RODRIGO SUZUKI OKADA

TECNOLOGIA ADAPTATIVA APLICADA A SISTEMAS HÍBRIDOS DE APOIO À DECISÃO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

São Paulo
2013

RODRIGO SUZUKI OKADA

TECNOLOGIA ADAPTATIVA APLICADA A SISTEMAS HÍBRIDOS DE APOIO À DECISÃO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

Área de Concentração:

Sistemas Digitais

Orientador:

Prof. Dr. João José Neto

São Paulo
2013

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 27 de março de 2013.

Assinatura do autor

Assinatura do orientador

FICHA CATALOGRÁFICA

Okada, Rodrigo Suzuki

Tecnologia adaptativa aplicada a sistemas híbridos de apoio à decisão/ R. S. Okada. – ed. rev. – São Paulo, 2013.

210 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Técnicas adaptativas 2. Aprendizado computacional 3. Tomada de decisão I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

Aos meus pais.
Seres trabalhadores, zelosos e sábios.
Honestos, amigos, e educadores.
Pessoas que admiro.

AGRADECIMENTOS

Aos meus pais, pelo apoio, dedicação e amor incondicional, responsáveis diretos pela minha formação, seja ela acadêmica ou como pessoa.

Ao professor Wilson V. Ruggiero, por conceder-me minha primeira oportunidade de pesquisa, pelos seus conselhos, ideias inovadoras e criticas construtivas.

Ao orientador João José Neto, não só pela sua orientação ao longo desta pesquisa, pelos seus conselhos e sugestões, mas também pelo constante estímulo e atenção, primordiais para execução deste trabalho.

Ao corpo docente e funcionário da Escola Politécnica da Universidade de São Paulo, que, de forma direta ou indireta, colaboraram com a realização desta pesquisa.

RESUMO

Este trabalho apresenta a formulação de um sistema híbrido de apoio à decisão que, através de técnicas adaptativas, permite que múltiplos dispositivos sejam utilizados de forma colaborativa para encontrar uma solução para um problema de tomada de decisão. É proposta uma estratégia particular para o trabalho colaborativo que restringe o acesso aos dispositivos mais lentos com base na dificuldade encontrada pelos dispositivos mais rápidos para solucionar um problema específico. As soluções encontradas por cada dispositivo são propagadas aos demais, permitindo que cada um deles agregue estas novas soluções com o auxílio de técnicas adaptativas. É feito um estudo sobre aprendizagem de máquina mediante incertezas para verificar e minimizar os impactos negativos que uma nova solução, possivelmente errônea, possa ter. O sistema híbrido proposto é apresentado numa aplicação particular, utilizando testes padronizados para compará-lo com os dispositivos individuais que o compõem e com sistemas híbridos de mesma finalidade. Através destes testes, é mostrado que dispositivos consolidados, mesmo que de naturezas distintas, podem ser utilizados de maneira colaborativa, permitindo não só calibrar um compromisso entre o tempo de resposta e a taxa de acerto, mas também evoluir de acordo com o histórico de problemas processados.

Palavras Chave: Adaptatividade. Tabela de decisão adaptativa. Tomada de decisão. Métodos multicritério. Raciocínio baseado em casos. Sistemas híbridos. Aprendizagem de máquina. Classificador da Bayes. k-Nearest Neighbor. Compromisso entre tempo de resposta e taxa de acerto.

ABSTRACT

This work presents a formulation of an hybrid decision-making system that employs adaptive techniques as a way to coordinate multiple devices in order to make a collaborative decision. The strategy proposed here is to restrict the use of slower devices, based on how difficult the specific problem is - easier problems may be solved on faster devices. Each device is able to learn through solutions given by the others, aggregating new knowledge with the aid of adaptive techniques. In order to evaluate and minimize the negative impact those new solutions may have, a study concerning machine learning under uncertainty is carried out. A particular application of this system has been tested and compared, not only to each individual device that is part of the system itself, but to similar hybrid systems as well. It is shown that even devices of distinct natures may be reused in a collaborative manner, making it possible to calibrate the trade-off between hit rate and response time, and to envolve according to the input stimuli received as well.

Keywords: Adaptivity. Adaptive decision table. Decision-making. Multi-criteria methods. Case-Based Reasoning. Hybrid systems. Machine learning. Naive Bayes. k-Nearest Neighbor. Speed-Accuracy Trade-off.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AHP *Analytic Hierarchy Process*

ATDA Ambiente de Tomada de Decisão Adaptativa

CBR *Case-Based Reasoning*

DSS *Decision Support System*

ILE *Interactive Learning Environment*

k-NN *k-Nearest Neighbor*

SAT *Speed-Accuracy Trade-off*

TDA Tabela de Decisão Adaptativa

TDAE Tabela de Decisão Adaptativa Estendida

LISTA DE SÍMBOLOS

a	Uma possível alternativa a ser utilizada como solução de um problema
$c_{1-\alpha/2}$	Intervalo de confiança do valor-verdade $\hat{v}$, com $(100 \cdot (1-\alpha/2))\,\%$ de certeza
cr	Credibilidade de uma fonte de realimentações
cr_0	Credibilidade inicial das informações recebidas via treinamento
$d\,(a,b)$	Função de distância (e.g., Euclidiana) entre instâncias a e b
$d_j\,(a,b)$	Função de distância entre os valores das instâncias a e b no j-ésimo critério
$I\,(a)$	Função indicadora - retorna 1 se a for verdadeiro, senão, retorna 0
m	Número de critérios utilizados pelo tomador de decisão
p	Número de alternativas utilizadas pelo tomador de decisão
$P\,(a\|b_1, b_2, \ldots)$	Probabilidade de a ser verdade quando $b_1, b_2, \ldots$ também são verdadeiros
$\bar{v}$	Valor-verdade quando calculado via média aritmética da verdades das realimentações
$\hat{v}$	Valor-verdade quando calculado via média aritmética desviada de Agresti-Coull
V	Contador de realimentações positivas recebidas para um caso particular

$w(a, b)$	No *k-NN*, representa a função de ponderação pela distância entre a e b
W	Contador de todas as realimentações recebidas para um caso particular
x	Vetor de características de m dimensões
$x_{[i]}$	Valor do vetor x associado ao i-ésimo critério
$y(x)$	Função que calcula o vetor μ (cada dispositivo implementa esta função a sua maneira)
$z_{1-\alpha/2}$	$(100 \cdot (1 - \alpha/2))$-ésimo percentil de uma distribuição normal
β_+	Valor de reforço relativo de realimentações positivas
β_-	Valor de reforço relativo de realimentações negativas
$\delta(a, b)$	Métrica de *overlap* - se $a = b$, retorna 0, senão, retorna 1
μ	Vetor que informa a expectativa de certeza de cada uma das p alternativas para uma instância x
$\phi^{-1}(p)$	Função quantil de uma distribuição qualquer

1 INTRODUÇÃO

Sistema de apoio à decisão (*DSS - Decision Support System*) é um artefato computacional utilizado para apoiar um tomador de decisão em sua tarefa de encontrar soluções para os problemas de sua área de atuação (SOL; TAKKENBERG; ROBBÉ, 1985), sendo originalmente proposto de forma teórica para solucionar problemas organizacionais (KEEN, 1978). Sua principal função é a de coletar informações úteis a partir de uma massa de dados pura, e inferir quais decisões aparentam ser as mais vantajosas com base neste conhecimento, disponibilizando-as de forma compreensível ao tomador de decisão. Apesar de não haver consenso sobre um modelo único e definitivo, sistemas de apoio à decisão são comumente compostos por uma base de conhecimento, um modelo de tomada de decisão e uma interface de usuário (SPRAGUE; CARLSON, 1982).

O conceito de sistema de apoio à decisão, contudo, não é completamente homogêneo. Desde sua introdução no final da década de 1960, seu significado associado tem sido alterado de acordo com os novos avanços em múltiplos ramos de pesquisa (ARNOTT; PERVAN, 2005), permitindo uma maior aderência dos sistemas a diferentes finalidades. Sistemas de apoio a negócios e sistemas de informação executiva são exemplos de derivados do sistema original, cada um utilizando teorias oriundas de diferentes especialidades, tais como estudos sociais e gerenciamento de banco de dados.

A partir dos avanços obtidos na área de aprendizagem de máquina, espera-se que um próximo passo do desenvolvimento dos sistemas de tomada de decisão seja um sistema *evolutivo*, capaz de assimilar novas informações de forma incremental, eliminando a necessidade de ser reconstruído toda vez que for preciso atualizar sua base de conhecimento (ARNOTT, 2004). O processo de aprendizado deste tipo de sistema deve funcionar de forma autônoma, livre de interferência do usuário, dependendo unicamente dos estímulos que recebe, provenientes do ambiente em que se encontra.

A Figura 1 apresenta uma linha do tempo com a evolução dos sistemas de apoio à decisão, bem como os ramos de pesquisas utilizados para sua fundamentação, conforme sugerido em (ARNOTT; PERVAN, 2005). De acordo com esta linha do tempo, os primeiros sistemas de apoio à decisão surgiram no início da década de 1970, tendo sido utilizados como ferramentas de auxílio para tomadas de decisões organizacionais de forma individual pelos diversos tomadores de decisão - note que este conceito está de acordo com as definições previamente citadas.

Este sistema foi utilizado como base fundamental para que outros sistemas, mais especializados, pudessem ser formalizados. Por exemplo, o advento de técnicas de mineração de dados permitiu especializar o sistema original para analisar o relacionamento entre informações armazenadas em uma base de dados - processo conhecido por *data warehousing*.

De forma semelhante, trabalhos relacionados à inteligência artificial permitiram a inclusão de agentes inteligentes nos sistemas de apoio à decisão, aumentando seu potencial para solucionar problemas mais complexos, que dificilmente seriam solucionados através de técnicas clássicas. Pesquisas relacionadas ao gerenciamento de conhecimento, enfim, permitiram que o conhecimento específico de um ramo de atividade - e.g., medicina e finanças -

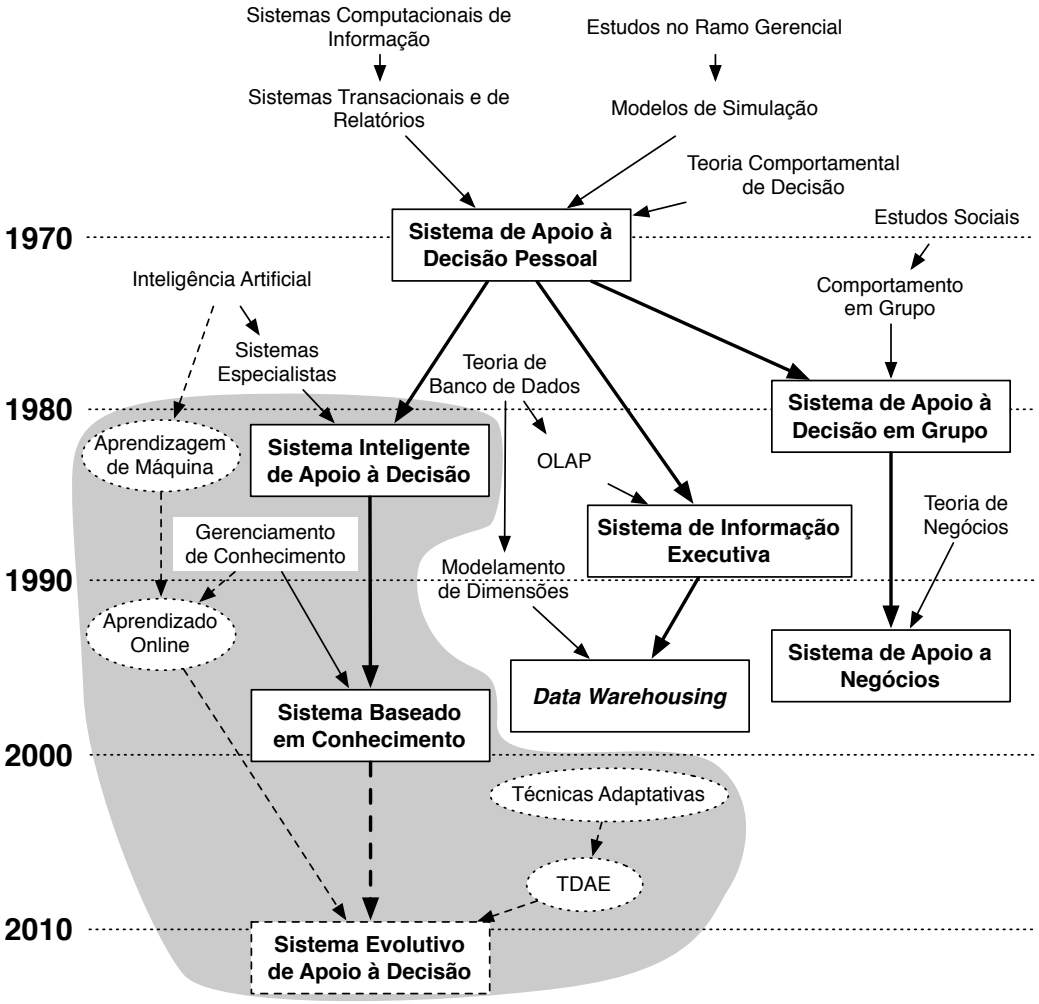


Figura 1: Evolução dos sistemas de apoio à decisão, conforme sugerido por Arnott (ARNOTT; PERVAN, 2005). Itens em pontilhado denotam ramos não previstos no diagrama original de Arnott, enquanto a região em destaque (cinza) denota as técnicas utilizadas por este trabalho e o sistema resultante.

pudesse ser transferido para um sistema de apoio à decisão, especializando-o para uma tarefa específica.

O sistema evolutivo proposto neste trabalho, então, é uma extensão dos sistemas de apoio à decisão baseados em conhecimento, incorporando técnicas adaptativas para que, durante sua execução, novas informações possam ser agregadas ao conhecimento já acumulado até então. Para esta finalidade, foram utilizadas as seguintes técnicas:

- **Aprendizagem de Máquina** é um ramo da inteligência artificial que estuda a aquisição de conhecimento feita por sistemas computacionais. Uma revisão bibliográfica deste conceito é feita na seção 2.4.

- **Aprendizado Online** é um paradigma de aprendizagem que estuda a aquisição de conhecimento de forma incremental, ao contrário das técnicas clássicas, que utilizam um *batch*. Uma breve introdução a este assunto é feita na seção 2.4.1.

- **Adaptatividade** é o termo usado para caracterizar sistemas capazes de tomar a decisão de se auto-modificar, em resposta a um estímulo externo. Sistemas desta natureza permitem que o sistema evolua de acordo com o ambiente em que se encontram. Uma revisão deste conceito está apresentada na seção 2.1.

- ***TDAE*** é a abreviação da *Tabela de Decisão Adaptativa Estendida*, que designa um sistema híbrido de tomada de decisão via adaptatividade - processo explorado neste estudo. Sua visão geral pode ser encontrada na seção 2.1.4.

Este trabalho representa um esforço inicial no desenvolvimento de um sistema desse tipo, generalizando a técnica híbrida proposta pela *TDAE* para outros dispositivos adaptativos, e utilizando teorias de aprendizagem para controlar e direcionar a adaptatividade, para que as mudanças sejam construtivas. Tal tipo de sistema possibilita obter ganhos na taxa de acertos ao longo do tempo, ou então, evitar que a mesma se deteriore para problemas de natureza dinâmica, em que as informações são passíveis de obsolescência.

1.1 Justificativa

Desde sua publicação por Neto em 2001 (NETO, 2001), dispositivos adaptativos têm sido alvo de estudo em variados campos de pesquisa além da construção de compiladores - área que impulsionou o desenvolvimento inicial da adaptatividade. Áreas como processamento de linguagens naturais se beneficiam do uso deste tipo de técnica explorando a capacidade de expressão das mesmas (MENEZES; NETO, 2002). De maneira mais prática, técnicas adaptativas estão sendo empregadas para solucionar problemas de navegação robótica (HIRAKAWA; SARAIVA; CUGNASCA, 2007), bem como reconhecimento de padrões, envolvendo áreas de visão computacional (COSTA; HIRAKAWA; NETO, 2002). Ainda, ferramentais como o AdapTools têm sido desenvolvidos não só para auxiliar a construção de dispositivos adaptativos, mas também como material pedagógico (PISTORI, 2003; JESUS et al., 2007).

Mesmo antes da formalização do dispositivo adaptativo geral em 2001, casos particulares de adaptatividade já haviam sidos utilizados para solucionar problemas específicos. Como exemplo, redes de Markov adaptativas foram empregadas para a construção de um software autônomo de composição musical (BASSETO; NETO, 1999). Autômatos de pilha estruturados adaptativos, a partir dos quais foi formalizado o dispositivo adaptativo geral, foram alvo de estudo em trabalhos anteriores, nos quais é provado que seu poder de representação é equivalente ao de uma Máquina de Turing (ROCHA; NETO, 2000).

O emprego de técnicas adaptativas voltadas para a tomada de decisão tem sido feito em dispositivos mais específicos. Ao formalizar o dispositivo adaptativo geral em 2001, conforme ilustra a Figura 2, uma tabela de decisão adaptativa (*TDA*) é também apresentada como exemplo de uso deste tipo de técnica (NETO, 2001). Esta mesma tabela foi alvo de aprofundamento por

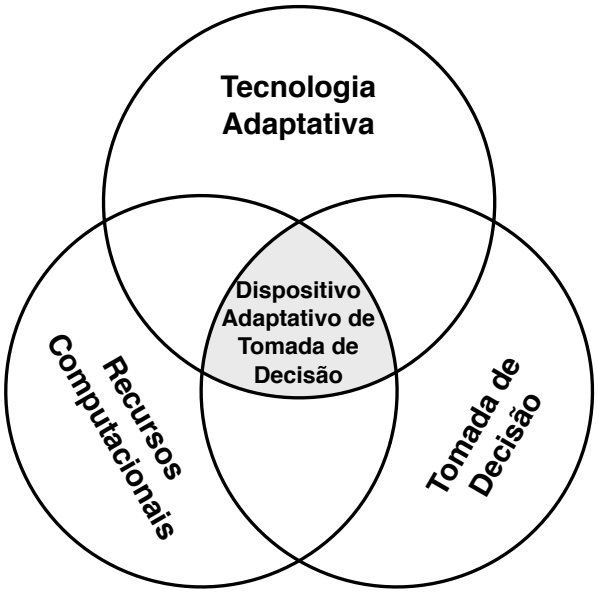


Figura 2: Diagrama conceitual de um dispositivo adaptativo de tomada de decisão (TCHEMRA, 2009).

Tchemra, que, em 2009, apresentou uma extensão à *TDA*, em que um segundo método de decisão (no caso, o *AHP - Analytic Hierarchy Process*) é utilizado para solucionar problemas que a tabela clássica de decisão não prevê em seu conjunto corrente de regras (TCHEMRA, 2009). A solução dada pelo método auxiliar, então, é incorporada pela tabela de decisão na forma de uma nova regra, caracterizando uma forma de aprendizado incremental. A esta tabela, foi dado o nome de *TDAE* (Tabela de Decisão Adaptativa Estendida).

A *TDAE* constitui um exemplo do uso de técnicas adaptativas em sistemas híbridos de apoio à decisão, e mostra ser possível realizar um trabalho colaborativo, ainda que restrito a dois dispositivos particulares. No entanto, o escopo das tabelas de decisão é restrito para problemas já previstos em seu conjunto de regras, e, em particular, elas não operam sobre problemas numéricos.

Outra restrição implícita da *TDAE* é o critério para decidir se o método auxiliar deve ser utilizado: em uma tabela de decisão, quaisquer problemas

não previstos em seu conjunto de regras devem ser repassados ao método auxiliar. Contudo, este critério é limitado para dispositivos guiados por regra, e não poderia ser utilizado, por exemplo, em dispositivos baseados em probabilidades, como é o caso do classificador de Bayes.

Estas limitações apresentadas pela *TDAE* incentivaram e motivaram o estudo e aplicação da tecnologia adaptativa como meio para coordenar uma tomada de decisão colaborativa entre dispositivos de naturezas distintas.

1.2 Objetivos

O principal objetivo deste trabalho é propor um sistema híbrido de apoio à decisão que, através de técnicas adaptativas, permite que múltiplos dispositivos de diferentes naturezas sejam utilizados de forma colaborativa para solucionar um problema de tomada de decisão. Este uso de múltiplos dispositivos é uma generalização do modelo introduzido pela *TDAE*, em que cada dispositivo agrega conhecimento através das soluções apresentadas pelos demais. Em especial, permite que dispositivos mais lentos, mas que possuem melhores taxas de acertos, propaguem suas soluções para dispositivos mais rápidos. Experimentos foram realizados através de *benchmarks* padronizados na área de aprendizagem de máquina - muitos deles oriundos de dados reais - para que o resultado possa ser comparado ao de sistemas congêneres.

O segundo objetivo deste estudo é sugerir formas de tratar incertezas sobre a veracidade de um conhecimento adquirido durante o aprendizado incremental. Enquanto os conhecimentos originados através de dados de treinamento, obtidos com a ajuda de humanos, podem ser tratados como verdadeiros com confiança, conhecimentos obtidos incrementalmente através de outros dispositivos são propensos a conter erros, e, consequentemente, me-

nos confiáveis. Porém, métodos clássicos de aprendizagem de máquina não distinguem as informações pela sua confiabilidade, tratando todas da mesma forma, o que pode fazer com que o aprendizado venha a comprometer a qualidade do conhecimento anteriormente acumulado. Embora não exista um modelo computacional consolidado que modele a influência de informações incertas sobre a confiança, é possível verificar, de forma isolada, a influência que certos fatores cognitivos podem exercer sobre a tomada de decisão. Tais fatores foram utilizados para estipular um modelo de confiança a ser utilizado pelo sistema adaptativo.

A partir deste sistema, também é verificada a relação de compromisso entre a taxa de acertos e o tempo de resposta de uma tomada de decisão, comumente chamada de *Speed-Accuracy Trade-off* (*SAT*). Tipicamente, quanto mais tempo é gasto para encontrar uma solução, mais precisa ela se torna - tal relação é ilustrada pela Figura 3. Seres humanos procuram encontrar um ponto de equilíbrio entre o tempo gasto e a taxa de acertos para maximizar seus ganhos - e.g., durante uma prova escolar, procura-se o obter o melhor resultado dentro de um dado intervalo de tempo (tempo é o fator limitante), enquanto a escolha de um novo imóvel deve ser feita minimizando os erros, mesmo que seja necessário sacrificar mais tempo (precisão é o fator limitante). Mesmo animais são capazes de efetuar tal escolha, sacrificando precisão para obter soluções mais rápidas em problemas em que o custo do erro é baixo (CHITTKA; SKORUPSKI; RAINE, 2009).

Porém, grande parte dos métodos clássicos de tomada de decisão apresentam algoritmos de comportamento linear, e não permitem regular o tempo a ser gasto, nem a taxa de acertos desejada. Encontrar uma forma de calibrar esta relação de compromisso é o terceiro grande objetivo deste trabalho. Fazendo uso da natureza híbrida do sistema proposto, são verificadas formas de

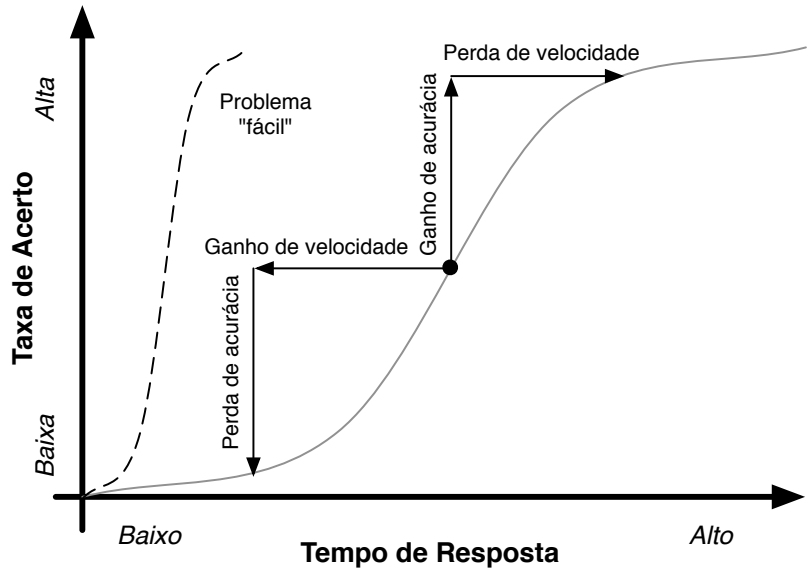


Figura 3: Curva *SAT* que relaciona o compromisso entre velocidade e taxa de acertos.

controlar a quantidade de problemas a serem solucionados em cada dispositivo: preferencialmente, problemas perceptualmente fáceis devem ser solucionados pelos dispositivos mais rápidos, enquanto problemas mais complexos devem ser solucionados pelos dispositivos mais precisos e, potencialmente mais lentos. O entendimento do conceito de dificuldade de um problema é discutido ao longo desta dissertação.

Segue abaixo um resumo dos resultados conceituais obtidos como produtos deste trabalho:

- Uma forma diferenciada de tomada de decisão, que, através de técnicas adaptativas, permite criar um sistema híbrido que integra dispositivos de diferentes naturezas.

- Generalização da extensão do *TDAE* para dispositivos que utilizem, no lugar da aplicação de regras, técnicas probabilísticas para obter soluções.

- Definição de um modelo que pondera as informações conhecidas pelo sistema através de sua confiança e permite distinguir um conhecimento consolidado de um conhecimento obtido via aprendizado incremental, potencialmente menos confiável.

Abaixo, seguem os principais resultados práticos obtidos:

- Criação de um *framework* extensível para desenvolvimento de sistemas híbridos de tomada de decisão, que permite que novos dispositivos sejam incorporados. Foi desenvolvido na linguagem de programação Java.
- Implementação de vários dispositivos de tomada de decisão, preparando-os para a utilização do modelo de confiança proposto.
- Validação da relação de compromisso entre taxa de acertos e velocidade com o auxílio de *benchmarks* padronizados.

1.3 Organização da Dissertação

Esta dissertação está organizada em quatro capítulos principais:

- **(cap. 2) Revisão Bibliográfica** - Introduz os conceitos utilizados por este trabalho para a formulação e definição do sistema híbrido de tomada de decisão, oferecendo uma revisão da literatura nas áreas de tomada de decisão, técnicas adaptativas, aprendizagem de máquina e questões relacionadas à confiança de uma decisão.
- **(cap. 3) Proposta** - Capítulo dedicado ao detalhamento da proposta de projeto deste estudo, apresentando o funcionamento do sistema de tomada de decisão, e descrevendo de que forma o modelo de confiança é utilizado para realizar um processo incremental de aprendizagem.

- **(cap. 4) Resultados** - Apresenta o banco de testes utilizado para verificar o funcionamento de um sistema-exemplo, composto por uma série de dispositivos definidos no capítulo anterior. Os testes se dividem em dois principais grupos: 1. verificar a evolução de um sistema à medida que os dispositivos passam a propagar suas soluções para os demais; e 2. verificar como a taxa de acertos e o tempo de resposta se comportam ao controlar a quantidade de problemas a serem solucionados em cada dispositivo - tal controle é realizado através de um limiar, cujo funcionamento é detalhado na proposta.

- **(cap. 5) Conclusão** - Capítulo de encerramento, destacando as conclusões, contribuições e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, serão abordados os principais tópicos relacionados a este trabalho, introduzindo uma visão geral, histórico e perspectivas para futuras pesquisas de cada conceito utilizado como base para o desenvolvimento deste estudo.

2.1 Técnicas Adaptativas

2.1.1 Visão Geral

O termo *adaptatividade* corresponde à capacidade de um sistema tomar a decisão de automodificar-se, alterando seu próprio comportamento como resposta a um estímulo de entrada sem a interferência de agentes externos (NETO, 2007). Tal característica permite que o sistema evolua de acordo com o histórico de entradas processadas, adaptando-se ao ambiente em que se encontra.

Apesar de ter sido formalizado em 2001 (NETO, 2001), estudos anteriores de estruturas de topologias variáveis podem ser encontrados na literatura em diferentes níveis de abstrações e generalizações. Um dos primeiros exemplos foi uma nota publicada por Agasandjan em 1967, idealizando um autômato de estrutura variável (AGASANDJAN, 1967). Na mesma época, Salomaa também apresentava um trabalho teórico sobre autômatos finitos variáveis no tempo

(SALOMAA, 1968). De maneira complementar, Christiansen também viria a trabalhar com teorias adaptativas, contudo, voltada para a compreensão das dependências de contextos em gramáticas (CHRISTIANSEN, 1985; CHRISTIANSEN, 1990).

A trajetória da formalização do dispositivo adaptativo, contudo, teve origem no autômato de pilha estruturado apresentado por Neto e Magalhães em 1981 (NETO; MAGALHÃES, 1981). Com base neste trabalho, Neto, em 1988, apresentava o funcionamento de um reconhecedor sintático com comportamento adaptativo (NETO, 1988). Este formalismo seria, em 1993, consolidado na forma de um autômato adaptativo (NETO, 1993), cujo poder de representação é equivalente a uma Máquina de Turing, conforme provado em (ROCHA; NETO, 2000). A generalização do dispositivo adaptativo veio em um trabalho publicado por Neto em 2001, que descreve o comportamento dos dispositivos guiados por regras, e como torná-los adaptativos, permitindo o reuso de dispositivos já conhecidos (NETO, 2001).

Dentro deste contexto, o termo *dispositivo* é utilizado para referenciar uma abstração lógica formal, capaz de solucionar um problema de entrada. Em particular, dispositivos guiados por regras são aqueles dirigidos por um conjunto finito de regras que descrevem a configuração em que o dispositivo se encontra, bem como seu comportamento ao receber estímulos externos (NETO, 2001). Exemplos de dispositivos incluem autômatos finitos, árvores de decisão, tabelas, redes de Markov, gramáticas e afins - e tipicamente, são imutáveis durante a execução, não apresentando um comportamento adaptativo.

De acordo com Neto (NETO, 2001), um dispositivo guiado a regras ND é descrito por $ND = (C, NR, S, c_0, A, NA)$, onde:

- C denota o conjunto finito de possíveis configurações que ND pode as-

sumir, enquanto c_0 é sua configuração inicial.

- S é o conjunto de todos os estímulos de entrada aceitos por ND. O valor nulo ϵ é membro deste conjunto.
- $A \subseteq C$ é o subconjunto de configurações de aceitação.
- NA é o conjunto finito de todos os símbolos de saída de ND. Estes são escritos como resposta à aplicação das regras presentes em NR. Do ponto de vista de um observador externo, um símbolo de saída $z \in NA$ pode ser interpretado como uma operação a ser executada.
- $NR \subseteq C \times S \times C \times NA$ é o conjunto finito de regras que define o comportamento do dispositivo ND. A aplicação de uma regra $r \in NR$, denotada por $r = (c_i, s, c_j, z)$, altera a configuração atual de c_i para c_j ao consumir o símbolo de entrada $s \in S$, escrevendo $z \in NA$ na saída.

Em contrapartida, um *dispositivo adaptativo* é capaz de alterar, de forma autônoma, seu próprio conjunto de regras como resposta a um estímulo de entrada. Este dispositivo pode ser separado em duas componentes: um dispositivo subjacente e uma camada adaptativa, responsável pelo controle das mudanças a serem realizadas (NETO, 2001). Esta configuração permite reutilizar formalismos já consolidados e torná-los adaptativos, ao custo de um pequeno acréscimo de complexidade (PISTORI, 2003). A disposição do dispositivo adaptativo está ilustrada na Figura 4.

O controle das alterações do conjunto de regras é feito através de *ações adaptativas* - sequência de operações elementares de consulta, inserção e remoção de regras - que são associadas a um subconjunto particular das regras existentes, denominadas *adaptativas*. O disparo de uma regra adaptativa faz com que sua ação associada seja executada, resultando na alteração do conjunto atual de regras.

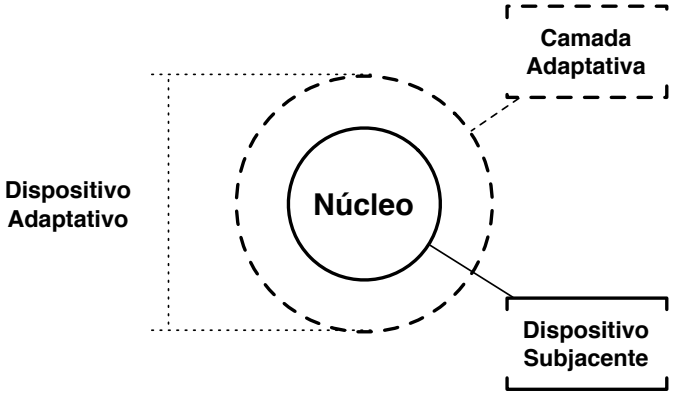


Figura 4: Estrutura típica de um dispositivo adaptativo, formado por um dispositivo subjacente, envolto por uma camada adaptativa.

