chamar funções adaptativas em situações em que não possui uma regra que se adeque ao problema, ou quando executa uma regra adaptativa. O método auxiliar utilizado por Tchemra, no caso, foi o AHP (*Analytic Hierarchy Process*) [13].

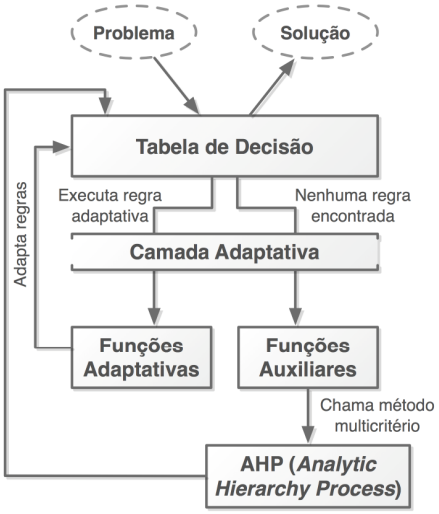


Figura 2. Fluxo de mensagens de uma Tabela de Decisão Adaptativa Estendida (*TDAE*).

Uma importante contribuição da TDAE foi o uso de tecnologia híbrida através da adaptatividade, permitindo que o dispositivo incorpore um novo conhecimento de forma incrementando, utilizando, para tanto, outro dispositivo, mais capaz em termos de potencial de decisão. Esta propriedade é base constituinte do ambiente aqui proposto, conforme está detalhado na seção III.

Outros exemplos de empregos deste tipo de técnica, além da tabela de decisão, podem ser encontrados em casos como processamento de linguagens naturais [14] e composição musical [15]. Ferramentais didáticos também foram desenvolvidos como forma de criar, aplicar e testar técnicas adaptativas, a exemplo do *AdapTools* [16].

B. Sistema de Apoio à Decisão:

Sistema de apoio à decisão (do inglês "*decision support system*") é um artifício computacional utilizado para dar apoio a um tomador de decisão, ajudando-o na tarefa de solucionar problemas [17]. Em geral, é formado por uma base de conhecimento, um modelo de decisão e uma interface gráfica [18], e deve ser treinado antes de ser efetivamente utilizado.

O treinamento de um sistema de suporte à decisão, classicamente, se dá através de um conjunto de amostras de casos conhecidos, fornecendo uma base de conhecimento inicial. O termo "*eager learning*" é dado aos métodos de aprendizados que tentam construir uma abstração durante o treinamento (ex: construção de uma árvore de decisão via ID3), enquanto o termo "*lazy learning*" é atribuído para os métodos que não fazem uma generalização a partir destes dados até que seja necessário durante sua execução [19] (ex: classificador *k-NN*). Em geral, o treinamento deve ocorrer quando se deseja atualizar a base de conhecimento, reconstruindo as abstrações criadas pelos métodos de aprendizado.

O sistema também deve levar em consideração os tipos de critérios que podem ser medidos e recebidos como entrada. Tipicamente, os problemas de classificação e de tomada de decisão utilizam critérios de dois tipos básicos: *nominais* (também chamados de *categóricos* ou *discretos*) e *contínuos*. Critérios nominais são aqueles que possuem um conjunto finitos de valores que podem ser atribuídos (ex: o critério "*tempo*" pode assumir valores tais como "*ensolarado*", "*chuvoso*", "*nublado*", etc.). Já critérios contínuos podem assumir valores numéricos quaisquer (ex: "*temperatura*" pode assumir valores reais acima de 0K – limite inferior conhecido).

Deve-se notar que os nem todos os métodos de aprendizado, classificação e de aprendizado têm capacidade de tratar ambos os tipos de critérios. Como exemplo, árvores de decisão construídas usando-se o algoritmo ID3 [20] tratam apenas atributos nominais, enquanto SVM (*Support-Vector Machines* [21]) opera apenas com valores numéricos. Métodos mais robustos, como C4.5 [22] e K* [23], lidam com ambos nativamente.

Em determinados casos, pode ser desejável converter os critérios de um tipo para outro. Um caso estudado foi o classificador de Bayes [24] o qual, ainda que consiga tratar os dois tipos por construção, obtém resultados melhores quando os atributos contínuos são convertidos para um conjunto finito de valores – passo conhecido como discretização. Uma das técnicas mais simples de discretização é o EWD (*Equal Width Discretization*), que divide o espaço de valores numéricos em múltiplas bandas de igual largura, cada banda correspondendo a um valor discreto [24]. Técnicas mais robustas incluem o RMEP (*Recursive Minimal Entropy Partitioning*) e uso de clusters [25].

O processo contrário, em transformar um critério nominal em contínuo, pode ser feito substituindo os valores nominais por um conjunto de critérios com valores binários mutuamente exclusivos. Cada valor binário representa uma categoria possível do critério nominal a ser convertido [26].

Inicialmente, os modelos de decisão eram imutáveis, sendo construídos uma única vez durante treinamento e mantidos até que um novo treinamento fosse realizado, com base em dados mais recentes. O uso de agentes inteligentes tem ganhado força neste ramo como maneira de aumentar o potencial do sistema [27], mas, em geral, não trata a questão da imutabilidade da abstração. Os conceitos *adaptativo* e *evolucionário* têm sido considerados como sendo os próximos passos do desenvolvimento dos sistemas de tomada de decisão [28].

C. Aprendizado Incremental:

O termo *aprendizado incremental* se refere à capacidade que apresenta um sistema de manter-se atualizado a partir de informações recebidas durante sua execução [29]. Esta

propriedade permite que o sistema consiga não só agregar conhecimentos antes desconhecidos, mas também se adequar a problemas dinâmicos, em que soluções anteriores podem não ser mais válidas para a situação atual [30].

Para métodos de aprendizado do tipo *lazy learning*, a capacidade incremental pode ser obtida com certa facilidade, uma vez que estes métodos não mantêm uma abstração e/ou generalização extraída do treinamento. Em especial, métodos baseados no aprendizado através de instâncias (IBL – *Instance-Based Learning*) são naturalmente incrementais por armazenarem apenas as instâncias treinadas previamente e utilizando-as como parâmetro de comparação (o termo instância se aplica a um conjunto de critérios-valores utilizados como entradas do sistema). Exemplos de métodos com *lazy learning* incluem o classificador de Bayes, *k-Nearest Neighbor* [31] e K* [23].

Contudo, adicionar capacidade incremental em métodos de aprendizado do tipo *eager learning* mostra-se mais complexa, uma vez que isso implica em alterar uma abstração de forma dinâmica – uma tarefa pouco trivial, considerando-se que as alterações dependem da estrutura criada pelo método de aprendizado. Um dos exemplos conhecidos neste tipo de aprendizado é o ITI (*Incremental Tree Inducer*), capaz de manter uma árvore de decisão atualizada de forma incremental, reestruturando-a a cada novo exemplo [32].

Nota-se também que, em conjunto com a capacidade de aprendizado incremental, é desejável obter o efeito contrário, de "esquecer" conhecimento de forma incremental, já que, em determinados problemas, soluções podem ser válidas apenas por um período de tempo, fazendo-se necessário um mecanismo para descartá-las após seu prazo de validade [5].

A capacidade de aprendizado incremental é utilizada como base para o modelo dos dispositivos a serem utilizados pelo sistema proposto. A adaptatividade ocorre na forma de inserção e remoção de conhecimento, cuja descrição é feita na seção III.

D. Raciocínio Baseado em Casos:

Quando confrontados com um novo problema, uma das técnicas frequentemente adotadas por humanos para solucioná-lo é reutilizar soluções de problemas passados que tinham semelhanças com o problema atual [33]. A este método de raciocínio, em que uma solução é adotada através de casos previamente enfrentados ao invés de derivar uma nova abstração, é dado o nome de Raciocínio Baseado em Casos (do inglês "*Case-Based Reasoning*", ou então, CBR).

O CBR é um método que se executa, tipicamente, em quatro etapas [34], conforme ilustra a Figura 3:

- ***Retrieve***: busca por casos cujo problema é similar ao caso atual;
- ***Reuse***: adaptação das soluções dos casos encontrados, tornando-os compatíveis com o novo problema;
- ***Revise***: aplicação da solução, verificando se os resultados foram aceitáveis;
- ***Retain***: memorização da nova solução na base de conhecimento, disponibilizando-a como um caso conhecido para futuros problemas.

Note que este método de raciocínio possui um modelo de aprendizado similar ao IBL, pois não mantém uma generalização a partir dos dados conhecidos. Isso permite utilizar um aprendizado incremental neste tipo de raciocínio, cuja aquisição de conhecimento ocorre após confirmar que a solução adotada estava correta.

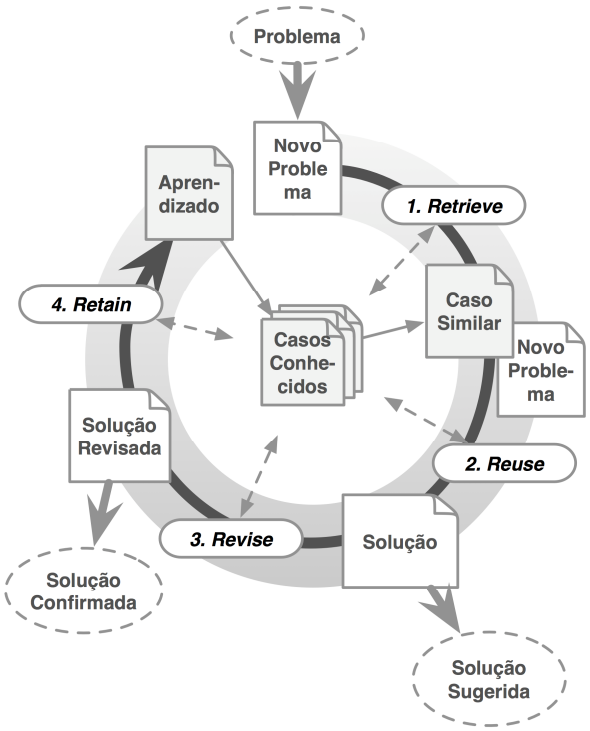


Figura 3. Metodologia adotada pelo raciocínio baseado em casos, conforme descrito por Aamodt [34].

Dois problemas surgem a partir deste raciocínio. Primeiramente, deve-se dispor de uma métrica para calcular a semelhança entre dois casos, que depende do problema a ser resolvido. Em seguida, deve-se notar que, em grande parte dos casos, não é possível assegurar que uma solução adotada está, de fato, correta, devido à possível latência temporal e espacial dos efeitos causados pela solução [35], ou pela falta de realimentação do usuário. Assim, o aprendizado incremental, diferentemente de um treinamento realizado por especialistas, deve lidar com um nível de incerteza muito maior. Este aspecto será tratado na seção seguinte.

E. Tratamento de Incertezas e Confiança de uma Decisão:

Um dos aspectos explorados neste trabalho é a capacidade do sistema decisório de estimar um nível de confiança para suas respostas. Confiar em uma solução significa acreditar que ela é, de fato, correta.

De maneira geral, quanto mais uma ação se repete, mais facilmente as pessoas acreditam que a mesma seja uma verdade. Crença também é fortemente influenciada por realimentações externas: opiniões que reforçam a ideia original aumenta a confiança, enquanto realimentações negativas

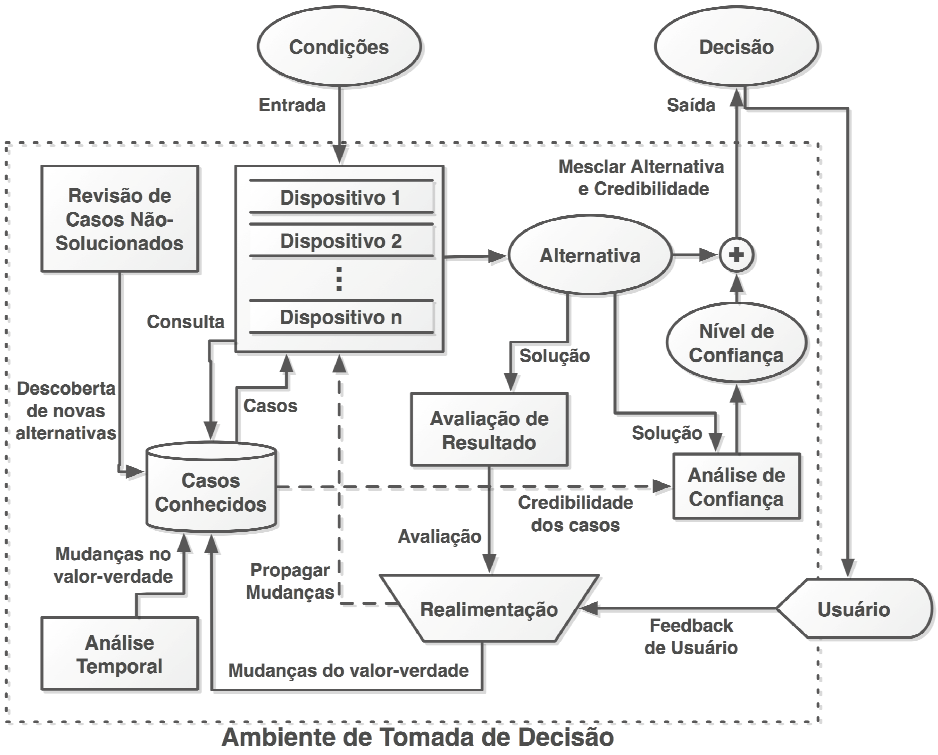


Figura 4: Visão Geral do Ambiente Adaptativo de Tomada de Decisão

alertam sobre a possibilidade de um erro [36].

Modelos que estimem a confiança baseada nas realimentações recebidas ainda não estão consolidados. Einhorn [37] sugere que a confiança C seja uma função baseado na soma F das realimentações:

$$C = f(F) \quad (1)$$

$$F = \beta_+ N_+ + \beta_- N_- \quad (2)$$

$$\beta_+ + \beta_- = 1.0, (\beta_+, \beta_- \geq 0.0) \quad (3)$$

Onde N_+ e N_- são as somas de realimentações positivas e negativas, respectivamente. Já β_+ e β_- são valores relativos de reforço das alimentações positivas e negativas. Note que a função f foi deixada em aberto em sua concepção original, mas espera-se um baixo valor para $F\sim0$, e deve aumentar com o módulo de F. Note que se $\beta_+ N_+ > \beta_- N_-$, temos maior confiança que o resultado está correto, enquanto $\beta_+ N_+ < \beta_- N_-$ aumenta a confiança de o resultado estar incorreto.

Uma consideração deve ser feita com relação às somas das realimentações: imaginando-se que o treinamento – do ponto de vista das fórmulas acima – é uma espécie de "realimentação", podem-se utilizar valores iniciais elevados para N_+ ou N_-, pois se imagina que o treinamento tem grande certeza de que a resposta está correta ou não. Contudo, para novas soluções, devem-se utilizar valores menores, para que a confiança inicial também seja baixa. Os incrementos realizados pelo aprendizado também devem ser ponderados de acordo com a credibilidade do mesmo: como exemplo, realimentações vindas de usuário ou especialista são mais confiáveis que uma realimentação obtida por uma mera repetição de ações. O uso de realimentações já pode ser observado em exemplos individuais de sistemas de busca, como o CBIR (*content-based image retrieval*) [38].

Nota-se que a confiança e a veracidade de uma solução, apesar de serem utilizadas em conjunto, não necessariamente são correlacionadas. Nada impede que o tomador de decisão, baseado em evidências incorretas, confie que sua solução seja correta, ainda que seja inverídica.

III. Ambiente Adaptativo de Tomada de Decisão

A. *Definições:*

Primeiramente, para fins deste projeto, o termo *instância* será utilizado como sendo o conjunto de valores de entrada do sistema – um para cada critério – de acordo com o termo utilizado pelo IBL. Cada instância k é representada, internamente, como sendo um vetor x_k de m dimensões, onde o i-ésimo representa o valor associado ao i-ésimo critério.

$$x_k = (x_{k1}, x_{k2}, ..., x_{ki}, ..., x_{km}) \quad (4)$$

Para critérios *contínuos*, seus valores devem ser números reais, enquanto para critérios *nominais*, serão utilizados nomes – referenciados como *categorias*. O número de categorias possíveis para cada critério deve ser finito.

Em contrapartida, o termo *caso* será usado como um par instância-alternativa, representando o problema e uma possível solução, indicada pela alternativa. Considerando que a alternativa de um caso j seja representada por y_j, temos:

$$caso_j = (x_j, y_j) \quad (5)$$

O termo *dispositivo* será utilizado para designar os métodos de tomada de decisão a serem utilizados pelo sistema,

devendo, assim, ser adaptativos. Sua estrutura interna está descrita na seção C.

Por fim, *realimentação* será compreendida como uma reação à solução encontrada pelo sistema, capaz de alterar seu comportamento para futuras novas soluções, podendo ser gerada pelo usuário e/ou por um mecanismo inteligente. Sua descrição encontra-se na seção D.

B. Visão Geral:

A partir do modelo utilizado pela TDAE, em que um segundo dispositivo é utilizado para auxiliar o aprendizado do dispositivo adaptativo [13], pode-se expandi-lo para que utilize um número arbitrário de dispositivos, dispostos em uma estrutura de fila. Este modelo é ilustrado pela Figura 5. Sua ideia intuitiva é semelhante da TDAE: ao receber um problema, o sistema o envia para o primeiro dispositivo da fila. Caso consiga solucioná-lo de forma confiável, retorna sua decisão; caso contrário, o problema é repassado ao próximo da fila. Este procedimento deve se repetir até que um dos dispositivos consiga solucionar o problema, ou então, até esgotar a fila. O funcionamento interno de um dispositivo está descrito na subseção C.

Para cada alternativa gerada pelos dispositivos, deve-se estimar um nível de confiança para a decisão tomada, utilizando, como base, as realimentações recebidas pelo sistema, conforme introduzido na seção II. E. Para tanto, também se faz necessário utilizar uma base de conhecimento que armazene as realimentações recebidas para cada caso conhecido, permitindo calcular, de alguma forma, a equação (1). A saída do sistema, inclusive, é composta pela união entre a alternativa e as informações de confiança.

A realimentação armazenada na base pode ser originada no próprio sistema – através de um bloco de avaliação de decisão – ou então, fornecido pelo próprio usuário. O bloco de realimentação deve receber estas informações e enviá-las à base, bem como propagar as mudanças para os dispositivos, permitindo que alterem sua estrutura interna. Como exemplo, a TDAE original recebia esta informação propagada do AHP, o que resultava na geração de novas regras [13].

Junto à base de dados, deve haver um bloco de análise temporal, cuja finalidade é verificar se a validade de algum caso conhecido expirou – isto é, deixou de valer após um período de tempo – permitindo tratar conceitos quem variem no tempo. Também deve estar presente um bloco de revisão, que verifica a possibilidade de existirem novas alternativas, antes desconhecidas. Esta possibilidade, em princípio, ocorre ao detectar que há um grande número de casos semelhantes em que nenhuma das soluções é considerada válida.

C. Dispositivo Adaptativo de Tomada de Decisão:

Internamente, cada dispositivo adaptativo de tomada de decisão, a exemplo do modelo original ilustrado pela Figura 1, deve conter um dispositivo subjacente, capaz de solucionar o problema de decisão vindo do dispositivo anterior, confirme ilustra a Figura 5. Isso deve ser feito calculando-se um *índice de mérito* relacionado com a probabilidade de cada alternativa ser solução do problema, de acordo com o método de decisão utilizado.

A partir do índice calculado, o dispositivo deve analisar se a solução obtida pode ser considerada "adequada", verificando se a solução de melhor nota é um bom candidato para solucionar o problema. Uma técnica simples para realizar tal operação é utilizar um limiar: se a melhor nota estiver acima do limiar, retorna sua solução como sendo resposta ao sistema. Caso contrário, repassa o problema ao próximo dispositivo. A condição de saída da fila está detalhada na subseção G.

A camada adaptativa deve permitir que realimentações – geradas a partir das soluções encontradas pelos dispositivos – alterem seu comportamento. Detalhes sobre a realimentação se encontram nas seções D. e E.

Deve-se notar que o dispositivo subjacente pode ter acesso à base de conhecimento para realizar consultas, uma vez que métodos de aprendizado baseados em instâncias dependem do armazenamento dos casos conhecidos. Tal estrutura permite utilizar uma única base, mesmo que seja utilizado mais de um método baseado em casos.

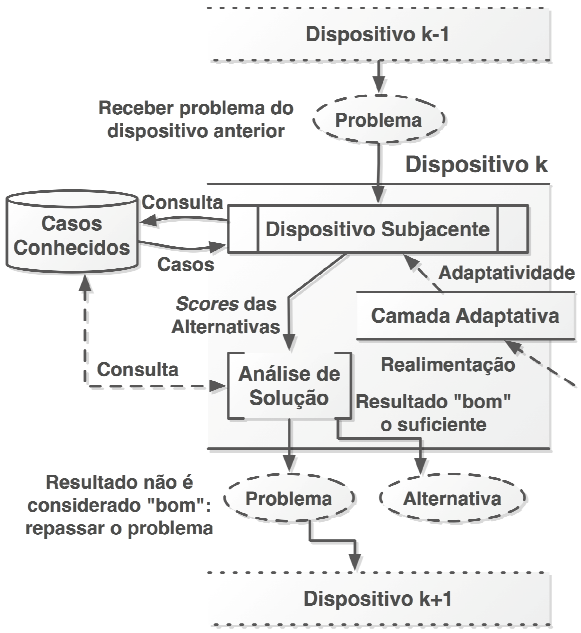


Figura 5: Modelo de um dispositivo adaptativo de tomada de decisão.