Formalmente, um dispositivo adaptativo AD é descrito por $AD_k = (ND_k, AM)$, tal que:

- $k \geqslant 0$ é um contador que é incrementado quando uma ação adaptativa é executada.
- ND_k é o dispositivo subjacente após k utilizações do dispositivo. ND_0 é o dispositivo subjacente inicial, enquanto ND_{k+1} é a evolução de ND_k após uma ação adaptativa de inserir uma nova informação. Este novo dispositivo terá um novo conjunto de regras NR_{k+1}, uma versão editada do conjunto anterior NR_k.
- AM é o mecanismo adaptativo responsável pela execução das ações adaptativas.

Expandindo a definição de ND em AD, temos que $AD_k = (C_k, AR_k, S, c_k, A, NA, BA, AA)$, tal que:

- C_k é o conjunto finito de estados após a execução de k ações adaptativas, enquanto c_k é a configuração inicial do dispositivo AD_k.

- Os conjuntos S, $A \subseteq C_k$ e NA permanecem com suas definições originais. Assim, os estímulos de entrada válidos, o conjunto de estados de aceitação e os símbolos de saída devem estar todos previstos no dispositivo inicial.

- BA e AA são conjuntos de ações adaptativas que podem ser executadas antes e depois da aplicação de uma regra, respectivamente. Ambos os conjuntos devem conter a ação nula, que não efetua alterações no conjunto de regras.

- O conjunto de regras de um dispositivo adaptativo é renomeado de NR para $AR_k \subseteq BA \times C_k \times S \times C_k \times NA \times AA$. Regras adaptativas são escritas na forma $ar_k = (ba, c_i, s, c_j, z, aa)$, onde $ba \in BA$ é a ação adaptativa a ser executada antes de aplicar a regra. Após executá-la, a porção não-adaptativa da regra (c_i, s, c_j, z) é aplicada, para que, em seguida, a ação adaptativa $aa \in AA$ seja executada. Caso a regra a ser aplicada seja removida devido à ação ba, sua aplicação é abortada, forçando o dispositivo a encontrar outra regra para consumir o estímulo de entrada s. Se $ba = aa = \epsilon$, então a regra será, essencialmente, não-adaptativa.

O Algoritmo 1 ilustra o funcionamento de um dispositivo adaptativo AD. O funcionamento básico deste algoritmo está baseado em um *loop*, que itera a cadeia de entrada $w = s_1, \ldots s_n$ tal que $s_i \in S \forall\, 0 < i \leqslant n$, extraindo um símbolo s_{atual} por vez e utilizando-o para selecionar o conjunto CR_T de regras aplicáveis a ser utilizado. Regras aplicáveis são aquelas cujos valores c_i e s são respectivamente compatíveis com a configuração $AD.c_{atual}$ e o símbolo s_{atual} que estiver correntemente sendo processado.

De maneira mais formal, $CR_T = \{ar \in AR_T \,|\, ar = (ba, c_i, s, c_j, z, aa)\,,\ c_i = c_{atual}, s \in \{s_{atual}, \epsilon\};\ c_i, c_j \in C_T;\ ba \in BA;\ aa \in AA;\ z \in NA\}$. A obtenção

Algoritmo 1 $executaAR\,(AD, w, out)$

Entrada
AD: Dispositivo adaptativo a ser executado
w: Cadeia de símbolos de entrada $s \in S$
Saída
out: Cadeia de saída, com símbolos $z \in NA$

```
1: AD.c_atual ← c_0
2: for all s_atual ∈ w do
3:    consumido ← false
4:    while not consumido do
5:       CR_T ← buscaRegra (AD, s_atual)
6:       if CR_T = ∅ then
7:          Rejeita w
8:       else if ||CR_T|| = 1 then
9:          ar ← CR_T [0]
10:         consumido ← executaD (AD, ar, out)
11:      else
12:         consumido ← executaND (AD, ar, out)
13:      end if
14:   end while
15: end for
16: if AD.c_atual ∈ AD.A then
17:   Aceita w
18: else
19:   Rejeita w
20: end if
```

deste conjunto está representada pela função $buscaRegra$ na linha 5, e é descrita pelo Algoritmo 2. Note que, para ser aceita, a regra deve ter c_i igual ao estado do dispositivo, enquanto seu símbolo s deve ser igual ao símbolo de entrada s_{atual}, ou então, nulo.

Note-se que esta notação presume que o símbolo seja unidimensional, isto é, um valor escalar. Se o símbolo for uma composição de múltiplos valores, isto é, $s = (s_1, \ldots, s_m)$, a condição para a regra ser aceita passa a ser $s \,|\, \forall\, s_i \in s \,:\, s_i \in \{s_{atual\ i}, \epsilon\}$, ilustrado pelo Algoritmo 3. Para o caso unidimensional, esta formulação é idêntica à condição original.

A cardinalidade de CR_T define a maneira com que o Algoritmo 1 deve

Algoritmo 2 $buscaRegra\,(AD, s)$

Entrada
AD: Dispositivo adaptativo a ser executado
s: Símbolo lido da cadeia de entrada = s_{atual}
Saída
Conjunto CR_T de regras aplicáveis

1: $CR_T \leftarrow \{\}$
2: **for all** $ar = (ba, c_i, s, c_j, z, aa) \in AD.AR_T$ **do**
3: **if** $ar.c_i = AD.c_{atual} \wedge subsimbolo\,(s, ar.s)$ **then**
4: Insere ar no CR_T
5: **end if**
6: **end for**
7: **return** CR_T

Algoritmo 3 $subsimbolo\,(s, rs)$

Entrada
s: Símbolo lido da cadeia de entrada
rs: Símbolo da regra
Saída
$true$ se símbolo deve ser aceito, $false$ caso contrário

1: **for all** $rs_i inrs$ **do**
2: **if** $rs_i \notin \{s_i, \epsilon\}$ **then**
3: **return** $false$
4: **end if**
5: **end for**
6: **return** $true$

proceder:

- Caso nenhuma regra seja encontrada, o algoritmo encerra sua execução, rejeitando a cadeia de entrada (linha 7).
- No caso de haver apenas uma única regra aplicável - i.e., $\|CR_T\| = 1$ - sua aplicação pode ser feita de maneira determinística, explicitada pela linha 10 e descrita pelo Algoritmo 4. Neste caso, a regra $ar = (ba, c_i, s, c_j, z, aa)$ encontrada é aplicada, o que resulta na execução da ação adaptativa ba. O prosseguimento depende do resultado de ba:
  - Caso a ação de ba resulte na remoção de ar do conjunto de regras

AR, o Algoritmo 4 encerra sua execução, retornando $false$ sem aplicar ar. Neste caso, o símbolo de entrada s_{atual} não é consumido, e o dispositivo é forçado a encontrar outra regra na nova instância do conjunto AR (Algoritmo 1, linha 4).

  - Caso contrário, o símbolo s_{atual} de entrada é consumido, alterando a configuração corrente de AD de c_i para c_j, e escrevendo o símbolo z na saída. Finalmente a ação aa é executada.

- Caso mais de um regra possa ser aplicada, todas são executadas ao mesmo tempo, de forma não-determinística (linha 12). Devido à natureza determinística dos sistemas existentes, uma ação deste tipo é simulada através de *backtracking*. Apesar de casos particulares terem sido estudados (PISTORI; NETO; PEREIRA, 2006; CASTRO JR.; NETO; PISTORI, 2007), o ideal é evitar este tipo de situação devido o alto custo computacional envolvido na simulação desse tipo de ocorrência. Neste trabalho, casos não-determinísticos não serão avaliados - logo, a função $executaND$ não será desenvolvida neste trabalho.

As linhas 2 e 10 do Algoritmo 4 ilustram a chamada das ações ba e aa, respectivamente, através do procedimento $adapta$, que deve ser implementado de acordo com o tipo de dispositivo. Cada ação adaptativa é definida através de uma função *adaptativa*, que definem a sequência de operações a serem executadas por tal ação. Tais funções devem possuir os seguintes itens (NETO, 2001):

- Um **nome** simbólico, utilizado para referenciar a função. As regras utilizam este nome para identificar quais as funções a serem chamadas.

- Um conjunto de **parâmetros** que deve ser passado como argumento durante a chamada da função. Cada parâmetro é representado por um

Algoritmo 4 $executaD\,(AD, ar, out)$

Entrada
AD: Dispositivo adaptativo a ser executado
ar: Regra adaptativa a ser executada
Saída
out: Cadeia de saída, com símbolos $z \in NA$

1: **if** $ar.ba \neq \epsilon$ **then**
2: $\quad adapta\,(AD, ar.ba)$
3: **end if**
4: **if** $ar \notin AD.AR$ **then**
5: $\quad$ **return** $false$
6: **end if**
7: $AD.c_{atual} \leftarrow ar.c_j$
8: Insere $ar.z \rightarrow out$
9: **if** $ar.aa \neq \epsilon$ **then**
10: $\quad adapta\,(AD, ar.aa)$
11: **end if**
12: **return** $true$

nome simbólico, utilizado para constituir a lógica da função. Durante a execução, os valores destes argumentos não podem ser alterados pela função.

- Um conjunto de **variáveis** locais, cujo valor é resolvido durante a execução da função adaptativa através de operações de consulta. Assim como os parâmetros, seus valores não podem ser modificados após serem preenchidos.

- Um conjunto de **geradores**, que geram novos valores cada vez que são utilizados. São representados por um nome simbólico, que pode ser referenciado durante a execução da função adaptativa.

- Finalmente, é necessário especificar o **corpo** da função adaptativa, que especifica as operações elementares a serem executadas para alterar o conjunto de regras. As três operações adaptativas elementares são:

  Consulta (?) Operação que busca um regra de acordo com um critério

de pesquisa, retornando um dos valores presentes na regra (configuração inicial / final ou o símbolo de entrada). Este valor deve ser atribuído a uma variável.

Inclusão ($+$) Operação que insere uma nova regra no conjunto de regras. Seu valor é baseado nos valores presentes nos parâmetros, nas variáveis e nos geradores definidos previamente.

Exclusão ($-$) Operação que exclui uma determinada regra.

Após o consumo de todos os símbolos, a cadeia de entrada é aceita se a configuração final do dispositivo pertence ao conjunto A.

2.1.2 Tabela de Decisão

Tabela de decisão é uma forma compacta para descrever, de forma ilustrativa, a lógica de uma tomada de decisão, associando uma série de condições a uma ação a ser tomada. A associação entre as condições e as ações é feita dividindo a tabela em quatro quadrantes conforme ilustra a Tabela 1: os quadrantes à esquerda listam, na forma de linhas, as condições a serem testadas e as possíveis ações a serem tomadas, enquanto os quadrantes à direita contêm a associação dos valores das condições para cada regra (coluna) e as ações a serem executadas como resposta à aplicação de uma regra. Uma célula vazia nos valores das condições indica que a condição desta linha não precisa ser testada nesta regra (coluna). A Tabela 2 mostra um exemplo de uma simples tabela de decisão que decide a aprovação de um aluno com base em sua média final e frequência nas aulas.

De maneira geral, as tabelas de decisão são estruturas de fácil formulação, compreensão e manutenção, sendo frequentemente empregadas por profissionais de áreas não técnicas - e.g., gerentes e executivos - para formular

		Regras					
		r_1	...	r_j	...	r_n	E
Condições	c_1	Valores das condições					
	...						
	c_i						
	...						
	c_m						
Ações	a_1	Ações a serem tomadas					
	...						
	a_k						
	...						
	a_p						

Tabela 1: Estrutura típica de uma tabela de decisão.

	r_1	r_2	r_3	r_4
Média ≥ 7	S	N	N	
$7 >$ Média ≥ 5		S	N	
Média < 5			S	
$Freq. \geq 70\%$	S	S		N
Aprovado	S			
Recuperação		S		
Reprovado			S	S

Tabela 2: Exemplo de tabela de decisão.

modelos de tomada de decisão (HUGHES; SHANK; STEIN, 1968). A transmissão de conhecimento através de tabelas mostra-se mais expressiva do que outras estruturas, tais como formas narrativas, formas escritas ou fluxogramas (TCHEMRA, 2009).

Em sua forma mais simples, uma tabela de decisão apresenta valores binários para suas condições, tipicamente *Sim* (S) e *Não* (N). Certas condições, porém, são mais facilmente representadas quando assumem valores discretos quaisquer. Como exemplo, a Tabela 2 apresenta três condições para *média*, ao passo que apenas uma linha seria necessária case a tabela utilizasse entradas discretas, conforme ilustra a Tabela 3. A este tipo de tabela, se dá o

nome de Tabela de Decisão de Entradas Estendidas (RAJARAMAN, 2004).

	r_1	r_2	r_3	r_4
Média	≥ 7	$[5-7[$	< 5	
$Freq. \geq 70\%$	S	S		N
Aprovado	S			
Recuperação		S		
Reprovado			S	S

Tabela 3: Exemplo de tabela de decisão com entradas estendidas.

Cada regra individual da tabela pode ser interpretada como uma série de sentenças **if** X **then** Y, onde X é o conjunto de condições a serem analisadas, enquanto Y é a ação a ser executada. As condições são normalmente compostas por uma série de operações do tipo *and*, apesar de que outros operadores lógicos também possam ser utilizados (TCHEMRA, 2009). Como exemplo, a regra $r2$ da Tabela 2 é aplicada quando três condições são verdadeiras:

- Média não é maior ou igual a 7;
- Média é maior ou igual a 5;
- Frequência é maior ou igual a 70%

Caso as três condições sejam verdadeiras, a ação escolhida será considerar que o aluno será reavaliado em um exame de recuperação. Note que a condição de a média ser inferior a 5 não é testada nesta regra - até porque, logicamente, a média só poderia assumir valores maior que 5 e menor que 7, definido pelas duas primeiras regras.

2.1.3 Tabela de Decisão Adaptativa

A aplicação da adaptatividade sobre a tabela de decisão foi introduzida em 2001 por Neto como exemplo da formalização do dispositivo adaptativo. Denominada Tabela de Decisão Adaptativa (*TDA*), ela permite que seu conjunto de regras seja modificado ao longo de sua utilização, o que inclui inserir novas regras, bem como excluir regras dentre as existentes (NETO, 2001). Tal comportamento dinâmico permite que a tabela consiga aprender de forma incremental, ajustando o conjunto de regras conforme novas informações são disponibilizadas (STANGE; NETO, 2011).

Esta seção apresenta as modificações feitas sobre a tabela de decisão convencional, além de um exemplo de uso deste dispositivo.

2.1.3.1 Definição

Utilizando a Tabela 1 como referência, a *TDA* incorpora novas colunas e linhas que representam a declaração das funções adaptativas, bem como as chamadas das mesmas por parte das regras. Seu modelo é ilustrado pela Tabela 4, que apresenta m condições, p ações e nf funções adaptativas.

Note que a tabela de decisão subjacente se encontra nos dois quadrantes superiores à direita, e não sofre alterações com relação à tabela original - cada coluna ainda representa uma regra, enquanto as linhas definem os valores individuais associados a cada regra, além das ações a serem executadas. Uma adição à tabela original é a linha de rótulo (*tag*), utilizada para identificar o significado associado a cada coluna. Como exemplo, regras são identificadas pelo rótulo R, enquanto os rótulos S e E são utilizados para delimitar o início e fim das colunas de regras.

A definição da *TDA* também insere um novo conjunto de linhas abaixo

		Operações Adapt.					Regras					
		...	AD_z			...		r_1	r_2	...	r_n	
Rótulos		...	H	$+$	$-$	...	S	R	R	...	R	E
Condições	c_1						Valores das condições					
	...											
	c_i											
	...											
	c_m											
Ações	a_1						Ações a serem tomadas					
	...											
	a_k											
	...											
	a_p											
Funções Adaptativas	FAD_1						funções adaptativas a serem chamadas					
	FAD_2											
	...											
	...											
	...											
	FAD_z						Referências a FAD_z					
	...											
	FAD_{nf}											

Tabela 4: Estrutura típica de uma tabela de decisão adaptativa (*TDA*), que define as funções adaptativas em novas colunas, e sendo chamadas através de um novo quadrante.

das ações, definindo as funções adaptativas a serem executadas devido à aplicação de uma regra:

- Uma nova linha é utilizada para cada função adaptativa existente, sendo identificada por um nome. Células marcadas nesta linha indicam que suas respectivas regras devem chamar esta função adaptativa.

- Uma nova linha deve ser associada para cada gerador, variável ou parâmetro de entrada utilizados pelas funções adaptativas. Caso uma regra chame uma função adaptativa qualquer, as linhas dos parâmetros utilizados pela função também devem ser preenchidos com os valores destes parâmetros. Os geradores e variáveis não são preenchidos nestas colunas.

As definições das funções adaptativas se fazem em uma série de colunas à esquerda da tabela subjacente. Cada coluna define as operações a serem executadas por cada função adaptativa, enquanto as linhas apresentam os parâmetros, variáveis e geradores utilizados por cada operação:

- De forma análoga à tabela subjacente, a primeira linha apresenta os rótulos de cada coluna:
  - O rótulo H é utilizado para assinalar o cabeçalho de uma função adaptativa. Nesta coluna, a célula que corresponde ao nome da função deve estar marcada com b ou a, assinalando que a função deve ser executada antes ou depois da aplicação da regra que a referencia. Caso a função faça uso de parâmetros, variáveis ou geradores, suas respectivas células devem estar marcadas com os indicadores P, V e G.
  - As colunas que se seguem após a coluna de cabeçalho definem as operações elementares a serem executadas pela função, sendo rotuladas de acordo com o tipo de operação: $+$ (inclusão), $-$ (exclusão), $?$ (consulta).
- Nas linhas das condições, cada operação assinala quais devem ser os valores a serem trabalhados - i.e., em uma operação de inclusão, diz quais devem ser as condições da regra a ser criada, enquanto em uma operação de exclusão diz quais são as condições da regra a ser excluída.
- As linhas de ações assinalam quais delas devem ser executadas pela regra a ser incluída, excluída ou consultada.
- Linhas com os nomes das funções adaptativas devem vir marcadas se a regra a ser incluída/excluída chama esta função.

- Assim como na tabela subjacente, se uma função adaptativa foi marcada, ela deve ser referenciada pela nova regra (ou, no caso de exclusão, indica que a regra a ser removida faz referência a esta função).

2.1.3.2 Exemplo Ilustrativo

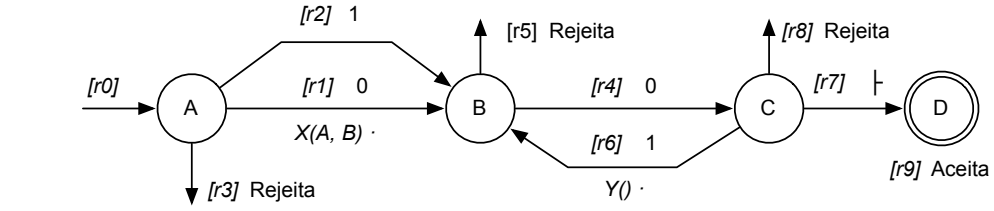
Um exemplo de uso da *TDA* é a simulação de um autômato adaptativo (NETO, 2001). Este autômato representa uma aplicação específica do formalismo adaptativo que utiliza autômatos de estados finitos como dispositivo subjacente (NETO, 1993), conferindo-lhe um aumento de poder de representação (ROCHA; NETO, 2000). A adaptatividade se manifesta na forma de funções adaptativas, que podem ser chamadas ao efetuar uma transição de estados.

A Figura 5 ilustra um exemplo de autômato de 4 estados e 10 transições, bem como a tabela de decisão utilizada para representá-lo. O alfabeto deste autômato é composto pelos símbolos $\{0, 1, \vdash\}$, onde $\vdash$ representa o fim da cadeia. Comparando esta tabela com a *TDA* ilustrada pela Tabela 4, observa-se que:

- Cada transição do autômato é representada por uma regra na tabela. A regra de rótulo S é associada à transição que define o estado inicial do autômato, enquanto as demais são marcadas com R. A regra rotulada com E não é associada a uma transição, apenas delimitando a tabela.

- As condições da tabela de decisão representam as condições necessárias para as transições do autômato serem executadas, i.e., o estado corrente e o símbolo lido da cadeia. Por exemplo, a transição $(B, 0) \rightarrow C$ é associada à regra *4*.

- As ações da tabela representam as ações a serem tomadas devido a execução de uma transição do autômato. No caso, informam o próximo



Regras		Cabeçalho de X	Corpo de X						0	1	2	3	4	5	6	7	8	9	10
Tags		H	+	+	+	-	H	-	S	R	R	R	R	R	R	R	R	R	E
Condições	Estado		p1	g1	g1	p1		C		A	A	A	B	B	C	C	C	D	
	Símbolo		0	0	1	0		1		0	1		0		1	⊢			
Ações	Próx. Est.		g1	p2	p1	p2		B	A	B	B		C		B				
	Get								✔	✔	✔		✔		✔				
	Aceita											X		X			X	✔	
Ações Adaptativas	X	b		✔		✔				✔									
	Y						a	✔							✔				
Parâmetros	p1	P		g1		p1				A									
	p2	P		p2		p2				B									
Geradores	g1	G																	

Figura 5: Exemplo de um autômato adaptativo e sua tabela de decisão correspondente.

estado do autômato ao executar tal transição. Também informam se o símbolo deve ser consumido, e se o autômato deve aceitar (✓) ou rejeitar a cadeia ($\times$).

- Este exemplo possui duas funções adaptativas:
  - A primeira função recebe o nome X, e faz uso de dois parâmetros $p1$ e $p2$, e de um gerador $g1$. Esta função é composta por quatro operações elementares: três delas de inclusão (marcadas com o rótulo $+$) e uma de exclusão ($-$). Quando chamada, esta função deve ser executada *antes* de executar as ações informadas pela regra (esta ordem de execução é definida pelo valor *b*, presente em seu cabeçalho).

– A outra função, referenciada por Y, não faz uso de nenhum parâmetro, gerador ou variável, e contém uma única operação de exclusão. Quando chamada, esta função deve ser executada *após* executar as ações informadas pela regra (esta ordem de execução é definida pelo valor *a*, presente em seu cabeçalho).

A partir desta tabela, a simulação do autômato adaptativo é feita através dos seguintes passos:

1. A regra inicial, marcada com o rótulo S, deve ser aplicada, simulando o início de operação do autômato, alterando seu estado corrente $c_{corrente}$ para o estado inicial informado pela ação da regra.
2. Para cada símbolo s da cadeia de entrada:
   (a) Busca-se uma regra aplicável para os parâmetros $(c_{corrente}, s)$. No autômato, isso equivale a verificar se, no estado corrente, existe uma transição que consuma o símbolo s.
   (b) Se a regra não existir, a simulação deve ser interrompida, pois não há transições válidas no autômato nestas condições.
   (c) Senão, verificar se a regra possui a ação de rejeitar a cadeia. Em caso positivo, a simulação é encerrada, com o autômato rejeitando a cadeia.
   (d) Se a ação não é de rejeição:
      i. Verifica se a regra possui uma função adaptativa a ser executada antes de executar as ações informadas pela regra. Caso exista, ela é executada.
      ii. Se a execução da função adaptativa fez com que a regra escolhida tenha sido removida, volta para o passo 2a.

iii. Executa as ações informadas pela regra - em especial, atualiza o valor do estado corrente do autômato.

iv. Verifica se a regra possui uma função adaptativa a ser executada após executar as ações informadas pela regra. Caso exista, ela é executada.

v. Se o símbolo lido é o indicador de fim de cadeia $\vdash$:

   A. Verifica qual é a ação de aceitação da regra. Este passo simula os estados de aceitação do autômato.

   B. Se a ação estiver marcada com ✓, cadeia é aceita;

   C. Senão, se a ação estiver marcada com $\times$, cadeia é rejeitada;

vi. Verifica se símbolo deve ser consumido. Caso negativo, retorna ao passo 2a.

Utilizando o automato da Figura 5 como exemplo, para uma cadeia de entrada cujo primeiro símbolo é 0, as seguintes ações são realizadas:

1. Executa as ações informadas pela regra inicial $r0$, mudando o estado corrente para A.

2. Pela tabela de decisão, a condição atual $(A, 0)$ é satisfeita pela regra $r1$. Também é observado que esta regra possui uma referência para X que deve ser executada com os parâmetros $p1 = A$ e $p2 = B$ antes de aplicar a regra. Assim, de acordo com suas operações elementares, as seguintes ações adaptativas são executadas:

   (a) Cria uma nova regra $r11$, cuja condição é $(p1, 0)$, e que mude o estado do autômato para um novo estado, referente ao gerador $g1$. Resolvendo os valores de $p1$ e $g1$, a nova regra representa uma transição $(A, 0) \rightarrow \#1$, onde $\#1$ é o valor gerado por $g1$.

(b) Cria uma nova regra $r12$ que representa a transição $(\#1, 0) \rightarrow B$. Esta regra é adaptativa, e chama a função X com $p1 = \#1$ e $p2 = B$.

(c) Cria uma nova regra $r13$ que representa a transição $(\#1, 1) \rightarrow A$.

(d) Remove a regra cuja condição é $(A, 0)$ e ação é atualizar o estado atual para B. Consultando a tabela, esta regra existe ($r1$), logo, é removida.

3. Após a ação da função adaptativa X, o autômato passa a ter uma nova configuração, ilustrada pela Figura 6.

4. Como a regra $r1$, que seria aplicada, foi excluída pela ação da função adaptativa, deve-se voltar ao passo 2a e procurar outra regra aplicável, agora, no dispositivo após as mudanças. Pela nova configuração, há outra regra aplicável, $r11$, que transita de A para $\#1$, consumindo o símbolo 0.

Iterando este processo até consumir toda a cadeia, a tabela deve chegar a uma ação de aceitação ou rejeição. Por exemplo, a cadeia $0110 \vdash$ é aceita, disparando as regras $r11$, $r13$, $r2$. $r4$ e $r7$, enquanto a cadeia 11 é rejeitada.

2.1.4 Tabela de Decisão Adaptativa Estendida

Em sua definição original, um dispositivo adaptativo é capaz de alterar seu conjunto de regras como resposta a um estímulo externo, utilizando, para tanto, funções adaptativas (NETO, 2001). Contudo, o poder de expressão destas funções é limitado a três operações adaptativas elementares, e dificulta a implementação de ações mais complexas (TCHEMRA, 2009).

Ainda, um dispositivo adaptativo típico interrompe sua execução quando é confrontado com uma situação em que não há regras aplicáveis ao problema. No exemplo de um autômato finito, a cadeia de entrada é rejeitada

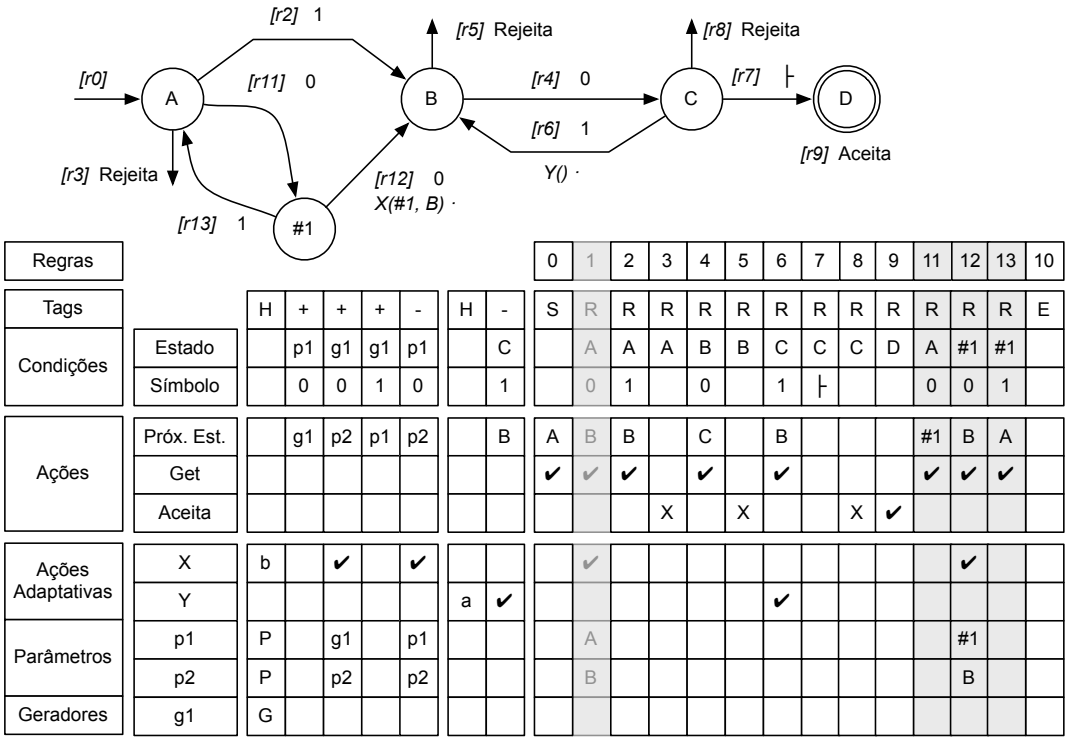


Figura 6: Evolução de autômato da Figura 5 após consumir a subcadeia 0. Em destaque, a regra $r1$ recém-excluída e as três novas regras.

(NETO, 2001), enquanto em dispositivos de operação contínua (i.e., sem configurações de aceitação e rejeição), esta situação indica que o problema não era previsto pelo dispositivo. Tal comportamento impede que o dispositivo se adapte a novos problemas, visto que nenhuma regra é aplicada e, portanto, nenhuma função adaptativa é executada.

A Tabela de Decisão Adaptativa Estendida (*TDAE*) é uma variante da *TDA* formalizada por Tchemra em 2009 em que a adaptatividade não se manifesta unicamente durante a aplicação de uma regra, mas também na ausência de regras aplicáveis: funções auxiliares são utilizadas para consultar um dispositivo auxiliar M. A solução obtida por este dispositivo, então, é incorporada à tabela na forma de uma nova regra, permitindo que, subsequentemente, o mesmo problema possa ser solucionado sem precisar utilizar o dispositivo secundário (TCHEMRA, 2009). O método estudado por Tchemra, no caso, foi o

AHP (Analytic Hierarchy Process), desenvolvido por Saaty na década de 1970 (SAATY; VARGAS, 2000).

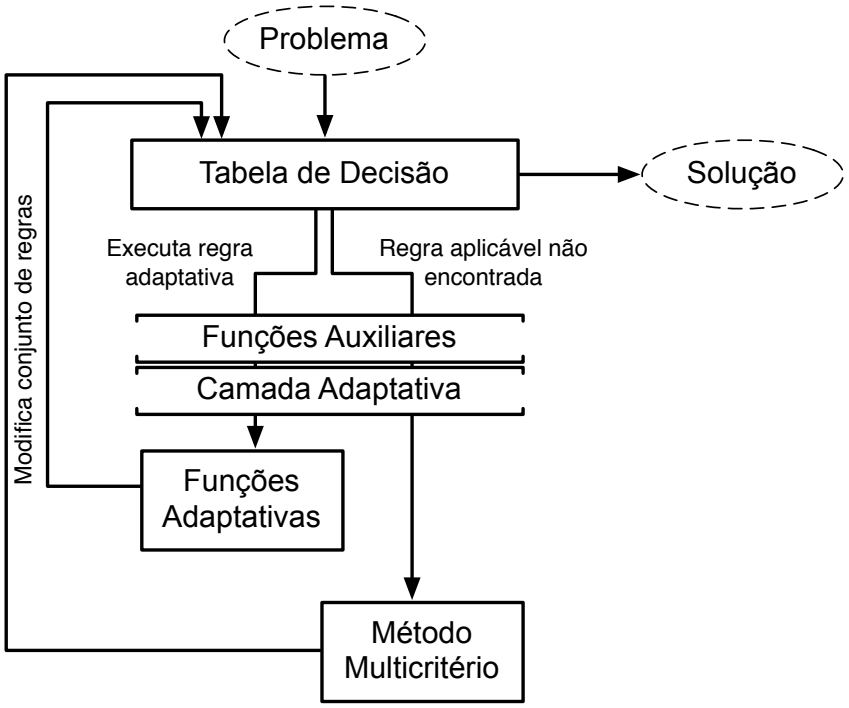


Figura 7: Diagrama conceitual de uma *TDAE*, ilustrando o uso das funções auxiliares e o método multicritério auxiliar

As funções auxiliares agem como um intermediário entre a tabela de decisão subjacente e sua camada adaptativa, permitindo que operações mais complexas possam ser executadas - em particular, permitem o acesso a um segundo dispositivo M - flexibilizando a descrição das operações a serem executadas pela camada adaptativa. O elemento M, no caso, é especificado de forma genérica, podendo assumir um método multicritério qualquer. Assim, a *TDAE* poderia ser descrita como $TDAE = (TDA, FM, M)$, onde FM representa o conjunto de funções auxiliares.

Nota: apesar de sua concepção estar voltada para casos em que nenhuma regra é aplicável, funções auxiliares podem, em teoria, ser chamadas por regras convencionais para executar as mesmas operações realizadas por fun-

ções adaptativas convencionais, mas de maneira mais simples. Ainda que o poder computacional não seja aumentado, a expressividade dos dispositivos (i.e., facilidade de utilização por aderência ao problema) aumenta consideravelmente.

		Ações Adaptativas					Regras					
		...	AD_z			...	r_1	...	r_j	...	r_n	E
Condições	c_1						Valores das condições					
	...											
	c_i											
	...											
	c_m											
Ações	a_1						Ações a serem tomadas					
	...											
	a_k											
	...											
	a_p											
F. Auxiliares	FM_1						Funções auxiliares a serem chamadas					
	...											
	FM_h											
	...											
	FM_q											
Funções Adaptativas	FAD_1						Ações adaptativas a serem tomadas					
	FAD_2											
	...											
	...											
	...											
	FAD_z						Referências a FAD_z					
	...											
	FAD_{nf}											

Tabela 5: Estrutura típica de uma tabela de decisão adaptativa estendida (*TDAE*), que define novas linhas para funções auxiliares.

A *TDAE* foi uma das primeiras aplicações da adaptatividade voltada para sistemas híbridos, em que múltiplos dispositivos, de naturezas distintas, são integrados para realizar a tomada de decisão de forma colaborativa. Sua estrutura permite que um dispositivo aprenda incrementalmente com as soluções obtidas através dos demais, evoluindo não só através de computações realizadas internamente - i.e., funções adaptativas pré-programadas - mas também através de eventos externos (note que, no lugar do *AHP*, poderia haver um

operador humano, que cumpriria o papel de Oráculo).

Estas contribuições são exploradas neste estudo como forma de criar um modelo para a construção de sistemas híbridos, em que um dispositivo, quando incapaz de solucionar um problema de forma confiável, consulta outro para obter uma solução, que então, é incorporada na forma de um novo conhecimento.

2.1.5 Estado Atual

Atualmente, a maior parte dos trabalhos relacionados à adaptatividade se concentra na sua aplicação em diversos ramos de pesquisa, utilizando as teorias já consolidadas como fundamentação - note que a teoria que abrange a adaptatividade, apesar de formalizada no caso geral em 2001, já havia sido estudada em casos particulares desde a década de 1980. Linguagens naturais, autômatos adaptativos, redes de Markov, aprendizagem de máquina e tomada de decisão são exemplos de bases teóricas que, hoje, são aplicadas para uso prático.

No campo de automação robótica, o uso da adaptatividade permite que robôs consigam navegar em um ambiente de forma autônoma através de autômatos concisos e de baixa complexidade computacional (HIRAKAWA; SARAIVA; CUGNASCA, 2007). O estudo de processamento de linguagens naturais também possibilita a construção de robôs capazes de gerenciar diálogos homem-máquina, demonstrando um comportamento supostamente criativo (ALFENAS; PEREIRA-BARRETTO, 2012).

Ramos de mineração de dados também tem utilizado a adaptatividade como forma de simplificar o problema de tratar um volume elevado de informações, como exemplo, na identificação de indivíduos através de suas carac-

terísticas (CEREDA; NETO, 2012). Também se torna possível identificar padrões através de operações de convolução, onde a escolha do núcleo da operação é feita baseada em soluções anteriores (CAMARGO; RAUNHEITTE, 2011). Foi também idealizado um sistema capaz de recomendar novos locais para um indivíduo visitar, baseado no contexto em que encontra (e.g., locais diferentes podem ser recomendados, dependendo se o indivíduo está trabalhando, ou então, de férias) (CRIVELARO; ROCHA, 2012). Também é possível extrair um comportamento preditivo, baseado em informações passadas, conforme mostra o sistema adaptativo de previsão de tempo descrito em (VALLE; ROCHA, 2011).

Finalmente, com o contínuo amadurecimento da adaptatividade, torna-se possível criar ambientes para o desenvolvimento de sistemas dotados de técnicas adaptativas. Um dos primeiros exemplos conhecidos é o *AdapTools*, que permite criar, inspecionar e executar autômatos adaptativos de forma visual (JESUS et al., 2007). Outro exemplo, mais recente, foi o desenvolvimento do *MMAdapt* (ALFENAS et al., 2012), uma plataforma de desenvolvimento de sistemas baseados em redes de Markov adaptativas - trabalho inspirado no sistema gerador de músicas em tempo real desenvolvido por Basseto, que também faz uso de redes de Markov, mas para um uso particular (BASSETO; NETO, 1999). Também se destaca a especificação de uma linguagem de programação de alto nível, capaz de expressar algoritmos adaptativos de forma nativa (SABALIAUSKAS; ROCHA, 2011; SILVA, 2011). Por fim, plataformas para fins didáticos também tem sido desenvolvidas, como forma de facilitar a introdução deste tipo de técnica em meios acadêmicos (SANTIBAñEZ; NETO, 2011).

2.2 Tomada de Decisão

Tomada de decisão é um processo cognitivo em que uma ação deve ser tomada como resposta a um estímulo externo, normalmente caracterizado por um conjunto finito de m *condições* - comumente descritos como *critérios* ou *atributos* (CROZIER; RANYARD, 1997). Normalmente, a melhor ação deve ser escolhida dentre um conjunto finito de p *alternativas*, determinando aquelas que melhor satisfazem o objetivo a ser alcançado.

Apesar de ser frequentemente mencionada em áreas de pesquisa de sistemas especialistas - casos como medicina e sistemas gerenciais - a tomada de decisão pode ser encontrada em situações do próprio dia-a-dia. Como exemplo, um problema de escolher o que vestir depende de inúmeros fatores: desde o local para onde estamos indo - vestuário para um casamento e para ir à praia são bem diferentes - até as condições climáticas.

Esta seção apresenta os conceitos acerca de tomada de decisão, bem como os aspectos psicológicos que influenciam a maneira com que as decisões são tomadas e que, frequentemente, são difíceis de serem representados em um modelo computacional.

2.2.1 Descrição

Formalmente, um problema de tomada de decisão pode ser descrito por um vetor m-dimensional $x = \left(x_{[1]}, \ldots, x_{[m]}\right)$, onde $x_{[i]}$ é o valor associado ao i-ésimo critério. A solução do problema - a ação a ser tomada - é dada por uma função $y\left(x\right)$, particular de cada método de tomada de decisão.

Tipicamente, os critérios podem ser divididos em dois tipos básicos: nominais (também ditos categóricos ou discretos) e contínuos (i.e. numéricos):

- Critérios nominais são aqueles que atribuem um rótulo (do inglês *label*) para definir seu valor. O número de rótulos que podem ser atribuídos deve ser finito, e definido a priori de acordo com o atributo medido. Como exemplo, o critério *tempo* poderia assumir valores tais como *ensolarado*, *nublado*, *chuvoso*, etc.

- Em comparação, critérios contínuos são associados a valores numéricos reais. Os limites superior e inferior são normalmente impostos pela grandeza que se está medindo. Como exemplo, o critério *temperatura* poderia assumir qualquer valor real, desde que acima de 0 K - teoricamente, o menor valor possível para a temperatura.

Deve-se notar que nem todos os métodos de classificação, tomada de decisão e de aprendizagem são capazes de tratar ambos os tipos de critérios. Como exemplo, árvores de decisão geradas pelo uso do algoritmo ID3 (QUINLAN, 1986) solucionam problemas que contenham apenas atributos nominais, ao passo que aquelas construídas através do algoritmo C4.5 - uma evolução do ID3 - lidam com ambos os tipos (QUINLAN, 1996). Em comparação, métodos como *SVM* (*Support-Vector Machines*) tratam apenas problemas estritamente numéricos (CORTES; VAPNIK, 1995).

Tipicamente, os métodos de tomada de decisão automaticamente rejeitam problemas cujos critérios utilizam tipos não suportados. Uma forma de evitar este problema é convertendo os critérios de tipos não suportados para outros, tratáveis pelo método:

- Valores numéricos podem ser categorizados para um conjunto finito de valores nominais, no qual cada nome representa um intervalo do espaço numérico. Este processo é conhecido como discretização. Uma técnica básica de discretização é o *EWD* (*Equal Width Discretization*) (YANG;

WEBB, 2002), que divide o espaço de valores numéricos em n bandas de igual largura. Outros métodos incluem o *RMEP* (*Recursive Minimal Entropy Partitioning*) e métodos de geração de clusters (DOUGHERTY; KOHAVI; SAHAMI, 1995).

  - A Figura 8 ilustra um exemplo de um espaço numérico discretizado em 5 bandas, cada uma representando uma categoria do novo critério nominal. Note que a largura de cada banda do *EWD* se mantém constante, enquanto no cluster, a largura depende da concentração de pontos.
  - Idealmente, é preferível utilizar métodos capazes de determinar um número ótimo de intervalos, bem como seus respectivos tamanhos e posições (KOTSIANTIS; KANELLOPOULOS, 2006). De maneira geral, tais métodos fazem uso de informações adicionais além da disposição espacial dos valores a serem discretizados, tal como a classe a que cada valor pertence, tornando-os supervisionados.

- O processo contrário, em transformar um critério nominal em numérico, pode ser feito substituindo-o por um conjunto de critérios numéricos com valores binários mutuamente exclusivos, cada um representando uma de suas categorias (WITTEN; FRANK, 2005). Como exemplo, um critério nominal formado pelas categorias *Alice*, *Bob* e *Carol* dá origem a 3 critérios numéricos binários. Se o valor do original era *Alice*, o valor do critério numérico que representa *Alice* tem valor 1, enquanto os demais recebem 0.

Mesmo métodos que tratem ambos nativamente podem se beneficiar de uma conversão: um caso estudado foi o classificador de Bayes, que produz resultados melhores quando seus atributos numéricos são discretizados (YANG;

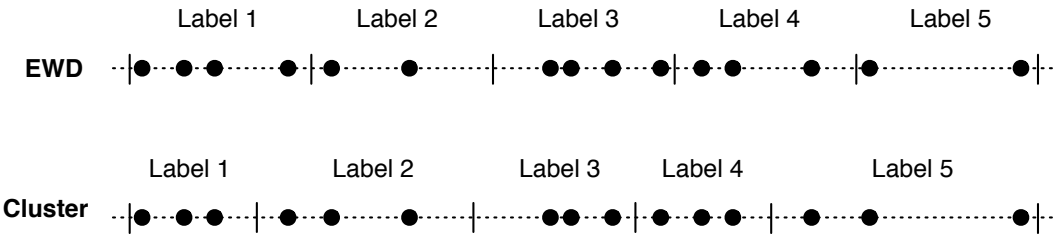


Figura 8: Exemplificação da diferença entre a discretização via *Equal Width Discretization* e clusterização, com 5 categorias.

WEBB, 2002) - note que o classificador de Bayes precisa conhecer a distribuição dos valores de cada critério numérico para operar com precisão, enquanto esta informação torna-se menos relevante em critérios nominais, estabilizando a tomada de decisão.

Pode-se salientar que o problema da tomada de decisão é semelhante ao problema da classificação, uma vez que ambos descrevem a entrada como um vetor x de m dimensões, devendo retornar um nome $y(x)$ como resposta - a tomada de decisão interpreta este nome como uma ação (alternativa) a, enquanto a classificação entende que é a classe c. Como exemplo, tabelas de decisão podem ser utilizadas em problemas de classificação, enquanto o *k-Nearest Neighbor*, um algoritmo tipicamente de classificação, pode ser utilizado para tomada de decisão. Contudo, deve-se verificar que se o retorno não é um nome, mas sim, um valor numérico, o problema passa a ser de regressão.

Por fim, vale observar que a notação da função $y(x)$ implica que apenas uma alternativa deve ser adotada como solução, não fornecendo informações a respeito da possibilidade de as demais alternativas também serem possíveis soluções. Uma forma de contornar este problema é retornar, ao invés de uma única solução com 100% de certeza, um vetor p-dimensional $\mu = (\mu_1, \mu_2, \ldots, \mu_p)$, onde cada valor individual μ_i representa a expectativa de

certeza da i-ésima alternativa. No caso particular de haver uma única solução, a alternativa adotada é $\arg\max(\mu_1, \mu_2, \ldots, \mu_p)$.

Para que as expectativas possam ser comparadas entre si de forma padronizada, o vetor μ deve ser normalizado - i.e., a soma de todos os elementos deve ser 1. Logo:

$$\sum_{i=1}^{p} \mu_i = 1 \tag{2.1}$$

2.2.2 Aspectos Psicológicos da Tomada de Decisão

Ainda que considerado um processo puramente racional, a tomada de decisão é um processo cognitivo (BELL; RAIFFA; TVERSKY, 1988), influenciável por inúmeros fatores mentais, tais como experiências passadas e valores pessoais (KOLODNER, 1991). Mesmo emoções demonstram grande influência, sendo fundamentais para tomar boas decisões mediante riscos ou incertezas (BECHARA, 2003). Entre os fatores emocionais que podem influenciar a tomada de decisão, pode-se citar memória seletiva (as pessoas comumente ignoram soluções que são contrárias a seus valores (PLOUS, 1993)), inércia (dificuldade de mudar de opinião), otimismo (CHUA et al., 2004) e preconceito.

Um exemplo pode ser encontrado no jogo do ultimato, cujas regras são simples: um jogador, referenciado como *líder*, recebe um prêmio da banca do jogo. Ele deve reparti-lo com um segundo jogador, o receptor, oferecendo-lhe uma proposta de divisão do prêmio. Se o receptor aceitá-la, ambos repartem o prêmio conforme combinado. Caso contrário, se o receptor não aceitar, ambos ficam sem o prêmio. Um raciocínio frio indica que, para o receptor, é mais vantajoso sempre aceitar a proposta, pois, por menor que seja a oferta, é melhor que recusar e ficar sem o prêmio. Contudo, experimentos mostram que ofertas consideradas injustas, em que o líder fica com a maior parte do

prêmio, são frequentemente rejeitadas (SANFEY et al., 2003).

Há também situações em que o resultado não depende unicamente da decisão do indivíduo, mas das decisões tomadas por terceiros. Assim, a tomada de decisão deve ser estratégica, de forma a confrontar estes oponentes e visando maximizar seu ganho próprio. A esta forma de tomada de decisão, dá-se o nome de teoria dos jogos (MYERSON, 1991). Situações modeladas pela teoria dos jogos demonstram que a decisão considerada mais racional nem sempre é a melhor.

Um exemplo conhecido da teoria dos jogos é o dilema dos prisioneiros (POUNDSTONE, 1992). Neste dilema, imagina-se uma situação em que dois suspeitos, A e B, são presos pela polícia e são mantidos em cárceres isolados, sem ter como realizar contato um com o outro. Como a polícia não tem provas suficientes para condená-los, oferece a ambos duas opções: testemunhar contra o outro prisioneiro ou manter-se em silêncio. Caso um prisioneiro testemunhe e o outro fique em silêncio, aquele que testemunhou é libertado, enquanto o outro é preso por 1 ano. Caso ambos testemunhem, os dois são presos por 3 meses, e por fim, se ambos manterem-se em silêncio, permanecem presos por 1 mês.

		Prisioneiro *B*	
		Silêncio	Testemunhar
Prisioneiro *A*	Silêncio	*A*: 1 mês de prisão *B*: 1 mês de prisão	*A*: 1 ano de prisão *B*: sai livre
	Testemunhar	*A*: sai livre *B*: 1 ano de prisão	*A*: 3 meses de prisão *B*: 3 meses de prisão

Tabela 6: Descrição do dilema dos prisioneiros.

Como o prisioneiro deseja minimizar seu prejuízo, independentemente da decisão do outro, a opção de testemunhar parece mais sensata, uma vez

que seu prejuízo, no máximo seria 3 meses de prisão, enquanto no melhor caso, ficaria até livre. Se ambos raciocinarem desta maneira, ambos ficarão 3 meses presos, enquanto, paradoxalmente, a decisão que minimiza a perda global é quando ambos se mantêm em silêncio, e neste caso, apenas 1 mês de prisão é dada. Mesmo que os prisioneiros pudessem negociar entre si para manter-se em silêncio, nada garante que cumpririam o acordo - neste caso, trair este acordo poderia ser muito vantajoso para sair livre (MULLEN, 1990).

Outro aspecto psicológico capaz de afetar a tomada de decisão é a aversão ao risco. Em geral, as pessoas preferem tomar decisões que sejam mais estáveis do que arriscar uma decisão em que existe a possibilidade de haver uma perda ou redução de ganhos (KAHNEMAN; TVERSK, 1979). Um caso simples é uma aposta com uma moeda, em que um jogador tem duas possibilidades: ganhar $100 imediatamente ou então, apostar em um jogo de cara ou coroa em que, caso ganhe, levará $300, mas se perder, nada receberá. Teorias normativas levariam a escolher a aposta, já que, na média, o lucro seria de $150, mas na prática, os jogadores preferem não arriscar e levam os $100 (MULLEN, 1990).

Por fim, um ponto de importância é relevar a idade de uma informação durante a tomada de decisão. Assim como a inercia dificulta a mudança de opinião quando uma informação é consolidada no conhecimento (i.e., eventos mais antigos têm maior importância em relação a informações mais novas), há casos em que as pessoas, por intuição, acabam dando mais valor para eventos mais recentes, ignorando ou até mesmo se esquecendo de eventos mais antigos (PLOUS, 1993). Este comportamento permite que a tomada de decisão se adeque a situações em que os conceitos se alterem ao longo do tempo - a esta propriedade dá-se o nome de desvio de conceito (*concept drift*). Caso estas mudanças não sejam levadas em consideração, a taxa de acerto de um

sistema de tomada de decisão acaba se deteriorando ao longo do tempo. No entanto, detectar tais mudanças não é trivial, uma vez que elas podem ocorrer pontualmente, gradativamente ou abruptamente. Podem até mesmo ser confundidas com ruído ou caso raro (KUNCHEVA, 2009). Levar em conta o desvio de conceito implica não só a necessidade de aprendizagem, mas também de um meio de se esquecer um conhecimento obsoleto - processo ainda pouco explorado.

Na área computacional, um dos desafios a serem superados é o desenvolvimento de um modelo de tomada de decisão semelhante ao do homem, capaz de considerar estes aspectos durante as fases de aprendizado e execução. Devido à sua complexidade, estudos passados exploram aspectos individuais para obter novos modelos de decisão. Em particular, um aspecto considerado na presente dissertação é o nível de confiança dado a uma informação conhecida, e como utilizá-lo para tomar decisões melhores, evitando utilizar informações pouco confiáveis.

2.2.3 Nível de Confiança

Tipicamente, processos computacionais de tomada de decisão partem do pressuposto de que o conhecimento analisado é correto, dispensando assim sua validação. Apesar de operar bem para informações obtidas via treinamento, esta premissa torna-se menos atraente para informações obtidas através de outros meios em que a veracidade da informação não possa ser verificada por completo. Humanos, naturalmente, têm uma crença de que a informação é correta, e a utilizam para avaliar se a informação pode ou não ser utilizada de forma segura. Segundo Parsloe (PARSLOE; WRAY, 2000), dentre os fatores que afetam esta confiança, destacam-se:

- **Repetição**: uma afirmação repetida múltiplas vezes torna-se mais facilmente aceita como verdadeira;
- **Realimentação**: opiniões (favoráveis ou contrárias) de agentes externos podem interferir na forma como cada um vê o problema;
- **Opinião própria**: cada indivíduo pode possuir uma opinião formada sobre a solução de um problema, seja ela por convicção própria, por desconhecimento de outra alternativas, ou por observações empíricas.

De maneira geral, quanto mais repetimos uma ação em relação a um determinado problema, mais acreditamos que tal ação seja a mais adequada. A este comportamento, dá-se o nome de viés de repetição (*repetition-bias*). Opiniões contrárias, por outro lado, nos ajudam a evitar que a aplicação de uma ação inadequada seja repetida no futuro (PARSLOE; WRAY, 2000). O inverso também é verdade: reforçar uma opinião é um incentivo para continuar com a crença de que tal opinião seja a correta.

Estudos passados sugerem que o nível de confiança possa ser modelado em função das realimentações (*feedbacks*) positivas e negativas recebidas. Einhorn sugere que a confiança C de uma informação é uma função baseada na soma F das realimentações recebidas para tal informação (EINHORN; HOGARTH, 1978):

$$\begin{cases} C = f(F) \\ F = \beta_+ N_+ + \beta_- N_- \\ \beta_+ + \beta_- = 1.0 \qquad (\beta_+, \beta_- \leqslant 1.0) \end{cases} \tag{2.2}$$

Neste modelo, os valores de N_+ e N_- são acumuladores de realimentações positivas e negativas, respectivamente. Já β_+ e β_- são valores relativos

de reforço, que ponderam a influência de cada tipo de realimentação. A função $f(F)$ não foi definida originalmente, mas espera-se que seja uma função que retorne uma confiança baixa quando $F \sim 0$, e devendo saturar em 1.0 quando $|F| \to \infty$. Caso $F > 0$, a confiança será de que a informação esteja correta, enquanto $F < 0$ indica que a confiança é de que a informação esteja incorreta. Note que F representa um estimador das experiências obtidas através das realimentações originadas a partir do uso da informação em questão (EINHORN; HOGARTH, 1978).

Além da quantidade de realimentações recebidas, outros dois fatores com forte influência no nível de confiança são a credibilidade da fonte da realimentação e a intensidade com que a realimentação é realizada - quanto maiores os efeitos causados por um evento, mais forte será a evidência a ser realimentada (GRIFFIN; TVERSKY, 1992). Em geral, eventos de alta intensidade e baixa credibilidade causam excesso de confiança, e então, um indivíduo superestima a possibilidade de estar certo. Do contrário, evidências de baixa intensidade e alta credibilidade fazem com que a possibilidade de estar certo fique subestimada.

Posteriormente, um experimento realizado por Westbrook, Gosling e Colera demonstra a dificuldade de relacionar o nível de confiança com a possibilidade de se estar correto, uma vez que nada impede que o tomador de decisão confie nas evidências que o levem a uma resposta incorreta, mesmo que ele próprio seja especialista no assunto - no caso, médicos, clínicos e enfermeiros (WESTBROOK; GOSLING; COIERA, 2005). Neste mesmo trabalho foi observado o efeito do excesso de confiança, em que médicos - quando comparados com enfermeiros - subestimavam a possibilidade de estarem errados.

O uso de realimentações para moldar a confiança de um resultado pode ser encontrado em determinadas aplicações que fazem uso de aprendizagem

de máquina. Um exemplo é o *CBIR* (*content-based image retrieval*), um sistema de busca de imagens que faz uso de *feedbacks* dos usuários para refinar sua pesquisa, retornando os resultados que sejam mais relevantes e eliminando possíveis erros (DING; LI; YANG, 2004). Contudo, deve-se ressaltar que não existe um modelo formal e consolidado de como modelar a confiança através de recursos computacionais, ainda que modelos individuais tenham sido formalizados.

2.3 Raciocínio Baseado em Casos

Case-Based Reasoning (raciocínio baseado em casos, ou *CBR*) é um paradigma de resolução de problemas em que, diferentemente das técnicas clássicas de inteligência artificial - estas fazem uso de um conhecimento voltado ao domínio do problema - utilizam-se informações obtidas através de experiências passadas (AAMODT; PLAZA, 1994). Novos problemas são solucionados buscando casos passados similares, e reutilizando suas soluções (SMYTH; CUNNINGHAM, 1992), partindo do princípio que problemas semelhantes devem, de alguma forma, apresentar soluções também semelhantes.

Estudos realizados indicam que o uso de experiências passadas tem grande importância na tomada de decisão humana, especialmente no início de um novo aprendizado, quando ainda não se tem uma compreensão adequada do problema (ANDERSON, 1983). Mesmo processos como inferência por analogia aparentam fazer uso de casos passados para solucionar novos problemas (GENTNER, 1983). Evidências empíricas também indicam que, na ausência de informações concretas, o uso de casos conhecidos é uma das ações mais frequentes para tomar uma decisão (BALAKRISHNAN; RATCLIFF, 1996). Por fim, problemas tipicamente pessoais, como os encontrados no cotidiano, de senso comum e de julgamento estético, também são solucionáveis aplicando-se a

técnica de raciocínio baseado em casos (KOLODNER, 1991).

Outra importante característica desta forma de raciocínio é a possibilidade de se realizar um processo de aprendizado incremental, no qual experiências são agregadas uma de cada vez, tornando-as disponíveis para novos problemas (AAMODT; PLAZA, 1994). Schank já idealizava uma memória dinâmica que permitia agregar conhecimento de forma incremental (SCHANK, 1982). Göker inclusive verifica que a aquisição incremental permite que o método se adapte a problemas dinâmicos, para os quais soluções previamente aceitas podem ficar obsoletas após um intervalo suficientemente grande de tempo (GÖKER; ROTH-BERGHOFER, 1999).

Em termos de metodologia, o raciocínio baseado em casos pode ser dividido em quatro etapas (AAMODT; PLAZA, 1994), conforme ilustra a Figura 9. Estas etapas são referenciadas como *4R* em função das iniciais de seus respectivos nomes:

1. **Retrieve** - Busca de casos semelhantes ao problema a ser solucionado. Um *caso* é tipicamente formado pelo problema previamente encontrado e a *solução* adotada;

2. **Reuse** - Reuso e adaptação dos casos passados para que sejam compatíveis com o novo problema. No caso mais simples, pode-se simplesmente reutilizar as soluções encontradas diretamente, sem adaptá-las;

3. **Revise** - Aplicar a solução adotada e, se possível, revisá-la, eliminando possíveis efeitos indesejáveis encontrados na solução;

4. **Retain** - Se a solução adotada é avaliada como útil, ela pode ser integrada ao conhecimento já adquirido, tonando-se uma nova referência para a resolução de problemas futuros.

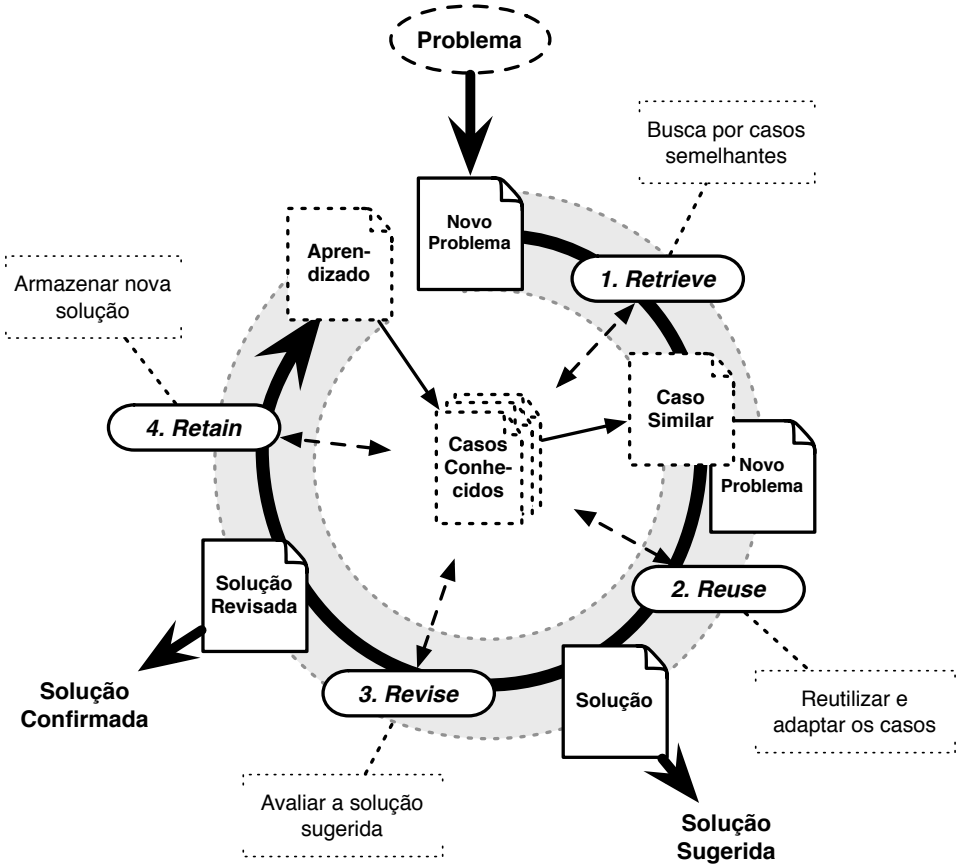


Figura 9: Ciclo de vida *4R* (*Retrieve*, *Reuse*, *Revise* e *Retain*) do *CBR*.

Note que as duas últimas etapas, ainda que desejáveis, não são indispensáveis para seu funcionamento, apesar de que sua ausência poderia ocasionar uma perda da capacidade evolutiva. Outras definições, como a apresentada por Leake, Kinley e Wilson (LEAKE; KINLEY; WILSON, 1995), utilizam apenas as duas primeiras etapas, sob os nomes de *Memory Search* e *Adaptation*, respectivamente.

Para que a busca seja feita, contudo, torna-se necessário criar uma métrica que estime a similaridade entre dois casos, o que pode ser dependente do domínio do problema a ser solucionado. De forma parecida, adaptar os casos encontrados requer algum conhecimento do problema, apesar de que, no caso mais simples, seja possível utilizar a solução diretamente, sem adap-

tações.

Revisar uma solução adotada também apresenta certas imposições: nem sempre é possível certificar-se de que a solução proposta foi realmente adequada, uma vez que pode haver uma latência até que eventuais efeitos colaterais causados pela solução se manifestem (BAKKEN, 1993). Também é preciso levar em consideração a forma como a avaliação deve ser feita, já que ela pode ser realizada por diversos agentes, desde inteligência artificial até operadores humanos (ou *experts*), cada qual com diferentes graus de credibilidade e disponibilidade. Assim, antes de a aquisição de novos casos ser feita, o método de raciocínio baseado em casos deve ser alimentado com informações sobre o quão confiável e correta a nova solução é, a fim de evitar um aprendizado errôneo. Este aspecto é abordado na seção 2.2.3.

2.4 Aprendizagem de Máquina

Aprendizagem de máquina é um ramo da inteligência artificial orientado para o desenvolvimento de algoritmos que permitam que sistemas computacionais consigam evoluir através da aquisição de novas informações ao longo do tempo. Evoluir, neste caso, significa que seu desempenho deve melhorar à medida que novas informações são incorporadas ao conhecimento. Uma definição frequentemente utilizada é a que foi proposta por Mitchell:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E (MITCHELL, 1997).*

Tipicamente, aprendizagem de máquina se traduz na tarefa de compilar dados obtidos empiricamente, organizando e extraindo informações úteis e

gerando uma abstração que represente, de forma compacta, a informação contida na massa de dados original (e.g., árvore de decisão). Este processo é comumente referenciado como *inferência*, e pode ser identificado em duas vertentes clássicas:

- O aprendizado pode ocorrer através de um treinamento **supervisionado**, em que um conjunto de instâncias rotuladas é utilizado como entrada - conforme ilustrado pela Figura 10. Cada *instância* é formado por:
  - Um vetor de características (*feature vector*) $x = \left(x_{[1]}, \cdots, x_{[m]}\right)$, em que $x_{[i]}$ é o valor associado ao i-ésimo critério;
  - Uma alternativa (saída ou rótulo) esperada a. Instâncias com alternativas conhecidas são chamadas de *rotuladas*.
- A partir deste conjunto de treinamento (também chamado de *batch*), deve-se inferir a relação entrada-saída, resultando na criação de uma abstração representada pela função $y\left(x\right)$ que retorna a alternativa correspondente para o vetor x.
- De forma contrária, aprendizado **não supervisionado** não especifica um rótulo para cada instância. O objetivo da inferência, neste caso, não é encontrar uma abstração que represente a relação entrada-saída - até porque os dados não são rotulados. A meta é encontrar padrões e/ou estruturas escondidas dentro das instâncias, que possam conter algum significado antes desconhecido. Geração de clusters é um exemplo clássico deste tipo de aprendizagem.

Na maior parte dos sistemas de apoio à decisão, a inferência da abstração inicial é feita a partir de um treinamento supervisionado, fazendo uso de instâncias obtidas de medidas reais, ou então, fornecidas por um especialista.

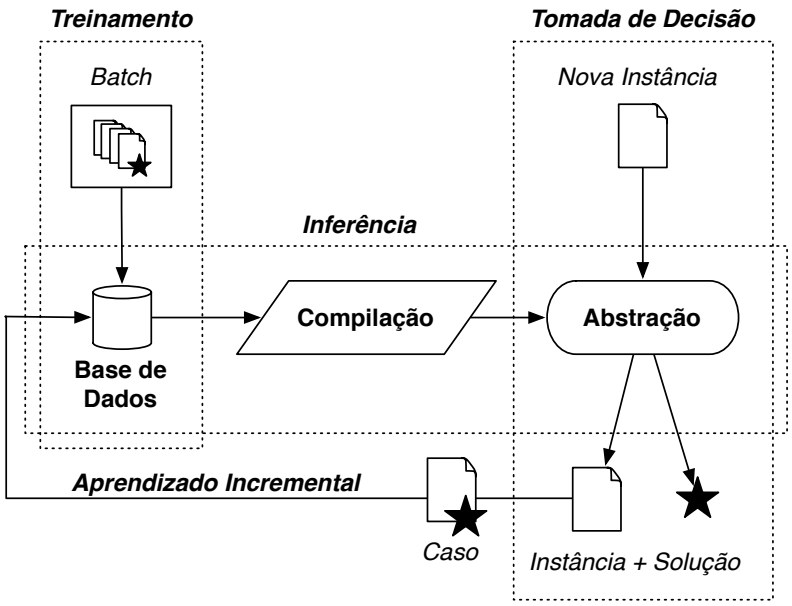


Figura 10: Relação entre aprendizado supervisionado e a tomada de decisão.