D. Realimentação:

Para este projeto, realimentação é uma relação que avalia um par instância-alternativa, informando se a alternativa é ou não uma solução para a instância do problema, de acordo com o analisador que a gerou. Uma instância k de realimentação é representada por uma tripla (v_k, β_k, c_k), tal que:

- v_k = valor verdade (0.0 ou 1.0) da dupla instância-alternativa avaliada por esta realimentação – valor 1.0 diz que a alternativa é solução, enquanto 0.0 diz o contrário;
- β_k = valor relativo de reforço, que depende se a realimentação é positiva (valor verdade 1.0) ou negativa (valor verdade 0.0) – ver equações (2) e (3);
- c_k = credibilidade do analisador que gerou esta

realimentação – deve ser um valor positivo maior que zero.

A realimentação, gerada após a tomada de uma decisão, pode ter origem em três tipos de fontes:

- **Usuários do sistema**: informam se as soluções foram o que esperavam inicialmente;
- **Analisadores Computacionais:** blocos dotados de "inteligência" (IA), capazes de verificar automaticamente a qualidade da resposta – representado pela "Avaliação de Resultado" da Figura 4;
- ***Repetition-bias*:** o sistema, naturalmente, envia realimentações com baixo valor de c_k ao repetir a execução de uma solução, acreditando que está correta (pois o sistema espera que, se estiver errado, alguém lhe dirá através de uma realimentação negativa);

Nota 1: apesar de não ser realimentação, o treinamento do sistema também pode ser compreendido como sendo uma tripla (v_k, β_k, c_k), exceto que a credibilidade c_k assume valores muito maiores. Para este projeto, ainda que seja conceitualmente diferente, o treinamento também utilizará esta notação.

Nota 2: o bloco de avaliação de resultado não possui uma implementação fixa, e deve ser especificamente desenvolvida para cada sistema em particular. A presença deste bloco no sistema é opcional (quando não estiver presente, não haverá realimentação através de IA).

A junção da credibilidade e do valor de reforço resulta no peso que uma realimentação terá frente às demais:

$$w_k = \beta_k \cdot c_k \tag{6}$$

Assim, de acordo com as definições dadas por Einhorn [37], deve-se estabelecer uma função que estime a confiança de uma solução a partir da soma das realimentações recebidas – no caso, utilizamos uma soma ponderada por w_k – o que se encontra descrito na seção seguinte.

E. Cálculo do Valor-Verdade e Intervalo de Confiança:

Um meio para calcular o valor-verdade de uma alternativa para uma instância, então, pode ser dado pela média ponderada dos valores verdades de todas as *n* realimentações recebidas que avaliaram este par instância-alternativa:

$$\bar{v} = \frac{V}{W}, \left(V = \sum_{k=1}^{n} v_k \cdot w_k, W = \sum_{k=1}^{n} w_k \right) \tag{7}$$

O valor *V* pode ser interpretado como número de vezes que a alternativa foi considerada correta dentre as *W* vezes que o par foi avaliado.

A partir destes valores, também se calcula um intervalo de confiança para este valor-verdade, fornecendo, assim, a medida de confiança comentada anteriormente.

Tipicamente, o cálculo do intervalo de confiança é realizado encontrando dois limites, v_{low} e, v_{upp} tais que a probabilidade de a verdadeira média μ estar neste intervalo seja $1-\alpha$. Assim:

$$P\left(v_{low} \le \mu \le v_{upp}\right) = 1 - \alpha \tag{8}$$

Como o valor-verdade, de acordo com o que fora definido anteriormente, só assume valores binários – verdadeiro (1.0) ou falso (0.0) – tem-se uma distribuição de Bernoulli, cujo intervalo de confiança, quando é aproximado por uma distribuição normal, é:

$$\bar{v} \pm z_{1-\alpha/2} \sqrt{\frac{\bar{v}(1-\bar{v})}{W}} \tag{9}$$

tal que $Z_{1-\alpha/2}$ é o $(100\times(1-\alpha/2))$-ésimo percentil de uma distribuição normal. Seu cálculo é feito pela inversa da função distribuição acumulada da distribuição normal – comumente referenciada como *probit function*. Nota-se que não existe uma forma fechada para esta função, devendo ser estimada por aproximações [39].

Este intervalo, contudo, pode não representar bem situações em que o número de amostras (no caso, o número de realimentações recebidas) é baixo, ou então, para situações em que a verdadeira média é muito próxima de 0 ou 1 [40].

Como exemplo, ao lançar uma moeda não-viciada três vezes, a possibilidade de sair três caras é factível. Caso fosse calculada, a partir desta amostra, a massa de probabilidade de um lançamento dar "cara", chega-se à conclusão que a probabilidade é 1.0 com intervalo zero, o que não representa a real situação.

Uma alternativa viável neste caso é utilizar outros métodos de cálculo de intervalo de confiança. Um exemplo conhecido é o método de *Agresti-Coull*, que insere, artificialmente, uma quantidade igual de amostras de valor 0 ou 1, movendo a média para mais próximo de 0.5, evitando assim que o intervalo seja zero. O efeito das amostras artificiais é reduzido conforme o número de amostras reais aumenta, fazendo com que a média de *Agresti-Coull* convirja para a média real.

Através deste método, a nova média e intervalos são calculados conforme as equações (10), (11) e (12):

$$\hat{W} = W + \left(z_{1-\alpha/2}\right)^2 \tag{10}$$

$$\hat{v} = \frac{V + z_{1-\alpha/2}^2 / 2}{\hat{W}} \tag{11}$$

$$\hat{v} \pm c_{1-\alpha/2} = \hat{v} \pm z_{1-\alpha/2} \sqrt{\frac{\hat{v}(1-\hat{v})}{\hat{W}}} \tag{12}$$

Nota-se que neste método, o número de amostras artificiais inseridas é $(Z_{1-\alpha/2})^2$, novamente, relativo ao percentil utilizado anteriormente. O efeito das amostras pode ser visualizado pelo gráfico da Figura 6. Assim, este sistema utilizará, como valor-verdade de um par instância-alternativa, a média desviada de *Agresti-Coull*. O intervalo de confiança será $C_{1-\alpha/2}$.

Nota 1: o método de Agresti-Coull é válido mesmo quando o número de amostras for zero, resultando uma média 0.5 e intervalo também igual a 0.5, englobando todo o espaço de valores possível.

Nota 2: o cálculo do valor-verdade e do intervalo de confiança pode ser realizado de forma incremental, armazenando apenas as somatórias W e V, incrementando-as conforme novas realimentações cheguem e aplicando as equações (11) *e* (12).

A cada recebimento de realimentação, a base de conhecimento deve notificar os dispositivos sobre a mudança

no valor-verdade para o caso analisado, permitindo que os mesmo se adaptem ao novo conhecimento.

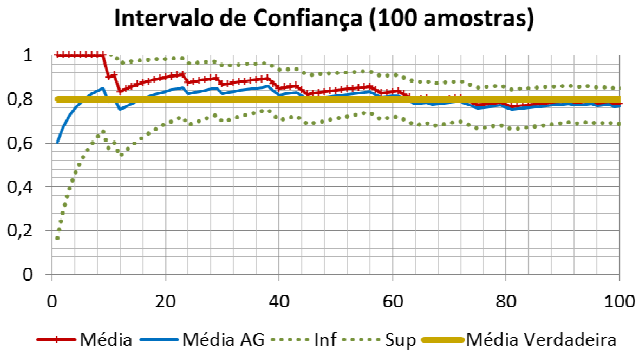


Figura 6: Gráfico da média de um valor que segue a distribuição de Bernoulli conforme o número de amostras aumenta. A média real é dada pela linha amarela (p=0.8). Nota-se que a média e o intervalo de Agresti-Coull (dado pelas linhas pontilhadas) se mantêm mais próximos da média real do que a média aritmética quando o número de amostras é baixo, mas convergem para o verdadeiro valor de p conforme o número de amostras aumenta.

F. *Base de Conhecimento:*

Conforme descrito pelo item anterior, é necessário armazenar os valores de V e W para cada caso possível para permitir estimar seu valor verdade. Como um caso é definido pela dupla instância-alternativa, pode-se utilizar a informação espacial da instância para rapidamente encontrar casos conhecidos, utilizando uma estrutura eficiente, como o *Locally Sensitive Hashing*, ou então, se todos os critérios forem numéricos, uma estrutura como o *R*-Tree* [41].

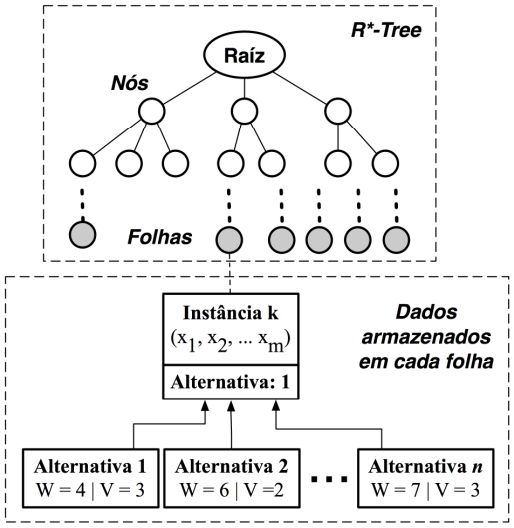


Figura 7: Exemplo de uma base utilizando uma *R*-Tree*. Em cada folha, deve estar armazenada a instância – utilizada como informação espacial, e todos os casos possíveis que utilizam tal instância – no caso, todas as alternativas possíveis, cada uma com seu valor W e V.

O uso destas estruturas se justifica para que consultas à base sejam eficientes, principalmente quanto for necessário encontrar os vizinhos mais próximos – uma necessidade para qualquer método de aprendizado baseado em instâncias. Nota-se que é possível armazenar estes dados em uma simples tabela, apesar de ser computacionalmente mais custoso.

Uma consideração, contudo, deve ser feita: critérios numéricos, ao contrário dos critérios nominais, podem assumir valores teoricamente infinitos. Para evitar um possível crescimento infinito da base de conhecimento, critérios numéricos devem ser discretizados, conforme as técnicas citadas na seção II. B.

G. *Condição de Saída da Fila de Dispositivos:*

A descrição do dispositivo utilizado pelo ambiente de decisão especifica a necessidade de decidir se a solução encontrada por um dispositivo é adequada, repassando o problema ao próximo em caso negativo. Para isso, o dispositivo deve calcular um *score* para cada alternativa.

Uma possível maneira de se decidir a parada é utilizar um *limiar de aceitação*, tal que a solução é aceita somente se o melhor *score* obtido está acima do limiar. Para isso, pode-se normalizar este escore para que a soma de todas as notas seja 1.0, o que permite escolher um limiar no intervalo $[0...1]$. A este índice de mérito normalizado, define-se o termo *probabilidade de classes* (baseado no termo utilizado pelo classificador de Bayes - *class probability*).

Supondo que p_i seja a probabilidade de classe da alternativa i, temos a seguinte lógica booleana para decidir a saída da fila:

$$aceita = \left(\max\left(p_1, p_2, ..., p_n\right) > thresh\right) \quad (13)$$

tal que *thresh* é o limiar escolhido (do inglês, *threshold*). Este método, porém, não considera a possibilidade de haver duas alternativas com probabilidades elevadas, o que poderia ocasionar ambiguidade. Para esta condição, poderia ser utilizado um segundo critério, em que se verifica a razão entre a segunda maior probabilidade e a primeira. Este valor deve estar abaixo de um limiar para que a solução seja considerada livre de ambiguidades. Assim:

$$ambig. = \left(\frac{\max_2\left(p_1, p_2, ..., p_n\right)}{\max\left(p_1, p_2, ..., p_n\right)} > thresh2\right) \quad (14)$$

$$aceita = \left(\max\left(p_1, p_2, ..., p_n\right) > thresh\right) \wedge \neg ambig. \quad (15)$$

tal que max_2 é uma função que calcula o segundo maior valor da lista e *thresh2* é o limiar de ambiguidade, devendo ser um valor menor que 1.0.

Nota: caso o dispositivo, por construção, não consiga resolver o problema, deve retornar probabilidade zero para todas as alternativas. Ainda que não cumpra com o quesito da normalização, não deve passar pela condição de saída.

H. *Treinamento:*

Inicialmente, deve-se treinar o sistema com um conjunto inicial de casos, informando-lhe quais são as alternativas corretas para um conjunto de instâncias conhecidas. A base de conhecimento deve armazenar, para cada instância, $V=W=w_0$ para a alternativa correta e $V=0$ para todas as demais. O valor w_0 representa um peso inicial para os casos de treinamento.

Nota: $w_0 = \infty$ faz o valor-verdade subir para 1.0 para a alternativa correta, e impede que este fato seja alterado por realimentações futuras.

Também deve ser definida a credibilidade c_k das realimentações de repetição – enviadas após cada decisão, reforçando a alternativa escolhida com $(v_k=1,\ \beta_k = \beta_+,\ c_k)$ e

negativando todas as demais com $(v_k=0, \beta_k = \beta_-, c_k)$. Para este trabalho, utiliza-se $c_k=0.01$. Nota-se que um valor muito elevado faz com que o sistema ganhe confiança rapidamente, diminuindo o impacto das realimentações de origem mais confiável.

IV. Dispositivos Adaptativos com Aprendizado Incremental

Para este projeto, foram desenvolvidos três dispositivos adaptativos, baseados em métodos desenvolvidos previamente, e que, por natureza, mostravam um bom potencial para adaptatividade: TDAE, k-Nearest Neighbor e classificador de Bayes. Cada um deles será descrito a seguir.

A. TDAE:

Como está descrito na seção II. , a tabela de decisão adaptativa estendida foi a base conceitual deste projeto, e é, também, um dos dispositivos com implementação disponível para uso pelo ambiente. Porém, diferentemente da formulação original, a tabela aqui utilizada faz uso de entradas estendidas (isto é, entradas podem assumir valores discretos arbitrários em lugar de valores binários). Os critérios numéricos, a exemplo da base de dados, também devem ser discretizados para serem inseridos na tabela.

A tabela de decisão, por definição, consegue tratar apenas instâncias já recebidas anteriormente, pois somente estas terão uma regra equivalente na tabela. Para estes casos, a tabela retorna probabilidade 1.0 para a alternativa prevista pela regra encontrada, e 0.0 para as demais.

Ao receber uma notificação de realimentação através de sua camada adaptativa, a TDAE pode prosseguir de duas maneiras:

- Se a instância avaliada já era conhecida pela base de dados, a TDAE deve verificar se a melhor alternativa mudou. Em caso positivo, deve alterar sua regra correspondente na tabela;
- Se a instância é nova, uma nova regra deve ser criada.

Nota-se que a tabela de decisão adaptativa permite executar uma adaptatividade ao executar uma regra. Neste projeto, não será utilizada a adaptatividade desta maneira, apesar de ser teoricamente possível.

B. k-Nearest Neighbor:

Um dos métodos de aprendizado baseado em instâncias utilizados foi uma extensão do popular *k-Nearest Neighbor* (k-NN). Originalmente, o classificador k-NN procura pelos k vizinhos mais próximos a uma nova instância t, cada um com uma solução y_i já conhecida. A solução da nova instância, então, é definida por [42]:

$$y_t = \underset{c\in\{c_1\ldots c_n\}}{\operatorname{argmax}} \sum_{i=1}^{k} I(y_i = c) \tag{16}$$

sendo que $I(x)$ é a função indicadora, retornando 1 se a alternativa correta da instância é c. Assim, a alternativa da nova instância é aquela que ocorre com maior frequência entre os vizinhos mais próximos. Esta função, porém, não considera a grau de semelhança entre os casos: instâncias mais próximas deveriam receber peso maior por, supostamente, terem maior influência na resposta final. Assim:

$$y_t = \underset{c\in\{c_1\ldots c_n\}}{\operatorname{argmax}} \sum_{i=1}^{k} I(y_i = c)\cdot w(x_t, x_i) \tag{17}$$

tal que $w(x_t,x_i)$ é uma função que pondera a instância x_i pela sua distância. Dentre os métodos utilizados, estão o método de Shepard [43] e a função de similaridade [42], dados pelas equações (18) e (19):

$$w(x_t, x_i) = \frac{1}{d(x_t, x_i)^p} \tag{18}$$

$$w(x_t, x_i) = 1 - \frac{d(x_t, x_i)}{d_{\max}} \tag{19}$$

onde $d(x_t,x_i)$ é uma função que calcula a distância entre duas instâncias, enquanto d_{max} é um fator de normalização para que a similaridade esteja no intervalo *[0...1]*.

Esta definição exige, contudo, que cada instância tenha uma única alternativa conhecida com 100% de confiança, explicitado pela função indicadora. No caso do ambiente adaptativo, as instâncias possuem valores-verdade ao invés de uma única verdade absoluta. Para adequar o k-NN a este problema, utiliza-se a equação (20):

$$y_t = \underset{c\in\{c_1\ldots c_n\}}{\operatorname{argmax}} \sum_{i=1}^{k} \hat{v}(x_i, c)\cdot w(x_t, x_i) \tag{20}$$

Esta nova definição substitui a função indicadora pelo valor-verdade de cada alternativa, conforme dado pela equação (11).

Já para a função de distância, classicamente, utiliza-se a distância Euclidiana. Para um caso com m critérios, temos:

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{m} \left| x_{1j} - x_{2j} \right|^2} \tag{21}$$

Contudo, esta função é somente válida em casos em que todos os critérios são numéricos. E ainda assim, como os valores de cada critério podem assumir ordens de grandeza diferentes, a influência de cada critério sobre o valor da distância acaba sendo desigual. Para mitigar este problema, normaliza-se a distância para que cada critério contribua com a distância de forma igualitária:

$$d(x_1, x_2) = \sqrt{\frac{\sum_{j=1}^{m} d_j(x_1, x_2)^2}{m}} \tag{22}$$

$$d_j(x_1, x_2) = \left| x_{1j} - x_{2j} \right| \Big/ d_{j\max} \tag{23}$$

onde $d_j(x_1,x_2)$ calcula a distância normalizada entre as instâncias x_1 e x_2 no critério j, enquanto $d_{j\ max}$ é a máxima distância possível neste critério.

Estas equações, contudo, continuam válidas apenas para critérios contínuos. Para tratar critérios nominais, pode-se estabelecer a seguinte equação:

$$d_j(x_1, x_2) = \delta(x_{1j}, x_{2j}) \tag{24}$$

A função $\delta(x_{1j}, x_{2j})$ retorna 0 se, e somente se, os valores x_{1j} e x_{2j} forem idênticos, retornando 1 caso contrário. Esta métrica é comumente definida como *Overlap Metric* [44]. Outra opção é utilizar uma métrica que considere as probabilidades de uma

alternativa aparecer com cada um dos valores da instância:

$$d_j(x_1, x_2) = \sum_{c=1}^{C} \left| P(c \mid x_{1j}) - P(c \mid x_{2j}) \right| \quad (25)$$

onde $P(c \mid x_{jj})$ é a probabilidade de a alternativa da instância x_i ser c, dado que seu valor para o j-ésimo critério é x_{ij}. Este valor pode ser obtido conhecendo-se a frequência com que cada alternativa aparece quando uma instância aparece com o valor x_j. Esta métrica recebe o nome de *Value Difference Metric* [44].

Note-se que o k-NN não requer manter uma abstração sobre as instâncias conhecidas, somente dependendo da base de dados.

C. Classificador de Bayes:

O classificador de Bayes é um método probabilístico de classificação que assume que os valores necessários dos critérios para definir uma alternativa são independentes entre si. Pela teoria de Bayes, a probabilidade de uma instância x_a ter a alternativa c como resposta é dada pela equação (26):

$$p(c \mid x_a) = \frac{1}{Z(x_a)} p(c) \prod_{i=1}^{m} p(x_{ai} \mid c) \quad (26)$$

onde $p(c)$ é a probabilidade a priori de uma instância x qualquer ter c como alternativa correta, enquanto $p(x_{ai}|c)$ é a probabilidade de o valor x_{ai} aparecer no i-ésimo critério quando uma instância tiver a alternativa c como solução. A princípio, Z é um fator de escala que denota a *evidência*, normalizando as probabilidades e fazendo com que sua soma seja 1.0:

$$Z(x_a) = \sum_{j=1}^{n} \left(p(c_j) \prod_{i=1}^{m} p(x_{ai} \mid c_j) \right) \quad (27)$$

Contudo, como Z é um valor comum a todas as probabilidades, chega-se à seguinte conclusão:

$$p(c \mid x_a) \propto p(c) \prod_{i=1}^{m} p(x_{ai} \mid c) \quad (28)$$

Supondo que a alternativa de uma instância seja aquela cuja probabilidade calculada seja a maior, temos:

$$y(x_a) = \underset{c \in (c_1, \ldots c_n)}{\operatorname{argmax}} \; p(c) \prod_{i=1}^{m} p(x_{ai} \mid c) \quad (29)$$

Para este trabalho, o valor de $p(c)$ é calculado através da soma dos valores-verdade de cada instância x para a alternativa c, e dividindo-o pelo total de valores verdades. Supondo que há s instâncias salvas na base, temos:

$$p(c) = \frac{\sum_{i=1}^{s} v(x_i, c)}{\sum_{j=1}^{n} \sum_{i=1}^{s} v(x_i, c_j)} \quad (30)$$

De forma similar, cada elemento $p(x_{ai}|c)$ pode ser calculado somando o número de vezes com que o valor x_i aparece quando uma instância tem a alternativa c como solução.

$$p(x_{ai} \mid c) = \frac{\sum_{j=1}^{s} u(x_j, x_a, i, c)}{\sum_{j=1}^{s} v(x_j, c)} \quad (31)$$

$$u(x_j, x_a, i, c) = v(x_j, c) \cdot I(x_{ji} = x_{ai}) \quad (32)$$

Alterações nos valores-verdade e no número de instâncias – informadas pela adaptatividade – devem alterar as probabilidades definidas acima. A atualização pode ser feita de forma eficiente se for armazenado o valor das três somatórias definidas nas equações (31) e (32), e apenas incrementá-las/decrementá-las conforme os valores-verdade utilizados por elas se alterem, utilizando a propriedade:

$$S_{t+1} = S_t - s_{ilt} + s_{slt+1} \quad (33)$$

onde S_t é uma somatória qualquer em um instante t, enquanto $s_{t \mid t}$ é o valor de um elemento da somatória em um instante t. Se este valor, em um instante $t+1$, for alterado, basta substituir o valor anterior e somar o novo.

Utilizando esta propriedade, tais somatórias acima podem ser atualizadas incrementalmente. Como exemplo, supondo que o valor de $v(x_b.c)_t$ seja o valor-verdade da alternativa c para a instância x_b em um instante t. Caso este valor seja alterado em um instante $t+1$:

$$p(c)_{t+1} = \frac{\left(\sum_{i=1}^{s} v(x_i, c) \right)_t - v(x_b, c)_t + v(x_b, c)_{t+1}}{\left(\sum_{j=1}^{n} \sum_{i=1}^{s} v(x_i, c_j) \right)_t - v(x_b, c)_t + v(x_b, c)_{t+1}} \quad (34)$$

O cálculo de $p(x_{ai}|c)_{t+1}$ pode ser feito de forma análoga, aplicando o mesmo conceito sobre $u(x_j,x_a,i,c)$ nos instantes t e $t+1$.

V. Resultados

A. Aprendizado Incremental:

Para verificar o funcionamento do aprendizado incremental, foi criado um sistema experimental, que utiliza dois dispositivos: TDAE, seguido por um k-NN, com $k=10$ e utilizando ponderação pelo método de Shepard, com $p=2$. Para calcular distâncias entre atributos nominais, utiliza-se o *Overlap Metric*, definido na equação 24. Para ponderar as realimentações positivas e negativas, é utilizado $\beta_+=\beta_-=0.5$. A discretização de valores numéricos foi realizada através do método *Equal Width Discretization*, descrito em [24], utilizando-se 100 valores discretos de igual comprimento.

Para testá-lo, foram utilizados *datasets* fornecidos pelo repositório de aprendizado de máquina da UCI [45]. Para cada conjunto de testes, 20% dos dados foram sorteados aleatoriamente para serem utilizados como treinamento inicial do sistema, utilizando $w_0 = \infty$. Após o treinamento, o sistema foi executado 100 vezes consecutivas, selecionando aleatoriamente 80% dos dados como entradas – não necessariamente o complemento dos 20% de treinamento. Em caso de o sistema retornar uma alternativa não esperada, há 2% de chance de o sistema receber uma realimentação $(v_k=0,\beta_k=\beta_-,c_k=1)$ para a alternativa retornada, e outra $(v_k=1,\beta_k=\beta_+,c_k=1)$ para a alternativa esperada.

A Tabela I mostra os resultados obtidos, ilustrando o aumento na taxa de acertos obtido para cada conjunto de dados.

TABELA I
EVOLUÇÃO DE APRENDIZADO APÓS 100 EXECUÇÕES CONSECUTIVAS

DATASET	TAXA DE ACERTOS INICIAL	TAXA DE ACERTOS FINAL
ABALONE	37.29%	87.13%
CAR EVALUATION	84.66%	96.96%
CREDIT	85.51%	96.92%
HABERMAN	72.95%	94.67%
IRIS	96.67%	99.17%
NURSERY	91.49%	98.37%
SHUTTLE	99.79%	99.97%
STATLOG	75.75%	94.50%
TIC-TAC-TOE	85.51%	97.52%
WINE	92.96%	99.30%
YEAST	61.92%	92.25%

Analisando os resultados individualmente, verifica-se que a maior parte dos erros ocorre com alternativas menos frequentes, e que, por consequência, possuem menos amostras no conjunto inicial de treinamento. Como exemplo, no *Car Evaluation*, a alternativa "*unacc*" domina as demais com uma frequência de 70%, e raramente é confundida com as demais, obtendo taxa de acertos superior a 99% logo após o treinamento. As demais alternativas, menos numerosas, possuem acertos iniciais inferiores, mas evoluem conforme recebem realimentações, conforme ilustra a Figura 8.

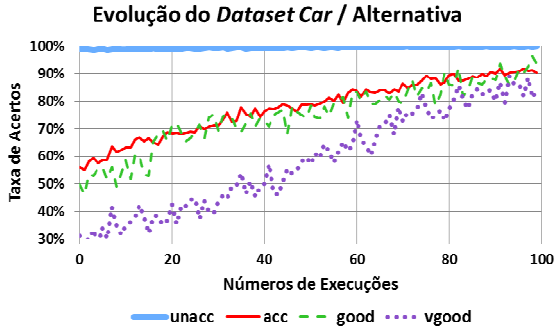


Figura 8: Evolução do *dataset Car Evaluation* para cada alternativa.

B. *Compromisso de Taxa de Acertos e Tempo de Resposta:*

Um aspecto analisado por este trabalho é o efeito do limiar de aceitação no tempo de resposta e acertos de um sistema de tomada de decisão, conforme definido na seção III. G.

Para efetuar esta análise, foi utilizado um sistema composto por três dispositivos, nesta ordem: TDAE, classificador de Bayes e k-Nearest Neighbor. A escolha destes dispositivos e de sua ordenação foi feita de forma que:

1. Casos conhecidos sejam solucionados de maneira imediata pela tabela de decisão;
2. Casos jamais vistos sejam solucionados de maneira rápida pelo classificador de Bayes;
3. Se o classificador de Bayes não conseguir um bom escore para a melhor solução, repassa o problema ao k-NN.

Note que o número de casos conhecidos, em geral, é muito superior ao número de critérios e de alternativas. Assim, por depender linearmente do tamanho da base de conhecimento, o k-NN deve ser mais lento que Bayes, que depende do número de critérios e alternativas.

Nota: exceto pela adição de Bayes e pelos dados de treinamento diferentes, este sistema possui a mesma configuração que o teste anterior, com k=10, p=2, etc.

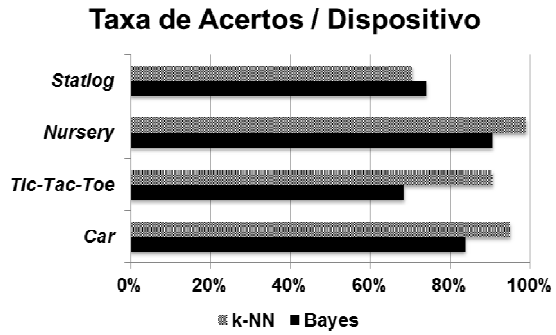


Figura 9: Taxa de acertos de 4 *datasets* quando testados com o classificador de Bayes e com o k-NN.

Como dados de testes, foram utilizados os mesmos *datasets* empregados anteriormente. Para cada conjunto, foi verificado o efeito do limiar sobre a taxa de acertos e tempo de resposta, variando-o de 0.3 até 1.0, em passos de 0.02. Para cada limiar, o sistema foi testado 10 vezes, sorteando aleatoriamente 70% dos dados como treinamento e utilizando os 30% remanescentes para validar o sistema, executando-o e verificando se a resposta foi aquela esperada pelo *dataset*.

Na maior parte destes conjuntos de testes, o k-NN – utilizando os parâmetros acima – obtém maior número de acertos do que o classificador de Bayes. Um dos raros exemplos contrários foi o *dataset Statlog*, conforme ilustra a Figura 9.

Os resultados do teste do limiar para quatro dos conjuntos utilizados podem ser visualizados pelos gráficos da Figura 10. De imediato, verifica-se que todos os exemplos apresentam uma piora no tempo de resposta com o aumento do limiar – comportamento esperado, uma vez que o k-NN passa a ser requisitado com maior frequência.

Outro resultado visível ocorre com os *datasets Car*, *Nursery* e *Tic-Tac-Toe*. Como k-NN consegue maior taxa de acertos do que Bayes nestes exemplos, seu uso faz com que os acertos do sistema cresçam conforme o limiar aumenta. Esta relação, então, pode ser utilizada para estimar um limiar que melhor satisfaça as necessidades do usuário (e.g. a escolha pode ser feita para "obter a maior número de acertos por unidade de tempo", ou "garantir uma taxa de acertos de x%").

Note-se também que, nestes exemplos, há um ponto de saturação, em que o aumento do uso do k-NN passa a não impactar em um aumento de acertos. Com isso, o uso deste limiar permite que o sistema consiga obter a mesma taxa de acertos do dispositivo mais capaz – o k-NN – utilizando, em média, apenas uma fração de seu tempo. O teste com o *dataset Car* é quem melhor evidencia este ganho: a taxa de acertos do sistema com limiar 0.5 é comparável ao k-NN, mas utilizando apenas 4ms por decisão.

O *Statlog* exemplifica um caso contrário, em que o uso

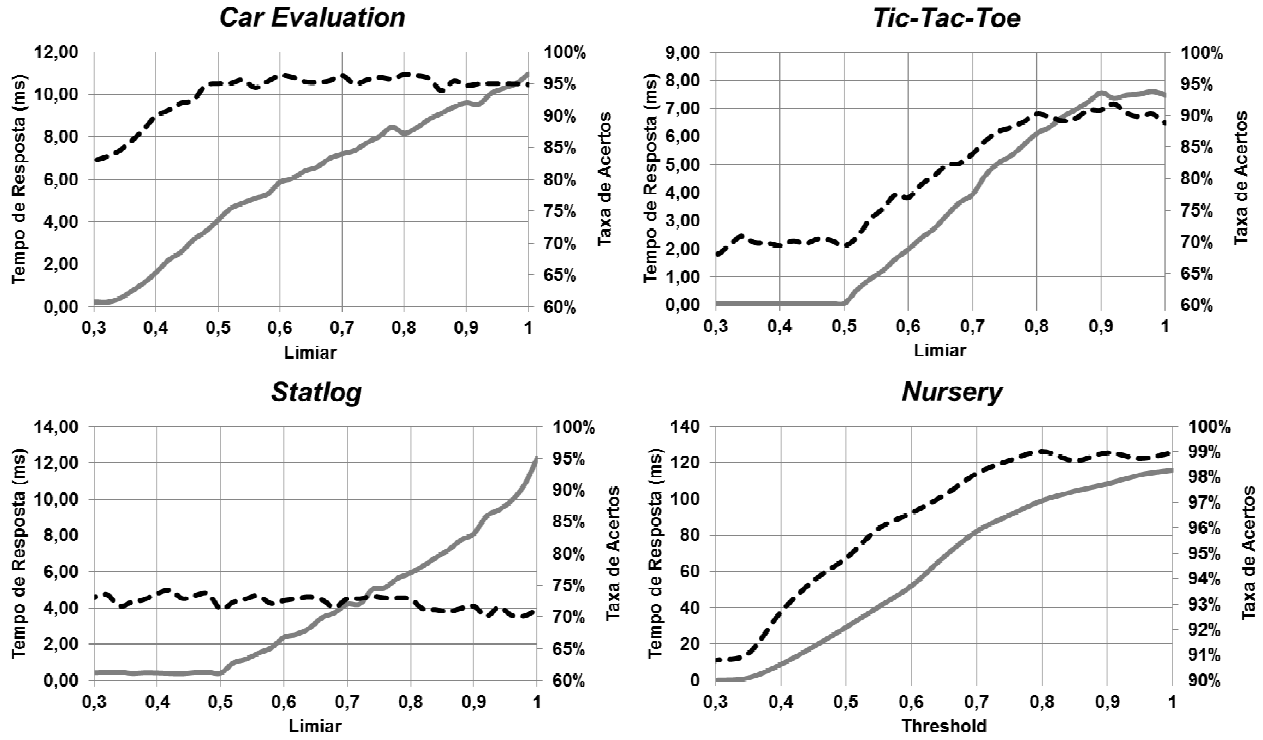


Figura 10: Resultados para o teste de compromisso entre acertos e tempo de resposta. A linha tracejada mostra a taxa de acertos do sistema para um determinado limiar, enquanto a linha em cinza indica, em média, o tempo de resposta utilizado para tomar cada decisão.

mais extensivo do k-NN resultou em resultados piores, não sendo, assim, vantagem utilizá-lo neste problema Deste resultado, pode-se verificar que a solução de um problema – ao invés de ficar a cargo do primeiro dispositivo que conseguir obter uma solução com índice de mérito acima do limiar – poderia ser determinada pela combinação das soluções obtidas por cada dispositivo, em uma tentativa de minimizar os erros via consenso.

VI. Conclusão

A. Contribuições:

Neste trabalho, foi criado um ambiente de tomada de decisão que, aliado às técnicas adaptativas, permite a criação de sistemas híbridos, que não só permitem utilizar um *trade-off* entre taxa de acertos e velocidade, mas também permitem utilizar métodos incrementais para que o sistema evolua de acordo com as entradas e realimentações recebidas.

Os resultados obtidos nestes primeiros experimentos se mostram promissores, permitindo o reuso de métodos clássicos. A junção dos métodos de k-Nearest Neighbor, tabela de decisão e Naive Bayes abrem a possibilidade de seu uso de maneira a tratar incertezas e aprender através de realimentações.

B. Trabalho Futuro:

Um aspecto a ser estudado é a validade temporal de uma decisão, para que seja possível conceber um ambiente capaz de tratar conceitos variáveis ao longo do tempo. Em especial, o processo de "esquecer" conhecimento anterior torna-se atraente para este fim. Também é desejável verificar uma maneira de o sistema verificar a possibilidade de haver novas alternativas – antes desconhecidas – que possam ser atribuídas para casos que não possuem solução definida.

Por fim, é desejável analisar mais uma série de dispositivos de diferentes naturezas, como exemplo, a árvore de decisão incremental [32] e o K* [23].

C. Mais Informações:

Informações relativas ao projeto – e.g. teorias e programas-exemplos – podem ser encontradas na seguinte URL:
http://sites.google.com/site/suzukiresearch/

Referências

[1] R. Crozier and R. Ranyard, "Decision Making: Cognitive Models and Explanations", in *Cognitive Process Models and Explanations of Decision Making*, *Decision Making*: *Cognitive Models and Explanations*, Saffron Walden: Routledge, 1997, pp. 5-20.

[2] D. E. Bell, H. Raiffa and A. Tversky, "Decision Making: Descriptive, Normative, and Prescriptive Interactions", in *Descriptive, Normative, and Prescriptive Interactions in Decision Making*, Cambridge: Cambridge University Press, 1988, pp. 9-30.

[3] B. D. Dunn, T. Dalgleish and A. D. Lawrence, "The Somatic Marker Hypothesis: A Critical Evaluation", Neuroscience and Biobehavioral Reviews, vol. 30, pp. 239–271, 2006.

[4] J. L. Kolodner, "Improving Human Decision Making Through Case-Based Decision Aiding", *AI Mag*, vol. 12, pp. 52-68, 1991.

[5] L. I. Kuncheva, "Using Control Charts for Detecting Concept Change in Streaming Data", Bangor University, UK, 2009.

[6] J. J. Neto, "Um Levantamento da Evolução da Adaptatividade e da Tecnologia Adaptativa", *Revista IEEE América Latina*, vol. 5, (7), pp. 496-505, Nov. 2007.

[7] H. Pistori, "Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações", Ph.D. dissertation, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 2003.

[8] J. J. Neto, "Adaptive Rule-Driven Devices - General Formulation and Case Study", in *CIAA '01: Revised Papers from the 6th International*

Conference on Implementation and Application of Automata, 2001, pp. 234-250.
[9] G. A. Agasandjan, "Automata with a Variable Structure", *Doklady Akademii Nauk SSSR*, vol. 174, pp. 529-530, 1967.
[10] H. Christiansen, "Recognition of Generative Languages", *Programs as Data Objects*, pp. 63-81, 1985.
[11] J. J. Neto, "Contribuições à Metodologia de Construção de Compiladores", Ph.D. dissertation, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 1993.
[12] 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 2000*, 2000, pp. 33-48.
[13] A. H. Tchemra, "Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério", Ph.D. dissertation, Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 2009.
[14] C. E. D. Menezes and J. J. Neto, "Um Método Híbrido para a Construção de Etiquetadores Morfológicos, Aplicado à Líingua Portuguesa, Baseado em Autômatos Adaptativos", in *Anais da Conferencia Iberoamericana em Sistemas, Cibernética e Informática*, 2002, pp. 19-21.
[15] B. A. Basseto and J. J. Neto, "A Stochastic Musical Composer Based on Adaptative Algorithms", in *Proceedings of the 6th Brazilian Symposium on Computer Music - SBC&M99*, 1999, pp. 105-113.
[16] J. Jesus, D. G. Santos, A. A. C. Junior and H. Pistori, "Adaptools 2.0: Aspectos de Implementação e Utilização", *Revista IEEE América Latina*, vol. 5, (7), pp. 527-532, 2007.
[17] H. G. Sol, C. A. T. Takkenberg and P. F. V. Robbé, "Expert Systems and Artificial Intelligence in Decision Support Systems", in *Proceedings of the Second Mini Euroconference*, 1985, pp. 17-20.
[18] R. H. Sprague and E. D. Carlson, *Building Effective Decision Support Systems*, Englewood Cliffs: Prentice Hall Professional Technical Reference, 1982, pp. 30-32.
[19] I. Hendrickx and A. Bosch, "Hybrid Algorithms with Instant-Based Classification", in *Machine learning - ECML 2005*, 2005, pp 158-169.
[20] J. R. Quinlan, "Induction of Decision Trees", *Machine Learning*, vol. 1, (1), pp. 81-106, Mar. 1986.
[21] C. Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning*, vol. 20, (3), pp. 273-297, 1995.
[22] J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5", *Journal of Artificial Intelligence Research*, vol. 4, pp. 77-90, Mar. 1996.
[23] J. G, Cleary and L. E. Trigg, "K*: An Instance-based Learner Using an Entropic Distance Measure", in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 108-114.
[24] Y. Yang and G. I. Webb, "A Comparative Study of Discretization Methods for Naive-Bayes Classifiers", in *Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop*, 2002, pp. 159-173.
[25] J. Dougherty, R. Kohavi and M Sahami, "Supervised and Unsupervised Discretization of Continuous Features", in *Machine Learning: Proceedings of the Twelfth International Conference*, 1995, pp. 194-202.
[26] I. H. Witten and E. Frank, *Data mining: Practical Machine Learning Tools and Techniques*, San Francisco: Morgan Kaufmann, 2005, pp. 393-399.
[27] T. Bui and J. Lee, "An Agent-Based Framework for Building Decision Support Systems", *Decision Support Systems*, vol. 25, (3), pp. 225-237, Apr. 2003.
[28] D. Arnott, "Decision Support Systems Evolution: Framework, Case Study and Research Agenda", *European Journal of Information Systems*, vol. 13, (4), pp. 247-259, Dec. 2004.
[29] D. A. Ross, J. Lim, R. S. Lin and M. H. Yang, "Incremental Learning for Robust Visual Tracking", *International Journal of Computer Vision*, vol. 77, (1-3), pp. 125-141, May 2008.
[30] A. Tsymbal, M. Pechenizkiy, P. Cunningham and S. Puuronen, "Dynamic Integration of Classifiers for Handling Concept Drift", *Information Fusion*, vol. 9, (1), pp. 56-68, Jan. 2008.
[31] D. W. Aha and D. Kibler, "Instance-based Learning Algorithms", in *Machine Learning*, 1999, pp. 37-66.
[32] P. E. Utgoff, N. C. Berkman and J. A. Clouse, "Decision Tree Induction Based on Efficient Tree Restructuring", *Machine Learning*, vol. 29, (1), pp. 5-44, Oct. 1997.
[33] J. L. Kolodner, "Improving Human Decision Making Through Case-Based Decision Aiding", *AI Mag.*, vol. 12, (2), pp. 52-68, 1991.
[34] A. Aamodt and E. Plaza, "Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", *AI Communications*, vol. 7, (1), pp. 39-59, 1994.
[35] B. E. Bakken, "Learning and Transfer of Understanding in Dynamic Decision Environments", Ph.D. dissertation, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA, 1993.
[36] E. Parsloe and M. J. Wray. *Coaching and Mentoring: Practical Methods to Improve Learning*. London: Kogan Page Publishers, 2000, pp. 33-40.
[37] H. J. Einhorn and R. M. Hogarth, "Confidence in Judgment: Persistence of the Illusion of Validity", *Psychological Review*, vol. 85, (5), pp. 395-416, Sep. 1978.
[38] D. Ding, Q. Li and J. Yang, "Multimedia Data Integration and Navigation Through Mediaview: Implementation, Evolution and Utilization", in *Lecture Notes in Computer Science*, 2004, vol. 2973, pp. 263-271.
[39] P. J. Acklam. "An algorithm for computing the inverse normal cumulative distribution function". Internet: home.online.no/~pjacklam/notes/invnorm/, Feb. 27, 2009 [Apr 5, 2011].
[40] L. D. Brown, T. T. Cai and A. DasGupta, "Interval Estimation for a Binomial Proportion", *Statistical Science*, vol. 16, (2), pp. 101-133, 2001.
[41] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, "The R*-Tree: an Efficient and Robust Access Method for Points and Rectangles", in *International Conference on Management of Data*, 1990, pp. 322-331.
[42] W. Liu and S. Chawla, "Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets", in *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 2011, pp. 345-356.
[43] D. Shepard, "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data", in *Proceedings of the 1968 ACM National Conference*, 1968, pp. 517-524.
[44] C. Li and H. Li, "A Survey of Distance Metrics for Nominal Attributes", *Journal of Software*, vol. 5, (11), pp. 1262-1269, 2010.
[45] A. Frank and A. Asuncion, "UCI Machine Learning Repository". Internet: http://archive.ics.uci.edu/ml, Sep. 13, 2011 [Sep. 29, 2011].