A avaliação do sucesso de um treinamento pode ser feita testando a abstração recém-criada com um conjunto de instâncias rotuladas. Um acerto ocorre quando a solução estimada por $y\,(x)$ coincide com a solução verdadeira a. A taxa de acerto obtida nesta avaliação fornece uma medida objetiva da qualidade do aprendizado realizado (WITTEN; FRANK, 2005).

Em contrapartida, o aprendizado não supervisionado pode ser utilizado para encontrar lacunas deixadas pela abstração inicial, buscando relações entre as instâncias cuja solução não pode ser obtida. Um padrão encontrado nesta etapa pode significar uma nova alternativa, antes não conhecida ou não especificada pelo treinamento inicial.

2.4.1 Aprendizado Online

Classicamente, sistemas de classificação, regressão e de tomada de decisão são treinados utilizando-se um *batch* - conjunto de instâncias já rotuladas com sua alternativa esperada. A inferência da abstração é feita buscando-se

um padrão que possa representar, de forma simplificada, o conjunto original de instâncias, com um erro mínimo nas respostas.

Porém, apesar de ser simples e direto, esse método não permite que a abstração seja atualizada durante sua execução, impedindo-a de tratar problemas dinâmicos, cujos conceitos são variáveis ao longo do tempo (KUNCHEVA, 2009). A única maneira de mantê-la atualizada é refazendo todo o processo de treinamento, desta vez utilizando um conjunto de instâncias mais recente. Devido ao alto custo computacional envolvido neste tipo de operação, não é viável executar esta tarefa com o sistema em atividade, fazendo-se necessário construir um novo sistema para substituir o anterior. Por ocorrer fora do ciclo normal de execução do sistema, este aprendizado é categorizado como *offline*.

Em contrapartida, *aprendizado online* é um paradigma de aprendizagem em que, ao invés de utilizar um *batch* para treinamento, permite que as instâncias rotuladas sejam fornecidas uma de cada vez. Apesar de serem conceitualmente mais complexos e desafiadores, sistemas treinados através deste paradigma podem receber atualizações de forma incremental à medida que novas instâncias forem sendo disponibilizadas (BABENKO; YANG; BELONGIE, 2009). O aprendizado online é especialmente atraente para sistemas com fluxo contínuo de dados, em que a resposta do sistema no instante seguinte depende fortemente das respostas nos instantes anteriores, fazendo-se necessário que o sistema se mantenha sempre atualizado (e.g., rastreamento visual de objetos).

O estudo de algoritmos com capacidade de aprendizado incremental é relativamente recente, concentrando-se na adaptação de algoritmos clássicos para que realizem um aprendizado online. Um exemplo conhecido é o *ITI* (*Incremental Tree Inducer*), que, ao contrário de técnicas como ID3 e C4.5,

permite construir árvores de decisão atualizáveis, apresentando operações de inserir e remover instâncias (UTGOFF; BERKMAN; CLOUSE, 1997). Outro exemplo é a versão online do popular *Random Forest*, que permite inserir novos nós durante sua execução (SAFFARI et al., 2009). Todavia, apresenta uma limitação de trabalhar apenas com valores numéricos, ao contrário do método original, que também permite o uso de critérios nominais.

Outros exemplos incluem versões online do *SVM* (*Support Vector Machine*), *AdaBoost* e *MIL* (*Multiple Instance Boosting*) (SAFFARI et al., 2009). Note que o termo *online* é costumeiramente utilizado em métodos que, em sua versão original, não apresentam comportamento incremental. Assim, não é comum utilizar este termo para algoritmos como classificador de Bayes ou *k-Nearest Neighbor*, que são incrementais por natureza.

2.4.2 *Lazy Learning* e *Eager Learning*

O processo de aprendizagem também pode ser categorizado pela forma com que os dados de treinamento são utilizados para criar a inferência:

- O termo *eager learning* é utilizado para caracterizar os métodos de aprendizado que buscam ativamente inferir sua abstração do conhecimento a partir das instâncias rotuladas de treinamento, disponibilizando-a antes do início da execução do sistema. Em geral, abstrações deste tipo são globais, i.e., atende a todas as instâncias de entrada da mesma forma, sem se adequar a cada uma delas.

- Em contrapartida, métodos do tipo *lazy learning* postergam a inferência até que ela seja necessária durante a execução do sistema (HENDRICKX; BOSCH, 2005). Esta característica permite que a inferência seja local, moldada para o problema a ser solucionado, mas requer manter todas as

instâncias armazenadas em uma base de conhecimento, demandando um espaço maior para armazená-las em relação ao espaço típico ocupado por uma abstração.

Por não manter uma abstração global, métodos do tipo *lazy learning* podem ser adaptados com certa facilidade para apresentar aprendizado incremental, visto que as abstrações são reconstruídas para cada problema a ser solucionado. Assim, para disponibilizar uma nova informação para os próximos problemas, basta inseri-la na base de conhecimento. Em particular, métodos baseados em instâncias (*IBL - Instance-Based Learning*) sequer mantêm uma abstração formal, limitando-se a consultar sua base de conhecimento para obter soluções de problemas passados que se assemelhem ao problema a ser solucionado. Exemplos de métodos *IBL* incluem o *k-Nearest Neighbor* (AHA; KIBLER, 1991) e *K** (CLEARY; TRIGG, 1995).

No entanto, adicionar capacidade incremental em métodos de aprendizado do tipo *eager learning* mostra-se uma tarefa mais complexa, já que se torna necessário dar manutenção a uma abstração global de forma dinâmica. Não existe uma regra simples para que esta manutenção seja feita, visto que ela depende da estrutura da abstração, e deve ser desenvolvida especificamente para cada método em particular. Devido a esta dificuldade, a maior parte dos métodos online costuma ser uma variante de um método do tipo *eager* que, originalmente, só trabalhava com treinamento via *batch* (note que grande parte dos métodos *lazy* é incremental por padrão e não são categorizados como online por não existir uma versão *offline*).

2.4.3 Considerações

Um aspecto pouco explorado nas publicações disponíveis é o processo de esquecer informações já armazenadas. Apesar de ser visto como um elemento indesejável, seu bom uso facilita a tarefa de dar manutenção a um sistema, eliminando informações irrelevantes ou incorretas que possam reduzir seu desempenho (MARKOVITCH; SCOTT, 1988; SMYTH; KEANE, 1995). Este aspecto torna-se vital em sistemas que devem lidar com incertezas, nos quais a possibilidade de absorver informações errôneas não é desprezível.

Experimentos empíricos também demonstram que o valor de uma informação varia de acordo com o conhecimento já adquirido: sua influência pode ser tanto positiva quanto negativa, em função do conhecimento previamente acumulado. São três as principais estratégias para tratar informações negativas: ignorar o problema, evitar que estas informações negativas sejam inseridas, ou remove-las do sistema assim que identificadas como negativas (MARKOVITCH; SCOTT, 1988). Porém, o valor de uma informação dificilmente pode ser medida no momento em que ela é adquirida, o que torna mais conveniente remover uma informação assim que sua não validade seja confirmada.

No entanto, são poucos os algoritmos que permitem a remoção de informações já armazenadas - a maioria opta por ignorar o problema completamente. Nem mesmo os métodos de aprendizado online apresentam, necessariamente, este tipo de funcionalidade (e.g., o *Online Random Forest* de Saffari permite apenas operações de expansão nas suas árvores (SAFFARI et al., 2009)).

Além disso, vale notar que nem sempre é trivial verificar a validade de uma informação, o que dificulta a distinção entre aquelas que são favoráveis ou contrárias, conforme citado na seção 2.3. Logo, torna-se difícil impedir que

informações não válidas acabem sendo incorporadas, o que reforça a necessidade de um método de exclusão de informações previamente memorizadas.

2.5 Métodos de Tomada de Decisão

Após a introdução dos principais conceitos aqui utilizados, segue abaixo a descrição dos algoritmos de tomada de decisão diretamente empregados adiante.

2.5.1 *k-Nearest Neighbor*

O *k-Nearest Neighbor* (*k-NN*) é um método de classificação de objetos baseado em uma métrica de distância que avalia a proximidade entre o objeto a ser classificado e os objetos conhecidos (i.e., com classificação conhecida). Partindo do pressuposto que objetos semelhantes devem possuir classificação semelhante, o classificador *k-NN* busca os k vizinhos mais próximos do novo objeto, e conta o número de ocorrências com que cada classificação aparece neles - aquela com maior frequência será atribuída ao novo objeto. Para o caso particular $k = 1$, o novo objeto é classificado da mesma forma que o objeto mais próximo.

O funcionamento deste algoritmo depende de três elementos fundamentais (WU et al., 2007):

- Um conjunto de instâncias rotuladas a ser utilizado para comparação;
- Uma métrica para computar a distância entre dois objetos;
- O valor de k, ou seja, o número de vizinhos mais próximos a serem considerados pelo algoritmo.

A definição e o uso destes elementos pelo *k-NN* e suas variantes são descritos nas seções seguintes.

2.5.1.1 Definição

Em sua forma original (LIU; CHAWLA, 2011), a classificação de um novo objeto x é expresso por:

$$y(x) = \underset{c \in c_1,\ldots,c_p}{\arg\max} \sum_{i=1}^{k} I(a_i = c) \tag{2.3}$$

em que $I(a_i = c)$ é a *função indicadora*, que retorna 1 se a classificação a_i da instância x_i é c, caso contrário, retorna 0. Assim, a classificação da nova instância é aquela que ocorre com maior frequência entre os k vizinhos mais próximos.

A Figura 11 ilustra um exemplo de classificação via *k-NN*, onde o pentágono é o objeto a ser classificado. Note que, com $k = 3$, o novo objeto será classificado como • (são dois círculos para apenas uma estrela), enquanto com $k = 5$, a classificação muda para ★ (3 estrelas para 2 círculos). Este problema evidencia a importância da escolha do valor de k: valores pequenos são sensíveis a ruídos, enquanto valores elevados dificultam a distinção das fronteiras entre as classes. Embora, informalmente, utiliza-se $k = \sqrt{n}$, onde n é o número de instâncias rotuladas, métodos como validação cruzada (*cross-validation*) são preferíveis para a determinar um valor ótimo para k (KOHAVI, 1995; OUYANG; LI; LI, 2006).

A definição original do *k-NN*, contudo, não considera o grau de semelhança entre dois objetos: supostamente, quanto menor a distância entre dois objetos, maior será a probabilidade de apresentarem a mesma classificação. Porém, pela definição, os k vizinhos são tratados de forma idêntica, não importa sua distância até o novo objeto. Uma extensão popular do *k-NN* é inserir

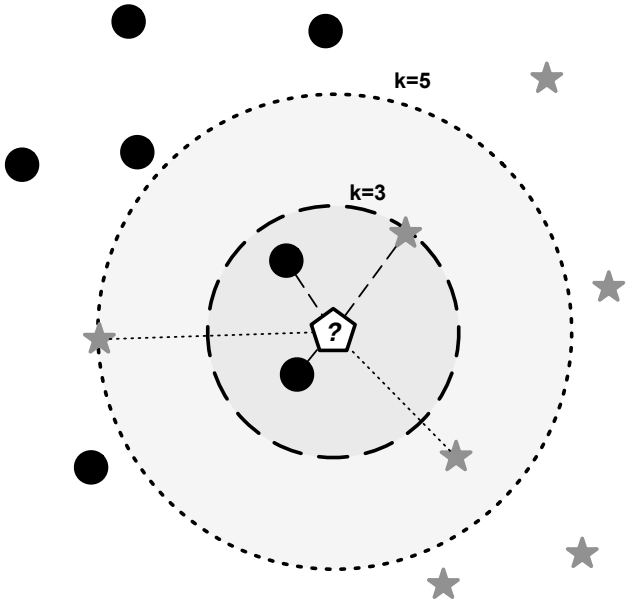


Figura 11: Exemplo de classificação *k-NN*, onde o objeto central (pentágono) é o objeto a ser classificado.

uma função de ponderação:

$$y(x) = \underset{c \in c_1,\ldots,c_p}{\arg\max} \sum_{i=1}^{k} I(a_i = c) \cdot w(x, x_i) \tag{2.4}$$

tal que $w(x, x_i)$ é uma função que pondera a influência de x_i pela sua distância até o objeto x a ser classificado (o peso deve aumentar conforme a distância diminui). Dentre as funções de ponderação conhecidas, destacam-se o método de Shepard (SHEPARD, 1968) e a função de similaridade (LIU; CHAWLA, 2011), ilustrados pelas equações (2.5) e (2.6), respectivamente:

$$w(x, x_i) = \frac{1}{d(x, x_i)^P} \tag{2.5}$$

$$w(x, x_i) = 1 - \frac{d(x, x_i)}{d_{max}} \tag{2.6}$$

Nota: ao utilizar a ponderação, nem sempre a classificação mais numerosa é a escolhida. Como exemplo, caso o problema da Figura 11 fosse solucionado com $k = 5$ utilizando ponderação, a classificação do novo objeto

poderia passar de ★ para ●, pois os dois círculos estão muito mais próximos do novo objeto que as três estrelas.

Nestas equações, $d(x, x_i)$ é uma função que calcula a distância entre dois conjuntos de valores x e x_i, enquanto d_{max} é um fator de normalização para que a distância esteja no intervalo $[0 \dots 1]$. Classicamente, utiliza-se a distância Euclidiana para esta finalidade. Para um espaço com m critérios, ela é representada por:

$$d(a, b) = \sqrt{\sum_{j=1}^{m} \left| a_{[j]} - b_{[j]} \right|^2} \tag{2.7}$$

A distância euclidiana, contudo, exige que o espaço também seja Euclidiano - apenas valores numéricos podem ser atribuídos em cada critério. Além disso, como estes valores podem assumir ordens de grandeza diferentes, a influência de cada critério sobre o valor da distância acaba sendo desigual. Uma forma de mitigar este problema é efetuar uma normalização da distância, uniformizando a contribuição de cada critério j:

$$d(a, b) = \sqrt{\sum_{j=1}^{m} \frac{d_j(a, b)^2}{m}} \tag{2.8}$$

$$d_j(a, b) = \frac{\left| a_{[j]} - b_{[j]} \right|}{d_{j\,max}} \tag{2.9}$$

onde $d_j(a, b)$ representa a distância normalizada entre os vetores a e b no j-ésimo critério, enquanto $d_{j\,max}$ é a máxima distância observada neste critério.

A definição da equação (2.9) ainda exige que os valores de cada critério sejam numéricos para que sua diferença seja calculada. Para valores nominais, uma possibilidade é utilizar a *Overlap Metric* (LI; LI, 2010), representada por $\delta(a_j, b_j)$. Como o nome sugere, esta função calcula a sobreposição de dois valores nominais. Esta distância deve ser zero para valores idênticos, e 1

para valores distintos:

$$d_j(a,b) = \delta\left(a_{[j]}, b_{[j]}\right) = \begin{cases} 0 & \text{se } a_{[j]} = b_{[j]} \\ 1 & \text{se } a_{[j]} \neq b_{[j]} \end{cases} \tag{2.10}$$

Esta métrica, contudo, não faz uso de informações adicionais sobre os valores - como exemplo, não relaciona como cada classificação é influenciada pelos valores de cada critério. Este tipo de informação permitiria estimar uma distância de forma mais precisa do que meramente verificando a sobreposição de valores. Um exemplo mais sofisticado é a *Value Difference Metric* (LI; LI, 2010):

$$d_j(a,b) = \sum_{c=1}^{p} \left| P\left(c|a_{[j]}\right) - P\left(c|b_{[j]}\right) \right| \tag{2.11}$$

onde $P\left(c|a_{[j]}\right)$ é a probabilidade da classificação de a ser c, dado que seu valor para o j-ésimo critério é $a_{[j]}$. Este valor pode ser obtido conhecendo-se a frequência com que cada classificação aparece quando uma instância aparece com o valor $a_{[j]}$.

2.5.1.2 Custo Computacional

Por ser um método baseado em instâncias, a complexidade computacional temporal do algoritmo *k-NN* depende fortemente do desempenho da busca utilizada para localizar os vizinhos mais próximos. Em uma busca linear, seria necessário calcular a distância entre todas as instâncias rotuladas e a nova instância, tornando-a computacionalmente custosa à medida que o número de instâncias armazenadas aumenta. Por outro lado, por não manter abstrações compactas do conhecimento, o problema de manter a base de conhecimento atualizada se reduz a um problema de inserção e remoção de instâncias na base de conhecimento.

Uma técnica muito empregada para reduzir o custo temporal da busca é o particionamento espacial, onde estruturas computacionais (e.g., árvores e clusters) são utilizadas para dividir o espaço em um conjunto finito de espaços (normalmente, disjuntos), permitindo que a busca pelos vizinhos mais próximos seja feito localmente, dentro do sub-espaço em que se encontra ou em sub-espaços próximos, removendo a necessidade de avaliar todo o espaço. Apesar das técnicas de particionamento estarem mais bem consolidadas em problemas puramente numéricos (e.g., *k-d tree* (BENTLEY, 1975) e *R*-Tree* (BECKMANN et al., 1990)), novas técnicas tem sido desenvolvidas para tratar critérios nominais (BOULLÉ, 2004; KIBRIYA; FRANK, 2007). Vale ressaltar que, através da discretização, problemas numéricos também podem ser particionados através deste tipo de técnica.

A Figura 12 ilustra uma *R*-Tree*, em que os pontos conhecidos são estruturados em uma árvore, tal que as folhas representam os pontos, e os nós, o menor sub-espaço (*bounding box*) que contém todos os pontos presentes em sua sub-árvore. A busca nesta árvore se resume a percorrer a árvore, visitando apenas os ramos próximos (i.e., com pequeno valor de distância) ao ponto utilizado como critério de busca e evitando ramos afastados. Este processo é mais eficiente do que efetuar uma busca linear. Neste exemplo, a busca com $k = 3$ só percorre os ramos $r13$ e $r31$.

2.5.1.3 Considerações

A popularidade do algoritmo *k-NN* fez com que seus resultados (e.g., taxa de acerto e tempo de resposta) sejam frequentemente utilizados como referência para novos algoritmos, e, de forma geral, apresenta boa taxa de acerto, apesar de que métodos mais modernos, como *random forests* e *boosted trees* (CARUANA; NICULESCU-MIZIL, 2006), apresentam melhores taxas de acerto e

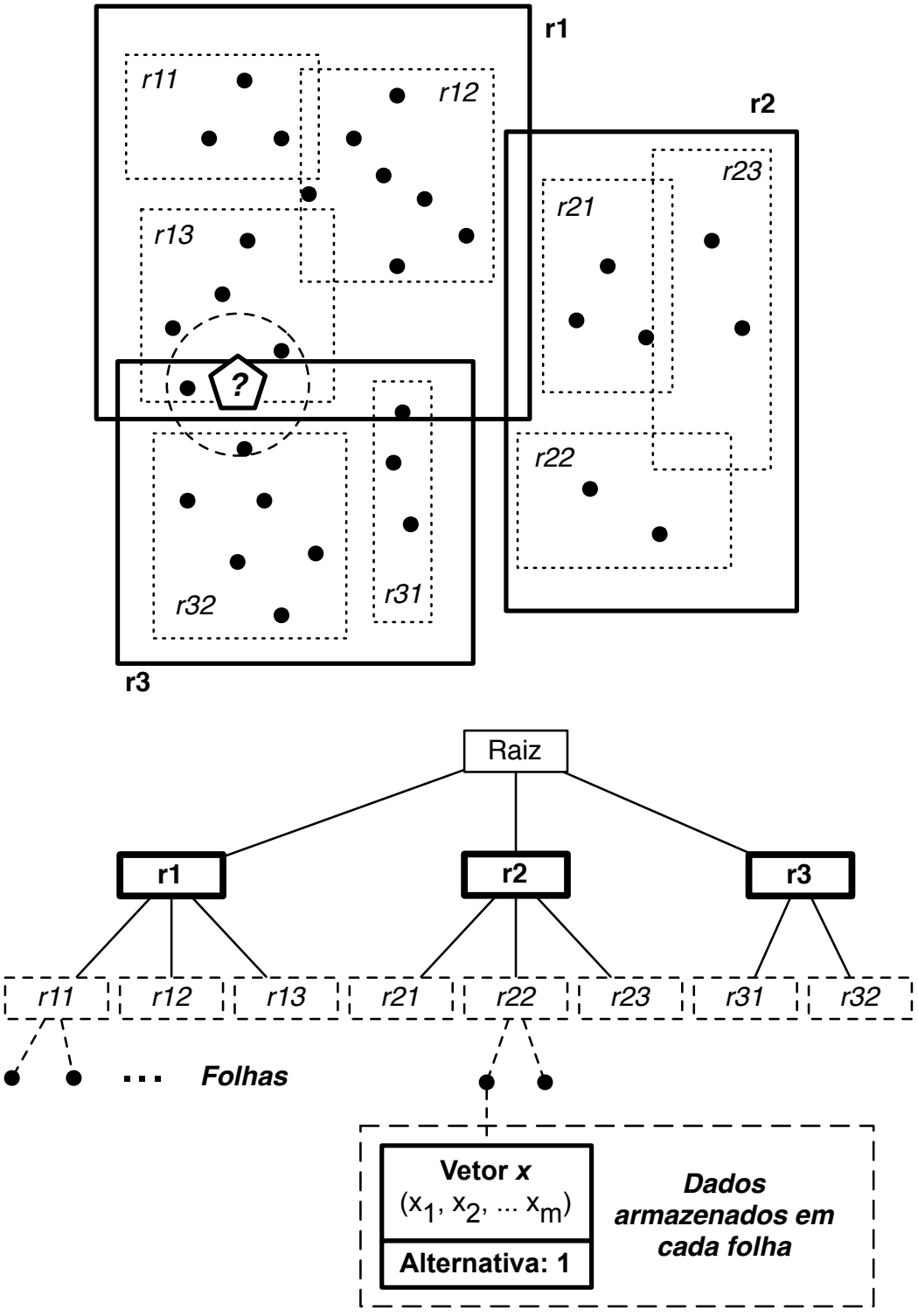


Figura 12: Representação gráfica de uma *R*-Tree*. A aplicação do *k-NN* pode ser feita visitando apenas parte dos pontos conhecidos - representados espacialmente por um vetor x.

custos computacionais menores, ainda que não estejam na mesmo nível de consolidação e maturidade.

As variantes deste algoritmo refletem as pesquisas realizadas nas épocas em que cada uma delas foi concebida. Como exemplo, destaca-se o uso de lógica fuzzy para tratar dados cuja expectativa de certeza de uma classe pode assumir valores reais - de 0 a 1 (KELLER; GRAY; GIVENS JR., 1985). Redução do espaço de instâncias rotuladas (i.e., condensar um conjunto de instâncias em um conjunto menor, mas que represente aproximadamente a mesma informação) é outra área de estudo deste algoritmo (WU et al., 2007). O estudo do tratamento de incertezas também impulsionou o desenvolvimento de variantes que lidam com informações cuja veracidade não é completa ou verificável (ZHANG et al., 2010). O estudo de algoritmos de *clusters* baseados em densidade - e.g., *DBSCAN* (ESTER et al., 1996) e OPTICS (ANKERST et al., 1999) - também teve influência na criação de variantes que utilizem a densidade das instâncias como forma de calcular a distância entre instâncias, confirme explicado em (VOULGARIS; MAGOULAS, 2008). A Figura 13 ilustra um exemplo de busca baseada em densidade, em que a estrutura espacial dos objetos é levada em consideração.

2.5.2 Classificador de Bayes

O classificador de Bayes é um método de classificação que estima qual deve ser a classe de um determinado vetor x de características baseando-se em técnicas probabilísticas. Caso seja conhecida a probabilidade $P\left(x_{[i]}|c\right)$ de um valor $x_{[i]}$ qualquer (associado ao i-ésimo critério) ocorrer em instâncias de classe c, é possível estimar qual é a classe mais provável do vetor de entrada (AKAHO, 2001). Porém, $P\left(x_{[i]}|c\right)$ raramente é conhecido, sendo comumente estimado através das instâncias de treinamento, calculando a frequência rela-

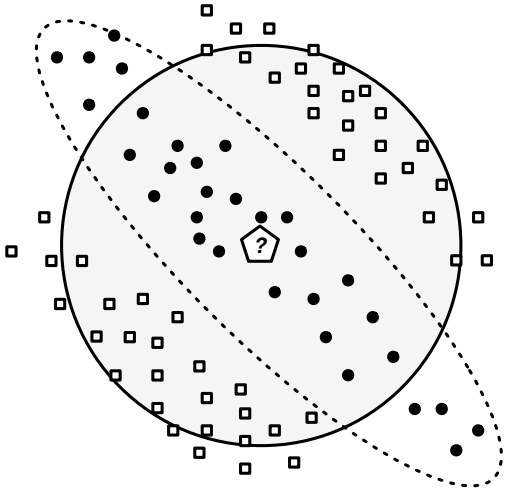


Figura 13: Comparação entre o *k-NN* clássico (linha contínua) e o k-NN baseado em densidade (linha tracejada).

tiva com que cada valor $x_{[i]}$ aparece nas instâncias que possuem classificação c.

Apesar de ser uma das formas mais antigas de classificação, sua simplicidade é muito atraente para uso em vários campos, como, por exemplo, classificação de textos e filtros de spam (WU et al., 2007). Sua construção não depende de iterações ou estimativa de parâmetros, o que simplifica o processo de treinamento. Sua execução é igualmente simples e de fácil interpretação, e ainda assim, mantém-se robusto para grande parte dos problemas.

2.5.2.1 Definição

Pela teoria de Bayes, a probabilidade de a alternativa c ser solução de uma instância com valores x é dada pela equação (2.12):

$$P\left(c|x\right) = \frac{1}{Z\left(x\right)} P\left(c\right) \prod_{i=1}^{m} P\left(x_{[i]}|c\right) \tag{2.12}$$

tal que $P(c)$ é a probabilidade a priori de c ser solução de qualquer vetor de características (este valor pode ser estimado verificando a porcentagem das instâncias que têm c como solução). Já $P\left(x_{[i]}|c\right)$ é a probabilidade de o valor $x_{[i]}$ aparecer no i-ésimo critério quando uma instância qualquer tem c como solução, e difere de acordo com o tipo do critério. Para critérios nominais, sua estimativa é feita calculando a porcentagem dos problemas que contêm $x_{[i]}$, dentre aqueles que têm a alternativa c como solução.

No caso de o critério ser numérico, o cálculo dependerá do tipo da distribuição que caracteriza o critério. Como exemplo, para a distribuição normal, utiliza-se:

$$P\left(x_{[i]}|c\right) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{\left(x_{[i]}-\mu_{ic}\right)^2}{2\sigma_{ic}^2}} \tag{2.13}$$

onde μ_{ic} é a média e σ_{ic} é o desvio padrão do valor do i-ésimo critério, considerando apenas instâncias cuja solução é c. Na prática, porém, é mais comum discretizar critérios numéricos para evitar o uso de funções dependentes da distribuição, conforme descrito na seção 2.2.1.

A função $Z(x)$ é um fator de normalização, comumente conhecido como *evidência*, que garante que a soma das probabilidades $P(c_j|x)$ de cada solução c_j seja igual a 1. Assim:

$$Z(x) = \sum_{j=1}^{s} \left(P(c_j) \prod_{i=1}^{m} P\left(x_{[i]}|c_j\right) \right) \tag{2.14}$$

A partir destes dados, o classificador de Bayes retorna a solução de maior probabilidade:

$$y(x) = \underset{c \in c_1,\ldots,c_p}{\arg\max} \frac{1}{Z(x)} P(c) \prod_{i=1}^{m} P\left(x_{[i]}|c\right) \tag{2.15}$$

Na maior parte dos casos, não é necessário calcular a evidência diretamente, uma vez que, por ser um fator de escala positivo, não altera a escolha

feita pelo operador $\arg\max$. Logo:

$$P\left(c|x\right) \propto P\left(c\right) \prod_{i=1}^{m} P\left(x_{[i]}|c\right) \tag{2.16}$$

$$y\left(x\right) = \underset{c \in c_1,\ldots,c_p}{\arg\max} P\left(c\right) \prod_{i=1}^{m} P\left(x_{[i]}|c\right) \tag{2.17}$$

2.5.2.2 Extensões

A popularidade e simplicidade do classificador de Bayes o torna muito receptivo à incorporação de extensões e modificações que possam melhorar seu desempenho em determinadas situações (WU et al., 2007). Em especial, grande parte destas extensões busca tratar a questão da independência dos critérios, ignorada pelo classificador original.

O *AODE* (*Averaged One-Dependence Estimators*) é um exemplo popular do classificador de Bayes, que busca tratar a dependência entre critérios avaliando as probabilidades com que cada valor aparece em conjunto com outros valores de outros critérios (WEBB; BOUGHTON; WANG, 2005). Formalmente:

$$P\left(c|x\right) = \begin{cases} \dfrac{\sum\limits_{i=1}^{m} P\left(c, x_{[i]}\right) \prod\limits_{j=1}^{m} P\left(x_{[j]}|c, x_{[i]}\right)}{\sum\limits_{k=1}^{p} \sum\limits_{i=1}^{m} P\left(c_k, x_{[i]}\right) \prod\limits_{j=1}^{m} P\left(x_{[j]}|c_k, x_{[i]}\right)} & \forall i \in \left[1, 2, \ldots, m\right], F\left(x_i\right) \geqslant mEst \\ \frac{1}{Z(x)} P\left(c\right) \prod\limits_{i=1}^{m} P\left(x_{[i]}|c\right) & \text{caso contrário} \end{cases} \tag{2.18}$$

onde $P\left(x_{[j]}|c_k, x_{[i]}\right)$ é a probabilidade do valor $x_{[j]}$ no j-ésimo critério aparecer um conjunto com o valor $x_{[i]}$ do i-ésimo critério quando a solução é c, enquanto $P\left(c, x_{[i]}\right)$ é a probabilidade de uma alternativa ter c como solução e apresentar o valor $x_{[i]}$. Já $F\left(x_{[i]}\right)$ é um contador de frequência com que cada valor aparecer. O AODE só será utilizado se todos os valores da instância x têm frequência maior que um limiar $mEst$. Caso contrário, utiliza-se o classificador

de Bayes convencional.

Apesar do acréscimo computacional e de memória (será necessário um novo contador de frequência para cada tripla $(x_{[i]}, c, x_{[j]})$), o *AODE* é mais estável que a versão original, possuindo uma variância de erros menor, bem como um pequeno ganho na taxa de acerto, aliviando a exigência da independência de critérios. Este classificador, a exemplo do original, também é capaz de aprender de forma incremental (WEBB; BOUGHTON; WANG, 2005) - característica desejada para os classificadores dinâmicos.

2.5.2.3 Comparação com Outros Algoritmos

Quando comparado com os demais métodos, o classificador de Bayes se destaca pela sua rapidez, tanto durante a execução como durante a fase de aprendizado. Porém, sua taxa de acerto costuma ser muito inferior à da maioria dos métodos de tomada de decisão, sejam eles mais modernos (e.g., *boosted trees*) ou algoritmos clássicos (e.g., *k-NN* e árvores de decisões) (CARUANA; NICULESCU-MIZIL, 2006). Mesmo suas variantes não costumam obter taxas de acerto significativas que justifiquem seu uso, ainda que sejam mais estáveis (WEBB; BOUGHTON; WANG, 2005).

Vale ressaltar que o classificador de Bayes assume que os critérios são independentes entre si - i.e., cada critério contribui com a probabilidade de uma classe ser resposta, independentemente dos valores dos demais critérios. Apesar de ser uma simplificação imprecisa, ela se mostra vantajosa em problemas que apresentem um elevado número de dimensões (i.e., o número de critérios). Tipicamente, espaços desta natureza apresentam problemas que não ocorrem em dimensionalidades menores, e.g., o volume de um objeto aumenta exponencialmente com o aumento do número de dimensões, e há uma tendência dos pontos se tornarem equidistantes (EVANGELISTA; EM-

BRECHTS; SZYMANSKI, 2006). A este fenômeno, dá-se o nome de maldição da dimensionalidade (*curse of dimensionality*), e representa um dos principais obstáculos enfrentados nas pesquisas relacionadas com aprendizagem de máquina. Apesar disso, o classificador de Bayes, ao assumir critérios independentes, torna-se menos sensível ao aumento do número de dimensões (AKAHO, 2001).

3 PROPOSTA

Este capítulo apresenta o sistema híbrido de tomada de decisão proposto neste trabalho, em que múltiplos dispositivos são utilizados colaborativamente para tomar uma decisão em conjunto. O emprego de técnicas adaptativas possibilita regular a relação de compromisso entre taxa de acerto e tempo de resposta de acordo com as exigências do problema. Adicionalmente, a adaptatividade permite que cada dispositivo aprenda, de forma incremental, através das soluções obtidas pelos demais, ou então, através de agentes externos, tratando informações cuja veracidade pode não ser necessariamente confirmada.

3.1 Visão Geral

Utilizando o mesmo conceito das etapas *4R* do raciocínio baseado em casos apresentado previamente na seção 2.3, propõe-se um sistema de tomada de decisão que seja capaz de não só solucionar problemas, mas também agregar as soluções obtidas na forma de um novo conhecimento, disponibilizando-as como novas referências para solucionar problemas posteriores.

A Figura 14 ilustra o fluxo a ser seguido pelo sistema proposto, enquanto a Figura 15 ilustra sua composição, destacando os itens a serem desenvolvidos neste trabalho. A partir delas, verifica-se que antes de serem solucionados,

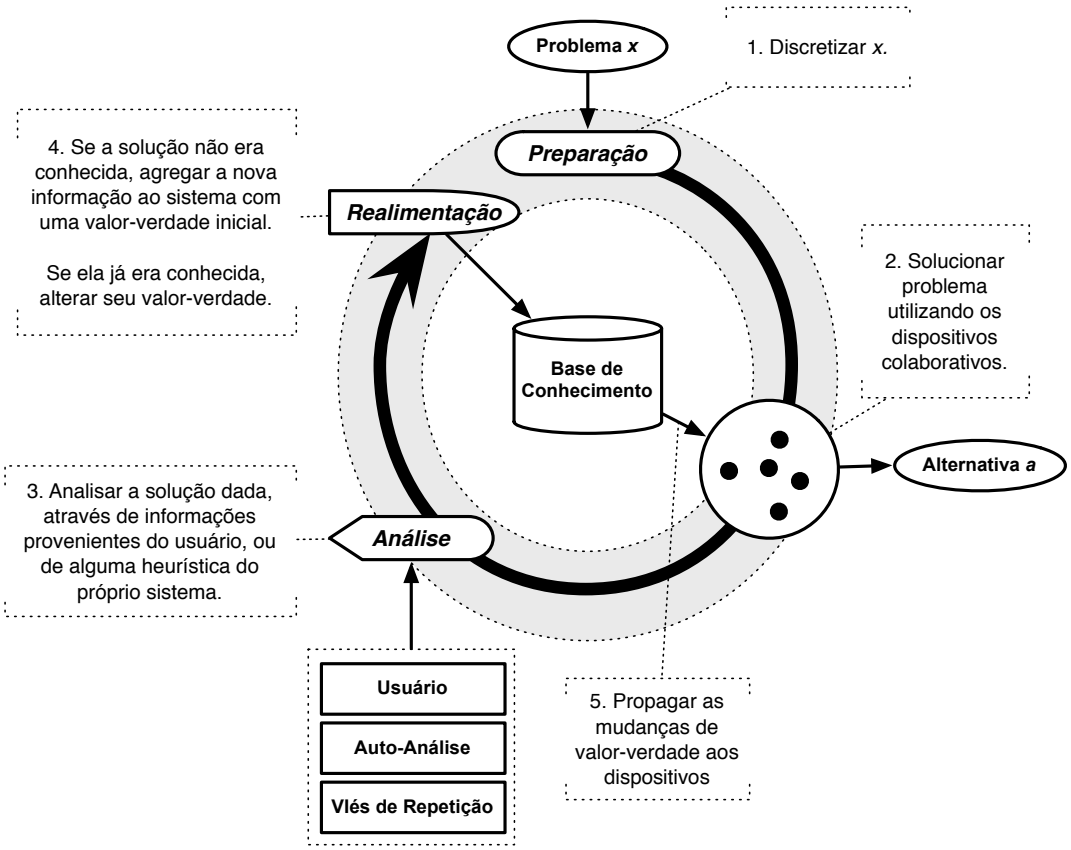


Figura 14: Fluxo do sistema de tomada de decisão proposto - baseado nas etapas *4R* do raciocínio baseado em casos.

os problemas de tomada de decisão, descritos por um vetor de características x, devem ter seus valores numéricos *discretizados* em uma etapa preliminar. Este processo flexibiliza a escolha dos dispositivos, permitindo que sejam utilizados aqueles que tratem exclusivamente de critérios nominais. Exemplos de métodos de discretização foram previamente apresentados na seção 2.2.1.

Após serem discretizados, os vetores de características são repassados a um conjunto de *dispositivos adaptativos* que, de forma colaborativa, trabalham em conjunto para encontrar uma solução comum, retornada na forma de uma alternativa a. Este trabalho colaborativo deve ser gerenciado através de algum processo que defina quais dispositivos devem ser utilizados, e como compor a solução final a partir das soluções encontradas por cada um deles. A seção

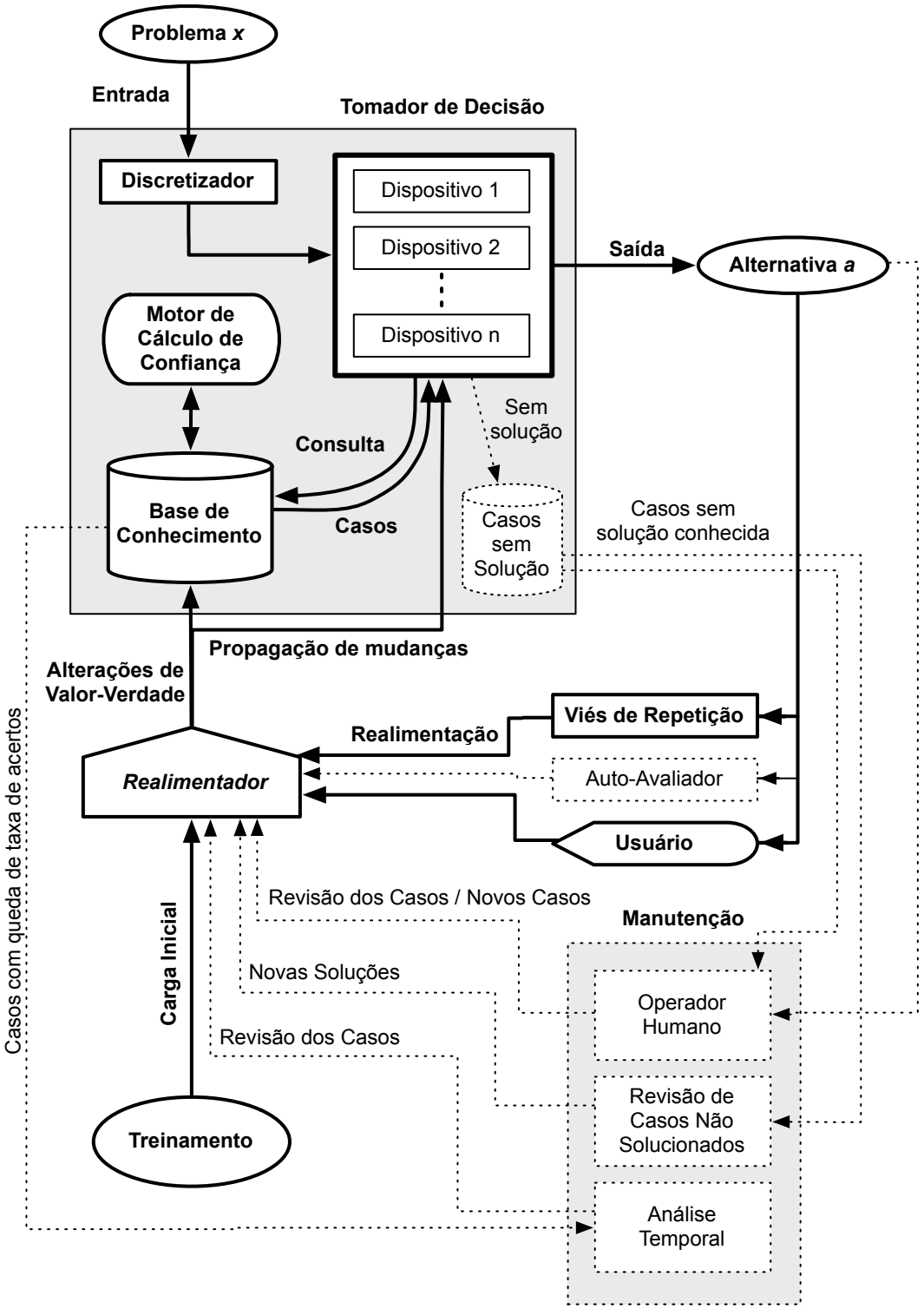


Figura 15: Visão geral do sistema de tomada de decisão proposto. O itens destacados em negrito são o foco deste trabalho.

3.2 detalha como este processo é realizado.

Para auxiliar os dispositivos em sua tarefa de encontrar soluções, o sistema proposto faz uso de uma *base de conhecimento*, responsável por armazenar as informações relevantes para a tomada de decisão. Este conhecimento é representado através de *casos*. Um caso é formado por um par (x, a), representando um vetor de características x e uma possível alternativa a. Apesar de ser estruturalmente similar ao conceito de *instância*, um caso não representa uma verdade, mas sim, uma possível solução de um problema.

Para cada caso (x, a), há um *valor-verdade* $\hat{v}$ associado, que empiricamente estima a probabilidade da alternativa a ser uma possível solução para x, com base na frequência com que esta alternativa solucionou corretamente este mesmo problema em momentos passados: alternativas aceitas com maior frequência têm maior valor-verdade. Esta frequência de acerto recebe o nome de *contador*, e é utilizada para calcular não apenas o valor-verdade associado, mas também uma estimativa de confiança deste valor.

Logo, o que a base de conhecimento efetivamente armazena são contadores para cada caso (x, a), permitindo que o valor-verdade associado seja calculado e disponibilizado aos dispositivos. Este recurso possibilita o uso de algoritmos de tomada de decisão que trabalhem com dados cuja verdade não é absoluta (e.g., o *k-NN fuzzy* citado na seção 2.5.1.3). Os processos de atualizar os contadores e calcular o valor-verdade estão descritos na seção 3.3.2.

Internamente, após os dispositivos encontrarem uma solução comum a para x, o par (x, a) é propagado de volta aos dispositivos, para que cada um deles aprenda com as soluções encontradas pelos demais. Mais especificamente, as soluções são enviadas à base de conhecimento, que, por sua vez, atualiza os contadores do caso (x, a) analisado e propaga as mudanças

ocorridas aos dispositivos, para que agreguem esta nova informação. Esta propagação recebe o nome de *realimentação*

Em sua forma mais básica, a solução encontrada é diretamente incorporada à base de conhecimento, reforçando que o caso (x, a) é válido. Este conceito está de acordo com o comportamento causado pelo *viés de repetição*, onde uma solução é dada como verdadeira enquanto não receber uma opinião contrária.

No entanto, em uma forma mais elaborada, esta realimentação também pode ter origem em um bloco de auto-avaliação que, com o auxilio de alguma inteligência (e.g., heurísticas, teste direto da solução), verifica se a solução a encontrada é realmente válida. Apesar de ser potencialmente mais confiável que o viés de repetição, este auto-avaliador depende fortemente da natureza dos problemas a serem solucionados, devendo ser implementado especificamente para cada sistema em particular. Outra fonte de realimentação, supostamente mais confiável, é o próprio usuário, que pode dar sua própria opinião, permitindo confirmar que a solução encontrada foi mesmo adequada.

Deve ser ressaltado que as realimentações podem ser de natureza tanto positivas como negativas: as positivas reforçam a validade de uma solução, enquanto as negativas apoiam a ideia de que a solução não é válida. Desta definição, nota-se que as realimentações oriundas do viés de repetição são sempre positivas, diferentemente daquelas oriundas das demais fontes, que dependem da análise feita em cima da solução.

O uso de diferentes fontes cria a necessidade de ponderar as realimentações, para que informações provindas de fontes de maior credibilidade tenham peso maior em relação às demais. Os processos de ponderar as realimentações, atualizar os contadores e calcular o valor-verdade estão detalhados na seção 3.3.

Deve ser ressaltado que estas realimentações apresentadas fazem parte do ciclo de vida da solução de um único problema, ocorrendo após a obtenção de suas soluções. Entretanto, elas também podem ser geradas fora deste ciclo, para possibilitar o tratamento de casos quaisquer em instantes de tempo arbitrários, e assim, dar manutenção ao sistema, procurando eventuais falhas e as corrigindo conforme necessário. A Figura 15 ilustra três possíveis módulos de manutenção:

- Um bloco para analisar o efeito do tempo sobre os casos conhecidos, com o objetivo de tratar o desvio de conceito, citado na seção 2.2.2.
- Um bloco para tratar problemas não solucionados, i.e., problemas que não puderam ser solucionados de forma confiável por nenhum dos dispositivos, e que, portanto, apresentam baixo valor-verdade em todas as alternativas. Neste caso, algoritmos de aprendizado não supervisionado, e.g., *clustering*, poderiam ser utilizados para encontrar possíveis padrões nestes problemas, e tentar inferir se elas representam uma nova alternativa não conhecida, ou então, encontrar suas alternativas verdadeiras.
- Um operador humano também poderia realizar uma manutenção por conta própria, avaliando os casos conhecidos e não solucionados, e corrigindo eventuais erros. Um treinamento *online* também poderia ser feito para esta finalidade.

Estes blocos de manutenção não serão tratados diretamente neste trabalho. A seção 5.2 discute como eles poderiam ser desenvolvidos, citando exemplos de técnicas que poderiam ser utilizadas para esta finalidade, bem como as possíveis dificuldades a serem encontradas.

As seções seguintes detalham cada uma destas etapas do processo de

tomada de decisão, descrevendo as estratégias empregadas em cada uma delas e justificando sua escolha.

3.2 Dispositivo Adaptativo Hierárquico

Esta seção apresenta a composição de um dispositivo adaptativo de tomada de decisão, descrevendo como cada um deles opera individualmente, bem como sugerindo estratégias para que trabalhem de forma colaborativa para encontrar uma solução comum, de acordo com o que fora apresentado previamente.

3.2.1 Generalização da *TDAE*

A partir do modelo híbrido de tomada de decisão utilizado pela *TDAE*, em que uma tabela de decisão ganha o auxílio de um segundo dispositivo para problemas que não é capaz de solucionar por conta própria, pode-se generalizá-lo para um número arbitrário de dispositivos, dispondo-os em uma fila. Seu funcionamento segue a mesma ideia utilizada pela *TDAE*: ao receber um problema (i.e., um vetor de características x), o primeiro dispositivo da fila se encarrega de solucioná-lo. Em seguida, deve ser verificado se sua solução é confiável, de acordo com algum critério de aceitação. Caso ela seja aceita, é retornada como solução do sistema, caso contrário, o vetor é repassado ao dispositivo seguinte da fila. Este procedimento se repete até que um dos dispositivos consiga solucionar x, ou então, até esgotar os dispositivos da fila. Após serem obtidas, as soluções são realimentadas aos dispositivos, para que cada um deles aprenda com as soluções dos demais. Este processo é ilustrado pela Figura 16.

Neste modelo, a ordem de execução dos dispositivos deve ser escolhida

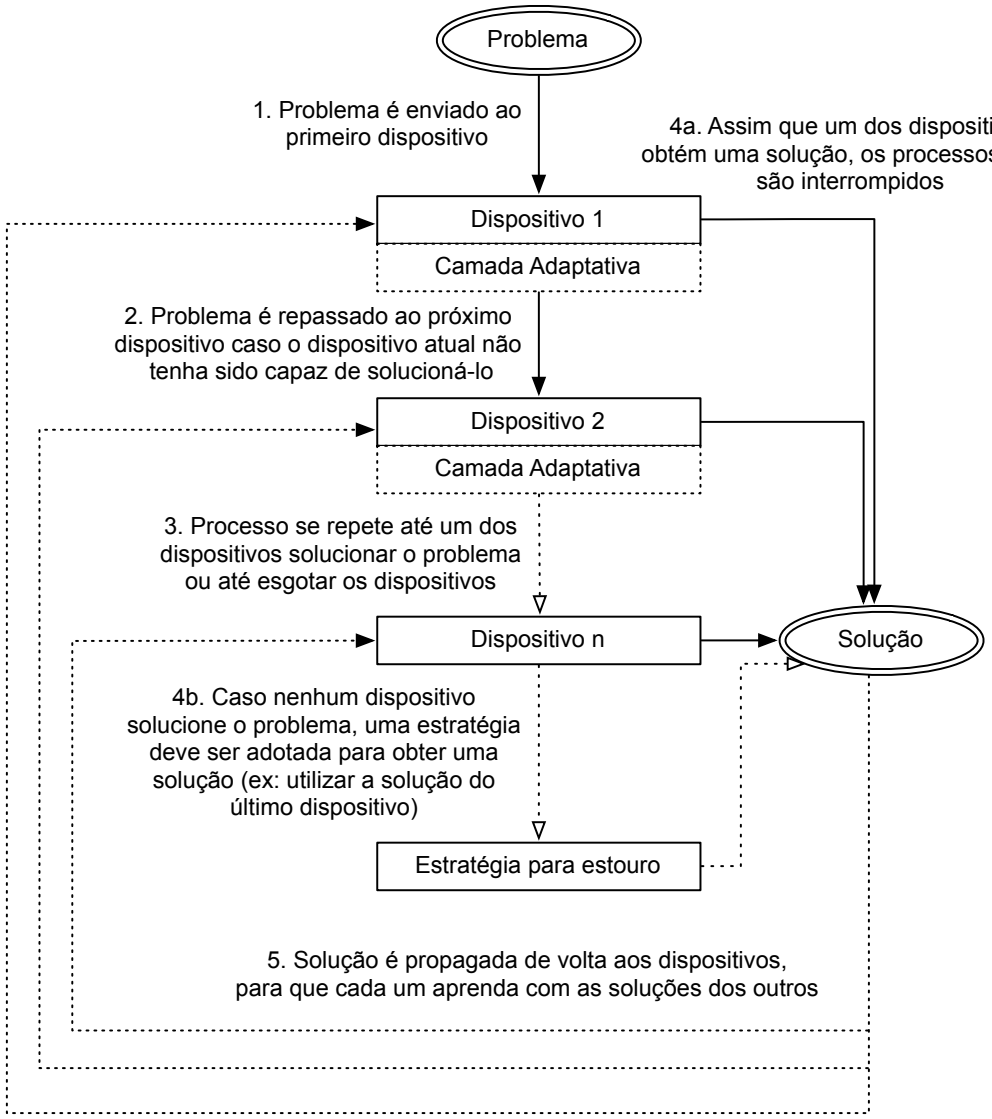


Figura 16: Princípio de funcionamento dos dispositivos em hierarquia.

de forma a minimizar o uso dos dispositivos mais lentos, utilizando-os apenas quando os demais não são capazes de obter uma solução aceita. Logo, sugere-se ordená-los em ordem crescente de tempo de resposta, mantendo os dispositivos mais rápidos no começo da fila. A velocidade de um dispositivo qualquer pode ser estimada de forma empírica, treinando-o com um conjunto de instâncias, executando-o e medindo seu tempo de resposta médio com um segundo conjunto de instâncias.

Entretanto, deve ser ressaltado que outras estratégias de colaboração en-

tre dispositivos poderiam ser empregadas - idealmente, um agente inteligente escolheria qual o dispositivo mais apropriado para cada vetor de características, evitando o uso daqueles cuja probabilidade de obter uma solução aceita é baixa. Entretanto, este trabalho dá ênfase à estratégia proposta acima, que, apesar de não apresentar um comportamento adaptativo, apresenta a possibilidade de regular a relação de compromisso entre taxa de acerto e tempo de resposta através do critério de aceitação, que controla a frequência de uso dos dispositivos mais lentos.

Supondo que seja utilizada esta estratégia de ordenação por tempo de resposta, a realimentação proposta permite que dispositivos mais velozes - e, em geral, menos precisos - aprendam através das soluções obtidas por dispositivos mais robustos. Potencialmente, à medida que estes dispositivos recebem novas informações, suas soluções passam a ser aceitas com maior frequência, resultando em uma queda no tempo de resposta. Em contrapartida, de forma análoga à curva *SAT* (apresentada na seção 1.2), este ganho de velocidade pode estar acompanhado de uma perda na taxa de acerto, causada pela menor frequência de acesso aos dispositivos mais robustos. Todavia, espera-se que esta perda seja minimizada pelo fato de o aprendizado ser feito com base em informações provindas de dispositivos mais robustos. A seção 4.6.1 apresenta resultados obtidos experimentalmente com relação a esta forma de aprendizado.

Diferentemente da *TDAE*, esta propagação de soluções não é a única fonte de informações de um dispositivo. Agentes externos podem também interferir na aprendizagem dos dispositivos da fila, fornecendo-lhes novas informações, ou então, atualizando o conhecimento já existente. O uso destes agentes flexibiliza o processo de aprendizagem, permitindo que outros dispositivos inteligentes fora desta fila possam analisar as soluções e contribuir

incorporando novas informações. Em particular, permite que um operador humano também consiga contribuir com novas informações.

Também deve ser destacado que este critério de aceitação é bastante subjetivo, dependendo fortemente do entendimento que se possui do que pode ser considerado uma solução confiável. Enquanto a decisão de utilizar o método auxiliar na *TDAE* é tomada quando nenhuma regra aplicável é encontrada, no caso geral, não é clara a fronteira que define quais soluções são válidas. A seção 3.2.2 detalha como esta confiabilidade é estimada e avaliada.

Com base neste princípio de funcionamento e na notação original da *TDAE* apresentada na seção 2.1.4, um *dispositivo adaptativo hierárquico*, neste trabalho, é definido como $DAH_k = (AD_k, FM, LM, j)$, onde:

- AD_k é um dispositivo adaptativo subjacente após a execução de k ações adaptativas. Porém, diferentemente dos dispositivos guiados por regra (definidos na seção 2.1.1), este dispositivo é responsável por estimar o vetor de expectativa de certezas μ para um determinado vetor de características x (atribui-se o nome $y\,(x)$ para a função que estima μ).

- LM é uma referência à fila de dispositivos, enquanto j é a posição do dispositivo nesta fila.

- FM representa um conjunto de funções auxiliares que estabelece a comunicação entre o dispositivo subjacente e sua camada adaptativa. Assim como na *TDAE*, estas funções são definidas externamente, não sendo diretamente explicitadas na definição do dispositivo AD_k.

Considerando as exigências da hierarquia de tomada de decisão, são três as funções auxiliares:

- Uma função $FM_{aceitacao}$ deve ser utilizada *após* a execução de um dispositivo para determinar se a solução obtida é válida de acordo com o critério de aceitação. Em caso positivo, a solução é retornada, caso contrário, o próximo dispositivo da fila LM deve ser chamado para solucionar o prolema.

- Uma segunda função $FM_{aprendizado}$ é responsável por receber informações de agentes externos - e.g, os demais dispositivos ou um operador humano - e agregá-las ao conhecimento do dispositivo.

- Por fim, $FM_{estouro}$ define a estratégia a ser seguida quando nenhum dispositivo da fila for capaz de solucionar um problema em particular.

Esta definição possui três diferenças conceituais com relação à *TDAE*. Primeiramente, enquanto as funções auxiliares da *TDAE* são executadas *antes* da aplicação de uma regra, $FM_{aceitacao}$ é utilizada *após* a execução do dispositivo, para que avalie sua solução com base no critério de aceitação. Este passo é desnecessário na *TDAE*, cujo critério para pedir auxílio de M é simplesmente a detecção da ausência de uma regra aplicável.

Em segundo lugar, destaca-se que o processo de aprendizado ocorre através de $FM_{aprendizado}$, separadamente da função responsável por acionar os dispositivos seguintes. Esta divisão ocorre para permitir que agentes externos também possam fornecer informações aos dispositivos sem passar pelo critério de aceitação, utilizado para controlar a colaboração entre dispositivos.

Outra diferença é a presença da função $FM_{estouro}$, inexistente na definição original da *TDAE*. Esta diferença é decorrente da possibilidade de nenhum dispositivo obter um solução aceita - possibilidade esta que não é considerada na *TDAE*, uma vez que as soluções encontradas através do método auxiliar M são sempre aceitas.

A Figura 17 ilustra como as funções auxiliares de FM são utilizadas pelos dispositivos, bem como a fila de dispositivos do ponto de vista da adaptatividade. Em comparação, a Figura 18 ilustra a composição do dispositivo adaptativo sob a ótica do processo de tomada de decisão, ilustrando as entrada e saídas do dispositivo.

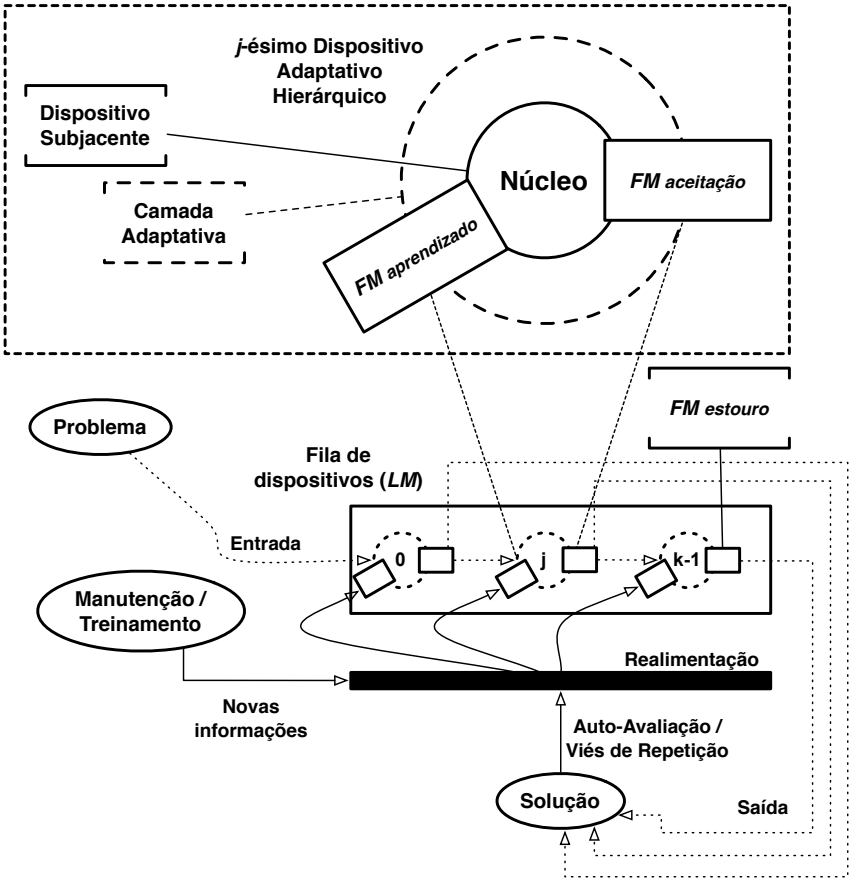


Figura 17: Funções auxiliares dos dispositivos adaptativos hierárquicos.

Nota: dispositivos adaptativos guiados por regra permitem que a aplicação de uma regra resulte na chamada de uma função adaptativa, responsável por alterar o conjunto de regras. Neste projeto, a adaptatividade não será utilizada desta maneira, restringindo-se ao comportamento adaptativo definido pela TDAE.

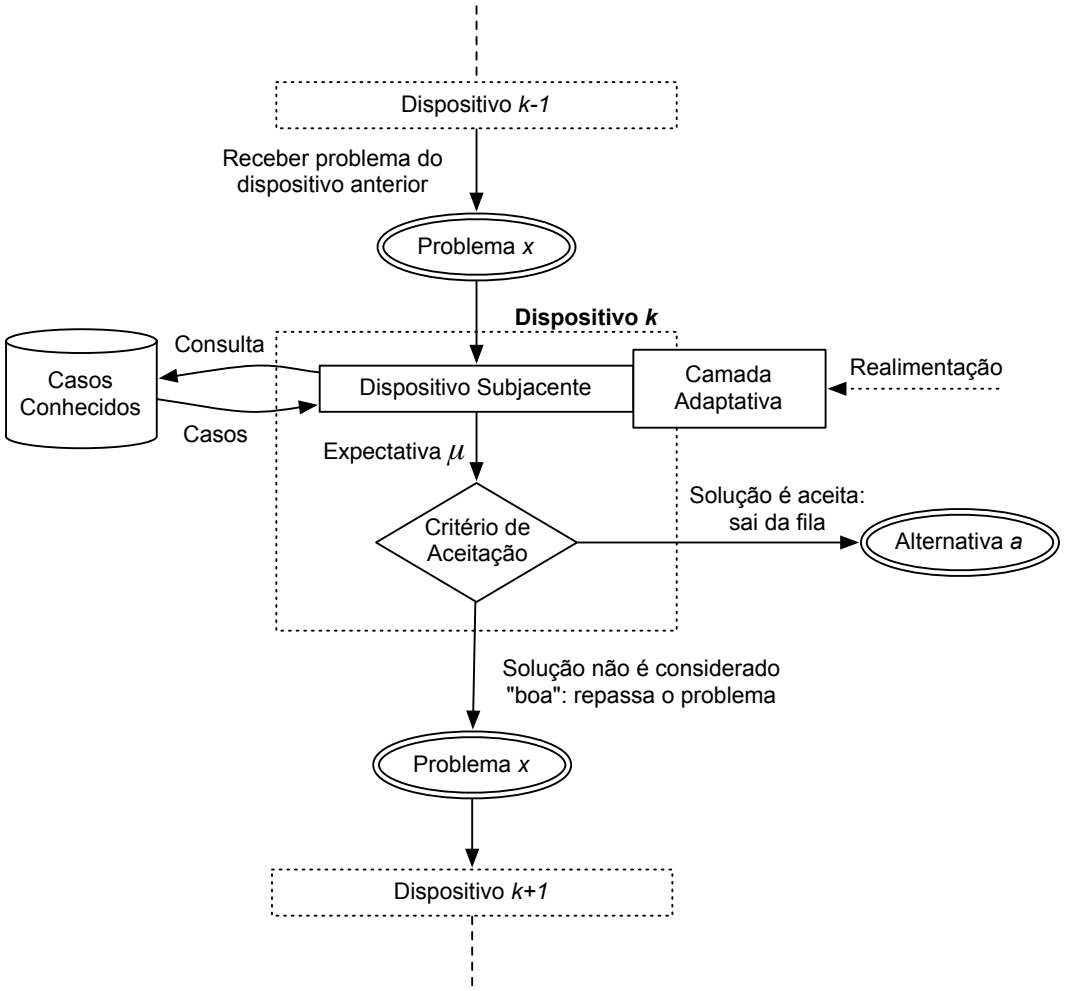


Figura 18: Composição de um dispositivo adaptativo hierárquico.

3.2.2 Critério de Aceitação

Com base na expectativa de certeza calculada pelos dispositivos, deve-se decidir se o resultado obtido deve ser considerado válido (o que significa sair da fila), ou se o vetor x deve ser repassado ao dispositivo seguinte. Para problemas em que apenas uma alternativa válida é esperada, uma forma de decidir a aceitação é verificar se a expectativa de certeza de maior valor está acima de um limiar, aceitando a alternativa correspondente em caso positivo. Logo:

$$aceita = \left(\max\left(\mu_1, \mu_2, \ldots, \mu_p\right) > thresh\right),\ 0 \leqslant thresh \leqslant 1 \qquad (3.1)$$

onde $thresh$ é o valor do limiar de aceitação. Entretanto, caso existam duas ou mais alternativas com expectativas acima do limiar e com valores muito próximos, esta condição torna-se pouco eficiente, pois nesse caso não fica clara a fronteira que define qual das alternativas é a solução mais adequada, tornando-as possivelmente ambíguas. Uma forma de contornar este problema seria utilizar um segundo critério, que verifica a razão entre as duas expectativas de maior valor - com a maior delas no denominador. Esta razão deve estar abaixo de um segundo limiar para que a solução seja considerada livre de ambiguidades. Assim:

$$ambig = \left(\frac{\max_2 (\mu_1, \mu_2, \ldots, \mu_p)}{\max (\mu_1, \mu_2, \ldots, \mu_p)} > thresh_{ambig} \right), \ 0 \leqslant thresh_{ambig} \leqslant 1 \quad (3.2)$$

$$aceita = (\max (\mu_1, \mu_2, \ldots, \mu_p) > thresh) \wedge \neg ambig \quad (3.3)$$

onde $thresh_{ambig}$ é o limiar de ambiguidade, enquanto $\max_2$ é a função que retorna o segundo maior valor dentre as expectativas. A aceitação, assim, passaria a depender de duas condições, conforme descrito em (3.3).

3.2.3 Algoritmo de Tomada de Decisão

A partir das definições feitas para o dispositivo adaptativo, o Algoritmo 5 esboça o processo de tomada de decisão hierárquica. Os dispositivos devem estar organizados como uma estrutura de lista D, e devem dispor de um método $expectativa\,(x)$ que calcule a expectativa de certeza de cada alternativa. Cada dispositivo deve incorporar sua própria lógica para este método, adequando-a de acordo com suas características. Note que o algoritmo é recursivo, incrementando o valor do índice k em cada recursão - o que significa utilizar o k-ésimo dispositivo para solucionar o problema corrente. A recursão é interrompida assim que uma solução é aceita pelo critério de aceitação, ou

então ao esgotar a fila de dispositivos.