Rodrigo Suzuki Okada é graduado em Engenharia Elétrica pela Escola Politécnica da Universidade de São Paulo (USP), Brasil, em 2008. Entre os anos de 2007 a 2008, participou de pesquisas relacionadas à simulação de eventos estocásticos através do Laboratório de Arquitetura e Redes de Computadores (LARC). Desde 2010 é aluno de Mestrado em Engenharia Elétrica, na área de técnicas adaptativa através do Laboratório de Linguagens e Técnicas Adaptativas (LTA) da Universidade de São Paulo (USP). Suas pesquisas se concentram nas técnicas adaptativas voltadas para o aprendizado de máquina, cognição e reconhecimento de padrões.

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 Técnicas Adaptativas 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.

Diseño de un sistema operativo reconfigurable para fines didácticos y prácticos

S. M. Martin, G. E. De Luca, and N. B. Casas

***Abstract*— This paper contemplates the design view of an operating system that allows for managing multiple low-level configurations without the need of recompiling. Most of the common commercial Unix-based and Windows OSs are designed to run with a particular configuration for performance, storage optimization, and/or hardware compatibility. However, in order to prioritize the didactic value of an OS, it can be useful to let the students experiment with different architectural configurations and compare their behavior. Finally, we will use this perspective to describe a design path towards an adaptive architecture OS.**

***Keywords*— Didactic Approach, Operating Systems, Configurable Architecture, Adaptive Devices**

I. Introducción

El diseño de un sistema operativo depende en gran medida de las necesidades específicas del mercado al que se destinará su distribución, del entorno (hardware, peopleware, orgware, y otros componentes software) en el que funcionará, y de los estándares que se deban respetar.

Aspectos como la inicialización de dispositivos, administración de memoria, la planificación de procesos e hilos, la administración de la seguridad de permisos, las interfaces kernel/usuario, las interfaces con los dispositivos, y los sistemas de archivos soportados, pueden contar cada uno con decenas de posibles implementaciones (o modos) diferentes; encontrar la combinación correcta para las necesidades específicas a satisfacer es, por lo tanto, una tarea que requiere un análisis exhaustivo. Una mala decisión en al menos uno de estos aspectos puede causar el fracaso en el producto final [1] [2].

La elección de una de estas combinaciones, es decir, una implementación particular para cada uno de estos aspectos, da por resultado la configuración a bajo nivel del sistema operativo a desarrollar —nos referiremos a esto simplemente como configuración del sistema operativo—.

Desde del punto de vista del proceso de software, esta decisión se toma en la etapa de diseño, y marca el camino a seguir a los programadores para el desarrollo del sistema operativo.

Salvo contadas excepciones [3], no existe necesidad de desarrollar varias alternativas para implementar uno o varios de estos aspectos, ya que esto demandará recursos y tiempo de desarrollo adicional, no solo para cada alternativa, sino también para el mecanismo que permita alternar entre ellas sin necesidad de re-compilar.

Si bien pueden existir configuraciones completamente diferentes entre sistemas operativos para computadores personales, para servidores, de tiempo real, o tolerantes a fallos, ninguno de éstos requiere ser reconfigurable durante su ejecución. Existen, sin embargo, casos en los que se proveen herramientas para mantener una retrocompatibilidad hacia versiones anteriores con diferentes configuraciones [4].

Más allá del punto de vista productivo, el diseño de un sistema operativo que soporte la administración de múltiples configuraciones sin necesidad de recompilar puede resultar un buen recurso para docentes y alumnos para la comprensión de su funcionamiento a bajo nivel.

Un alumno puede experimentar el transcurso de interacciones, acciones en lote, o la ejecución de un proceso, observando diferentes resultados según la configuración que se haya elegido. Los diferentes resultados pueden incluir: falta/sobra de recursos, diferencias de rendimiento, errores y excepciones predecibles, abrazos mortales, entre otros. Esto ayuda al alumno a adquirir una visión general del diseño de sistemas operativos, teniendo en cuenta todas las alternativas en los aspectos que determinan su configuración.

A su vez, el docente tiene la posibilidad de enriquecer sus clases teóricas utilizando casos de prueba preparados para cada las diferentes implementaciones. Por ejemplo, se pueden analizar diferentes planificadores corriendo el mismo lote de procesos de prueba para cada uno con solo cambiar la variable de entorno correspondiente.

Desde el punto de vista didáctico, el factor distintivo del enfoque de diseño planteado en este artículo respecto a utilizar diferentes sistemas operativos de única configuración precompilados es el análisis *ceteris paribus* de los cambios efectuados sobre la configuración; todo, salvo lo que se cambia, se mantiene inalterado, aislando sus consecuencias y, por lo tanto, dando una idea más clara —o menos contaminada— de su funcionamiento.

En este artículo seguiremos la siguiente disposición: en la sección II daremos una introducción al proyecto SODIUM; en la sección III proporcionaremos los definiciones y conceptos que utilizaremos en el resto del trabajo; en la sección IV mostraremos el diseño reconfigurable de administración de memoria aplicadas en SODIUM a modo de ejemplo; en la sección V se analizará el diseño de un sistema operativo autoconfigurable y adaptable; y finalmente, en la sección VI, se encontrarán las conclusiones y un mapa de trabajo para la inclusión de nuevos aspectos hacia una arquitectura reconfigurable unificada en el futuro.

S. M. Martin, Universidad Nacional de La Matanza (UNLaM), Buenos Aires, Argentina, smartin@unlam.edu.ar

G. E. De Luca, Universidad Nacional de La Matanza (UNLaM), Buenos Aires, Argentina, gdeluca@unlam.edu.ar

N. B. Casas, Universidad Nacional de La Matanza (UNLaM), Buenos Aires, Argentina, ncasas@unlam.edu.ar

II. PROYECTO SODIUM

SODIUM (Sistema Operativo del Departamento de Ingeniería de la Universidad de la Matanza) es un proyecto de desarrollo conjunto entre alumnos y profesores de la materia Sistemas Operativos de la carrera de Ingeniería en Informática de la Universidad Nacional de La Matanza de un sistema operativo didáctico.

Se trata de un sistema operativo de multiprogramación basado en la arquitectura Intel IA32. Sumado a esto, las perspectivas a futuro incluyen soportar multiprocesamiento, e instrucciones de 64 bits.

Iniciado en el año 2005, y con el objetivo de darles una oportunidad a los alumnos, no solo de aprender los conceptos teóricos, sino de poder involucrarse de manera práctica del desarrollo de un sistema operativo, SODIUM fue evolucionando a partir de la entrega de trabajos prácticos consistentes en módulos agregables, por ejemplo, un driver de interfaz IDE o el mecanismo de arranque utilizando un pendrive.

A final de cada año, todos los trabajos prácticos son sometidos a diferentes casos de prueba, y los mejores son adaptados e integrados al sistema operativo creando una nueva versión base para el año siguiente.

Uno de los primeros dilemas surgidos de esta metodología de trabajo surgió, predeciblemente, cuando se solicitaron diferentes implementaciones de un aspecto particular del sistema operativo. Por ejemplo, contando con un SODIUM base con un administrador de memoria basado en particiones equitativas, un trabajo práctico destinado al desarrollo de un administrador basado en particiones variables o paginación, debía forzosamente reemplazar al existente. Llevar esto a cabo implicaba una pérdida de valor —didáctico— importante ya que, si bien la administración de particiones variables es superior productivamente [5], se estaría perdiendo la oportunidad de compararlas dentro de la misma compilación de SODIUM. Tampoco serviría utilizar las versiones base de otros años para efectuar la comparación, ya que se perdería el valor *ceteris paribus*, es decir, podrían haber cambiado también otros aspectos del sistema operativo que también hayan cambiado alterando el resultado de la prueba [ver Figura 1].

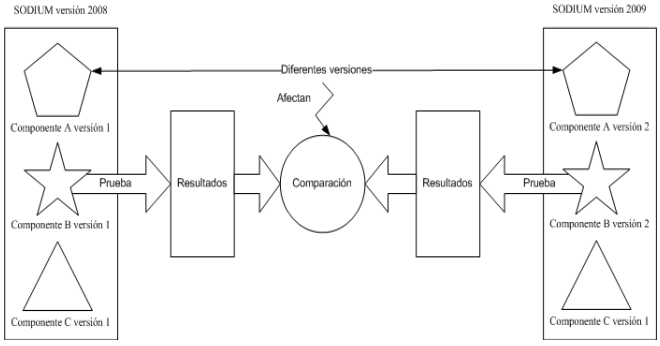


Figura 1. La comparación entre diferentes versiones del componente B es alterada por los cambios en A

Ante esta situación, los docentes de la cátedra tomaron la decisión de adoptar una arquitectura que tienda a maximizar la cantidad de aspectos del sistema operativo que puedan configurarse en tiempo de ejecución —los diferentes momentos en los que puede cambiarse un aspecto se describen en el capítulo III—.

Ambos trabajos prácticos se han integrado, entonces, para permitir que la selección del modo de administración de memoria en SODIUM pueda realizarse mediante la utilización de un menú [6] que aparece luego de la carga del kernel en memoria, y antes de iniciar la interfaz de línea de comandos.

Se han logrado avances sobre otros aspectos del sistema operativo como, por ejemplo, la posibilidad de cambiar el modo de planificación, carga y vinculación de procesos e hilos en memoria.

Sin embargo, hasta el momento todas las integraciones reconfigurables hechas han sido realizadas *ad hoc*, luego de presentarse una implementación nueva de algún aspecto del sistema operativo y exigiendo realizar un trabajo de reingeniería para agregar la nueva alternativa, o nivel de separación conceptual entre ellas.

Debido a esto, surge la necesidad de diseñar y planificar una arquitectura que contemple *a priori* los aspectos que serán reconfigurables, y sus posibles implementaciones candidatas, y utilizarlo como mapa de ruta para futuros desarrollos y consignas de trabajos prácticos para los alumnos de la cátedra.

III. CONCEPTOS DE UN DISEÑO RECONFIGURABLE

A continuación presentaremos conceptos que utilizamos para evaluar el diseño de un sistema operativo reconfigurable.

A. Tiempos para la configuración

Los niveles temporales para la configuración se utilizarán para determinar en qué estado de ejecución del sistema operativo se podrá cambiar el modo de funcionamiento de un aspecto del mismo, y se pueden dividir en los siguientes niveles:

- **Nivel 1 - Configuración en Tiempo de Compilación:**
  En este nivel se sitúan los aspectos que son configurables únicamente vía código y pueden ser cambiados luego de recompilar el sistema operativo modificado, y ejecutando las rutinas de arranque. A diferencia de los siguientes niveles que contarán con todo el código y/o las bibliotecas necesarias para cada modo disponible, el sistema operativo contará con lo necesario para la implementación elegida en particular.

- **Nivel 2 - Configuración en Tiempo de Arranque:**
  En este nivel se sitúan los aspectos que son configurables mediante un menú o archivo de configuración. La configuración se establecerá antes de ceder la ejecución a la consola de línea de comandos o interfaz gráfica, y se define una única vez por sesión. Para cambiar el modo de funcionamiento para el aspecto en cuestión se deberá

reiniciar el sistema operativo y seleccionarlo.

- **Nivel 3 - Configuración en Tiempo de Ejecución Limitada:** En este nivel se sitúan los aspectos que permiten ser configurados de manera diferente para cada proceso, pero que no pueden ser reconfigurados para el mismo una vez que inició su ejecución.

- **Nivel 4 - Configuración en Tiempo de Ejecución Pura:** En este nivel se sitúan los aspectos que permiten ser configurados en cualquier momento, y aprovechados para y por un proceso durante la ejecución del mismo.

El diseño de una arquitectura reconfigurable debería intentar llevar al mayor nivel temporal posible cada una de las transiciones entre modos de los aspectos del sistema operativo dado que menos elementos del contexto se verán afectados por la reconfiguración.

Cualquiera de los niveles citados anteriores, a partir del nivel 2, son además considerados modificables si permiten a un usuario administrador agregar, remover o modificar modos de funcionamiento para el aspecto evaluado. Tal es el caso de muchos sistemas operativos tipo Linux [7] cuyo código de kernel y bibliotecas de usuario pueden ser modificados, recompilados y puestos en marcha, en diferentes niveles temporales, según el aspecto a modificar.

B. Transiciones entre modos en tiempo de ejecución

Llamaremos transiciones a cada cambio entre un modo de funcionamiento a otro diferente de un aspecto del sistema operativo.

Como se ha dicho, estas transiciones deberían minimizar consecuencias secundarias sobre tareas, recursos, o dispositivos que vayan más allá de la naturaleza del cambio.

Por ejemplo, el cambio de un modo de administración de memoria particionado a paginado podría realizarse de manera transparente, sin embargo, no es posible garantizar que el cambio inverso sea compatible; podría haber más espacio asignado a páginas en espacio virtual que no cabrían en un contexto particionado de solo memoria física. Sin embargo, la transición es tomada en cuenta ya que nos permitiría llegar a la conclusión de la gravedad de la problemática presentada.

Es por esto que el diseño debe contemplar también cada una de las transiciones entre modos para cada aspecto, ya que no todos son compatibles entre sí. Esto puede ser representado mediante un grafo dirigido [ver Figura 2] en el cual la presencia de una flecha uniforme indica una transición compatible (transparente), una flecha intermitente indica una transición con posible pérdida de información o desperdicio de recursos, y una flecha intermitente tachada indica una transición incompatible.

Nótese que se han omitido las relaciones transitivas perfectamente compatibles, por ejemplo, pasar de “modo de particiones equitativas” a “modo paginado con swapping”, ya que, al tratarse en definitiva del diseño de un producto software, contemplar todas las transiciones posibles ―en definitiva, casos particulares― puede aumentar las líneas de código, tiempo de desarrollo, y dificulta su mantenimiento.

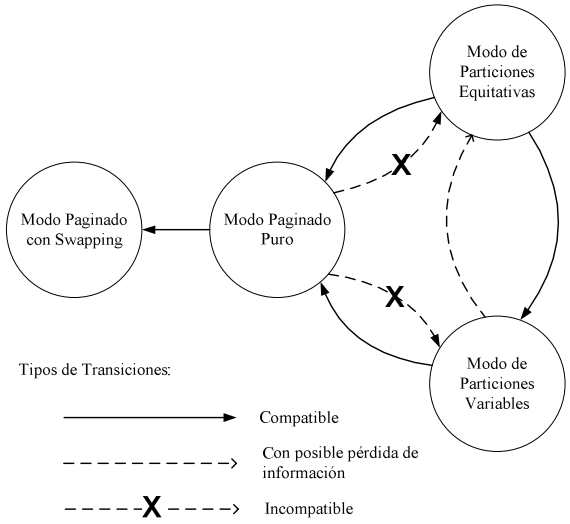


Figura 2. Ejemplo de diagrama de transiciones entre modos de administración de memoria en tiempo de ejecución

Por lo tanto, dada la necesidad de reducir el número de transiciones sin perder caminos posibles, resulta válido eliminar algunas relaciones transitivas y realizar los cambios de manera escalonada.

En el caso del ejemplo, es posible definir una jerarquía de modo que el pasaje de cualquier modo de particiones al modo de paginación pura funciona como primer paso hacia el pasaje a modo paginado con swapping.

Sin embargo, no resulta conveniente eliminar todas las relaciones transitivas ya que algunas representan cambios entre estados de un mismo nivel jerárquico (por ejemplo, entre los modos de particiones equitativas y variables), y no existe un orden lógico que indique desde cuál de ellas debe pasarse al siguiente.

Por un lado, Las transiciones incompatibles o de pérdidas irrecuperables difícilmente permitan superar el nivel temporal 2, por lo que sería válido dejar esas opciones como configurables en tiempo de arranque (no se incluyen en el diagrama). Por otro lado, Las transiciones con posible pérdida de información o desperdicio de recursos pueden llegar a realizarse hasta el nivel temporal 4, sin embargo, deben implementarse con verificaciones que avisen al usuario de los posibles conflictos que puedan surgir del cambio.

C. Facilidad de configuración

Dado que estamos evaluando el diseño de una arquitectura reconfigurable para fines didácticos, las interfaces que permitan el cambio entre implementaciones de los aspectos del sistema operativo deben ser fáciles de comprender y utilizar ya que estas no forman parte del sistema en sí, sino que son el medio por el cual efectuar los cambios.

En los sistemas basados en Unix, la mayor parte de la configuración se realiza a través de archivos de configuración [8] lo cual facilita la creación de scripts que automaticen gran parte del trabajo de administración.

Sin embargo, la utilización de archivos de configuración debería, en el caso de un sistema operativo didáctico, servir únicamente como medio de almacenamiento de la configuración, y no como interfaz para modificarla.

El diseño del sistema operativo didáctico debe contar con interfaces de intuitiva interacción como por ejemplo menús gráficos o variables de entorno para cambiar el modo de funcionamiento de alguno de sus aspectos.

D. Pasos para realizar el diseño

Finalmente, para realizar el diseño deben seguirse los siguientes pasos:

- Analizar la situación actual para cada transición —si es que existe alguna—, incluyendo factibilidad, interfaz por la cual se puede realizar, y nivel temporal.
- Enumerar los modos posibles y efectuar el análisis gráfico de los modos disponibles para descubrir nuevas transiciones posibles durante la ejecución.
- Realizar el diseño reconfigurable indicando detalles de implementación, nivel temporal, y elementos a verificar por posibles pérdidas de información para cada transición.
- Diseñar las interfaces de usuario que mejor se adecuen a la facilidad de configuración del caso.

IV. Ejemplo de Uso: Diseño de un Administrador de Memoria Reconfigurable

Para aplicar este método de diseño sobre el administrador de memoria de SODIUM primero hemos limitado los alcances de este aspecto a la administración del espacio disponible para ubicación de los segmentos de proceso y otros recursos de sistema en memoria ─principal y/o secundaria─, independientemente del modo de funcionamiento (la posibilidad de usar segmentación, el enlazado dinámico, y los modos de carga se verán en la siguiente sección).

Existe sobrada bibliografía sobre los diferentes modos de administración de memoria [9] [10] por lo que nos ocuparemos directamente de evaluar las transiciones entre ellos y los pasos para lograr el mayor nivel temporal para cada caso.

A. Situación actual

Actualmente, SODIUM permite la elección de cualquiera de los modos de memoria en la fig. 2, con un menú interactivo durante el arranque del sistema en el que el usuario puede elegir el deseado utilizando las flechas y la tecla de retorno del teclado.

Luego de efectuada la selección, no es posible realizar un cambio de modo sin reiniciar el sistema, por lo que todas las transiciones son de nivel temporal 2 (no tendría sentido diagramar este caso, ya que no habrían flechas a graficar).

B. Modos y transiciones posibles en tiempo de ejecución

Dado que SODIUM funciona en modo protegido y multiprogramación, no se contempla el uso de un único segmento de memoria, ya que éste solo aplica para sistemas operativos monoprogramados. Los modos disponibles para la administración de memoria en SODIUM son entonces:

- Modo de Particiones Equitativas (PE)
- Modo de Particiones Variables (PV)
- Modo Paginado Puro (PgP)
- Modo Paginado con Swapping (PgS)

El gráfico de transiciones está presentado en la Figura 3, por lo que las transiciones posibles son:

- PE → PV
- PE → PgP
- PV → ~PE
- PV → PgP
- PgP → PgS

C. Diseño reconfigurable

Como hemos indicado en los pasos para realizar el diseño, en este punto analizaremos cada transición para revisar la implementación, nivel temporal y posibles pérdidas para cada transición.

PE → PV: La transición desde el modo de particiones equitativas a la utilización de particiones variables debe realizarse creando tantas particiones variables como equitativas ocupadas existían al momento del cambio, y asignarles el tamaño de estas últimas.

También puede reducirse las particiones al tamaño del proceso que se encuentre en ella indicando fehacientemente el total de la memoria recuperada (de la fragmentación interna).

Una vez efectuada la conversión, tanto el posible espacio no asignado al final como las particiones libres contiguas pueden pasar a ser espacio libre para la asignación de nuevas particiones de tamaño variable [ver Figura 3].

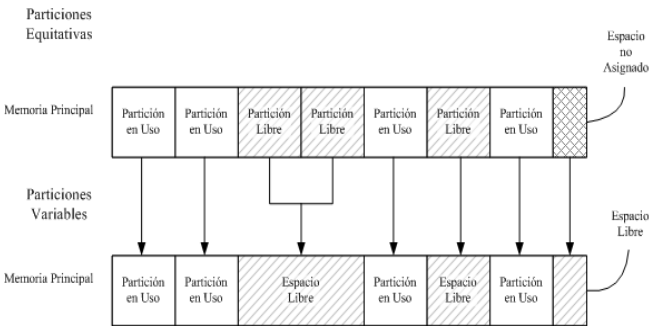


Figura 3. Ejemplo de conversión de modo de particiones equitativas a variables

Podrían existir otras formas de cambiar de modo que, al mismo tiempo eliminen la fragmentación de memoria y resuelvan el espacio libre directamente, pero ésta efectúa un cambio transparente ya que garantiza que las particiones queden en la ubicación física original.

Esta conversión no causa pérdidas de información ni desperdicio de recursos y puede ser realizada en cualquier momento mediante un llamado a sistema, y no afecta al funcionamiento del proceso, por lo que el nivel temporal es 3.

PV → ~PE: Para la transición desde el modo de particiones variables al de particiones equitativas se debe tomar el tamaño de la mayor partición existente como nuevo tamaño fijo

El resultado de este pasaje puede provocar que existan espacios no asignables si el tamaño de la partición más grande no es un divisor natural del tamaño total de la memoria [ver Figura 4].