Algoritmo 5 $decisaoHierarquica\,(D, k, x, a, P)$

Entrada
D: Lista de dispositivos
k: Índice do dispositivo a ser utilizado
x: Vetor de características
Saída
a: Alternativa escolhida
P: Lista com as expectativas de certeza calculadas para cada dispositivo

1: $\mu \leftarrow D\,[k]\,.expectativa\,(x)$
2: $P\,[k] \leftarrow \mu$
3: **if** $aceitaDecisao\,(\mu)$ **then**
4: $\quad a \leftarrow \arg\max\,(\mu_1, \ldots, \mu_p)$
5: **else if** $k + 1 < D.length$ **then**
6: $\quad decisaoHierarquica\,(D, k + 1, x, a, P)$
7: **else**
8: $\quad decisaoFinal\,(D, x, a, P)$
9: **end if**

Nota: se o vetor μ apresentar dois ou mais valores máximos idênticos, deve-se utilizar algum critério de desempate para a função $\arg\max$ apresentada na linha 4. Neste trabalho, no caso de empate, este operador retorna a alternativa de menor índice.

A função $aceitaDecisao\,(\mu)$ determina a condição de aceitação de uma decisão, devendo ser implementada de acordo com o critério adotado. Como exemplo, o Algoritmo 6 ilustra como seria este método utilizando a condição descrita pela equação (3.1), representando a função auxiliar $FM_{aceitacao}$ definida previamente.

Algoritmo 6 $aceitaDecisao\,(\mu)$

Entrada
μ: vetor com as expectativas de certezas a serem testadas
Saída
$true$ ou $false$

1: **return** $\max\,(\mu_1, \ldots, \mu_p) > thresh$

Em contrapartida, a linha 8 do Algoritmo 5 representa a função auxiliar $FM_{estouro}$, i.e., a estratégia utilizada quando nenhum dispositivo foi capaz de obter uma solução aceita pela condição imposta por $aceitaDecisao\,(\mu)$. A escolha desta estratégia deve ser feita de acordo com as exigências do problema. A estratégia mais simples seria retornar uma solução nula, deixando claro que o sistema não foi capaz de solucionar o problema. Caso seja exigida uma solução não-nula, outras estratégias poderiam ser empregadas, tais como utilizar um sistema de votação com os valores de μ obtidos em cada dispositivo, ou então, utilizar a solução do último dispositivo da fila - esta última estratégia é descrita pelo Algoritmo 7.

Algoritmo 7 $decisaoFinal\,(D, x, a, P)$

Entrada
D: Lista de dispositivos
x: Vetor de características
Saída
a: Alternativa escolhida
P: Lista com as expectativas de certezas calculadas para cada dispositivo

1: $\mu \leftarrow D\,[D.lastElement]\,.expectativa\,(x)$
2: $i \leftarrow \mathrm{index\,of}\,(\max\,(\mu))$
3: $a \leftarrow a_i$

Finalmente, o Algoritmo 8 ilustra o ponto de entrada do sistema, e a maneira como o processo de decisão hierárquica se relaciona com o processo de realimentação, responsável pelo aprendizado incremental (descrito nas seções seguintes). Deve ser notado que, após consultar os dispositivos, a solução obtida é realimentada, dando margem a cada um dos dispositivos para aprender através das soluções dos demais - em especial, permite que dispositivos mais rápidos aprendam com as soluções obtidas através daqueles com maiores taxas de acerto, potencialmente aumentando a frequência com que suas soluções são aceitas pelo critério de aceitação, e consequentemente, diminuindo o tempo médio de resposta do sistema.

Estas realimentações podem ser causadas por um bloco de auto-análise, ou então, por um viés de repetição. Os valores $cr_{repetition}$ e $cr_{analise}$ representam a credibilidade destas realimentações, e são definidas na seção 3.3.1, enquanto a função $realimenta\,(BD, D, x, a, cr)$, responsável pelo aprendizado incremental, é descrita na seção 3.4.1. Em contrapartida, a função $analise\,(x, a)$ deve ser responsável por realizar a auto-análise e, caso não valide a solução original, retorna uma outra solução, a ser realimentada.

Nota: apesar de especificado, este bloco de análise não será empregado neste trabalho, conforme citado na seção 3.1. Assim, apenas o viés de repetição será analisado como forma de auto-realimentação - realimentações originadas por uma lógico do próprio sistema.

Algoritmo 8 $decisao\,(x, a)$

Entrada
x: Vetor de características
Saída
a: Alternativa encontrada

1: $P \leftarrow \{\}$
2: Discretiza x
3: $decisaoHierarquica\,(D, 0, x, a, P)$
4: **if** $a \neq \epsilon$ **then**
5: **if** Existe em módulo de análise? **then**
6: $a_{analise} \leftarrow analise\,(x, a)$
7: $cr_{analise} \leftarrow credibilidadeAnalise\,()$
8: $realimenta\,(BD, D, x, a_{analise}, cr_{analise})$
9: **else if** $cr_{repetition} > 0$ **then**
10: $realimenta\,(BD, D, x, a, cr_{repetition})$
11: **end if**
12: **end if**
13: **return** a

A Figura 19 resume o processo hierárquico de tomada de decisão.

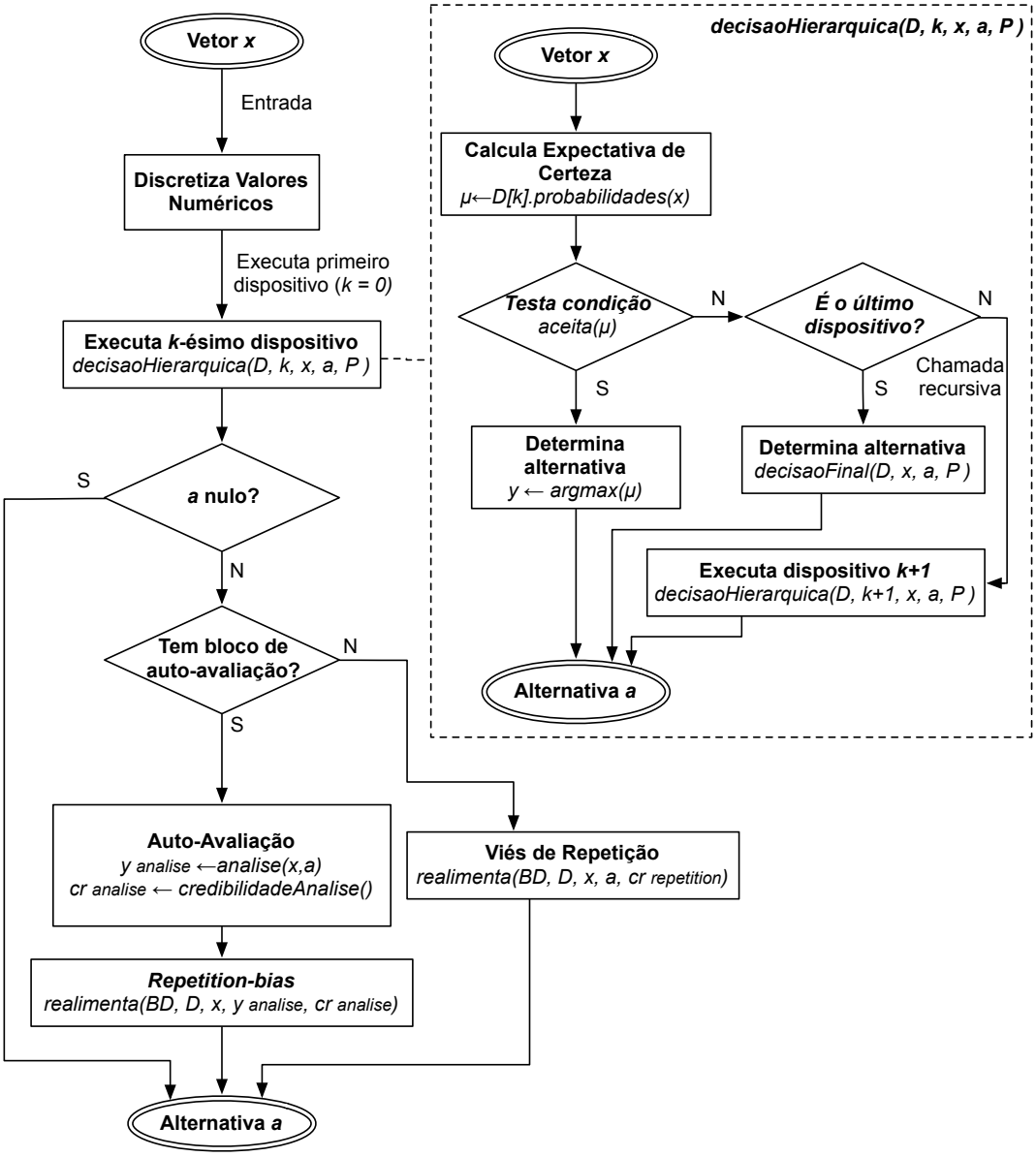


Figura 19: Fluxo resumido do processo hierárquico de tomada de decisão.

3.3 Gerenciamento de Conhecimento

Esta seção descreve a forma com que o sistema trata o armazenamento de casos conhecidos - processo iniciado a partir de informações recebidas de realimentações, e que resulta em um modelo de base de conhecimento capaz de estimar o valor-verdade de um caso já visto, bem como sua confiabilidade.

3.3.1 Realimentação

Conforme descrito anteriormente, realimentação é uma relação que avalia um caso (x, a) através de um valor-verdade v, informando se a alternativa a é ou não uma solução adequada para o vetor de características x. Uma instância de realimentação é representada por (x, a, v, β, cr), tal que:

- x: vetor de características analisado.
- a: alternativa analisada.
- v: valor-verdade que indica se a é uma solução válida para x. Realimentações positivas têm $v = 1$, enquanto realimentações negativas têm $v = 0$.
- β: valor relativo de reforço, para ponderar a influência das realimentações positivas e negativas. Semelhante ao peso proposto por Einhorn, descrito pela equação (2.2).
- cr: credibilidade do gerador desta realimentação. Deve ser um valor positivo maior que zero.

A escolha do valor de cr é feito de forma a ponderar a influência de cada fonte de realimentação em relação às demais. Valores maiores são atribuídos para fontes consideradas mais confiáveis:

- Partindo do pressuposto de que seres humanos são menos propensos a erros do que as máquinas, realimentações vindas diretamente do *usuário* devem exibir um valor de cr relativamente maior do que os das demais fontes. O mesmo vale para informações vindas de um operador humano.

- Por outro lado, informações recebidas por avaliações de *máquina* devem exibir credibilidades menores, pois estão mais sujeitas a erros. Módulos como a auto-avaliação, análise temporal e análise de casos sem solução se enquadram nesta categoria.

- Na ausência de realimentações para uma solução, pode ser utilizado um bloco genérico de *viés de repetição*. Este bloco reforça a alternativa escolhida com $v = 1$, baseando-se na crença de que se estiver errado, alguém lhe corrigirá. Por não ser uma fonte de informações confiável, utiliza-se um valor pequeno para cr - no caso, uma constante $cr_{repetition}$ pré-definida.

Conceitualmente, apesar de não ser uma realimentação, o treinamento do sistema também é representado por (x, a, v, β, cr), exceto que sua credibilidade cr apresenta valores muito maiores, pois assume que o treinamento é isento de erros. Portanto neste projeto, ainda que seja conceitualmente diferente, o treinamento também utilizará esta notação.

O peso w da realimentação é obtido a partir dos valores de reforço β e credibilidade cr:

$$w = \beta \cdot cr \tag{3.4}$$

A partir de um conjunto de realimentações recebidas, deve-se estabelecer uma forma de estimar a confiança de uma solução, conforme sugerido por Einhorn (EINHORN; HOGARTH, 1978). No caso, utiliza-se uma soma ponderada por w, descrito a seguir.

3.3.2 Cálculo do Valor-Verdade e do Intervalo de Confiança

Como citado na seção 3.1, o valor-verdade é uma estimativa empírica da probabilidade de uma alternativa a ser solução de um vetor x. Baseando-se na frequência com que a alternativa foi considerada válida, este valor é calculado a partir de uma média ponderada do valor-verdade das n realimentações recebidas para o caso (x, a):

$$\bar{v} = \frac{V}{W}, \quad \left(V = \sum_{k=1}^{n} v_k \cdot w_k, \quad W = \sum_{k=1}^{n} w_k \right) \tag{3.5}$$

onde V pode ser interpretado como número de vezes que a alternativa foi considerada correta dentre as W vezes em que o caso foi avaliado - estes são os contadores citados na seção 3.1. Assim, para cada caso (x, a), se faz necessário armazenar estes dois somatórios para que a estimativa da verdade $\bar{v}$ da alternativa a com relação ao vetor x possa ser calculada - supostamente, a alternativa de melhor verdade é a mais adequada. Note que estes somatórios podem ser facilmente atualizados conforme novas realimentações são recebidas.

Em conjunto, pode-se estimar o intervalo de confiança deste valor-verdade, permitindo determinar, de forma objetiva, a confiabilidade do valor calculado - quanto menor o intervalo, mais preciso deve ser o valor-verdade. Tipicamente, a estimativa do intervalo é realizada determinando dois limites, v_{low} e v_{upp}, tal que a probabilidade da média verdadeira μ (i.e., o verdadeiro valor-verdade) estar entre estes dois valores seja $1 - \alpha$. Assim:

$$P\left(v_{low} \leqslant \mu \leqslant v_{high}\right) = 1 - \alpha \tag{3.6}$$

onde α representa o nível de significância do intervalo de confiança. Valores típicos de α incluem 10%, 5%, 1%, $0,5\%$ e $0,1\%$.

A estimativa do intervalo de confiança de qualquer parâmetro de uma po-

pulação depende fortemente da maneira com que os valores das instâncias variam entre si - i.e., depende da distribuição característica do parâmetro medido. Como as verdades das realimentações assumem valores binários - verdadeiro (1) ou falso (0) - tem-se uma distribuição de Bernoulli, cujo intervalo de confiança em torno da média $\bar{v}$, quando aproximado por uma distribuição normal, é:

$$\bar{v} \pm z_{1-\alpha/2}\sqrt{\frac{\bar{v}\,(1-\bar{v})}{W}} \tag{3.7}$$

onde $z_{1-\alpha/2}$ é o $(100 \cdot (1-\alpha/2))$-ésimo percentil de uma distribuição normal, enquanto α é a porcentagem esperada da amostra que devem ficar fora do intervalo calculado. O cálculo do percentil é bastante conhecido, e se encontra descrito no Anexo A.

Todavia, estudos estatísticos demonstram que este intervalo de confiança pode não representar a real situação da média calculada, especialmente quando o número de exemplos amostrais é baixo, ou então, se a média verdadeira é muito próxima de 0 ou 1 (BROWN; CAI; DASGUPTA, 2001). Como exemplo, ao lançar uma moeda não-viciada três vezes, a probabilidade de sair três vezes caras é bastante significativa - no caso, 12,5%. No entanto, para esta amostra, a estimativa da massa de probabilidade de um lançamento dar cara seria 1 ± 0, muito distante do valor real.

A literatura propõe alguns testes para verificar se o intervalo de confiança pode ser utilizado (BROWN; CAI; DASGUPTA, 2001). Abaixo, alguns exemplos destes testes (dado que p é a média calculada, e n, o número de exemplos amostrais):

- $np \geqslant 5$ e $n\,(1-p) \geqslant 5$
- $np(1-p) \geqslant 5$
- $n \geqslant 50$, a não ser que p seja próximo de zero

No entanto, foi demonstrado que a significância estatística destes testes é muito baixa, mesmo quando $n \to \infty$ (BROWN; CAI; DASGUPTA, 2001). Para contornar este problema, utiliza-se alternativas para estimar a média e seu respectivo intervalo de confiança. Uma destas alternativas é o método de *Agresti-Coull*, que artificialmente insere uma quantidade igual de elementos de valor 0 e 1. Tal ação move a média estimada para um valor mais próximo de $0,5$ e alarga o intervalo de confiança, especialmente quando a amostra possui poucos elementos. Os efeitos desta amostra artificial se dispersam conforme a amostra aumenta, fazendo com que a média de Agresti-Coull convirja para a média convencional.

Utilizando este método, a nova estimativa do valor-verdade $\hat{v}$ e seu intervalo de confiança $c_{1-\alpha/2}$ são dados através das seguintes funções:

$$\hat{W} = W + \left(z_{1-\alpha/2}\right)^2 \tag{3.8}$$

$$\hat{v} = \frac{\hat{V}}{\hat{W}} = \frac{V + \frac{\left(z_{1-\alpha/2}\right)^2}{2}}{\hat{W}} \tag{3.9}$$

$$\hat{v} \pm c_{1-\alpha/2} = \hat{v} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{v}\left(1-\hat{v}\right)}{\hat{W}}} \tag{3.10}$$

A equação (3.8) mostra que o número de elementos artificiais é $\left(z_{1-\alpha/2}\right)^2$, enquanto a equação (3.9) mostra que metade delas são verdadeiras, enquanto as demais são falsas. Por fim, a equação (3.10) é idêntica à equação original (3.7), mas utiliza os novos valores $\hat{v}$ e $\hat{W}$.

A Figura 20 ilustra um exemplo em que é estimada a média de uma variável que segue uma distribuição de Bernoulli com $p = 80\%$ e $\alpha = 5\%$, conforme o tamanho da amostra aumenta. Note que a média real esteve dentro do intervalo de Agresti-Coull a todo o momento - algo que não ocorre com o intervalo

convencional quando a amostra só continha 8 elementos (todas elas tinham valor 1, logo, média estimada foi 1 com intervalo zero, não abrangendo a média verdadeira). Vale também ressaltar que ambas as médias convergem para a média real conforme a amostra aumenta.

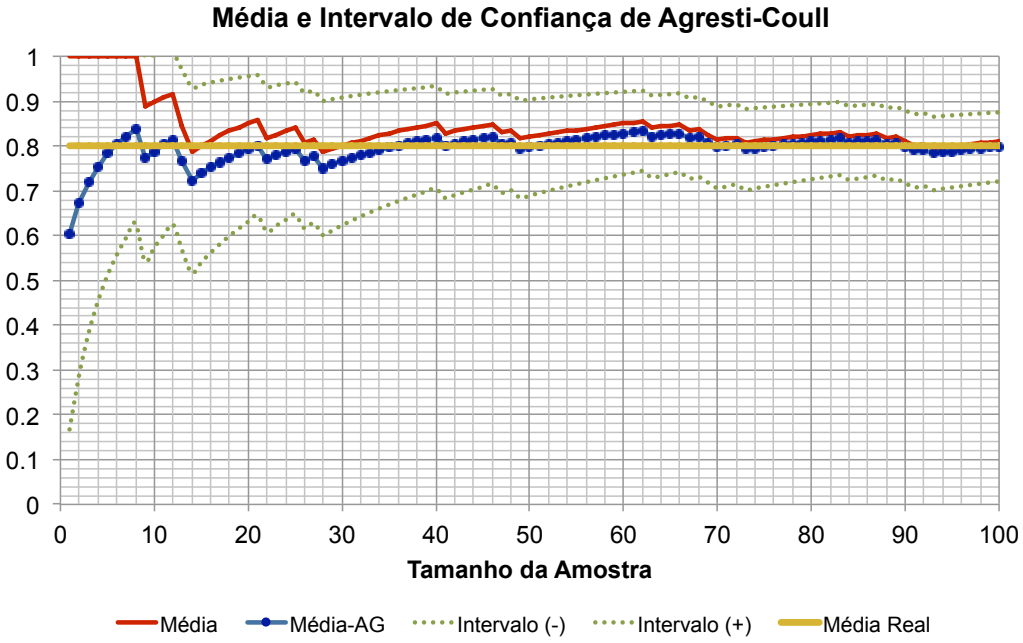


Figura 20: Gráfico comparativo da estimativa da média convencional (linha contínua) com a média de Agresti-Coull (linha marcada com •) de uma amostra que segue uma distribuição de Bernoulli com $p = 80\%$

Nota: o método de Agresti-Coull é funcional mesmo para $W = 0$*. No caso,* $\hat{v} = c_{1-\alpha/2} = 0,5$*, englobando todo o intervalo de valores válidos. Portanto, estes são os valores retornados para casos que nunca foram realimentados.*

Para cada realimentação recebida, os somatórios V e W do respectivo caso devem ser atualizados, bem como os valores-verdade e intervalos de confiança. Estas mudanças devem ser notificadas aos dispositivos, para que se adaptem ao novo conhecimento - processo descrito na seção 3.4.1.

3.3.3 Base de Conhecimento

Conforme descrito na seção 3.1, a base de conhecimento é responsável pelo armazenamento dos casos conhecidos, para que as informações de valor-verdade e confiança sejam disponibilizadas aos dispositivos - mais especificamente, armazena os contadores V e W de cada caso (x, a) , necessários para o cálculo do valor-verdade associado.

Obrigatoriamente, esta base de conhecimento deve ser capaz de receber inserções, modificações e exclusões de informações, para que seja possível agregar novos conhecimentos e eliminar informações parasitárias, e, idealmente, deve estar organizada para permitir buscas espaciais de forma otimizada, sem a necessidade de consultar toda a base para encontrar os vizinhos mais próximos de um vetor x, conforme descrito na seção 2.5.1.2. A Figura 21 ilustra um exemplo de base de conhecimento utilizando uma estrutura de árvore *R*-Tree*.

A otimização da busca, entretanto, depende fortemente dos critérios envolvidos, e.g., *R*-Tree* trata apenas problemas numéricos (BECKMANN et al., 1990). Para simplificação, será utilizada uma busca linear neste trabalho, para que o cálculo de distância seja independente dos critérios, em detrimento do tempo de resposta.

Uma consideração, contudo, deve ser feita: enquanto critérios nominais possuem uma quantidade finita de valores válidos, critérios numéricos podem assumir, teoricamente, valores reais infinitos. Para problemas que contenham, ao menos, um critério numérico, torna-se inviável armazenar todos os casos conhecidos, pois a base cresceria indefinidamente. Para limitar o tamanho da base a um valor finito, discretiza-se os valores numéricos, transformando-os em valores inteiros finitos. Assim, do ponto de vista da base, cada vetor

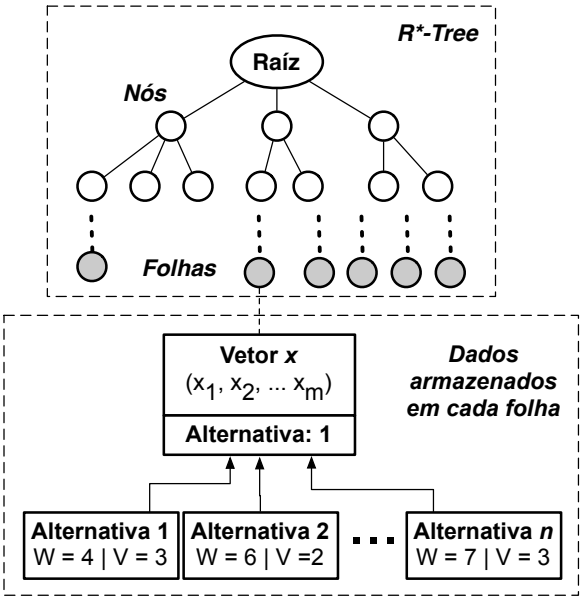


Figura 21: Um possível modelo para a base de conhecimento utilizando uma *R**-Tree*. Cada folha possui a representação x de um vetor e os contadores V e W de cada alternativa.

x é formada por valores nominais ou discretizados. A seção 2.2.1 descreve algumas técnicas de discretização comumente utilizadas.

Logo, quando for necessário consultar o valor-verdade e o intervalo de um caso (x, a) qualquer, ou então, ao atualizar os contadores W e V deste caso - através da função $BD.realimenta$ presente no Algoritmo 10 - serão utilizados os valores discretizados de x para realizar a consulta.

3.4 Aprendizado

Tradicionalmente, todo sistema de tomada de decisão possui uma fase de treinamento para adequá-lo a um problema em particular, e, em muitos casos, é a única forma de transmitir conhecimento ao sistema. Sistemas ditos como evolutivos permitem que novas informações sejam incrementalmente incorporadas durante suas execuções.

Esta seção apresenta estes dois processos de aprendizado, e como eles são utilizados neste trabalho.

3.4.1 Aprendizado Incremental

Conforme citado na seção 3.2.1, cada dispositivo obtém novas informações através de sua camada adaptativa. Isso é feito informando-o sobre as mudanças ocorridas no valor-verdade de um caso e no intervalo de confiança associado, permitindo que sua abstração seja atualizada. O Algoritmo 9 ilustra como as realimentações são geradas a partir de uma solução - no caso, é uma realimentação para cada alternativa, com $v = 1$ e $\beta = \beta_+$ para a solução adotada a, e $v = 0$ e $\beta = \beta_-$ para as demais. Já o Algoritmo 10 mostra como a propagação de uma realimentação é realizada.

Algoritmo 9 $realimenta\,(BD, D, x, a, c)$

Entrada
BD: Base de conhecimento com os contadores de cada caso
D: Lista de dispositivos
x: Vetor de características
a: Alternativa escolhida pela fonte realimentadora
c: Credibilidade da fonte

1: **for all** $a_i \in A$ **do**
   {Verdade da realimentação e valor de reforço relativo}
   {Ver seção 3.3.1}
2: $v \leftarrow 0$
3: $\beta \leftarrow \beta_-$
4: **if** $a_i = a$ **then**
   {Alternativa a_i era a escolhida: valor-verdade e β positivos}
5: $v \leftarrow 1$
6: $\beta \leftarrow \beta_+$
7: **end if**
8: $propaga\,(BD, D, x, a_i, v, \beta, c)$
9: **end for**

Nestes algoritmos, a função $BD.verdade\,(x, a)$ retorna o valor-verdade de uma alternativa a com relação ao vetor de características x utilizando a equa-

Algoritmo 10 $propaga\,(BD, D, x, a, v, \beta, c)$

Entrada
BD: Base de conhecimento com os contadores de cada caso
D: Lista de dispositivos
x: Vetor de características
a: Alternativa a ser realimentada
v: Valor-verdade da alternativa, segundo a fonte realimentadora
β: Valor relativo de reforço
c: Credibilidade da fonte

{Calcula valor-verdade $\hat{v}$ e intervalo de confiança ic de a em x}
1: $\hat{v} \leftarrow BD.verdade\,(x, a)$
2: $ic \leftarrow BD.intervalo\,(x, a)$
3: $BD.realimenta\,(x, a, v, \beta, c)$
{Calcula diferença no valor-verdade e intervalo de confiança}
4: $dif_v \leftarrow BD.verdade\,(x, a) - \hat{v}$
5: $dif_{ic} \leftarrow BD.intervalo\,(x, a) - ic$
6: **for all** $d \in D$ **do**
{Propaga diferenças aos dispositivos}
7: $d.aprende\,(x, a, dif_v, dif_{ic})$
8: **end for**

ção (3.9), enquanto $BD.intervalo\,(x, a)$, de forma análoga, calcula o intervalo de confiança através da equação (3.10). A realimentação em si é feita através do método $BD.realimenta\,(x, a, v, \beta, c)$, que faz uso das equações (3.4) e (3.5) para atualizar os contadores, e, portanto, o valor-verdade.

Após as realimentações, as alterações no valor-verdade e no intervalo de confiança são propagadas aos dispositivos através da função $d.aprende\,(x, a, dif_v, dif_{ic})$, referente à função auxiliar $FM_{aprendizado}$ definida previamente. Esta chamada permite que cada dispositivo atualize sua abstração de acordo com as alterações sofridas. A lógica desta função deve ser implementada para cada dispositivo em particular, de acordo com a estratégia utilizada para atualizar sua abstração.

Note que a escolha da credibilidade afeta a velocidade com que uma informação se torna confiável. Valores elevados fazem com que novas informações sejam rapidamente consolidadas como verdadeiras na base de conheci-

mento, potencialmente amplificando os efeitos de informações ruidosas. Por exemplo, com $v = 1$, $c = 4$ e $\beta = 1/2$, um novo caso passa de $v = W = 0$ para $V = W = 10$ em apenas 5 realimentações. Com base nestes contadores, o valor-verdade salta de $0,5$ para $\hat{v} \approx 0,86$ com intervalo de $c_{1-\alpha/2} \approx 0,18$ - um valor bastante elevado para poucas realimentações (e, portanto, propensa a erros), considerando que uma verdade absoluta tem valor-verdade igual a 1.

3.4.2 Treinamento

O treinamento do sistema de tomada de decisão é feito de forma supervisionada, i.e., através de instâncias rotuladas com a alternativa esperada. A partir deste treinamento, duas ações devem ser realizadas pelo sistema:

- A base de conhecimento, inicialmente vazia, deve ser carregada com os valores das instâncias rotuladas, criando os contadores iniciais para cada caso (x, a).
- A partir destes dados recém-carregados, os dispositivos devem inferir suas próprias abstrações para tomada de decisão.

Devido à semelhança entre o treinamento e o processo de realimentação (conforme citado na seção 3.3.1), pode-se reutilizar o Algoritmo 9 para esta finalidade: para cada instância de treinamento, aciona-se este método com o vetor x e a alternativa correspondente a. O valor cr_0 representa a credibilidade do treinamento, devendo assumir um valor muito maior do que de outras realimentações (supondo que as instâncias iniciais são corretas). Note que se $cr_0 = \infty$, o valor-verdade, calculado através da equação (3.9), torna-se 1 para a alternativa correta e 0 para as demais, e impede que este valor seja alterado por realimentações futuras.

3.5 Exemplos de Dispositivos Adaptativos

Neste trabalho, foram desenvolvidos três dispositivos adaptativos baseados em métodos desenvolvidos em trabalhos anteriores, e que, por natureza, mostravam-se potencialmente adequadas para a incorporação de recursos de adaptatividade: *TDAE*, classificador de Bayes e *k-Nearest Neighbor*. Cada um deles está descrito a seguir.

3.5.1 *TDAE*

A Tabela de Decisão Adaptativa Estendida, utilizada como uma das bases conceituais deste trabalho, é também um dos dispositivos utilizados para validar o sistema de tomada de decisão proposto. No entanto, a tabela utilizada neste trabalho apresenta duas diferenças com relação à definição original:

- Ao invés de utilizar valores binários, as entradas da tabela são estendidas (i.e., entradas podem assumir valores discretos arbitrários) para que critérios nominais possam ser utilizados mais facilmente. A exemplo da base de conhecimento, a tabela também requer que os valores numéricos sejam discretizados.

- Não serão permitidas regras reduzidas (i.e., regras que contenham valores nulos associados a um critério), nem regras conflitantes, portanto, uma regra tem apenas um único caso equivalente. Apesar de esta limitação impedir que múltiplas regras sejam condensadas em um conjunto menor, a equivalência entre regra e caso simplifica o processo de atualização das regras.

Utilizando a notação da seção 2.1.1, o vetor x é representado por um símbolo m-dimensional s na tabela, enquanto a alternativa escolhida é represen-

tada pelo símbolo de saída z da regra. Vale também ressaltar que a aplicação de um regra da *TDAE* não altera diretamente sua configuração atual c_{atual}. Caso o conjunto de regras venha a ser alterada, as mudanças serão resultado do recebimento de realimentações, que deve chamar a função $aprende$ correspondente à *TDAE* (Algoritmo 12, definido mais adiante).

Dentro do sistema de tomada de decisão proposto, $\mu_i = b/n$ para a i-ésima alternativa, tal que n é a quantidade de regras encontradas, enquanto b é a frequência com que a alternativa a_i aparece nas regras encontradas. Este cálculo é ilustrado pelo Algoritmo 11, e utilizado pelo Algoritmo 5 apresentado previamente. Vale ressaltar que esta definição permite que as soluções obtidas através de dispositivos guiados por regra sejam reescritas na forma de probabilidades.

Se nenhuma regra for encontrada, $\mu = (0, \ldots, 0)$. Ainda que este vetor não seja normalizado, ele não deve ser aceito pelo critério de aceitação de soluções, forçando, assim, que o próximo dispositivo da fila seja utilizado.

Nota: neste trabalho, por não ser permitido o uso de regras reduzidas nem de regras de mesma condição, o número de regras encontradas nunca será maior que 1*, o que faz com que a alternativa prevista pela regra tenha* $\mu_i = 1$*. Contudo, a formulação acima é mais flexível e extensível a outros dispositivos guiados a regra que poderiam apresentar mais de uma regra aplicável.*

Quanto ao aprendizado, as regras das tabelas devem ser atualizadas à medida que novas realimentações são recebidas, alterando o valor-verdade dos casos. Caso a alternativa de maior valor-verdade seja alterada, o valor z da regra correspondente deve ser atualizada com esta nova alternativa. Novos casos também devem ser memorizados na forma de novas regras, retornando o alternativa de maior valor-verdade. O Algoritmo 12 ilustra este processo de aprendizado. A função $BD.melhorAlternativa\,(x)$ deve retornar a alternativa

Algoritmo 11 $TDAE.expectativa\,(x)$

Entrada
x: Vetor de características
Saída
Vetor μ com as expectativas de certezas

1: $\mu \leftarrow (0, 0, \ldots, 0)$
2: $CR_T \leftarrow buscaRegras\,(TDAE, x)$
3: $n \leftarrow |CR_T|$
4: **for all** $r \in CR_T$ **do**
5: $\quad a \leftarrow r.z$
6: $\quad i \leftarrow \text{index of}\,(a)$
7: $\quad \mu_i \leftarrow \mu_i + 1/n$
8: **end for**

a_i de maior valor μ_i para um determinado vetor x. Note que, neste dispositivo, a única informação necessária é a alternativa de maior valor-verdade, o que faz com que os argumentos a, dif_v e dif_{ic} não sejam utilizados neste caso.

3.5.2 Classificador de Bayes

Neste trabalho, o classificador de Bayes, descrito previamente na seção 2.5.2, constitui um dispositivo de menor precisão, mas que pode ser executado rapidamente, sendo assim, utilizado em casos em que o custo do erro é baixo. Por ser um método probabilístico, seu funcionamento pode ser adaptado ao sistema de tomada de decisão proposto atribuindo o valor da probabilidade $P\,(a_i|x)$ para a expectativa μ_i da alternativa a_i correspondente.

Dada a formulação original, expressa na equação (2.12), é necessário definir a forma com que os valores $P\,(c)$ e $P\left(x_{[i]}|c\right)$ serão calculados em função do valor-verdade. Para o valor de $P\,(c)$, o cálculo é realizado somando os valores-verdade de todas os casos que contém c como alternativa, e dividindo pela soma de valores-verdade de todos os casos. Supondo que há n vetores

Algoritmo 12 $TDAE.aprende\,(x, a, dif_v, dif_{ic})$

Entrada
x: Vetor de características
a: Alternativa avaliada
dif_v: Diferença no valor-verdade da alternativa a com relação à x
dif_{ic}: Diferença no intervalo de confiança da alternativa a com relação à x

$a \leftarrow BD.melhorAlternativa\,(x)$
$CR_T \leftarrow buscaRegras\,(TDAE, x)$
if $|CR_T| = 0$ **then**
  {Não existia uma regra para x: criar uma}
  {Notação abaixo segue o padrão descrito na seção 2.1.1}
  $r \leftarrow (\epsilon, c_{atual}, x, c_{atual}, a, \epsilon)$
  Insere r no conjunto de regras AR
else if $|CR_T| = 1$ **then**
  {Atualizar regra existente}
  $r \leftarrow CR_T\,[0]$
  if $r.z \neq a$ **then**
    $r.z \leftarrow a$
  end if
  Reinsere r no conjunto de regras AR
else
  Situação de erro
end if

de características e p alternativas salvas na base de conhecimento, temos:

$$P\,(c) = \frac{S_v\,(c)}{S_{total}}, \quad S_v\,(c) = \sum_{i=1}^{n} \hat{v}\,(x_i, c)\,, \quad S_{total} = \sum_{j=1}^{p} \sum_{i=1}^{n} \hat{v}\,(x_i, c_j) \tag{3.11}$$

onde $\hat{v}\,(x, c)$ é a função que calcula o valor-verdade da alternativa c para x, utilizando a equação (3.9). De forma similar, cada elemento $P\left(x_{[i]}|c\right)$ pode ser consultado somando o número de vezes que o valor $x_{[i]}$ aparece quando um vetor de características tem a alternativa c como solução. Assim:

$$P\left(x_{[i]}|c\right) = \frac{S_u\left(x_{[i]}, c\right)}{S_v\,(c)}, \quad S_u\left(x_{[i]}, c\right) = \sum_{j=1}^{n} \hat{v}\,(x_j, c)\, I\left(x_{j\,[i]} = x_{[i]}\right) \tag{3.12}$$

onde x_j é o j-ésimo vetor da base de conhecimento, enquanto $x_{j\,[i]}$ é seu valor associado ao i-ésimo critério.

Na prática, os três somatórios definidos nas equações (3.11) e (3.12) são

armazenados em memória, para que os cálculos destas probabilidades possam ser realizados com apenas uma única divisão. O valor de cada somatório deve ser atualizado sempre que houver uma mudança em algum de seus elementos - i.e., sempre que algum valor-verdade for alterado. A atualização de qualquer somatório pode ser feita utilizando a seguinte propriedade:

$$S_{t+1} = S_t + dif_i = S_t + \left(s_{i|t+1} - s_{i|t}\right) \tag{3.13}$$

onde S_t é um somatório qualquer em um instante t, enquanto $s_{i|t}$ é o valor do i-ésimo elemento do somatório em um instante t. Se este valor, em um instante $t+1$, tiver sido alterado, a contribuição antiga $s_{i|t}$ deve ser removida, substituindo-a pelo novo valor $s_{i|t+1}$.

Como o sistema fornece as diferenças no valor-verdade aos dispositivos, pode-se simplesmente somar o efeito desta diferença em cada um dos três somatórios. Logo, se o valor-verdade de um caso (x_b, c_b) for alterado, temos:

$$S_v\left(c_b\right)_{t+1} = S_v\left(c_b\right)_t + dif_v\left(x_{b\,[i]}, c_b\right) \tag{3.14}$$

$$S_u\left(x_{[i]}, c_b\right)_{t+1} = S_u\left(x_{[i]}, c_b\right)_t + dif_v\left(x_b, c_b\right) I\left(x_{b\,[i]} = x_{[i]}\right) \tag{3.15}$$

$$S_{total\ t+1} = S_{total\ t} + dif_v\left(x_b, c_b\right) \tag{3.16}$$

onde $dif_v\left(x_b, c_b\right)$ é diferença do valor-verdade verificado no caso citado. Note que esta é a diferença estimada pelo Algoritmo 6, e pode, inclusive, ser negativa, caso o valor-verdade tenha diminuído.

Os Algoritmos 13 e 14 ilustram como os somatórios são utilizados para o cálculo das expectativas de certeza, e como são atualizados a partir das alterações causadas por realimentações.

Algoritmo 13 $Bayes.expectativa\,(x)$

Entrada
x: Vetor de características
Saída
Vetor μ com as expectativas de certeza

1: $\mu \leftarrow (0, 0, \ldots, 0)$
2: **for** $j = 1 \rightarrow p$ **do**
{Calcular $P\,(a_j)$}
3: $P \leftarrow S_v\,(a_j)\,/S_{total}$
{Calcular $\prod_{i=1}^{m} P\left(x_{[i]}|a_j\right)$}
4: $prod \leftarrow 1$
5: **for** $i = 1 \rightarrow m$ **do**
6: $prod \leftarrow prod \cdot S_u\left(x_{[i]}, a_j\right)/S_v\,(a_j)$
7: **end for**
8: $\mu_i \leftarrow P \cdot prod$
9: **end for**
{Calcular evidência $Z\,(x)$}
10: $Z = 0$
11: **for** $j = 1 \rightarrow p$ **do**
12: $Z \leftarrow Z + \mu_i$
13: **end for**
{Normalizar μ pela evidência}
14: **for** $j = 1 \rightarrow p$ **do**
15: $\mu_i \leftarrow \mu_i/Z$
16: **end for**

Algoritmo 14 $Bayes.aprende\,(x, a, dif_v, dif_{ic})$

Entrada
x: Vetor de características
a: Alternativa avaliada
dif_v: Diferença no valor-verdade da alternativa a com relação à x
dif_{ic}: Diferença no intervalo de confiança da alternativa a com relação à x

1: $S_{total} \leftarrow S_{total} + dif_v$
2: $S_v\,(a) \leftarrow S_v\,(a) + dif_v$
3: **for** $i = 1 \rightarrow m$ **do**
4: $S_u\left(x_{[i]}, a\right) \leftarrow S_u\left(x_{[i]}, a\right) + dif_v$
5: **end for**

3.5.3 *k-Nearest Neighbor*

Por fim, o método *k-NN* proporciona, neste trabalho, um dispositivo com tempo de resposta muito maior que os demais, porém, de maneira geral, mais preciso. Apesar de que algoritmos mais rápidos e precisos existam - e.g., *random forests* (CARUANA; NICULESCU-MIZIL, 2006) - a difusão e popularidade do *k-NN* o torna atraente como parâmetro de comparação.

Utilizando o *k-NN* ponderado da equação (2.4) como base, a probabilidade não-normalizada de um vetor x ter c como solução, baseado nos k vizinhos mais próximos, é descrita conforme a equação (3.17).

$$P\left(c|x\right) = \sum_{i=1}^{k} I\left(x, c\right) \cdot w\left(x, x_i\right) \tag{3.17}$$

Todavia, diferentemente desta definição, que faz uso da função indicadora, a disponibilidade do valor-verdade faz com que um vetor x não tenha uma única solução, mas sim, graus de pertinência em cada solução. Além disso, o intervalo de confiança possibilita realizar uma ponderação dos casos, favorecendo aqueles com intervalos menores. Assim, substituindo-se a função indicadora por $\hat{v}\left(x, c\right)$ e considerando a contribuição do intervalo de confiança $c\left(x, c\right)$ (calculado através da equação (3.10)), tem-se:

$$P\left(c|x\right) = \sum_{i=1}^{k} \hat{v}\left(x_i, c\right) \cdot w\left(x, x_i\right) \cdot b\left(x_i, c\right) \tag{3.18}$$

$$b\left(x, a\right) = 1 - 2c\left(x, a\right) \tag{3.19}$$

onde $0 \leqslant b\left(x, a\right) \leqslant 1$ é uma função peso que faz uso do intervalo de confiança $c\left(x, a\right)$, calculado conforme mostra a equação (3.10). O valor deste intervalo varia entre $1/2$ e 0.

Após o cálculo de $P\left(c|x\right)$ de cada alternativa, a expectativa de certeza

pode ser extraída normalizando-se estas probabilidades:

$$\mu_i = \frac{P\left(a_i|x\right)}{\sum_{j=1}^{p} P\left(a_j|x\right)} \tag{3.20}$$

O Algoritmo 15 ilustra como o dispositivo *k-NN* está internamente estruturado. O método $BD.nn\left(x,k\right)$ retorna os k vizinhos mais próximos de x, de acordo com alguma função de distância. A escolha do valor k deve ser feita através de algum critério que melhor se adeque ao problema a ser solucionado, conforme mostrado na seção 2.5.1.1.

De forma similar, a escolha da função de distância deve ser feita respeitando os requisitos de desempenho do sistema, escolhendo aquele com melhor custo-benefício - uma avaliação de cada função de distância deve ser realizada para determinar qual delas é a mais adequada para o problema a ser solucionado. Neste trabalho, utiliza-se a distância Euclidiana previamente definida pela equação (2.7), fazendo uso do *Value Difference Metric* para o cálculo de distância entre valores nominais, conforme definido na equação (2.11).

Igualmente semelhante, a escolha da função de ponderação $w\left(a,b\right)$ também deve ser feita por sistema. A função utilizada neste trabalho foi o método de Shepard descrita em (2.5), com $P = 2$.

Note que a base de conhecimento consultada para a busca dos vizinhos mais próximos, armazena apenas valores discretizados para os valores numéricos. Como o cálculo de distância depende dos valores numéricos verdadeiros, alguma aproximação deve ser feita para transformar o valor discretizado de volta em um valor numérico na escala verdadeira. Para o *EWD* (apresentado na seção 2.2.1), uma possibilidade é utilizar o valor central do intervalo que o valor discretizado representa. Assim, para um determinado valor dis-

creto i, seu valor aproximado na escala verdadeira pode ser calculado através da equação (3.21):

$$ap\left(i\right) = \frac{lim_{+}\left[i\right] + lim_{-}\left[i\right]}{2} \tag{3.21}$$

onde $lim_{+}\left[i\right]$ e $lim_{-}\left[i\right]$ são os limites superiores e inferiores do intervalo que o valor discreto i representa. Como exemplo, para um valor discreto que representa o intervalo $[5 - 25]$, seu valor aproximado na escala verdadeira será $5 + 25/30 = 15$. A seção 4.3.2 analisa os efeitos desta aproximação sobre o desempenho do sistema.

Enfim, deve-se notar que o *k-NN* não mantém abstrações próprias, dependendo exclusivamente das informações provenientes da base de conhecimento. Esta característica faz com que a função $aprende\left(x, a, dif_v, dif_{ic}\right)$ para o *k-NN* seja nula.

Algoritmo 15 $kNN.expectativa\left(x\right)$

Entrada
x: Vetor de características
Saída
Vetor μ com as expectativas de certeza

1: $P \leftarrow (0, 0, \ldots, 0)$
2: $\mu \leftarrow (0, 0, \ldots, 0)$
3: $Vizinhos \leftarrow BD.nn\left(x, k\right)$
4: $total \leftarrow 0$
5: **for** $j = 1 \rightarrow p$ **do**
6: **for** $i = 1 \rightarrow k$ **do**
7: $x_i \leftarrow Vizinhos\left[i\right]$
8: $v \leftarrow \hat{v}\left(x_i, a_j\right)$
9: $b \leftarrow 1 - 2 \times c\left(x_i, a_j\right)$
10: $w = peso\left(x, x_i\right)$
11: $P_j \leftarrow P_j + v \times w \times b$
12: **end for**
13: $total \leftarrow total + P_j$
14: **end for**
15: **for** $j = 1 \rightarrow p$ **do**
16: $\mu_j \leftarrow P_j / total$
17: **end for**

4 RESULTADOS

Este capítulo apresenta os experimentos realizados para validar a proposta apresentada neste trabalho, analisa seus resultados e os compara com os obtidos aplicando-se outras técnicas de tomada de decisão.

4.1 Sistema Experimental

O sistema experimental utilizado para validar a proposta é composto por dois dos dispositivos descritos anteriormente, na seguinte ordem:

1. O ponto de entrada da fila de dispositivos é o classificador de Bayes. Para critérios numéricos, ao invés de utilizar funções específicas por distribuição (e.g., função (2.13) para valores que seguem uma distribuição normal), seus valores são discretizados e utilizados como categorias de um critério nominal, conforme descrito na seção 2.2.1.

2. Caso a solução do classificador Bayes não satisfaça a condição de saída da fila, o método *k-NN* é utilizado em seu lugar para buscar uma solução. Para simplificação, utiliza-se um valor fixo $k = 10$ para o número de vizinhos a ser considerado.

3. A métrica para critérios nominais é o *Value Difference Metric*, enquanto a função de ponderação escolhida é o método de Shepard, conforme está descrito na seção 3.5.3.

Nota: ainda que seja preferível utilizar métodos mais robustos para a escolha de um valor ótimo para k em cada dataset a ser utilizado para validação (KOHAVI, 1995; OUYANG; LI; LI, 2006), um valor fixo permite analisar o problema sem inserir variáveis adicionais que possam interferir nos resultados.

Este sistema também apresenta a seguinte configuração inicial:

- O valor de reforço relativo entre realimentações positivas e negativas é o mesmo, i.e., $\beta_+ = \beta_- = 1/2$.
- O método de discretização utilizado é o *Equal Width Discretization* (*EWD*), dividindo o espaço de valores numéricos em 10 intervalos de mesma largura.
- A condição de saída da fila é dada pela função de limiar definida em (3.1). O valor do limiar, contudo, é definido por experimento.
- Caso nenhum dispositivo consiga uma solução que satisfaça a condição de saída da fila, a solução obtida com o uso do último dispositivo será utilizada, conforme descrito no Algoritmo 7.
- Partindo do pressuposto que as instâncias rotuladas destes *datasets* são isentas de erros, a credibilidade do treinamento inicial é $cr_0 = \infty$.

Caso algum teste faça uso de valores diferentes destes citados acima, tais diferenças serão explicitamente descritas no texto, juntamente com a apresentação do experimento. Outras propriedades - tais como a quantidade de instâncias rotuladas utilizadas para treinamento, o valor do limiar de saída da fila e o uso das realimentações - são definidas particularmente para cada experimento.

Os testes foram realizados utilizando *datasets* fornecidos pelo repositório de aprendizado de máquina da UCI (FRANK; ASUNCION, 2011), frequente-

mente utilizado como base de comparações entre algoritmos de classificação, regressão e aprendizado de máquina.

4.2 Metodologia

Para a realização de cada experimento, foi adotado um método de Monte-Carlo (CHEN, 1995) para obter os parâmetros de desempenho do sistema, conforme descrito abaixo:

1. Determinar os parâmetros de saída a serem medidos - neste caso, taxa de acerto e tempo de resposta;
2. Determinar a configuração inicial do sistema;
3. Determinar a quantidade de instâncias a ser utilizada para treinamento;
4. Para cada *dataset* a ser testado:
   (a) Treinar o sistema um subconjunto das instâncias rotuladas (conjunto de treinamento), escolhidas de forma aleatória;
   (b) Validar o sistema com o restante das instâncias (conjunto de testes):
      i. Para cada instância rotulada $(x, a_{instancia})$ do conjunto de testes:
         A. Executar o sistema com x;
         B. Verificar se a alternativa retornada $a_{retorno}$ é igual à alternativa esperada $a_{instancia}$;
      ii. Totalizar o número de acerto;
      iii. Totalizar o tempo de execução;
   (c) Iterar item 4b m vezes para o mesmo *dataset*;

(d) Extrair a média da taxa de acerto. Para isso, dividir o total de acerto pelo total de instâncias testadas;

(e) Extrair a média do tempo de resposta. De forma análoga, dividir o tempo total gasto pelo total de instâncias testadas;

(f) Analisar os resultados obtidos neste *dataset*;

5. Analisar os resultados globais do experimento.

O treinamento para um *dataset* em particular é feito alimentando o sistema com um subconjunto das instâncias sorteadas aleatoriamente, compondo seu conhecimento inicial. Para isso, utiliza-se o algoritmo 9, acionando-o múltiplas vezes - uma vez para cada instância $(x, a_{instancia})$ sorteada - com valor de credibilidade cr_0. A porcentagem de instâncias utilizadas para treinamento deve ser definida previamente. Um valor habitualmente utilizado nos estudos envolvendo aprendizado de máquina é 70% (esta partição também é conhecida como 70/30).

Os 30% restantes das instâncias devem ser utilizados para medir a taxa de acerto e o tempo de resposta do sistema. Para cada instância, seu conjunto x é enviado ao algoritmo 8, na esperança que sua saída seja igual à alternativa correta $a_{instancia}$. A quantidade de acertos dividida pelo total destas instâncias resulta na taxa de acerto, enquanto o tempo de resposta é a média do tempo gasto por instância. Tal medição permite analisar como o desempenho do sistema é afetado em função dos parâmetros de entrada - ou ainda, verificar como estes parâmetros de saída se relacionam, levantando uma *curva de desempenho*, que mostra a taxa de acerto em função do tempo de resposta - similar à curva *SAT* citada na seção 1.2.

O número de iterações m deve ser escolhido de forma a minimizar o erro padrão das estimativas da média da taxa de acerto e do tempo de resposta.

Este erro padrão é dado por $\epsilon = \frac{\sigma}{\sqrt{m}}$, onde σ é o desvio padrão da média do parâmetro estimado (HOLTON, 2004). Logo, para reduzir o erro padrão para uma fração f do desvio padrão, deve-se escolher $m = (1/f)^2$. Como exemplo, adotando um valor arbitrário $f = 5\%$, deve-se utilizar $m = 400$ - este é o valor adotado neste trabalho.

Note que o sorteio destas instâncias é refeito em cada iteração do *data-set*, para que a medição do desempenho seja feita com base em subconjuntos de treinamento distintos, tornando seu resultado estatisticamente mais significante - uma propriedade importante do método de Monte-Carlo.

4.3 Análises Preliminares

Esta seção apresenta uma série de experimentos realizados sobre cada dispositivo de forma individual, permitindo obter resultados a serem utilizados como base de comparação para o sistema proposto neste trabalho. Em particular, é realizada uma comparação entre os desempenhos do classificador de Bayes e do *k-NN*, além de uma análise individual do desempenho da *TDAE*.

4.3.1 Comparativo de Desempenho entre o Classificador de Bayes e o *k-NN*

Como base de comparação, o *k-NN* e o classificador de Bayes foram testados individualmente com uma série de *datasets* a fim de obter suas respectivas taxas de acerto e tempos de resposta. Esta análise preliminar permite verificar se é verdadeira a premissa de que o dispositivo mais lento (*k-NN*) possui, na maior parte dos casos, melhor taxa de acerto. Para este fim, dois sistemas experimentais foram construídos conforme as especificações apresentadas na seção 4.1. Em cada um deles, contudo, é utilizado apenas um

dispositivo (*k-NN* em um caso, classificador de Bayes no outro), para que seja possível analisar seu comportamento individual.

Em cada *dataset*, o conjunto de treinamento é composto por 70% das instâncias rotuladas (sorteadas de forma aleatória), enquanto os demais 30% são utilizados para validação, permitindo estimar a taxa de acerto e o tempo de resposta. Este processo é iterado 400 vezes, conforme descrito na seção 4.2.

Como o sistema é composto por apenas um dispositivo, não é definido um valor para o limiar para a condição de saída da fila (i.e., o critério de aceitação). Além disso, para evitar que o aprendizado incremental interfira nos resultados deste experimento, nenhum bloco de auto-análise será utilizado (ver Algoritmo 8). De maneira complementar, é utilizado $cr_{repetition} = 0$ para remover a influência do viés de repetição sobre o aprendizado.

A Tabela 7 mostra as características de cada *dataset*, informando a quantidade de instâncias presentes em cada um deles, além da quantidade de critérios (tanto nominais como numéricos) e de alternativas.

Em comparação, a Tabela 8 mostra as medidas obtidas pelos dispositivos em cada um dos *datasets*, enquanto o gráfico da Figura 22 compara as taxas de acerto entre os dispositivos.

Percebe-se que, na maior parte dos *datasets*, a premissa mostra-se verdadeira: apesar de mais lento, o *k-NN* obteve taxas de acerto superiores em 17 dos 20 *datasets* analisados. Também pode ser notado que *datasets* que fazem uso de critérios nominais são proporcionalmente muito mais lentos no *k-NN*. Este comportamento já era esperado, pois o cálculo de distância para critérios nominais através do *Value Difference Metric* possui complexidade superior quando comparada com o cálculo de distância para critérios numéricos.

	Instâncias	Critérios		Alternativas
		Numéricos	Nominais	
Blood Transfusion	748	4	0	2
Car Evaluation	1728	0	6	4
Cardiotocography Class	2126	21	0	10
Cardiotocography NSP	2126	21	0	3
Contraceptive Method	1473	2	7	3
Credit Aproval	690	6	9	2
Ecoli	336	7	0	8
Glass Identification	214	9	0	7
Hayes-Roth	132	4	0	3
ILPD	583	9	1	2
Iris	150	4	0	3
Nursery	12960	0	8	5
Page Blocks	5473	10	0	5
Pima Indians Diabetes	768	8	0	2
Shuttle	58000	9	0	7
Statlog (German Credit)	1000	7	13	2
Teaching Assistant	151	1	4	3
Tic-Tac-Toe	958	0	9	2
Vertebral Column	310	6	0	3
Yeast	958	8	0	10

Tabela 7: Descrição dos *datasets* testados.

Para verificar a significância deste resultado, é realizado um teste de hipótese que verifica se a taxa de acerto do *k-NN* é, de fato, maior nestes *datasets*. Para isso, é armazenada, em cada uma das 400 iterações, a diferença entre as taxas de acerto do *k-NN* e do classificador de Bayes. Em seguida, são estipuladas as seguintes hipóteses:

- H_0 (hipótese nula): Média das diferenças é menor ou igual a zero
- H_1 (hipótese alternativa): Média das diferenças é maior que zero

A partir desta amostra com as 400 diferenças, são calculados a média e o erro padrão correspondentes. Em seguida, o valor-p é calculado e comparado ao nível de significância α escolhido (neste caso, $\alpha = 5\%$). Se o valor-p é menor, H_1 é aceita, caso contrário, a hipótese H_0 não pode ser rejeitada. O Anexo B mostra como este cálculo é realizado.

	Bayes		*k-NN*	
	Acertos	**Tempo (ms)**	**Acertos**	**Tempo (ms)**
Blood Transfusion	74,9684%	0,0066	75,7893%	0,0526
Car Evaluation	83,5116%	0,0201	95,3998%	1,2393
Cardiotocography Class	66,3127%	0,1068	76,2955%	3,8177
Cardiotocography NSP	80,8871%	0,0677	91,0925%	3,8033
Contraceptive Method	49,8931%	0,0313	50,7534%	3,8033
Credit Aproval	83,9087%	0,0140	84,6702%	0,8141
Ecoli	79,9317%	0,0274	85,7436%	0,1571
Glass Identification	54,0123%	0,0205	66,8369%	0,1156
Hayes-Roth	77,8050%	0,0063	71,9313%	0,0340
ILPD	63,9063%	0,0162	67,4274%	0,3248
Iris	92,2522%	0,0069	95,6152%	0,0430
Nursery	90,4538%	0,0106	98,9265%	15,6695
Page Blocks	91,0566%	0,0182	92,1772%	0,3705
Pima Indians Diabetes	75,1316%	0,0235	73,2476%	0,3766
Shuttle	87,8446%	0,0209	97,5672%	0,1183
Statlog (German Credit)	74,8601%	0,0213	71,0631%	1,9723
Teaching Assistant	51,8679%	0,0058	53,0337%	0,0597
Tic-Tac-Toe	70,2066%	0,0117	90,0660%	0,8602
Vertebral Column	70,7170%	0,0125	75,4085%	0,1184
Yeast	57,0370%	0,0534	57,0426%	0,5928

Tabela 8: Médias da taxa de acerto e tempo de resposta de cada dispositivo nos *datasets* testados.

A Tabela 9 apresenta os resultados do teste de hipótese para cada *dataset*. Percebe-se que a maioria dos testes favorece amplamente a ideia de que a diferença entre as taxas de acerto é maior que zero, pois o valor-p é inferior ao α escolhido. Nos *datasets Hayes-Roth*, *Pima Indians Diabetes* e *Statlog*, por outra lado, a diferença é negativa, o que faz com que não seja possível rejeitar a hipótese nula. Por fim, apesar de a diferença ser positiva no *Yeast*, ela é mínima, e também não permite concluir que a média é maior que zero.

Nos quatro *datasets* em que não é possível concluir que a diferença é maior que zero, realiza-se um segundo teste que verifica se a diferença é menor que zero. Logo:

- H_0 (hipótese nula): Média das diferenças é maior ou igual a zero

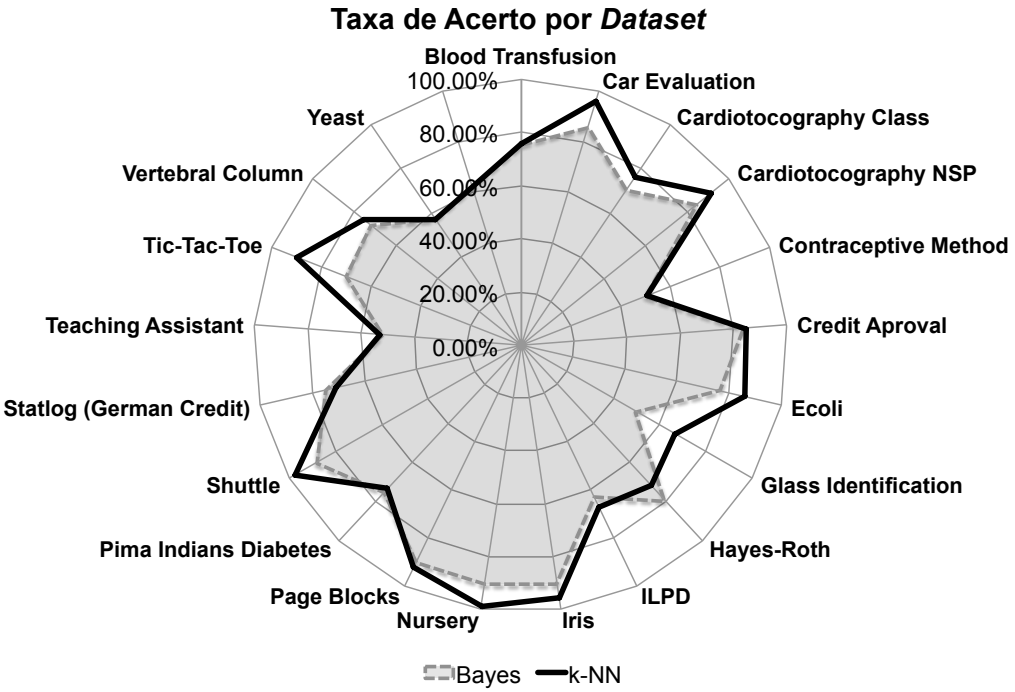


Figura 22: Média da taxa de acerto de cada dispositivo nos *datasets* utilizados para testes.

- H_1 (hipótese alternativa): Média das diferenças é menor que zero

A Tabela 10 mostra que os três *datasets* com diferença negativa, de fato, favorecem a hipótese de que a diferença é menor que zero. Por outro lado, também mostra que *Yeast* é o único *dataset* em que não é possível afirmar se a diferença era maior ou menor que zero - note que este resultado não implica necessariamente que a média é zero.

A partir destes resultados, pode-se afirmar, com alta significância, que o método *k-NN* é mais preciso do que o classificador de Bayes na maior parte dos *datasets* testados - mais especificamente, 16 dentre os 20. Tal resultado valida a premissa inicial necessária para realizar os experimentos a seguir.

	Média	E. Padrão	valor-p	Rejeita H_0
Blood Transfusion	0,0082	0,00388	$1{,}85 \times 10^{-02}$	Sim
Car Evaluation	0,1198	0,00233	$1{,}76 \times 10^{-73}$	Sim
Cardiotocography Class	0,0998	0,00252	$8{,}27 \times 10^{-63}$	Sim
Cardiotocography NSP	0,1021	0,00218	$9{,}96 \times 10^{-70}$	Sim
Contraceptive Method	0,0086	0,00296	$2{,}25 \times 10^{-03}$	Sim
Credit Aproval	0,0076	0,00310	$7{,}89 \times 10^{-03}$	Sim
Ecoli	0,0581	0,00471	$4{,}79 \times 10^{-22}$	Sim
Glass Identification	0,1282	0,00770	$9{,}22 \times 10^{-31}$	Sim
K*Hayes-Roth*	-0,0587	0,01018	$\approxeq 1$	Não
ILPD	0,0352	0,00457	$5{,}31 \times 10^{-12}$	Sim
Iris	0,0336	0,00461	$3{,}51 \times 10^{-11}$	Sim
Nursery	0,0819	0,00050	$8{,}94 \times 10^{-123}$	Sim
Page Blocks	0,0112	0,00118	$6{,}00 \times 10^{-16}$	Sim
Pima Indians Diabetes	-0,0188	0,00353	$\approxeq 1$	Não
Shuttle	0,0972	0,00303	$2{,}08 \times 10^{-54}$	Sim
Statlog (German Credit)	-0,0380	0,00310	$\approxeq 1$	Não
Teaching Assistant	0,0117	0,00292	$3{,}56 \times 10^{-05}$	Sim
Tic-Tac-Toe	0,1986	0,00324	$6{,}95 \times 10^{-81}$	Sim
Vertebral Column	0,0469	0,00591	$1{,}58 \times 10^{-12}$	Sim
Yeast	$5{,}6 \times 10^{-5}$	0,00249	$4{,}91 \times 10^{-01}$	Não

Tabela 9: Resultados do teste de hipótese da diferença entre as taxas de acerto do classificador Bayes e do *k-NN* ser maior que zero.

	Média	E. Padrão	valor-p	Rejeita H_0
Hayes-Roth	-0,0587	0,01018	$4{,}55 \times 10^{-08}$	Sim
Pima Indians Diabetes	-0,0188	0,00353	$2{,}93 \times 10^{-07}$	Sim
Statlog (German Credit)	-0,0380	0,00310	$6{,}95 \times 10^{-22}$	Sim
Yeast	$5{,}6 \times 10^{-5}$	0,00249	$5{,}09 \times 10^{-01}$	Não

Tabela 10: Resultados do teste de hipótese da diferença entre as taxas de acerto do classificador Bayes e do *k-NN* ser menor que zero.

4.3.2 Efeitos da Discretização

Dentre as características apresentadas na seção 4.1, o método de discretização é aquela que tem maior influência sobre todos os dispositivos utilizados. Enquanto algumas características são inerentes apenas ao sistema em si (e.g., a escolha do limiar), ou então, são particulares de um dispositivo individual (e.g., a escolha do valor k para o *k-NN*), a discretização afeta tanto o classificador de Bayes como o *k-NN*, alterando seus desempenhos individuais,

e, consequentemente, o desempenho global do sistema.