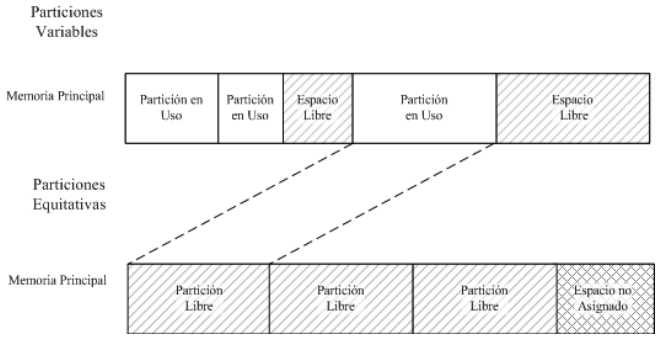


Figura 4. Ejemplo de conversión de modo de particiones variables a equitativas

El número total de particiones final será el resultado entero de la división entre el tamaño total de la memoria disponible por el tamaño fijo obtenido [ver Figura 5].

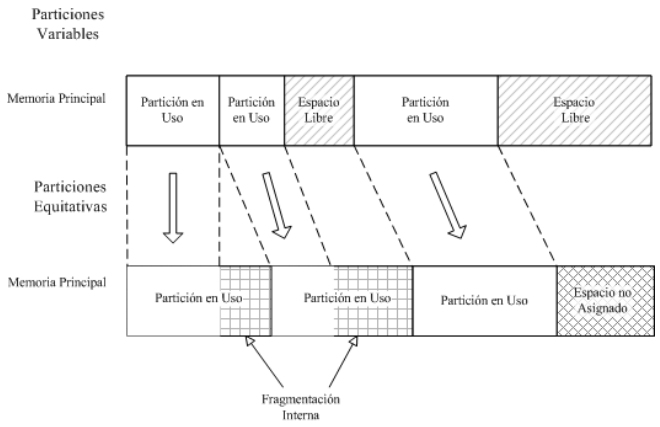


Figura 5. En la asignación de particiones puede producirse fragmentación interna

Como se puede observar en la figura 5, si existieran más de una partición y fueran de distinto tamaño, al efectuar la copia de memoria hacia su partición equitativa de destino, existirá una parte de ella que quedará desperdiciada para todos los casos menos uno, el de la partición más grande.

En el caso de que la cantidad de particiones equitativas resultantes fuera menor al de las particiones variables existentes, se estaría incurriendo en una pérdida de información ya que alguna de las particiones deberá ser descartada.

A pesar de causar un potencial desperdicio de recursos y/o pérdida de información, este cambio se puede implementar con un nivel temporal 3 utilizando mecanismos e interfaces que contemplen todos los casos.

PE → PgP y PV → PgP: El análisis de la transición particiones a modo paginado puro es similar para ambos casos por lo que podremos unificar el análisis tomando particiones equitativas como punto de partida.

Al activar el uso de particiones, cada proceso contará con un espacio virtual similar al espacio físico disponible de memoria principal usado en particiones.

Este espacio virtual es privado de cada proceso por lo que cada uno tendrá solamente una partición que podría conservar la misma ubicación absoluta —esto no es estrictamente necesario—.

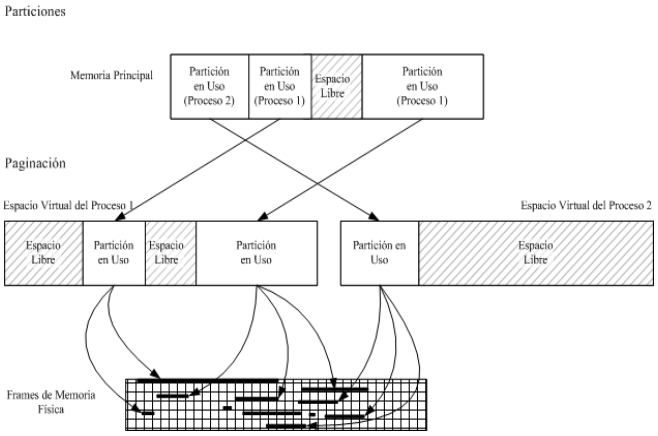


Figura 6. Ejemplo de conversión de modo de particiones variables a equitativas

Las áreas libres en los espacios virtuales de cada proceso no representan un desperdicio, a pesar de ser mayor ahora que cuando se utilizaba particiones, ya que esto no se traduce a un desperdicio a nivel físico; las páginas libres no están aún asignadas a frames —porciones de espacio del mismo tamaño que una página— de memoria [ver Figura 6].

Como la activación de paginación siempre tiende a ampliar el espacio virtual disponible no puede haber pérdida de información en estas transiciones. El espacio físico disponible se conserva, teóricamente, durante la transición aunque podría reducirse dado el tamaño de los directorios de páginas por proceso; sin embargo, si el número de particiones por proceso es muy elevado, podría mantenerse.

La transición podría realizarse en cualquier momento, sin embargo, no será de utilidad sino hasta que se instancie un nuevo proceso, por lo que el máximo nivel temporal es de 3.

PgP → PgS: La activación del uso de swapping en paginación habilita a la ampliación del espacio físico disponible (utilizando memoria secundaria) para la creación de más espacios virtuales, sin embargo, los espacios virtuales se mantienen del mismo tamaño.

Para realizar esto, se deben reemplazar los mecanismos de asignación de páginas para permitir definirlas como *no presentes* en memoria principal, habilitar las excepciones de falta de página, y los mecanismos que leen/escriben los frames correspondientes del soporte externo [ver Figura 7].

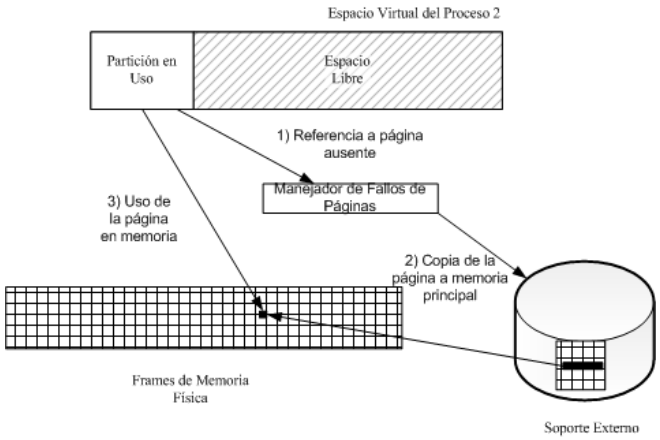


Figura 7. Pasos a implementar para activar paginación con swapping

Esta transición no causa pérdidas de información ni desperdicia recursos y afecta el transcurso de la ejecución de los procesos ampliando la memoria física disponible, y afectando el rendimiento por los accesos a dispositivos físicos por lo que el nivel temporal es 4.

Para el resto de las transiciones se puede mantener el nivel temporal 2 debiéndose reiniciar para realizar el cambio de modo.

Transiciones Incompatibles: El análisis de un cambio de modo de paginación a uno particionado resultó en una pérdida segura de información y en una posterior inestabilidad del sistema en el transcurso del pasaje. Es útil, sin embargo, dejar estas posibilidades contempladas en las interfaces, ya que significa un valor didáctico al alumno provocar la inestabilidad e investigar sus causas.

D. Interfaces para el cambio

Puede mantenerse el menú interactivo durante el proceso de arranque para la elección inicial de cualquiera de los modos. Dado que SODIUM no cuenta aún con una interfaz gráfica, puede implementarse una interfaz de texto auxiliar independiente de la consola de comandos en la que se puede cambiar el modo de memoria y otros aspectos del sistema utilizando los botones de función.

Esta interfaz debe informar y solicitar confirmación en caso de que la transición PV → ~PE pueda implicar pérdidas de información, o desperdicio de recursos, y explicar las causas de esto. Además deberá contemplar las transiciones incompatibles o de nivel temporal igual o menor a 2, indicando la futura inestabilidad, pérdida de datos, las causas que llevan a esto, e indicar fuentes de información a las que el alumno pueda recurrir para entenderlas.

V. Hacia un Sistema Operativo Adaptable

Si bien hasta aquí hemos pensado el diseño de un sistema operativo cuyos aspectos puedan ser cambiados en tiempo de ejecución de manera manual, esta metodología abre las puertas a que la configuración pueda ser realizada automáticamente, dependiendo de las necesidades de espacio, procesamiento, e interacción con los dispositivos conectados.

Las ventajas de un sistema operativo autoconfigurable es que en cada momento puede aprovechar el mejor rendimiento de aquellos modos de un aspecto que son menos compatibles pero de mayor rendimiento, y cambiar de modo si, habiendo variado la demanda del uso del mismo, se precisa utilizar un modo más compatible.

Dado que para cada modo diferente de un aspecto se precisan que ciertas condiciones sean cumplidas, y que se ejecute una rutina al cambiarse de modo, hemos considerado que las herramientas de análisis que mejor sirven para describir este comportamiento son las *tablas de decisión* [11].

A. Usando una tabla de decisión estática

En el caso del ejemplo, podemos construir un autómata finito a partir del diagrama de transiciones utilizado para realizar el diseño reconfigurable [ver Figura 2] cuyos estados sean cada uno de los posibles modos del aspecto en cuestión a analizar, en este caso, de la administración de memoria.

Deberemos luego definir las condiciones necesarias a cumplir para cada uno de los modos recorriendo inversamente las transiciones incompatibles partiendo del modo más compatible (**PgS**) en dirección inversa hacia los nodos menos compatibles (**PE** y **PV**).

La necesidad de retornar a los modos menos compatibles es que, tanto en el caso de la administración de memoria como en otros aspectos del sistema operativo, la compatibilidad implica complejidades que pueden afectar al rendimiento, o a la facilidad de administración.

Las transiciones entre los estados deberán contemplar los siguientes aspectos:

- Las condiciones necesarias que, dado un cambio en el contexto, pasen a (o dejen de) ser cumplidas, para realizar la transición.
- Las acciones necesarias para pasar de un modo a otro representarán un mensaje emitido.
- Por simplicidad en el ejemplo, los datos de entrada (input) serán únicamente los sucesos de proceso saliente **PrS** (liberación de memoria), y proceso entrante **PrE** (requerimiento de memoria).

Nótese que en el autómata resultante [ver Figura 8], puede darse el caso en que las transiciones que, al realizarse de modo manual —dada la incapacidad del usuario de tener en cuenta todas las variables internas del sistema—, eran de nivel 2 ahora pueden ser realizados en niveles 3 o 4, dado que el sistema puede detectar automáticamente las condiciones específicas en las que no se incurrirá en pérdida de datos.

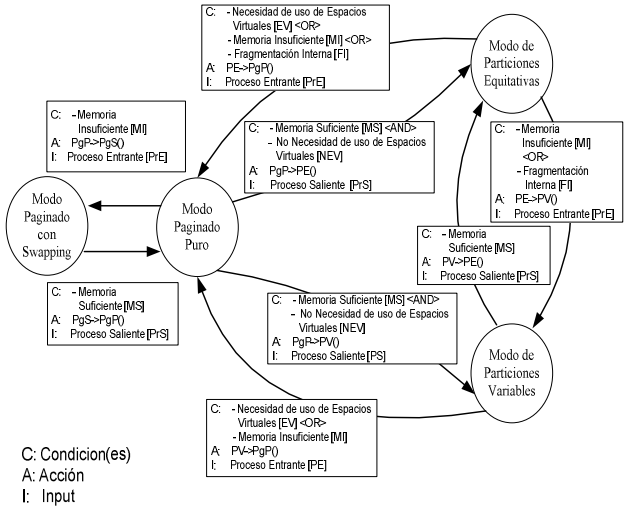


Figura 8. Autómata para cambios automáticos de modo de administración de memoria

Nótese también que, por lo general, las transiciones paralelas y opuestas, es decir, que van y vienen de los mismos dos estados, en sentido contrario, cuentan, en el mismo sentido, con condiciones opuestas. Por ejemplo, aquellas transiciones que se den por falta de memoria, podrán regresar al estado anterior solo si existe memoria suficiente.

Si bien los inputs y estados son también condiciones, la separación hecha responde a agregarle claridad a la aplicación de estas tablas a la visión de sistemas operativos.

El aporte didáctico de experimentar con una configuración automática será de mucha utilidad al indicar mediante mensajes a pantalla el preciso instante en el que se ha efectuado una transición y las condiciones que la propiciaron.

La decisión sobre cuál será la regla inicial estará dada por una elección de nivel temporal 2, ya sea por elección manual o archivo de configuración. En la práctica no existirá más estado final que aquel en el que se encuentre el sistema al apagarlo.

En la tabla de decisión obtenida [ver Figura 9], puede observarse que para aquellas transiciones con condiciones conectadas por disyunción lógica (OR), existen reglas con la misma acción; esto no significa que se produzca un indeterminismo, sino que diferentes causas pueden provocar la acción.

El indeterminismo sí se produce en las reglas que devuelven de **PgP** a **PV** o **PE** dado que el mismo conjunto de condiciones pueden llevar a efectuar cualquiera de las acciones, o en el caso de estar en **PE** y se de la condición de memoria insuficiente ante un proceso entrante.

En estos casos, se pueden probar las n-acciones posibles y evaluar si alguna de ellas sigue sin satisfacer las necesidades de memoria del proceso entrante.

Regla N°		1	2	3	4	5	6	7	8	9	10	11	12
Condiciones	MI	x				x					x	x	
	MS			x	x				x				x
	FI		x				x						
	EV							x		x			
	NEV				x				x				
Inputs	PrE	x	x			x	x	x		x	x		x
	PrS			x	x				x			x	
Estado	PE	x	x			x	x	x					
	PV			x						x			
	PgP				x				x				x
	PgS										x	x	
Acciones	PE->PV()	x	x										
	PV->PE()			x									
	PgP->PE()				x								
	PE->PgP()					x	x	x					
	PgP->PV()								x				
	PV->PgP()									x			
	PgS->PgP()										x	x	
	PgP->PgS()												x

Figura 9. Tabla de decisión estática para el ejemplo simplificado de administración de modos de memoria[1]

B. Usando una tabla de decisión adaptable

El caso del ejemplo, con cuatro estados y sus transiciones, es una simplificación a fines prácticos de presentar el método propuesto, de todo el mapa posible de estados y transiciones posibles para los modos de administración de memoria que puede haber, y habrá en SODIUM.

El problema de utilizar una tabla de decisión estática es que se parte de la hipótesis de que existe la(s) decisión(es) tomadas para cada combinación de condiciones serán siempre las más convenientes y, por lo tanto, se pueden definir *a priori*.

Sin embargo, durante la ejecución de un sistema operativo, pueden darse distintas opciones posibles de cambio de modo para un aspecto, o conjuntos de cambios de diferentes aspectos a la vez que den respuesta a una necesidad única y particular planteada por el propio uso del mismo.

La complejidad de un mapa de transiciones de modos resultante tal hace que sea no solo impracticable sino también ineficiente contemplar todas las combinaciones de condiciones-input que puedan darse.

Una alternativa que puede dar solución a un sistema de tal complejidad es el uso de las *tablas de decisión adaptables* [11] [12]. Tal como se describen en [11], las tablas de decisión adaptables son una extensión de las tablas de decisión convencionales con el agregado de meta-acciones —similares en el sentido estricto a las acciones convencionales— que pueden ser ejecutadas antes o después de las existentes, y cuyo fin es modificar, es decir, agregar, modificar, o eliminar reglas, la misma tabla para adaptar el comportamiento del

[1] MI – Memoria Insuficiente; MS – Memoria Suficiente; FI – Fragmentación Interna; EV – Necesidad de Espacios Virtuales; NEV – No Necesidad de Espacios Virtuales

sistema en base a los eventos que vayan ocurriendo y el contexto dado.

La elaboración de la tabla de decisión para un sistema operativo adaptable —y SODIUM, en el futuro— contará con las siguientes premisas:

- Pueden definirse acciones por defecto a tomar en cada regla mediante el método definido en la Sección III, sin evaluar casos específicos, excepciones, ni relaciones con otros aspectos del sistema operativo.
- Se definirán acciones adaptativas de ejecución previa que evalúen la compatibilidad de las reglas de transición actuales, elimine aquellas que no sean compatibles con el estado actual del sistema, y agregue aquellos estados que sí lo son.
- Otra acción adaptativa de ejecución previa determinará en base a un sistema de métricas la acción adecuada en cada momento ante cada evento de input.
- Se definirán acciones adaptativas de ejecución posterior para la toma de métricas sobre transición posible (rendimiento de ejecución, costo de transición, frecuencia de cambio, frecuencia de compatibilidad, etc.).
- El estado de la tabla se guarda antes de apagar el sistema y se toma como tabla inicial en la siguiente sesión.

El esquema de métricas para la determinación de las mejores reglas es un campo que merece la pena ser evaluado en posteriores estudios en el camino hacia un sistema operativo adaptable.

Las premisas para la creación de la tabla de decisión adaptable determinarán la capacidad de aprendizaje del sistema operativo para determinar cuáles serán las mejores transiciones en función de cómo es usado. Esto implicará que dos usuarios con necesidades de uso diferentes contarán, luego de un lapso de uso razonable, con una configuración del sistema operativo personalizada que se adapta de manera óptima y diferente para cada caso.

VI. Conclusión

Mediante el método expuesto, hemos podido contemplar el diseño reconfigurable de un aspecto de un sistema operativo didáctico que hasta el día de hoy, si bien puede ser cambiado luego de reiniciar el sistema, no permite ser cambiado mientras el sistema se encuentra en ejecución.

Este enfoque nos permitirá efectuar el diseño de la reconfigurable para el resto de todos los aspectos de SODIUM que aún permanecen definidos por compilación y que serían de mayor valor didáctico si pudieran cambiarse durante la ejecución.

El administrador de memoria ha sido el primer aspecto reconfigurable en el que hemos aplicado esta metodología, obteniendo como resultado una mayor comprensión *a priori* Mejorando el entendimiento —tanto de docentes como de alumnos— de los mecanismos e interfaces respecto al enfoque *ah hoc* utilizado otrora.

La combinación de este enfoque, con las tecnologías de dispositivos adaptables existentes nos acerca la posibilidad de obtener un sistema operativo que responda de manera diferente acorde a las necesidades de uso de distintos usuarios.

El potencial práctico y didáctico del desarrollo por parte de profesores y alumnos de un sistema operativo adaptable es, de por sí, motivo suficiente para continuar tanto con la investigación en este sentido, como con la aplicación práctica sobre SODIUM de los avances logrados.

Referencias

[1] J. C. Dvorak, *Vista's 11 Pillars of Failure*, PCMag.com, 2008.

[2] Steven J. Vaughan-Nichols, "The 10 Worst Operating Systems of All Time", *PC World Magazine*, 2009.

[3] A. C. Veitch; N. C. Hutchinson, "Kea-a dynamically extensible and configurable operating system kernel", Configurable Distributed Systems, 1996. Proceedings., Third International Conference on.

[4] Backwards Compatibility – Microsoft Versus, http://www.msversus.org/backwards-compatibility.html

[5] W. Stallings, *Sistemas Operativos*, Capítulo 7: "Gestión de Memoria", Prentice Hall, 4ª Edición, 2001 .

[6] N. Casas; G. De Luca; S. Martin; H. Ryckeboer; M. Cortina; G. Puyo; W. Valiente, "Algoritmo de Administración de Memoria en un Sistema Operativo Didáctico", *Anuario de Investigaciones, Resúmenes extendidos 2009*, Universidad Nacional de La Matanza, ISBN: 978-987-1635-24-8.

[7] Kurt Wall et ál, *Programación en Linux Al descubierto*, Prentice Hall, 2ª Edición, 2001.

[8] A. S. Tanenbaum, *Sistemas Operativos Modernos*, Prentice Hall, 1ª Edición, 1993.

[9] Mark E. Russinovich; D. A. Solomon, *Microsoft Windows Internals*, Microsoff Press, Fourth Edition, 2005, págs. 251-255 y 615-654.

[10] A. S. Tanenbaum; A. S. Woodhull, *Operating Systems, Design and Implementation*, Prentice Hall, 2ª Edición, 1997.

[11] J. J. Neto, "Adaptive Rule-Driven Devices - General Formulation and Case Study" Lecture Notes in Computer Science, 2002, Volume 2494/2002. Springer

[12] T. Pedrazzi, A. Tchemra, R. Rocha "Adaptive Decision Tables A Case Study of their Application to Decision-Taking Problems" Adaptive and Natural Computing Algorithms 2005, Part III. Springer.

Sergio Miguel Martin se graduó en la carrera de Ingeniería en Informática en la Universidad Nacional de La Matanza (UNLaM) de Buenos Aires, Argentina el año 2010. Es docente ayudante de las materias de Sistemas Operativos desde el 2010; de Autómatas y Lenguajes Formales desde el 2011; y de Métricas de Software desde el 2011. Actualmente se encuentra cursando la Maestría en Ingeniería en Informática dictada en la Escuela de Posgrado de UNLaM. Sus campos de investigación se centran en el diseño y la enseñanza de Sistemas Operativos.

Graciela Elisabeth De Luca nació en la ciudad de Buenos Aires, Argentina, 13 de Junio de 1959, título Analista Universitaria de Sistemas de La Universidad Tecnológica Nacional y Licenciada en Informática de la Universidad Católica de Salta. Desde el año 2005 pertenece al grupo de Investigación de la Universidad Nacional de la Matanza, en el área de Arquitectura y Sistemas Operativos. Trabajo de tesis en curso de la maestría en Informática de la Universidad Nacional de la Matanza.

Nicanor Blas Casas, nació en la ciudad de Buenos Aires, Argentina. Título Analista Universitario de Sistemas de la Universidad Tecnológica Nacional, Ingeniero en Informática de la universidad Católica de Salta. Desde el año 2005 pertenece al grupo de Investigaciónde la Universidad Nacional de la Matanza, en el área de Arquitectura y Sistemas Operativos. Trabajo de tesis en curso de la maestría en Informática de la Universidad Nacional de la Matanza.

Implementação do Jogo Pedra-Papel-Tesoura Aplicando Tecnologia Adaptativa

F. S. Santana and D. C. Passos

***Abstract*— The paper-scissor-rock, PSR, game has been applied to develop models to analyze evolutionary computing, biodiversity, and decision making in several research areas. PSR mathematical approach establishes the random solution as the best winning strategy to this game. However, recent studies analyzing players and results of PSR championships indicate new possibilities based on human behavior patterns. This paper describes the implementation of a software solution based on one of these strategies, applying adaptive technology and state machines. The results of this work allow the evaluation of the presented solution and other interesting studies in human behavior patterns and adaptive technology application in the development of modeling methods.**

***Keywords*— Game Theory, Adaptive Technology, Paper-scissor-rock.**

I. INTRODUÇÃO

O JOGO pedra-papel-tesoura, PPT, também chamado de "jan-ken-pon", é usualmente considerado uma brincadeira de crianças. Em alguns casos, o PPT também costuma ser aplicado na tomada de decisão em conflitos sem importância, que em teoria são decididos pelo fator "sorte" [5], de forma que o resultado do jogo faz com que não exista um responsável direto pela decisão tomada.

No entanto, com a aplicação de resultados de pesquisa em teoria dos jogos em outras áreas do conhecimento, o PPT vem ganhando importância na medida em que tem sido aplicado na criação de modelos para a abstração de diversos problemas relevantes [5]. Entre eles, destacam-se a computação evolucionária [3], os sistemas de aprendizado [6], as soluções envolvendo a criação de modelos para tratar a biodiversidade [1] e aplicações de modelos quantitativos para a tomada de decisão em economia [5] e neurociência [4].

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/]), entre outras, registram jogadores profissionais e promovem competições periódicas deste jogo, com a distribuição de consideráveis prêmios em dinheiro aos vencedores. Nessas competições, embora os resultados matemáticos afirmem que a melhor estratégia para se vencer o jogo de PPT é sempre jogar aleatoriamente [5], é natural que os jogadores profissionais busquem constantemente por estratégias para aumentar as probabilidades de vencer este jogo.

A estratégia não aleatória para se vencer o PPT proposta por [9,10], da Sociedade Internacional de PPT, baseia-se na hipótese de que o comportamento humano é afetado por aspectos de personalidade e por aspectos emocionais e, portanto, as decisões tomadas não são puramente aleatórias, obedecendo a padrões pré-determinados de comportamento. Segundo [9,10], estes padrões permitem definir técnicas capazes de aumentar as probabilidades de vitória no PPT.

Esta estratégia foi modelada em [7] na forma de uma máquina de estados, à qual foi incorporada uma tabela de decisão adaptativa. A tabela de decisão adaptativa foi utilizada para redefinir o comportamento da máquina de estados a partir do perfil corrente do jogador oponente. Este perfil deve ser reavaliado antes de cada jogada, considerando o contexto do jogo, o potencial estado emocional do jogador oponente e o histórico de jogadas.

Este trabalho apresenta a implementação da solução apresentada em [7], seguido de um pequeno experimento para apresentar a funcionalidade do sistema resultante.

Os resultados aqui obtidos permitem a realização de experimentos sucessivos para avaliar a eficácia da proposta do ponto de vista prático, a partir da análise estatística dos resultados obtidos. Após a realização de uma quantidade significativa de experimentos, a comparação entre os resultados dos jogos com os resultados que seriam obtidos se a estratégia puramente aleatória estivesse sendo aplicada, deve indicar se desvios probabilísticos importantes ocorreram. Estes desvios, por sua vez, poderiam indicar uma potencial eficácia probabilística da estratégia.

Além disso, a análise computacional do histórico de jogadas pode levar à descoberta de outros padrões não aleatórios que não foram previstos originalmente por [9,10]. Estes padrões, se existirem, podem ser incorporados à solução proposta por [7], tornando-as mais eficaz. Desta forma, será possível construir modelos mais confiáveis do que os disponíveis atualmente para a tomada de decisão baseados no jogo de PPT.

É importante notar que, caso se comprove a existência de uma estratégia adaptativa para vencer o jogo de PPT mais eficaz do que a solução aleatória, este resultado representará um estudo de caso onde se mostra que é possível tratar padrões de comportamento humano usando tecnologia adaptativa.

Finalmente, deve-se notar que este raciocínio potencialmente poderá ser estendido para o estudo de outros problemas em teoria dos jogos.

Fabiana Soares Santana, Universidade Federal do ABC, Santo André, São Paulo, Brasil, fabiana.santana@gmail.com.

Deborah Cristina Passos, Universidade Federal do ABC, Santo André, São Paulo, Brasil, deborah.cristina.passos@gmail.com.

II. MATERIAL E MÉTODOS

O PPT se joga com as mãos. Os gestos descritos nas Figuras 1, 2 e 3 representam, respectivamente, pedra, papel e tesoura.

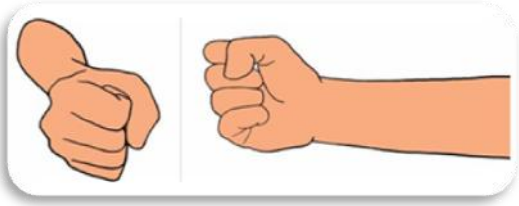


Figura 1 – Representação manual de pedra (extraída de http://worldrps.com/game-basics).

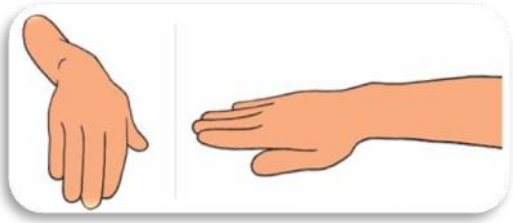


Figura 2 – Representação manual de papel (extraída de http://worldrps.com/game-basics).

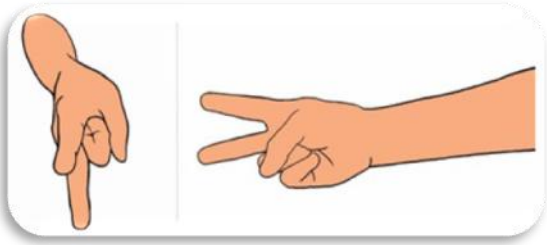


Figura 3 – Representação manual de tesoura (extraída de http://worldrps.com/game-basics).

As regras do jogo de PPT adotadas nas competições internacionais são as seguintes [5]:

- 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.

Figura 4 – As regras básicas do jogo: "pedra" vence "tesoura"; "tesoura" vence "papel" e "papel" vence "pedra".

- Os jogadores devem combinar previamente um determinado número de rodadas (que também pode ser pré-estabelecido em uma competição). Para este trabalho, este número será definido como *n*, sendo *n* um número inteiro, finito e ≥ 1.

[10] defende que PPT não é baseado no fator sorte, pois o ser humano não é capaz de jogar de maneira puramente aleatória. O seu comportando torna-se previsível ao longo do tempo e passa a obedecer a padrões pré-determinados. Aspectos de personalidade e de caráter psicológico definem as ações do jogador no PPT, principalmente no caso de jogadores iniciantes ou que estejam em desvantagem. Portanto, o conhecimento destes padrões permitiria alterar a distribuição de probabilidade entre as jogadas possíveis, aumentando consideravelmente as chances de vencer o jogo [9,10]. Naturalmente, este raciocínio só se aplicaria em rodadas sucessivas de PPT, como ocorre nas competições internacionais [http://worldrps.com/game-basics], pois uma simples jogada não seria suficiente para se estabelecer um padrão de jogo.

Os aspectos implementáveis da estratégia proposta por [9,10] são os seguintes:

1. Homens, principalmente iniciantes, têm a tendência de escolher "pedra" na 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 e iniciante, 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 para "tesoura", ou "tesoura", caso em que ocorre um empate.
3. Quando o adversário for 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 [9,10], 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.

[9,10] também supõe que o comportamento entre homens e mulheres é diferente. Este raciocínio, na verdade, é suportado por outros pesquisadores em biodiversidade. [8] 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.

Os demais aspectos da estratégia proposta por [9,10] consideram a análise histórica do perfil de um jogador.

A solução proposta por [7] foi definida a partir destas hipóteses, sendo que algumas decisões devem ser tomadas de acordo com uma tabela de decisões adaptativa. Esta solução ainda classifica os jogadores em três níveis: iniciante, médio e experiente, que traduzem o comportamento e as possibilidades de vitória de um jogador. Na primeira vez, o jogador será considerado como iniciante. Seu desempenho definirá a possibilidade de ele ser promovido para os níveis superiores, sendo que a reclassificação deve ser feita no início de cada jogada.

III. Implementação do PPT

A solução proposta em [7] considera que existem sempre dois jogadores. Um dos jogadores é a própria solução de software, chamada de *Jogador*. O outro jogador deve ser obrigatoriamente um ser humano, chamado de *Oponente*.

A solução sugere a aplicação de uma tabela de decisão simples para definir o vencedor de cada jogada, como mostra a Tabela 1. Na tabela, 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.

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**

Da mesma forma que ocorre nas competições oficiais, um jogo deve ser composto por *n* jogadas. Esta solução prevê o armazenamento das jogadas em uma estrutura de dados adequada, a fim de permitir a análise do histórico de jogadas.

A estrutura de dados utilizada para armazenar as jogadas é apresentada na Figura 5. São registradas as jogadas do Jogador e do Oponente e, ao término de cada jogada, é calculada a porcentagem de vezes em que cada jogador escolheu "pedra", "papel" ou "tesoura". Nesta implementação, a mesma estrutura foi definida também para um terceiro jogador, chamado de *Jogador Aleatório*, para o qual os resultados são sorteados a cada instante. Embora o Jogador Aleatório não participe do jogo, seus resultados serão utilizados para fazer a comparação entre a estratégia proposta e a solução aleatória.

Como o sexo do Oponente pode interferir na estratégia, um formulário inicial é apresentado ao usuário antes do início do jogo, onde esta informação é solicitada. A solução implementa o controle de usuários a fim de melhor analisar o seu perfil, mesmo que o jogo seja interrompido e retomado em outra ocasião. Se o jogador preferir não se identificar, será considerado como iniciante.

A máquina de estados da solução proposta por [7] está apresentada nas Figuras 6, 7, 8 e 9. A Figura 6 apresenta o início de cada *k*-ésima jogada, $1 \leq k \leq n$, onde o Oponente pode ser reclassificado de acordo com a sua pontuação. A Figura 7 apresenta o tratamento para Oponente iniciante, a Figura 8 apresenta o tratamento para o Oponente de nível médio e a Figura 9 apresenta o tratamento para o Oponente experiente. A reclassificação é necessária, pois interfere nas chances de vitória.

A cada *k-ésima* jogada, a máquina de estados determina a próxima jogada do Jogador. A pontuação é utilizada para redefinir o nível do Oponente e, conforme ele obtém resultados positivos, ele é considerado como um jogador de melhor nível, aqui chamado de Experiente. A pontuação é definida de modo simples, onde cada vitória vale um ponto.

A análise histórica do Oponente é realizada através da implementação de uma tabela de decisão adaptativa, que permite alterar o comportamento do Jogador de acordo com o padrão de comportamento do Oponente. A definição do perfil permite que a máquina de estados decida o que fazer, a fim de aumentar as chances de se obter, no mínimo, um empate.

Por exemplo, a *Condição1* pode ser "Oponente perde jogada *k-1*". A perda aumentaria o nível de frustração do Oponente e uma reação comum, especialmente em jogadores iniciantes ou estressados, seria escolher a opção que teria vencido a jogada *k-1* (Walker, 2006). Portanto, *Ação1* deve ser "jogar perfil que teria perdido a rodada *k-1*". Porém, se a tendência do Oponente de escolher determinada jogada ultrapassar os 50%, deve-se alterar a estratégia e jogar simplesmente de forma a vencer esta tendência. Isto deve ser traduzido por uma função adaptativa F_1 capaz de alterar *Ação1* para uma nova ação, capaz de traduzir a nova estratégia.

Após *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.

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.

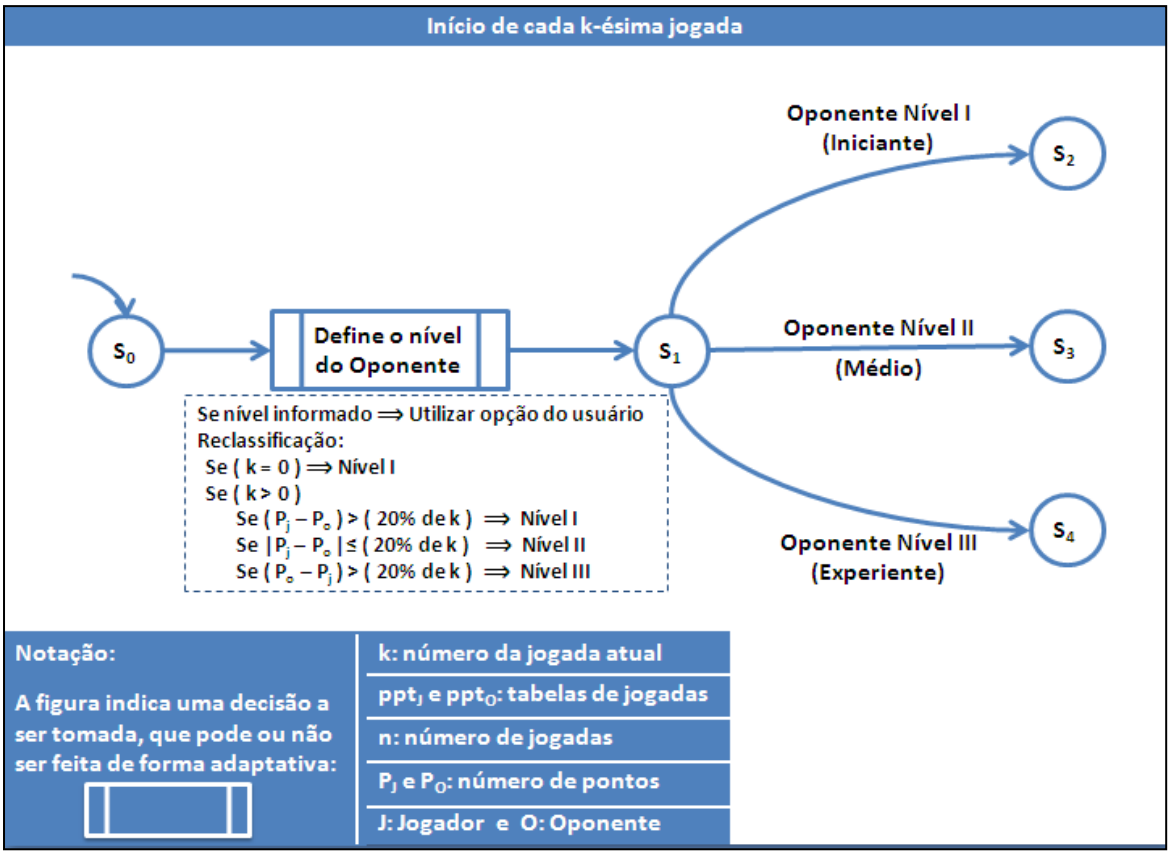


Figura 6 – Máquina de estados: início de cada *k*-ésima jogada.

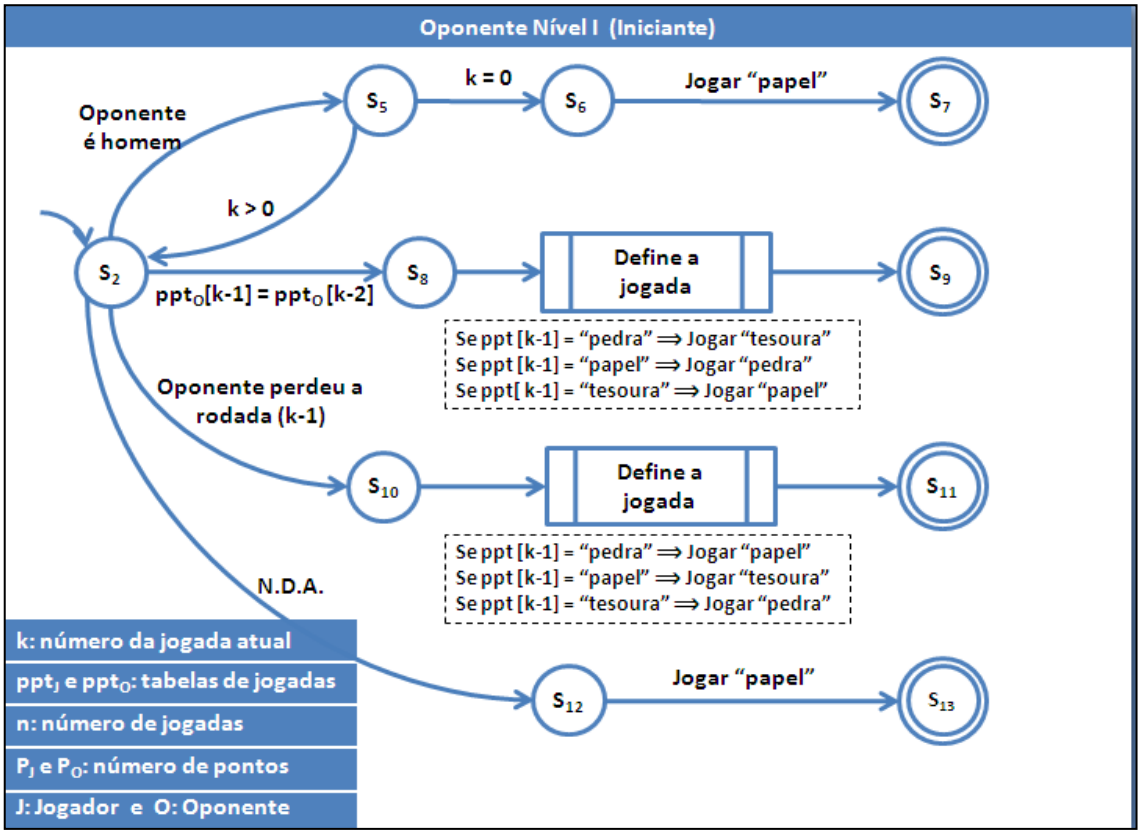


Figura 7 – Máquina de estados: tratamento para o Oponente iniciante.

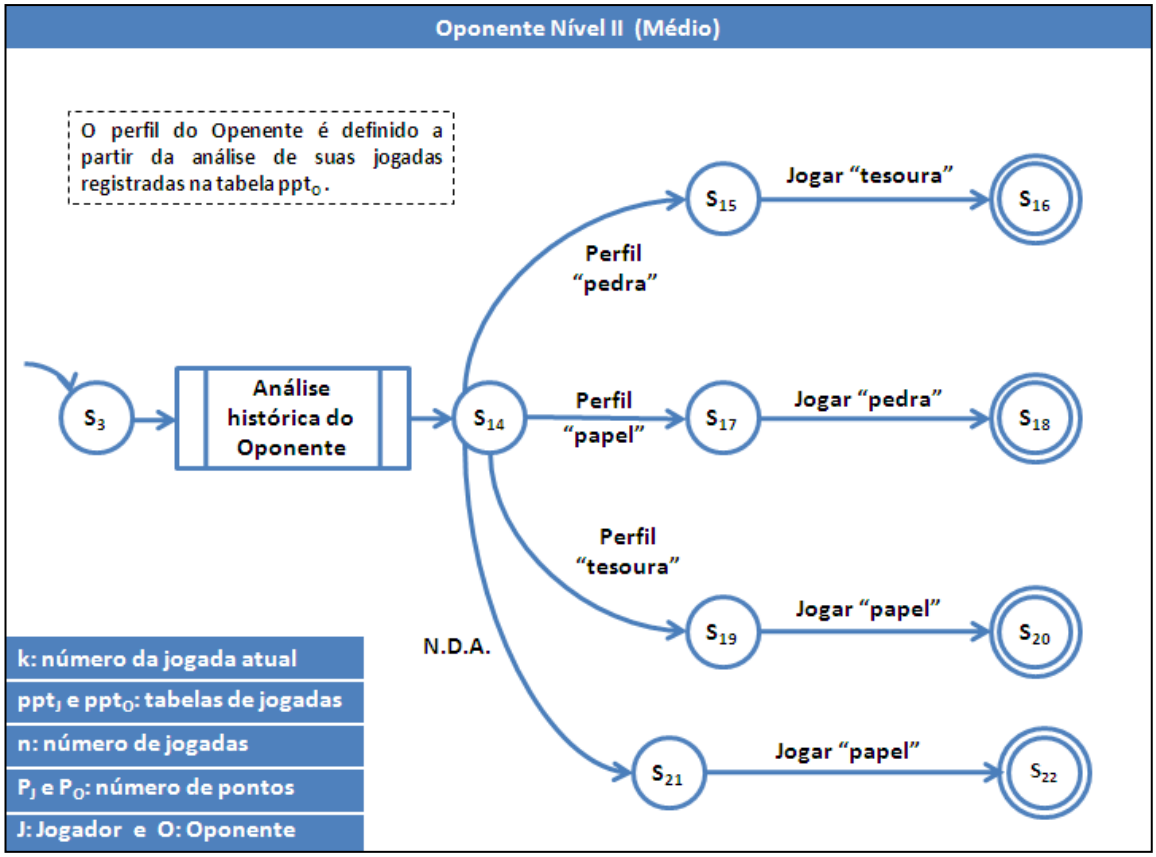


Figura 8 – Máquina de estados: tratamento para o Oponente de nível médio.

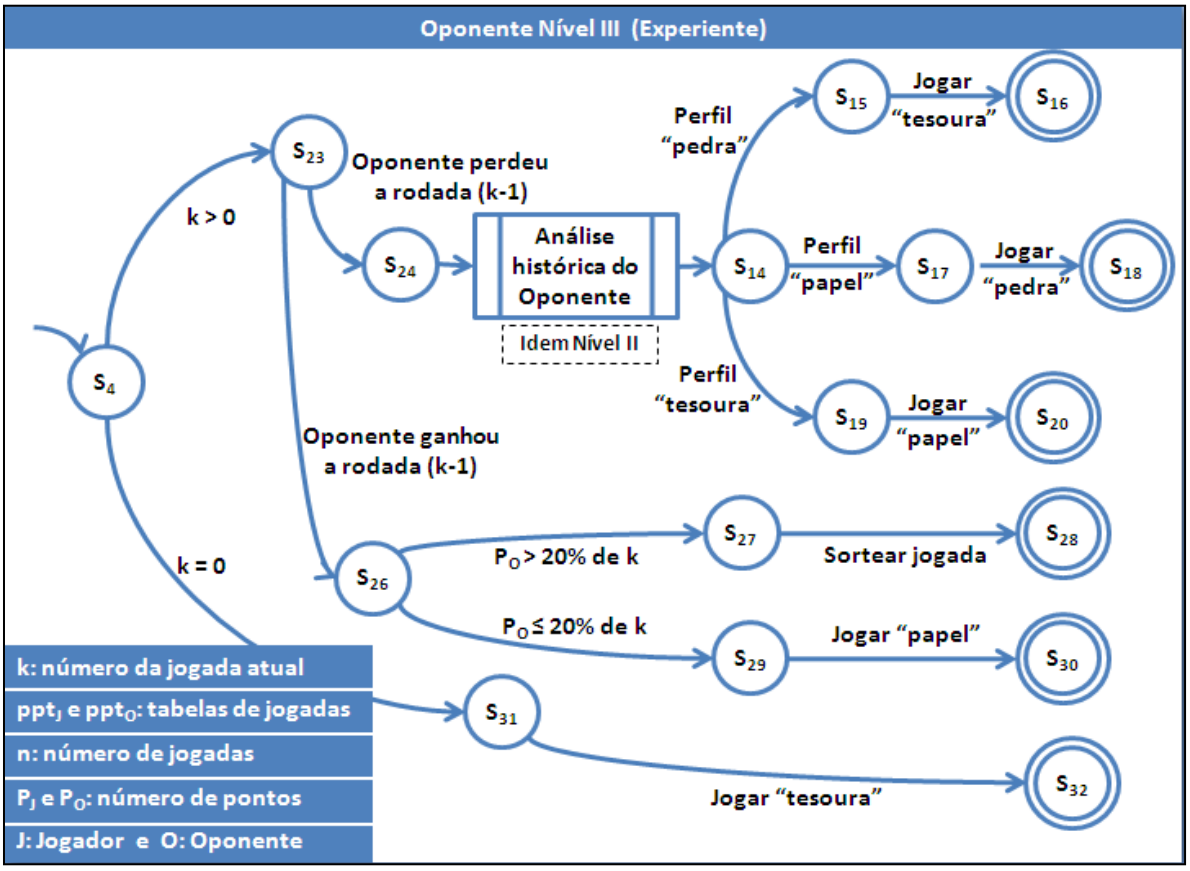


Figura 9 – Máquina de estados: tratamento para o Oponente experiente.

IV. Resultados

As figuras a seguir mostram a implementação da versão preliminar. A Figura 10 apresenta um formulário que solicita o nome, idade, sexo e nível do Oponente.

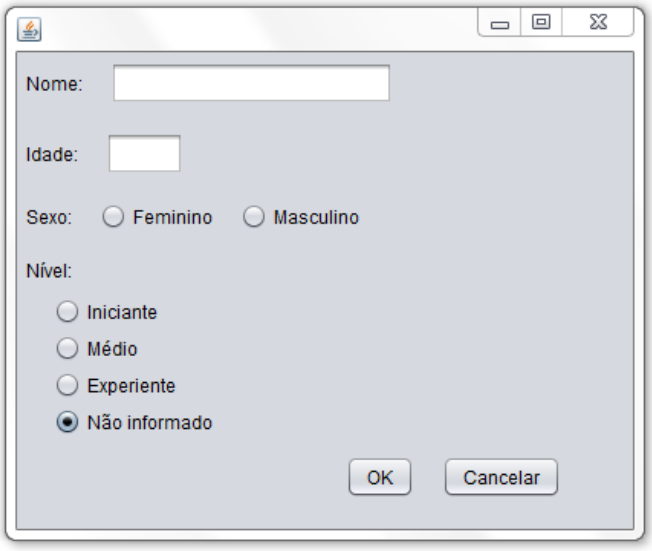


Figura 10 – Definição dos dados do Oponente.

O Oponente não necessita informar os atributos nome e idade obrigatoriamente. Os dados de idade têm a função de analisar estatisticamente os jogadores por idade, enquanto o nome permite salvar os dados do Oponente, de modo que o mesmo possa retornar a jogar em outra ocasião. Contudo, o sexo e nível do Oponente devem ser informados. Caso a opção de nível escolhida seja "não informado", o Oponente é considerado iniciante, como descrito anteriormente.

A Figura 11 mostra uma jogada e a Figura 12 mostra um possível resultado de uma jogada. Finalmente, a Figura 13 apresenta o histórico de jogadas para uma partida realizada.

V. Conclusão

Este trabalho apresenta a implementação de uma estratégia para aumentar as probabilidades de vencer o jogo PPT baseado em máquinas de estado e tecnologia adaptativa, a partir de uma solução proposta em [7]. Esta solução está baseada na análise de padrões de comportamento de jogadores apresentada em [9,10].

Os resultados obtidos permitem a realização de experimentos com usuários distintos, a fim de avaliar a eficácia da proposta do ponto de vista prático. Esta solução deve ser disponibilizada na Internet e divulgada entre grupos aleatórios, para que a quantidade e diversidade de

experimentos permitam a análise estatística dos resultados obtidos. Esta implementação prevê ainda a comparação com a estratégia aleatória, para melhor identificar desvios.

É importante notar que, 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.

Caso se comprove a existência de uma estratégia adaptativa para vencer o jogo de PPT mais eficaz do que a solução aleatória, este resultado representará um estudo de caso onde se mostra que é possível tratar padrões de comportamento humano usando tecnologia adaptativa.

Espera-se que os resultados desta avaliação, neste momento de caráter puramente exploratório, direcionem novas pesquisas neste sentido, oferecendo uma alternativa para a modelagem matemática e quantitativa de problemas encontrados em diversas áreas do conhecimento.

Referências

[1] 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.

[2] CHANG, Y-H e KAELBLING, L. P. Playing is believing: the role of beliefs in multi-agent learning. M.I.T. Artificial Intelligence Laboratory.

[3] 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

[4] 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.

[5] 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.

[6] NAMATAME, A. Emergence of Desired Collectives with Evolving Synchronized Coupling Rules. Dept. of Computer Science, National Defense Academy, Yokosuka, 239-8686, JAPAN.

[7] SANTANA, F. S., BARBERATO, C. e SARAIVA, A.M. Avaliação de uma estratégia para vencer o jogo de “pedra, papel e tesoura” usando técnicas adaptativas. Memórias do WTA 2010.

[8] 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.

[9] WALKER, G. The Official Rock Paper Scissors Strategy Guide. 2004.

[10] 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]

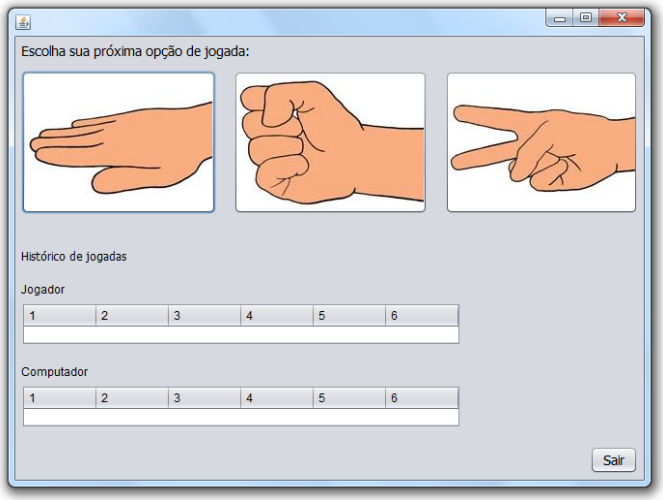


Figura 11 – Tela de jogada.

Figura 12 – Possível resultado de uma jogada.

Jogador	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Registro de jogadas	Papel	Papel	Papel	Pedra	Tesoura	Papel	Pedra	Tesoura	Papel	Pedra	Tesoura	Papel	Pedra	Tesoura	Papel
Tendência			Pedra					Papel					Tesoura		
			26,7%					46,6%					26,7%		
Oponente	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Registro de jogadas	Pedra	Tesoura	Tesoura	Papel	Pedra	Tesoura	Papel	Tesoura	Pedra	Papel	Tesoura	Pedra	Papel	Tesoura	Papel
Tendência			Pedra					Papel					Tesoura		
			26,7%					33,3%					40%		

Figura 13 – Histórico de Jogadas.

Adaptive Search with multiple Unmanned Aerial Vehicles (UAVs)

A. N. Chaves, P. S. Cugnasca, J. J. Neto

***Abstract*— Despite there being some researches about collaborative unmanned aerial vehicle, operations with this kind of robots are not yet occurring. Using adaptative concepts, a model to investigate search operations applied to multiple UAVs is proposed. The adaptative concepts fit very well with the dynamism present in search operations in unknown environment. Two mechanism of adaption are purposed. The first leads to a prioritization of the most likely cells in the probabilistic map and the second leads to a dynamic redivision of the subareas initially allocated. Theoretical calculations show an increase in efficiency by about 100%. The utility of collaborative UAVs is not only the increase in efficiency, but also the reduction of costs and risks for the crew. Furthermore, related works, UAV history, conclusions and suggestions for future researches are also presented.**

***Keywords*— Adaptative systems, Unmanned Aerial Vehicle, multiple UAVs, search operations, SAR.**

I. INTRODUÇÃO

VANTs (Veículos Aéreos Não Tripulados) são aeronaves capazes de serem operadas por controle remoto ou autonomamente. Esses robôs são ideais para operações longas (que expõem a tripulação à fadiga extrema) e operações em que há risco para o piloto (tanto para a saúde quanto risco de morte) – são as chamadas *dull, dirty and dangerous missions*. Logo, uma importante aplicação desses veículos diz respeito às operações de busca envolvendo múltiplos VANTs [1], [2], [3], [4], pois aumentando o número de aeronaves pode-se aumentar a cobertura da busca e o utilização de VANTs torna viável que a operação perdure por muito mais tempo. Além disso, com o barateamento das tecnologias de controle e de comunicação, a utilização de múltiplos VANTs se torna viável e bastante atrativa. Espera-se que a combinação de esforços das partes colaborativas supere a soma dos esforços das partes isoladas, minimizando o tempo e os recursos necessários para localizar um ou mais alvos. Também em favor da utilização de VANTs em operações de busca, o Manual Internacional de Busca e Salvamento [5] aponta que mais de uma aeronave tripulada não deve fazer buscas na mesma subárea, pois essa situação gera um estado de alerta prejudicial ao sucesso da operação, fazendo com que a tripulação tenha a sua atenção desviada das buscas visuais para a navegação e coordenação com outras aeronaves. Portanto, a utilização de VANTs nesse tipo de operação se faz necessária.

Este trabalho recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) através do programa de Bolsas de Mestrado.

A. N. Chaves, Escola Politécnica da Universidade de São Paulo (Poli-USP), São Paulo, Brasil, aquila.chaves@usp.br.

P. S. Cugnasca, Escola Politécnica da Universidade de São Paulo (Poli-USP), São Paulo, Brasil, paulo.cugnasca@poli.usp.br.

J. J. Neto, Escola Politécnica da Universidade de São Paulo (Poli-USP), São Paulo, Brasil, paulo.cugnasca@poli.usp.br.

As operações de buscas representam a fase mais importante das operações de busca e salvamento [6], mais conhecidas na literatura pela sigla SAR (*search and rescue*). Nesse contexto, a utilização de tecnologias adaptativas mostra-se atrativa, pois permite que os VANTs se adaptem a novas informações provenientes de outros VANTs, permitindo o compartilhamento do mesmo espaço de busca e agregando dinamismo às buscas, contribuindo, assim, para aumentar a eficiência da operação.

O objetivo deste trabalho é apresentar uma proposta de modelagem de um sistema de VANTs colaborativos, aplicada a operações de busca, utilizando os conceitos de tecnologia adaptativa.

O artigo está organizado da seguinte forma: a seção II apresenta o histórico dos VANTs até chegar à visão de operação envolvendo múltiplos VANTs e como a colaboração de VANTs é abordada em trabalhos relacionados. A seção III apresenta o mecanismo de busca adaptativa, definindo o cenário de busca, a estrutura da informação trocada entre os VANTs e os mecanismos de adaptação em si. Por fim, as seções IV e V apresentam, respectivamente, os resultados esperados com base em cálculos teóricos e as considerações finais deste trabalho.

II. MÚLTIPLOS VANTS

Existem diversas abordagens que tratam da colaboração de múltiplos VANTs. A subseção a seguir apresentará um breve histórico da tecnologia em VANTs, mostrando a evolução desde o seu surgimento até a visão de futuro com múltiplos VANTs colaborativos, contexto em que este trabalho se insere.

A. Da origem à visão de futuro

Também conhecidos como *Unmanned Aerial Vehicles* (UAVs), *Uninhabited Aerial Vehicles* [7], *Unmanned Aircraft Systems* (UASs), ou simplesmente *drones* [8], os VANTs possuem histórico associado, principalmente, a operações militares.

Enquanto o conceito de voo não tripulado foi introduzido formalmente por Nicola Tesla em 1915 [9], os primeiros testes sem tripulação datam de 1916, quando Lawrence e Elmer Sperry construíram uma aeronave com navegação automática – batizada de "torpedo aéreo" [7]. Mas, somente em 1917 o primeiro VANT foi desenvolvido, durante a Primeira Guerra Mundial. Até aquele momento, os VANTs não eram confiáveis e sua utilidade não era reconhecida pela maioria dos militares e líderes políticos [8]. No entanto, muito antes disso, o ser humano já se aventurava a construir "engenhocas" que, pela definição atual, já poderiam ser consideradas

VANTs. Em 1849, austríacos utilizaram balões interligados a cabos elétricos para lançar bombas sobre o inimigo [10].

A era moderna dos VANTs inicia-se na década de 1970, com os Estados Unidos e Israel desenvolvendo projetos de VANTs de pequeno porte, menos velozes e mais baratos que os utilizados até aquele momento. O sucesso das operações israelenses na guerra do Líbano, utilizando essas aeronaves, em 1982, deu origem a um novo sistema que foi utilizado com sucesso nas operações no Iraque, em 1991 e em 2003 [7]. Portanto, pode-se dizer que foram após as operações em 1991, quando o VANT Pioneer foi utilizado em 300 missões durante a operação *Desert Storm*, que a utilização de VANTs deslanchou [8] [9].

Atualmente, a maior parte dos VANTs ainda são semiautônomos, ou seja, dependem de um operador humano para serem guiados [11]. Nesse contexto, este trabalho também contribui para o avanço na pesquisa de VANTs verdadeiramente autônomos.

Após décadas de avanço, os principais desafios referentes aos VANTs estão relacionados ao voo colaborativo [8]. Apesar de os VANTs atuais apresentarem baixa autonomia, a visão é que, no futuro, múltiplos robôs aéreos sejam capazes de atuar de modo colaborativo. Os VANTs funcionarão como uma rede de sensores, devendo ser coordenados para cumprir missões complexas [12]. É nessa direção que este trabalho visa majoritariamente contribuir.

B. Abordagens em relação a Múltiplos VANTs

Ryan *et al*. [13] apontam que, apesar do barateamento dos sistemas computacionais e do crescente interesse pela utilização de VANTs colaborativos, alguns desafios ainda persistem, tais que: (i) detecção do alvo por câmeras ou sensores; (ii) mecanismos para evitar colisão com outros VANTs ou com obstáculos fixos; (iii) reconfiguração de formação, quando os VANTs voam em formação; (iv) controle transparente do grupo de VANTs, que visa à operação de múltiplos VANTs sem a necessidade de operar individualmente cada VANT; e (v) limitações de hardware e de comunicação. Nesse trabalho, os autores abordam, principalmente, o desafio (iv), sugerindo que uma aplicação de VANTs colaborativos deve fornecer uma interface gráfica simplificada para que um operador insira os objetivos da missão. Assim, os VANTs devem cooperar para dividir e realizar as tarefas de modo autônomo. Será visto, a seguir, que a proposta de busca adaptativa atende à sugestão desses autores.

Luotsinen, Gonzalez e Boeloeni [14] propuseram um modelo em que a colaboração de VANTs ocorre pela troca de informações sobre os riscos existentes na exploração de um ambiente hostil. Para isso, um mapa de ocupação é utilizado e compartilhado entres os VANTs. O mapa identifica as regiões hostis e é utilizado na navegação de cada VANT, de modo a evitar essas regiões. Nesse caso, a adaptação ocorre para desviar de áreas de risco. Segundo os autores, o principal conceito utilizado foi o *Context Based Reasoning framework* (CxBR) [15], cuja função é modelar o comportamento dos agente (que exerce papel de VANT) baseando-se no conhecimento prévio desse agente e na sua experiência adquirida ao longo da operação. Assim, a cada percepção, o agente define um conjunto finito de consequências e atualiza seu conhecimento.

Outro aspecto importante da aplicação de VANTs em operações de busca e salvamento é a detecção de pessoas por meio do processamento de imagens. Nesse contexto, Doherty e Rudol [3] abordaram esse problema, refinando algoritmos de identificação de corpos humanos. Além disso, desenvolveram um *framework* para cooperação baseado em delegação de metas e sequência de ações. Aqui, observa-se uma cooperação por meio da distribuição e divisão de tarefas.

Em determinados cenários de busca, também pode ser necessário considerar regiões de sombra enquanto a operação de busca é realizada. Por exemplo, em aéreas urbanas, a presença de prédios faz com que seja necessário observar determinados pontos no solo de vários ângulos diferentes. Abordando esse problema, Waharte e Trigoni [2] e Jakob *et al*. [16] compararam a eficiência de alguns algoritmos de busca, considerando regiões de sombra.

Por fim, há trabalhos que utilizam protótipos reais para estudar a colaboração de VANTs. Por exemplo, Ryan *et al*. [17] implementaram um sistema que realiza missões colaborativas supervisionadas por um único operador. Nesse trabalho, VANTs de pequeno porte foram utilizados e a cooperação também se baseia na distribuição de tarefas de patrulha ou de procura de invasor.

Portanto, verifica-se que a pesquisa de VANTs colaborativos realmente autônomos é uma área de interesse multidisciplinar que ainda se encontra no seu nascedouro. A seção a seguir apresenta a proposta de busca adaptativa, foco principal deste artigo.

III. Busca Adaptativa

Outros trabalhos também utilizam adaptatividade para operações de busca. No entanto, abordam diferentes cenários de busca que o abordado neste trabalho. Rubio, Vagners e Rysdyk [18], por exemplo, utilizaram conceitos de adaptatividade para adequar a trajetória de busca aos reflexos do sol e trabalharam com um objeto de busca por vez.

A seguir, o cenário de busca deste trabalho é definido.

A. Cenário de busca

Sabe-se que, para cada cenário, há uma estratégia de busca mais adequada [6]. Portanto, a identificação do cenário de busca é essencial para o sucesso de uma operação de busca. O cenário proposto é a perda de uma aeronave em alto mar com o desconhecimento de qualquer informação sobre sua posição. Sabe-se apenas a região onde é possível que estejam as partes que se espalharam, representada pela região retangular na figura 1.

Os objetos de busca (*search objects*) serão espalhados a partir de um foco central de acordo com uma distribuição gaussiana, conforme apresentado na figura 1, que está coerente com a distribuição de probabilidades apresentada pelo Manual Internacional de Busca e Salvamento [5]. Inicialmente, os VANTs desconhecem qualquer informação sobre a localização

dos objetos perdidos. Esses objetos de busca (também chamados de alvos na literatura) representam destroços e eventuais sobreviventes.

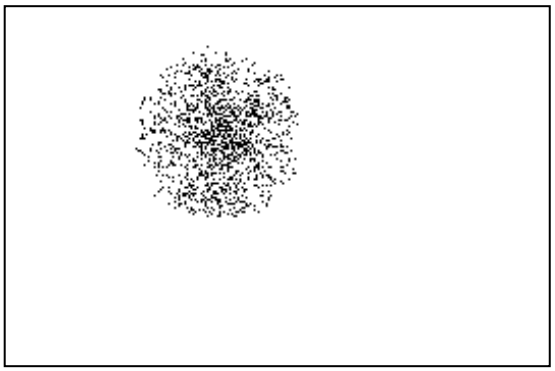


Figura 1. Cenário de busca.

Em acidentes como esse, é possível que haja aproximadamente 600 objetos entre bagagens, fuselagens, partes do avião e, principalmente, sobreviventes espalhados. Tudo isso dentro de uma área total de busca de aproximadamente 350 mil quilômetros quadrados [19].

Assim, a busca colaborativa é bastante atraente nesse cenário, pois, diferentemente de buscas tripuladas, os VANTs podem sobrevoar a área interruptamente, durante muitas horas e, numa mesma altitude. A adaptatividade agrega ainda mais na medida em que, quando ocorre detecção dos primeiros objetos, os VANTs podem trocar informações e se adaptar às novas informações-, tentando inferir onde está a região de espalhamento.

B. Discretização da região de busca

De acordo com Chung e Burdick, os métodos probabilísticos são os mais apropriados para avaliar a evolução da maioria das tarefas de coleta de informações, pois eles são capazes de representar imprecisões na área busca [20]. Na proposta de busca adaptativa deste artigo, os VANTs compartilham o conhecimento sobre o ambiente e sobre a evolução da operação de busca na forma de um mapa de probabilidades. Essas probabilidades representam a chance de se detectar um objeto em cada célula desse mapa.

Dessa forma, a busca adaptativa utiliza um mapa de busca discretizado. Ou seja, cada VANT mantém o conhecimento da área de busca de forma discretizada, em que cada célula contém a probabilidade dela conter um objeto de busca. A figura 2 ilustra a representação do conhecimento do VANT.

Cabe ressaltar que a figura 2 apresenta apenas uma pequena parte do espaço de busca, que pode conter milhares de células. Nota-se que em cada célula há a probabilidade *P* dela conter um objeto de busca, cujo valor pode variar entre 0 a 1.

O modelo de navegação utilizado aqui é o mesmo utilizado em [21], em que a navegação do VANT entre as células se restringe às direções norte, sul, leste e oeste. No referido trabalho, Lin e Goodrich concluem que esse modelo é bem próximo das capacidades de um VANT de pequeno porte, pois, em voos reais, um VANT faz uma curva de 90 graus (cobrindo três células) no mesmo tempo em que dois deslocamentos perpendiculares (também cobrindo três células) ocorreriam. Portanto, esse modelo de navegação é bastante adequado a mini ou micro-VANTs e a VANTs de asas rotativas. Além disso, utilizando uma câmera com rotação em três eixos, mais conhecida como *gimbal camera*, é possível fazer com que a busca realizada pela câmera faça a curva em 90 graus.

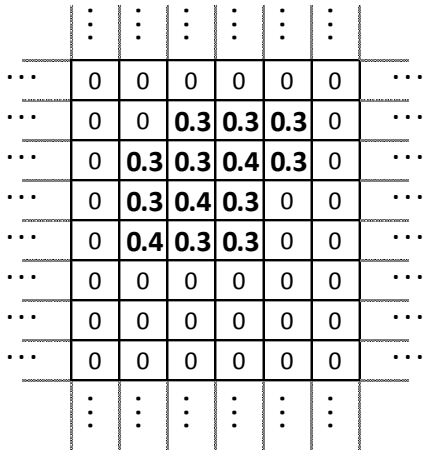


Figura 2. Discretização da área de busca.

C. Adaptação

Tendo o mapa do espaço de busca discretizado em células, a probabilidade de cada célula conter um objeto é inicializada com zero, pois não há qualquer conhecimento sobre a provável localização dos objetos no espaço de busca.

Nesse estado, os VANTs utilizam o padrão de busca "Rotas Paralelas" como algoritmo de navegação. A figura 3 ilustra esse padrão de busca, que é definido pelo DECEA [6] como o mais indicado em situação que não se tem informações sobre o objeto de busca.

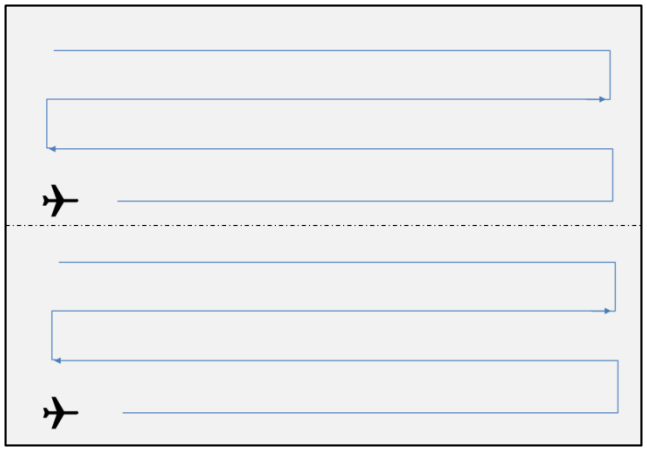


Figura 3. Buscas iniciais utilizando o padrão de Rotas Paralelas para dois VANTs, explorando por varredura todo o espaço de busca, que foi dividido em duas subáreas.

Quando um dos VANTs (chamado de primeiro) detecta um objeto, ele atualiza o seu mapa de probabilidade (aumentando as probabilidades das células ao redor da célula em que este objeto foi encontrado) e o transmite ao outro VANT (chamado de segundo). Assim, as células dentro do círculo pontilhado (figura 4), excetuando as que já foram visitadas, têm as probabilidades aumentadas.

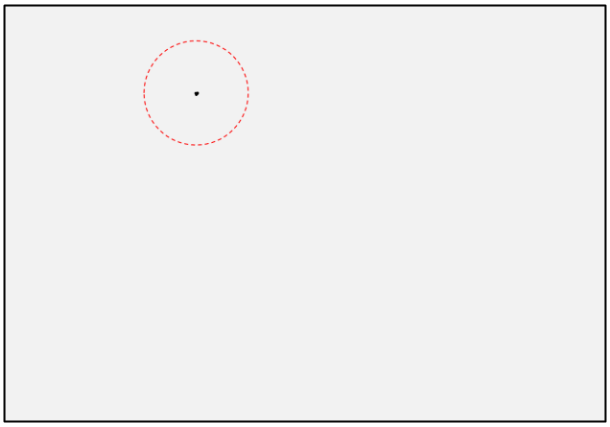


Figura 4. Exemplo de detecção do primeiro objeto. As células que estão dentro do círculo pontilhado têm suas probabilidades aumentadas.

Ao receber a atualização do mapa, o segundo VANT, que ainda não detectou nada, altera o seu padrão de busca, adaptando-se a essa à nova situação. A partir desse momento, a busca do segundo VANT passa a se comportar da seguinte forma:

1. Uso do algoritmo A* para chegar até a célula de maior probabilidade ou até a fronteira da região de maior probabilidade (em caso de haver mais de uma célula aglutinada com maior probabilidade). Aqui, as células já sobrevoadas e com detecções negativas ganham pesos suficientemente altos para evitar que o VANT perca tempo sobrevoando células onde já foi feita a busca. Ao mesmo tempo, os pesos devem ser suficientemente baixos para que não impeça a passagem do VANT por essas células, gerando um caminho demasiadamente longo. Células com maior probabilidade de conter objetos (mas com probabilidade menor do que a probabilidade da célula de destino), que ainda não foram sobrevoadas, também influenciam na redução do peso dos nós intermediários durante o calcula do caminho pelo algoritmo A*.

2. Ao chegar à região de maior probabilidade, o VANT efetua uma busca local.

A busca local baseia-se em, a cada movimento, deslocar o VANT para a célula vizinha mais próxima de maior probabilidade. Se houver alguma outra célula de maior probabilidade que as células vizinhas, utiliza-se o algoritmo A* para navegar até essa célula, de acordo com o exposto anteriormente. Esse processo é repetido até que toda a região seja varrida.

Outras heurísticas mais elaboradas poderiam ser utilizadas na busca local. No entanto, isso não faz parte do escopo deste trabalho.

Enquanto isso, considerando que o primeiro VANT está numa região mais provável de conter os objetos (pois já houve uma detecção), ele continua a sua varredura pelo padrão "Rotas Paralelas" até finalizá-la em sua subárea. Apenas quando ele finaliza essa busca é que ele altera o seu comportamento, agindo da mesma forma que o segundo VANT e ajudando-o em buscas locais. Para evitar colisão entre VANTs, quando o primeiro VANT está em navegação no padrão "Rotas Paralelas", todas as células que estão na mesma coordenada *y* se tornam não navegáveis pelo segundo VANT.

A cada nova detecção realizada por qualquer um dos VANTs, o mapa probabilístico é atualizado e os passos 1 e 2 são realizados pelo(s) VANT(s) responsável(is) pela busca local. Observa-se pela figura 5 que, conforme as detecções vão ocorrendo, o mapeamento da área dos objetos perdidos vai se tornando mais nítido.

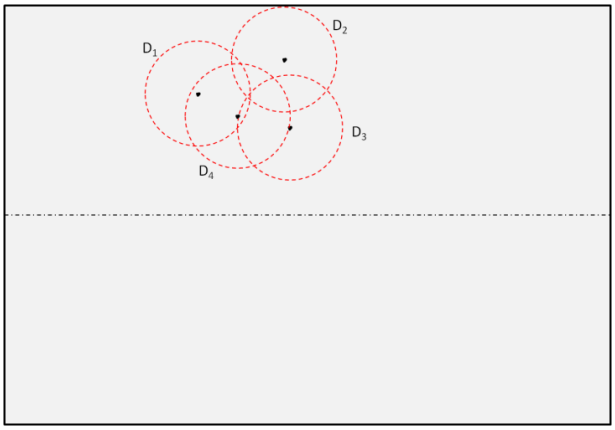


Figura 5. Quatro detecções de objetos e, consequentemente, quatro atualizações das probabilidades do espaço de busca.

Assim, essa constante adaptação atua como se fosse uma "força de atração" que direciona o(s) VANT(s) responsável(is) pela busca local sempre para a região de maior probabilidade de detecção. Além desse mecanismo adaptativo, que atua fortemente na priorização de áreas, a busca com múltiplos VANTs exige outros mecanismos de adaptação que atuam na coordenação entre os VANTs.

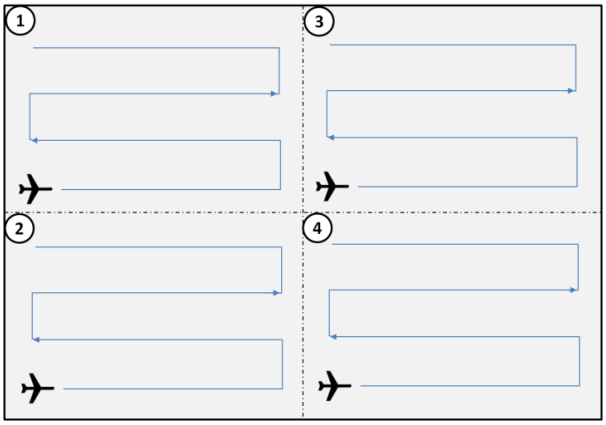
Numa busca realizada por quatro VANTs, por exemplo, inicialmente o espaço de busca retangular é dividido em quatro partes iguais, alocadas a cada um dos quatro VANTs, conforme figura 6a. Posteriormente, assumindo que o VANT da área 1 encontrou um objeto, o VANT que estiver mais próximo abandona a sua subárea (neste caso o segundo VANT – figura 6b) e segue para realizar a busca local. Até aqui, foi introduzido apenas a coordenação para alocar o VANT mais próximo. Nesse momento, os outros dois VANTs, que ainda não assumiram a responsabilidade da busca local, devem se

adaptar à nova situação e continuar a varredura na subárea deixada pelo segundo VANT que seguiu para a busca local.

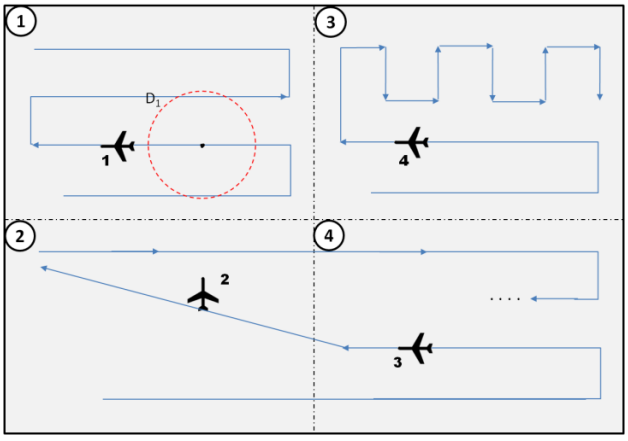
A adaptação proposta atua da redivisão dinâmica das subáreas e ocorre da seguinte forma: quando o segundo VANT segue para realizar a busca local, os outros VANTs das áreas 3 e 4 (figura 6) realizam as seguintes manobras:

- O VANT da área 3 adapta o padrão de busca de modo a priorizar as células que estão mais próximas de onde foi detectado o primeiro objeto.
- O VANT da área 4 adapta-se, da mesma forma, aumentando sua subárea de busca. Nesse último caso, a saída do segundo VANT faz com que a área 2 deixe de ser espaço proibido para o VANT da área 4.

Nessa segunda adaptação, as regras utilizadas foram: (i) o VANT mais próximo do VANT que assumiu a busca local assume a sua subárea; (ii) após a primeira detecção, atualizar o padrão de busca de modo que a varredura ocorra paralelamente à subárea em que o objeto foi detectado e que a progressão da varredura ocorra no sentido de afastamento dessa subárea.



(a) Busca adaptativa com quatro VANTs no estado inicial.



(b) Busca adaptativa com quatro VANTs após o segundo mecanismo de adaptação proposto.

Figura 6. Busca adaptativa com quatro VANTs.

IV. Resultados

Este trabalho faz parte de uma pesquisa em andamento cuja simulação, embora já demonstre alguns resultados, continua em fase de desenvolvimento. Por esse motivo, até o momento, analisou-se apenas o cenário com dois VANTs. Partindo do modelo adaptativo proposto e considerando os seguintes parâmetros:

- câmera com angulação de 30º [22],
- altitude de busca de 150 metros [6],
- área total de varredura de 10.000 km² e objetos espalhados numa área de raio de 5 km de lado – espalhamento próximo ao de um caso real obtido em [23],

pode-se obter o resultado esperado.

Dada a angulação e a altitude, obtêm-se células de dimensões de aproximadamente 100m x 100m. Logo, considerando uma velocidade média do VANT de 20m/s (velocidade média atingida por um mini-VANT), dois VANTs levariam cerca de 7 dias (174 horas) voando ininterruptamente para varrer exaustivamente a área.

Utilizando os mecanismos adaptativos propostos neste trabalho, obtém-se um valor médio teórico de 82 horas. Totalizando um aumento de eficiência de aproximadamente 100%. Ou seja, houve redução de aproximadamente 50% no tempo de busca. O cálculo foi feito tirando uma média entre o melhor caso (um dos VANTs encontra o primeiro objeto imediatamente após o início da varredura) e o pior caso (um dos VANTs encontra o primeiro objeto no fim da sua varredura).

Simulações foram conduzidas com dois VANTs e os resultados demonstraram, após 30 simulações, uma média de 55% de redução no tempo de busca (comparando com o tempo necessário para realizar buscas em 100% da área), com desvio padrão de 21%.

Novos cenários ainda serão analisados e os resultados serão apresentados em trabalhos futuros.

V. Considerações Finais

Este trabalho apresentou um modelo de busca adaptativa que pode ser utilizado em qualquer tipo de robô. Porém, a utilização em grupos de VANTs é bastante atrativa de modo que os VANTs podem ser empregados em áreas de difícil acesso. Além disso, os VANTs podem realizar as buscas ininterruptamente enquanto houver combustível e, ao invés de câmeras, podem utilizar sensores térmicos, permitindo também buscas noturnas. Ainda que em aeronaves tripuladas também seja possível fazer uso de sensor térmico, há um aumento do risco para a tripulação, em se tratando de voos noturnos.

O modelo propõe adaptatividade no que se refere à priorização de áreas com maior probabilidade de se encontrar objetos de busca e à redivisão das subáreas. Nesse segundo caso, além do redimensionamento dinâmico das subáreas que estão sendo varridas (emprego do padrão de busca “Rotas

Paralelas"), o sentido da progressão da varredura também é ajustado ao novo mapa de conhecimento.

Para uma operação de busca utilizando dois VANTs no cenário proposto, cálculos teóricos mostram redução de 50% no tempo de busca. Num segundo momento, utilizando o método Monte Carlo, simulações serão conduzidas com o objetivo de demonstrar a potencialidade do método aqui proposto.

Trabalhos futuros também podem explorar a coordenação dos VANTs, utilizando a teoria multiagente, principalmente na fase de busca local e redimensionamento das subáreas. A teoria multiagente é aderente à autonomia que se deseja dar aos VANTs e fornece diversas técnicas de coordenação e colaboração que podem ser muito bem aplicadas nesta situação.

Diferentes algoritmos e heurísticas também devem ser explorados e estudados na busca local.

A discretização do mapa e o fato de os VANTs se guiarem apenas pelo mapa probabilístico também contribuem para a solução do desafio de controle transparente abordado em [13]. O mecanismo permite que apenas um operador controle todos os VANTs (não importa quantos), apenas manipulando as informações do mapa probabilístico.

Além disso, diversos outros cenários podem ser analisados, como, por exemplo, o caso em que se deseje iniciar as buscas pela rota planejada da aeronave perdida e, por fim, as simulações devem ser flexibilizadas de modo que a colaboração de *n* VANTs possa ser observada.

REFERÊNCIAS

[1] Seng Keat Gan and Salah Sukkarieh, "Multi-UAV target search using explicit decentralized gradient-based negotiation," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shangai, China, 2011, pp. 751-756.

[2] S. Waharte and N. Trigoni, "Supporting Search and Rescue Operations with UAVs," in *International Conference on Emerging Security Technologies (EST)*, Canterbury, UK, 2010, pp. 142-147.

[3] Patrick Doherty and Piotr Rudol, "A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization," in *AI 2007: Advances in Artificial Intelligence*, Mehmet Orgun and John Thornton, Eds.: Springer Berlin / Heidelberg, 2007, vol. 4830, pp. 1-13.

[4] Home Land Security NewsWire. (2010, Julho) Home Land Security NewsWire. [Online]. http://www.homelandsecuritynewswire.com/uavs-perform-autonomous-search-and-rescue-operations

[5] IMO/ICAO, "IAMSAR Manual - International Aeronautical and Maritime Search and Rescue Manual," IMO/ICAO, London/Montreal, 2003, 2003.

[6] DECEA, "Manual de Busca e Salvamento (SAR)," Ministério da Defesa - Comando da Aeronáutica, 2009.

[7] Timothy H. Cox, Christopher J. Nagy, Mark A. Skoog, and Ivan A. Somers, "Civil UAV Capability Assessment," NASA and CSM, Inc., Draft Version 2004.

[8] Kimon P. Valavanis, *Advances in Unmanned Aerial Vehicles: State of the Art and the Road to Autonomy*, Kimon P. Valavanis, Ed.: Springer, 2007.

[9] U.S. Army, "Unmanned Aircraft System - Roadmap 2010-2035," U.S. Army UAS Center of Excellence, Fort Rucker, Alabama, USA, 2010.

[10] Scientific American, "More about Ballons," *Scientific American*, vol. 4, no. no. 26, p. 205, Março 1849.

[11] L.N. Long, S.D. Hanford, O. Janrathitikarn, G.L. Sinsley, and J.A. Miller, "A Review of Intelligent Systems Software for Autonomous Vehicles," in *IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2007)*, Honolulu, HI, USA, 2007, pp. 69-76.

[12] George Vachtsevanos, Liang Tang, and Johan Reimann, "An Intelligent Approach to Coordinated Control of Multiple Unmanned Aerial Vehicles," in *In Presented at the American Helicopter Society 60th Annual Forum*, Baltimore, MD, 2004.

[13] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J.K. Hedrick, "An overview of emerging results in cooperative UAV control" , vol. 1, 2004, pp. 602-607.

[14] Linus J. Luotsinen, Avelino J. Gonzalez, and Ladislau Boeloeni, "Collaborative UAV Exploration of Hostile Environments," *Intelligent Autonomous Vehicles*, vol. Spring 2008, 2004.

[15] Avelino J. Gonzalez and Robert Ahlers, "Context-based representation of intelligent behavior in training simulations," *Trans. Soc. Comput. Simul. Int.*, vol. 15, pp. 153-166, 1998.

[16] Michal Jakob, Eduard Semsch, Dusan Pavlicek, and Michal Pechoucek, "Occlusion-aware Multi-UAV Surveillance of Multiple Urban Areas," in *6th Workshop on Agents in Traffic and Transportation (ATT 2010)*, 2010.

[17] A. Ryan *et al.*, "Decentralized Control of Unmanned Aerial Vehicle Collaborative Sensing Missions," in *American Control Conference (ACC '07)*, New York City, USA, 2007, pp. 4672-4677.

[18] Juan Carlos Rubio, Juris Vagners, and Rolf Rysdyk, "Adaptive path planning for autonomous uav oceanic search missions," in *AIAA 1st Intelligent Systems Technical Conference*, Chigado, Illinois, 2004.

[19] Centro de Comunicação Social da Marinha. (2009, Junho) Força Aérea Brasileira (FAB). [Online]. http://www.fab.mil.br/portal/capa/index.php?mostra=3311

[20] T.H. Chung and J.W. Burdick, "Multi-agent probabilistic search in a sequential decision-theoretic framework," in *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.*, Pasadena, California, USA, 2008, pp. 146-151.

[21] L. Lin and M.A. Goodrich, "UAV intelligent path planning for Wilderness Search and Rescue," in *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, 2009, pp. 709-714.

[22] Michael A. Goodrich *et al.*, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89-110, 2008.

[23] Folha Online. (2009, Junho) Folha Online - Ministro afirma que avião da Air France caiu a cerca de 700 km de Fernando de Noronha. [Online]. http://www1.folha.uol.com.br/folha/cotidiano/ult95u575492.shtml

Parte II

Transcrição da mesa redonda

A transcrição da mesa redonda está em fase de revisão. Em breve, esta seção será substituída pelo texto adequado.