Para verificar a extensão desta influência, foi realizado um experimento idêntico ao apresentado na seção 4.3.1, contudo, utilizando 100 intervalos para o *EWD* no lugar de 10. Em seguida, as taxas de acerto e os tempos de resposta em cada dispositivo são comparados com os resultados anteriores. No caso particular do *k-NN*, também foi feito um teste sem discretização, no qual as instâncias numéricas são armazenadas na base com seu valor real.

A Figura 23 mostra como a taxa de acerto do classificador de Bayes é afetada pela escolha do número de intervalos. Na maior parte dos *datasets*, um acréscimo no número de intervalos acarretou uma perda na taxa de acerto (e.g., perda de 36,27% no *Ecoli*), uma vez que isso leva a um aumento no número de possíveis combinações de valores discretos. Como muitas destas combinações acabam não recebendo instâncias de treinamento, o poder de decisão nestes espaços vazios acaba sendo prejudicado. A Figura 24 ilustra um exemplo de como estes espaços vazios acabam surgindo.

Uma exceção, contudo, é a taxa de acerto do *Shuttle*, em que houve um ganho de 7,46%. Note que este é o *dataset* com maior número de instâncias, fazendo com que o número maior de intervalos seja favorável para melhor discernir valores numéricos.

Já a taxa de acerto do método *k-NN* mostra-se muito menos sensível ao número de intervalos: ela se mantém estável, muito próximo da taxa obtida pelo *k-NN* original (i.e., sem discretização), conforme mostra a Figura 25. Comparando a taxa entre do *k-NN* com discretização em 10 intervalos em relação ao original, a diferença é superior a 2% em apenas 4 *datasets*: *Blood Transfusion* (-2,66%), *Page Blocks* (3,64%), *Shuttle* (2,33%) e *Teaching Assistant* (4,24%).

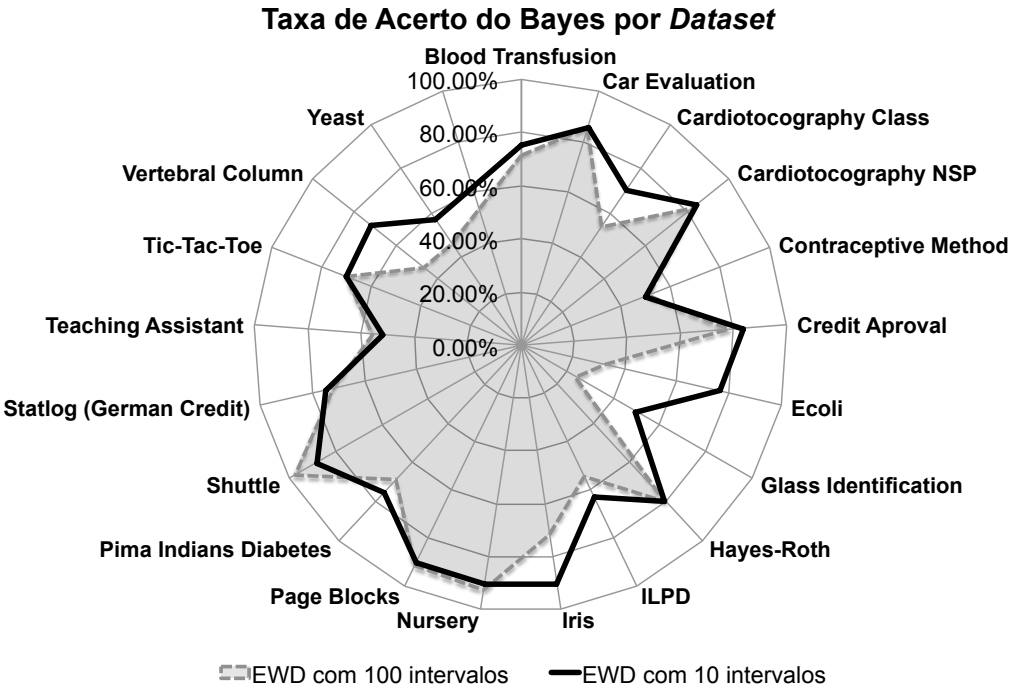


Figura 23: Média da taxa de acerto do classificador de Bayes de acordo com a discretização utilizada.

Esta estabilidade, contudo, se inverte quando o tempo de resposta é utilizado como parâmetro de comparação: um aumento no número de intervalos faz com que a quantidade-limite de casos armazenados pela base de conhecimento aumente, potencialmente aumentando a quantidade de casos armazenados para um mesmo conjunto de treinamento, e, consequentemente, tornando o *k-NN* mais lento. No exemplo da Figura 24, a primeira discretização gera, no máximo, 9 combinações, enquanto a segunda discretização pode gerar até 36 para o mesmo conjunto de treinamento - note que apenas 25 combinações realmente estão presentes na base de conhecimento, mas ainda assim, deve apresentar maior lentidão com relação ao exemplo com 9 combinações.

A Figura 26 mostra a razão entre o tempo de resposta obtido pelos dispositivos com 100 intervalos e o tempo de resposta obtido pelos mesmos dispositivos, mas com 10 intervalos. Note que, no classificador de Bayes, esta

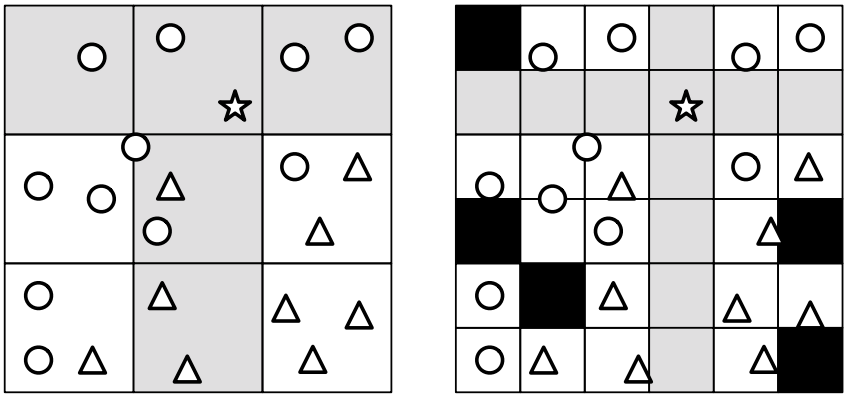


Figura 24: Exemplo bidimensional de discretização, com duas alternativas: círculo e triângulo. Para 3 intervalos discretos, a instância marcada com estrela pode ser facilmente classificada, enquanto para 6 intervalos, é impossível solucioná-la, pois não há nenhuma instância em nenhum dos dois eixos. Outros espaços vazios estão pintados de preto.

razão se mantém próxima de 1, enquanto no *k-NN*, esta razão sempre foi maior que 1, i.e., o *k-NN* com 100 intervalos nunca foi mais rápido que o *k-NN* com 10 intervalos. Esta diferença é mais evidente nos *datasets* numéricos com uma quantidade maior de instâncias: *Page Blocks* (razão de 15,887) e *Shuttle* (razão de 271,967).

Estes resultados evidenciam a importância da escolha de um número adequado de intervalos para a discretização, uma vez que tanto o tempo de resposta como a taxa de acerto podem ser significativamente prejudicados por uma escolha inadequada.

4.3.3 Taxa de Acerto da *TDAE*

Diferentemente do classificador de Bayes e do *k-NN*, uma tabela de decisão é capaz de solucionar apenas casos já previstos em seu conjunto de regras, limitando seus acertos às instâncias previamente treinadas. Como os conjuntos de treinamento e de testes gerados pelo processo definido na

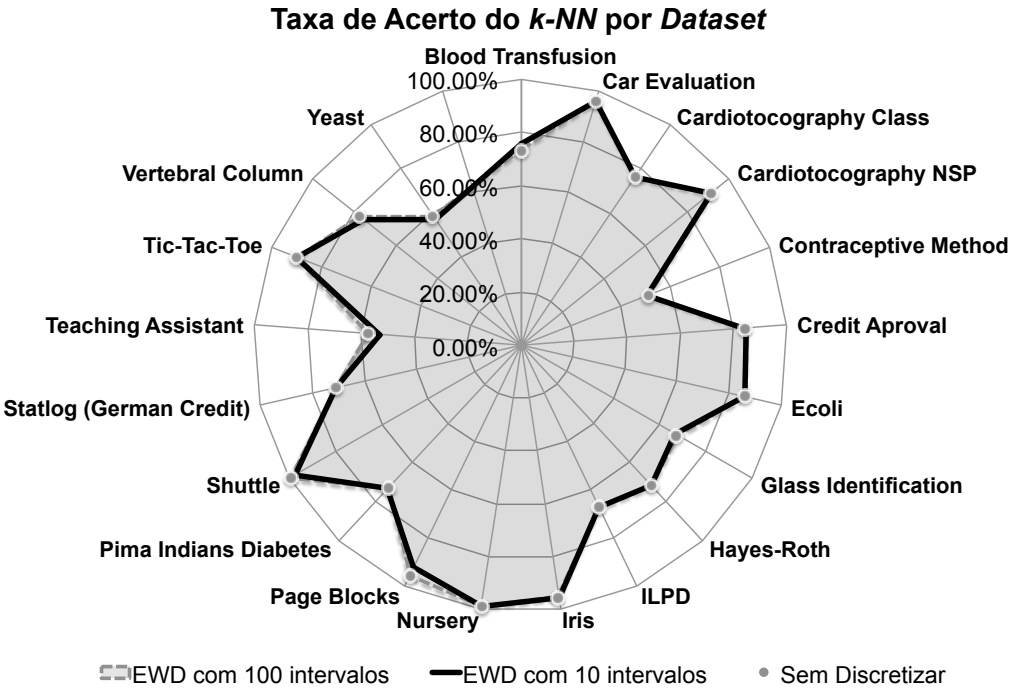


Figura 25: Média da taxa de acerto do *k-NN* de acordo com a discretização utilizada.

seção 4.2 são disjuntos (i.e., sem elementos em comum, partindo do pressuposto de que os *datasets* não contêm instâncias repetidas), a medição da taxa de acerto da *TDAE* (sem a ajuda de um método auxiliar) não aparenta ser justificável - de fato, para um *dataset* puramente nominal, a taxa de acerto é zero.

Contudo, para problemas que utilizem critérios numéricos, os valores armazenados nas regras não representam categorias, mas sim, valores discretizados que, por sua vez, representam intervalos de valores reais. Logo, uma regra, mesmo que completa (i.e., com todos os seus valores preenchidos), pode abranger uma quantidade teoricamente infinita de instâncias - lembrar que existem infinitos números reais dentro de qualquer intervalo real com comprimento maior que zero. Como exemplo, a Tabela 11 representa a discretização da Tabela 2 apresentada anteriormente. Nela, as instâncias $x_1 = (5, 80\%)$ e $x_2 = (6, 100\%)$ são discretizadas para um mesmo valor, resultando em $(1, 1)$

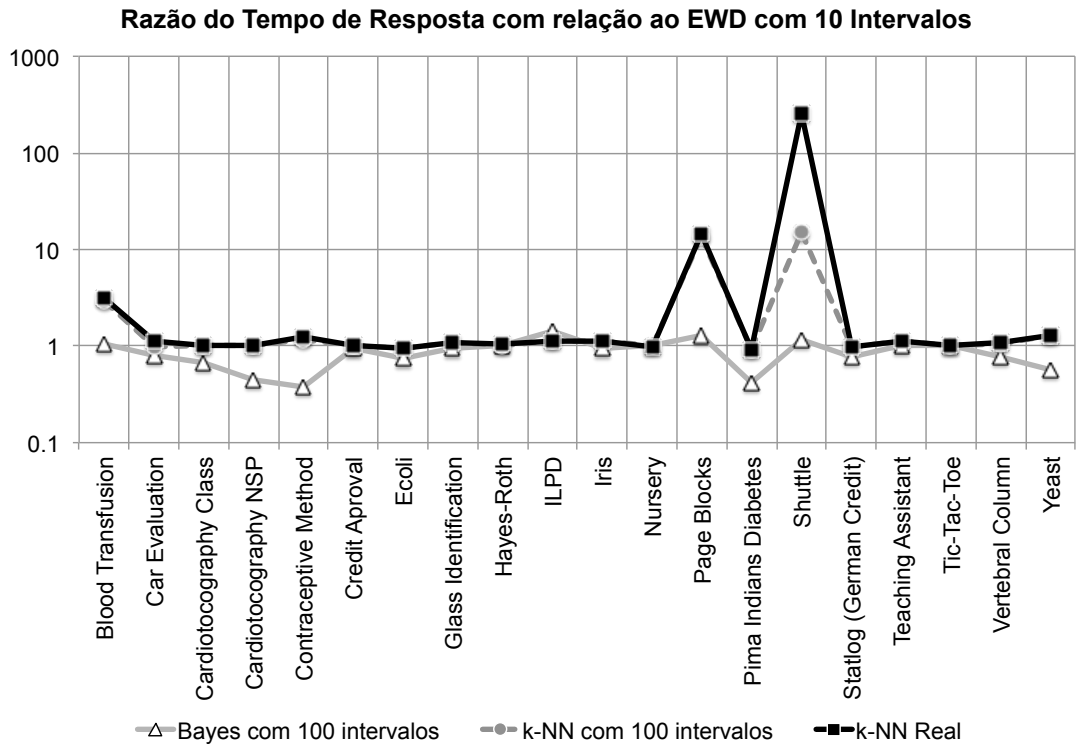


Figura 26: Razão entre o tempo de resposta dos dispositivos com 100 intervalos e o tempo do mesmo dispositivo, mas com 10 intervalos.

e caem em uma mesma regra r_2, ainda que seus valores numéricos originais sejam distintos.

	r_1	r_2	r_3	r_4
Média	2 (≥ 7)	1 ($[5-7[$)	0 (< 5)	
Frequência	1 ($\geq 70\%$)	1 ($\geq 70\%$)		0 ($< 70\%$)
Aprovado	S			
Recuperação		S		
Reprovado			S	S

Tabela 11: Exemplo de tabela de decisão com valores discretos. Entre parênteses, o intervalo real que cada valor discreto representa.

Dessa forma, para problemas com critérios numéricos, é possível obter uma taxa de acerto maior que zero, dependendo da distância entre as instâncias treinadas e do método de discretização utilizado. A Tabela 12 mostra a taxa de acerto e tempo de resposta da TDAE para os mesmos *datasets*

testados anteriormente, utilizando o *EWD* com 10 intervalos conforme explicado na seção 4.1. De forma geral, aqueles que possuem critérios numéricos conseguiram uma taxa de acerto maior que zero (destaque para o *Shuttle*, cuja taxa de acerto é de quase 95%, impulsionada pela grande quantidade de instâncias que preenchem todo o espaço de valores possíveis), enquanto problemas estritamente nominais tiveram taxa de acerto zero, conforme esperado. Já o tempo de resposta varia de acordo com a quantidade de instâncias, mantendo-se abaixo do tempo obtido pelo *k-NN*, contudo, acima do classificador de Bayes.

	TDAE		**Critérios**	
	Acertos	**Tempo (ms)**	**Numéricos**	**Nominais**
Blood Transfusion	68,1813%	0,0118	4	0
Car Evaluation	0,0000%	0,2497	0	6
Cardiotocography Class	6,6873%	0,7710	21	0
Cardiotocography NSP	7,1207%	0,7672	21	0
Contraceptive Method	12,7223%	0,1517	2	7
Credit Aproval	2,4952%	0,0905	6	9
Ecoli	10,8059%	0,0559	7	0
Glass Identification	9,5300%	0,0273	9	0
Hayes-Roth	50,2625%	0,0109	4	0
ILPD	17,4354%	0,0736	9	1
Iris	36,9761%	0,0135	4	0
Nursery	0,0000%	5,0152	0	8
Page Blocks	82,5776%	0,0263	10	0
Pima Indians Diabetes	2,7281%	0,1064	8	0
Shuttle	94,1146%	0,0256	9	0
Statlog (German Credit)	0,2814%	0,2793	7	13
Teaching Assistant	35,7821%	0,0122	1	4
Tic-Tac-Toe	0,0000%	0,1792	0	9
Vertebral Column	14,0638%	0,0352	6	0
Yeast	21,4955%	0,1818	8	0

Tabela 12: Taxa de acerto e tempo de resposta da *TDAE* nos *datasets* utilizados para testes, sem a ajuda de um método auxiliar.

Nota: o dataset Teaching Assistant possui instâncias repetidas, o que causa um aumento na taxa de acerto, pois caso uma destas instâncias seja utilizada para treinamento, a outra instância, de testes, deve ser corretamente

solucionada, a menos que outras instâncias de treinamento caiam no mesmo valor discreto e alterem a alternativa da regra.

4.4 Compromisso entre Taxa de Acerto e Tempo de Resposta

O primeiro experimento efetivamente realizado neste trabalho foi verificar a possibilidade de controlar a relação de compromisso entre taxa de acerto e tempo de resposta do sistema proposto, e, em caso positivo, verificar como estes parâmetros de saída se relacionam.

O tempo de resposta do sistema está diretamente relacionado à frequência com que os dispositivos mais lentos são acessados: quanto mais frequentemente forem utilizados, maior deve ser o tempo médio de resposta. Logo, a *estriticidade* imposta pela condição de saída da fila de dispositivos deve reger quanto tempo, em média, deve ser gasto em cada instância de entrada. O termo estriticidade, neste contexto, indica o quão provável uma solução é rejeitada - quanto maior o número de soluções rejeitadas, maior será a frequência de uso dos outros dispositivos.

De maneira similar, a taxa de acerto também depende da frequência com que cada dispositivo é acessado: o número de acertos deve aumentar ao repassar uma quantidade maior de problemas aos dispositivos mais precisos. Caso seja verdadeira a afirmação de que dispositivos mais lentos são mais precisos, condições de saída mais restritivas fazem com que tanto a taxa de acerto como o tempo de resposta aumentem em decorrência do uso mais frequente destes dispositivos.

Partindo da função de aceitação 3.1, a estriticidade da condição de saída da fila está diretamente ligada ao valor do limiar escolhido: valores maiores

diminuem a probabilidade de uma solução qualquer ser aceita, aumentando a frequência de acesso aos dispositivos mais lentos. Dessa forma, o que este experimento efetivamente avalia é o efeito da escolha deste limiar sobre a taxa de acerto e o tempo de resposta, utilizando os *datasets* citados previamente.

De maneira similar ao experimento preliminar, não é utilizada nenhuma forma de realimentação durante a tomada de decisão, para evitar que ela interfira nos resultados.

4.4.1 Sistema Experimental

Em um primeiro teste, verifica-se a relação entre taxa de acerto e tempo de resposta do sistema experimental proposto, composto pelo classificador de Bayes e pelo *k-NN*. Pela análise preliminar da seção 4.3.1, percebe-se que o *k-NN* é não só o dispositivo mais lento, mas também aquele de maior taxa de acerto na maior parte dos *datasets* avaliados, satisfazendo, assim, as premissas necessárias para a execução deste experimento.

Para cada *dataset*, o processo definido na seção 4.2 é repetido com diferentes valores para o limiar - entre $0,00$ e $1,00$, em passos de $0,02$. Em cada limiar, o sistema experimental é testado 400 vezes, sorteando aleatoriamente 70% das instâncias para treinamento e utilizando os 30% remanescentes para medir a taxa de acerto e o tempo de resposta neste limiar. A partir destes resultados, são gerados dois gráficos por *dataset*:

- Primeiramente, a taxa de acerto e o tempo de resposta são postos em função do limiar. Uma análise deste gráfico permite escolher o limiar que melhor se adeque às necessidades do problema - e.g., maximizar a taxa de acerto, utilizando, em média, 1 ms por instância.
- Também é possível criar um gráfico da taxa de acerto em função do

tempo de resposta. Esta curva ilustra o desempenho do sistema experimental, possibilitando compará-lo com o desempenho de outros sistemas, independentemente das técnicas que utilizem.

A Figura 27 apresenta as curvas de desempenho obtidas em cada um dos *datasets* testados previamente. No limiar zero, percebe-se que a taxa de acerto e o tempo de resposta são iguais aos apresentados pelo classificador de Bayes - um comportamento já esperado, visto que, neste limiar, a condição imposta pela função (3.1) sempre será satisfeita, logo, apenas o classificador de Bayes é efetivamente utilizado.

Analogamente, com o limiar em seu valor máximo $(1,00)$, o tempo de resposta apresentado pelo sistema é igual à soma dos tempos de resposta apresentados pelo classificador de Bayes e *k-NN*, pois a condição de saída da fila nunca será satisfeita, levando o sistema a utilizar todos os dispositivos para toda instância de entrada. Por não conseguir satisfazer esta condição, o sistema sempre opta pela solução apresentada pelo Algoritmo 6, que, por sua vez, opta pela solução do último dispositivo, neste caso, do *k-NN*. Logo, neste limiar, a taxa de acerto é aproximadamente igual àquela apresentada pelo *k-NN*.

Também se observa que nos limiares próximos de zero, não há variações significativas nas taxas de acerto e nos tempos de resposta. Estas variações só ocorrem em limiares superiores a $1/p$, onde p é o número de alternativas. Pela equação (2.1), nota-se que este é o menor valor que pode ser assumido pela expectativa de certeza máxima (a menos que todos os elementos do vetor tenham valor zero), logo, para limiares menores que $1/p$, as soluções do classificador de Bayes sempre serão aceitas, i.e., o *k-NN* nunca será consultado nestes limiares.

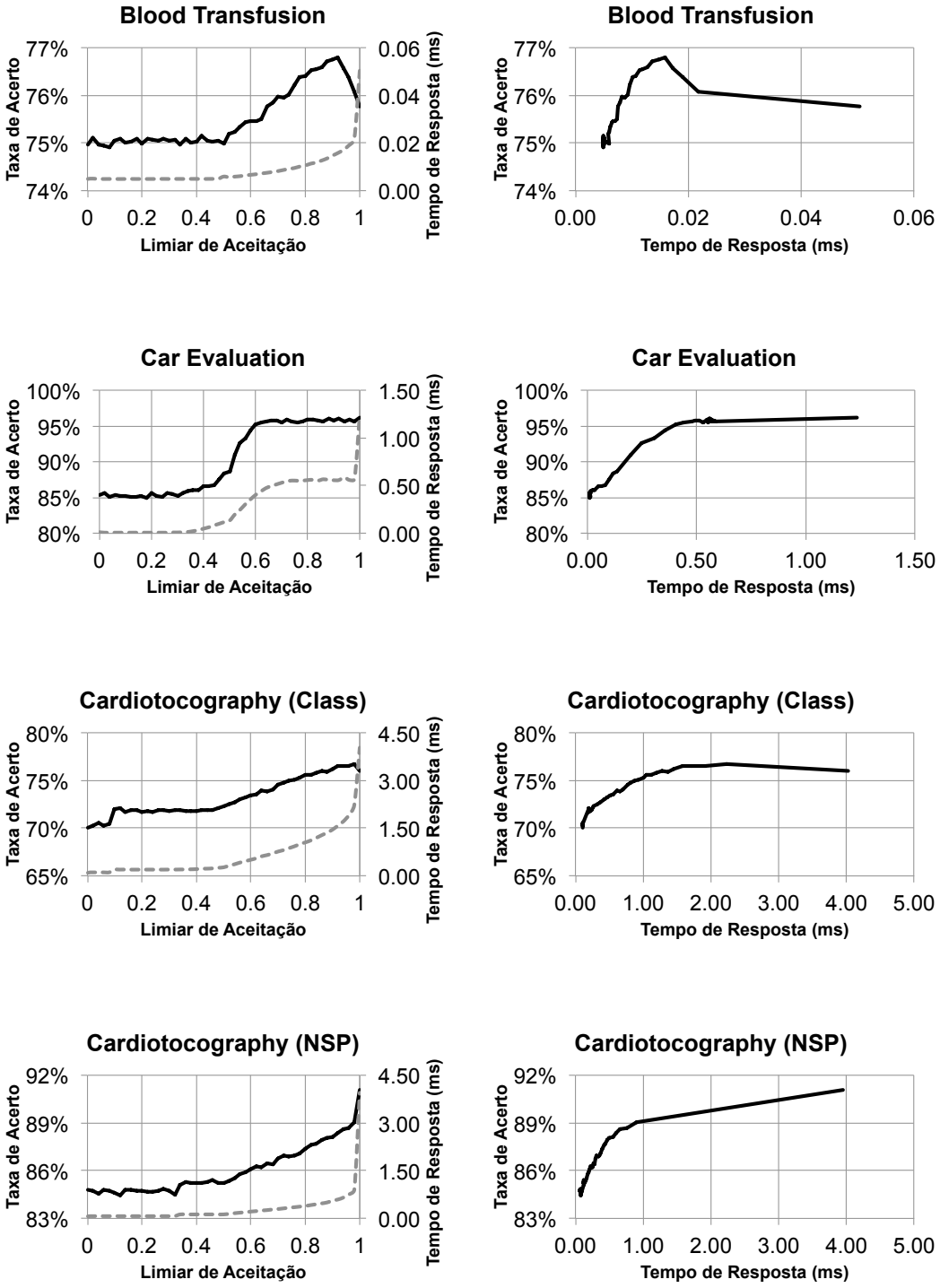


Figura 27: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema experimental em função do limiar. À direita, a curva de desempenho (taxa de acerto em função do tempo de resposta). (cont.)

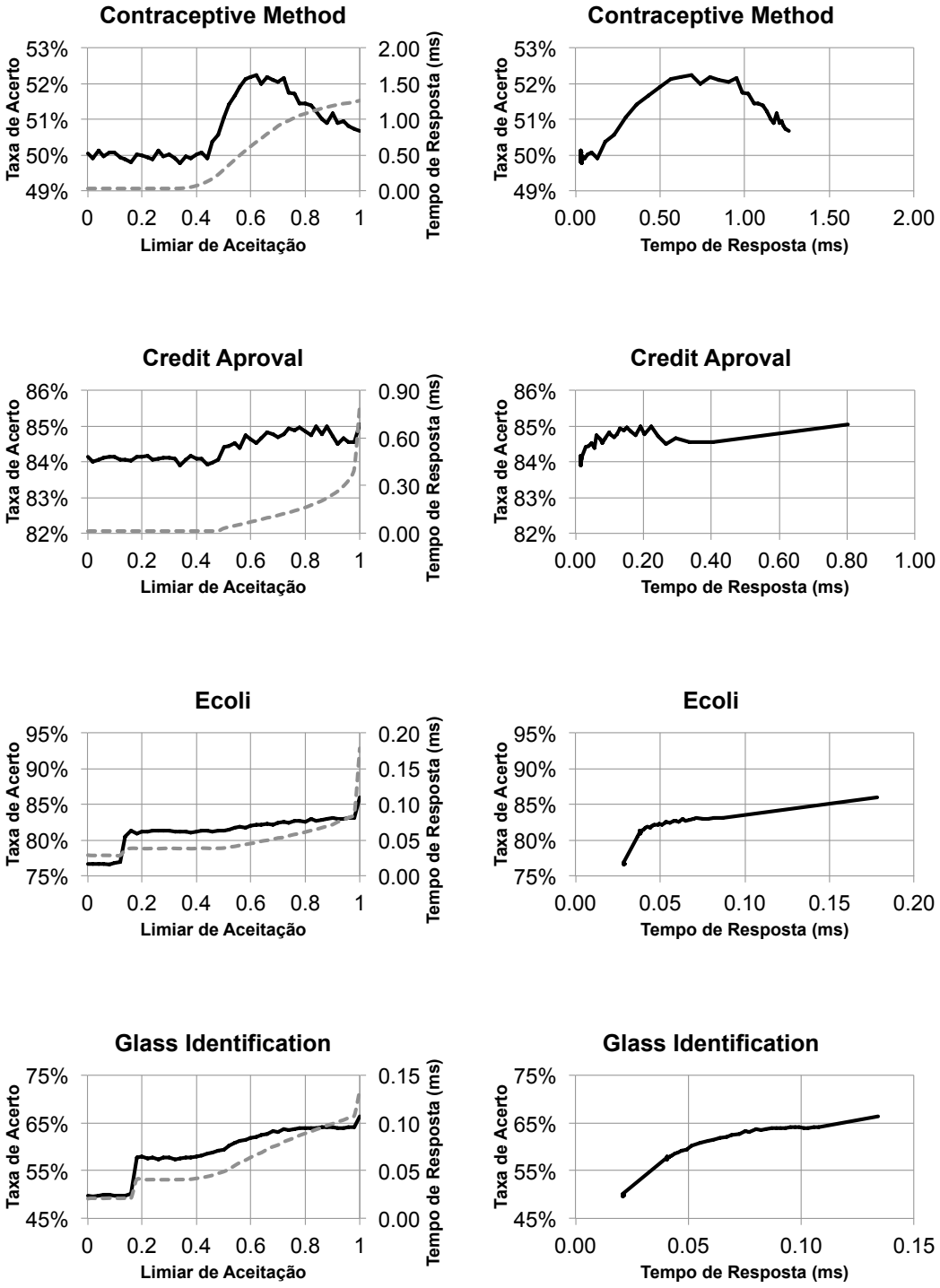


Figura 27: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema experimental em função do limiar. À direita, a curva de desempenho (taxa de acerto em função do tempo de resposta). (cont.)

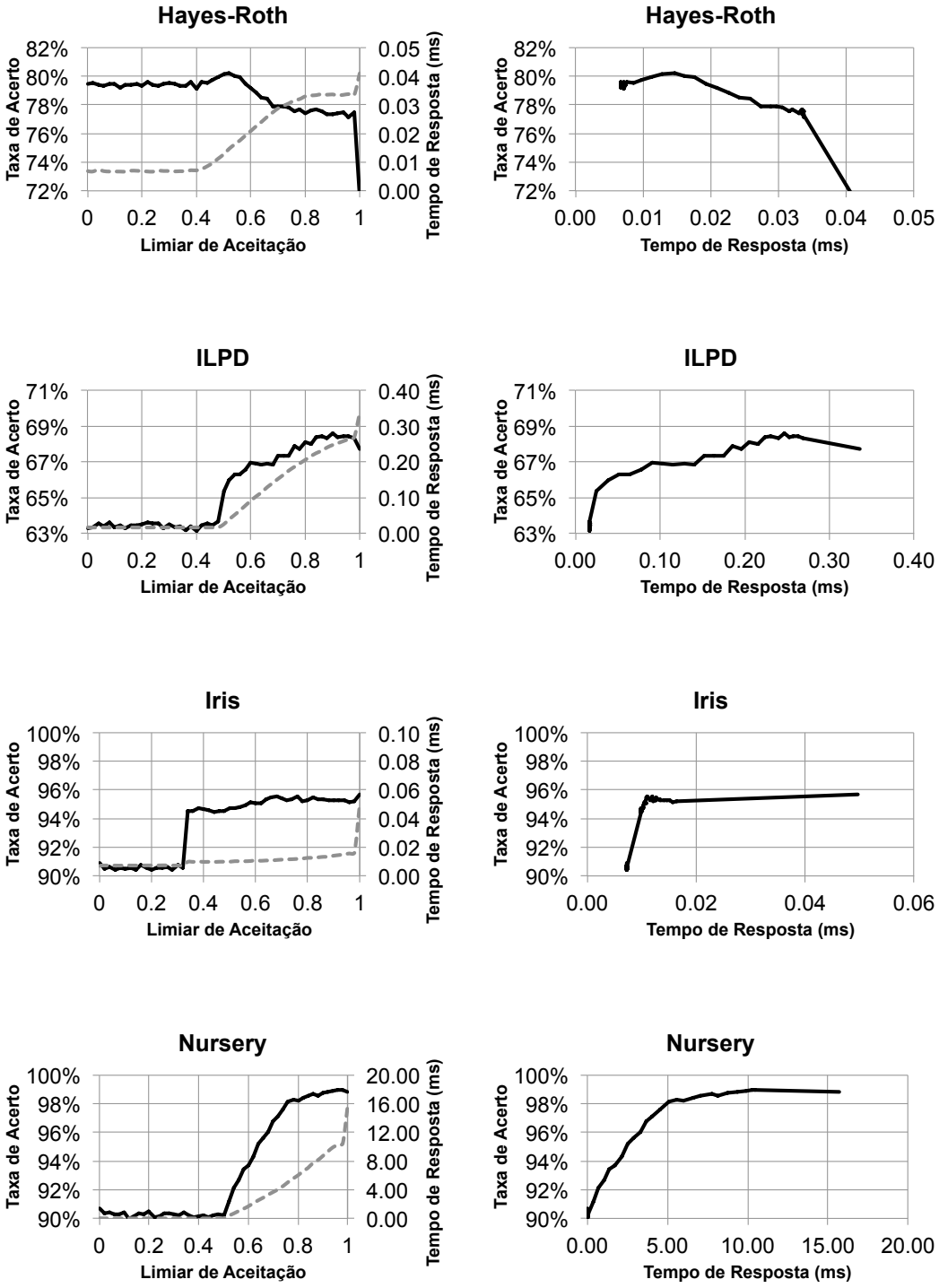


Figura 27: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema experimental em função do limiar. À direita, a curva de desempenho (taxa de acerto em função do tempo de resposta). (cont.)

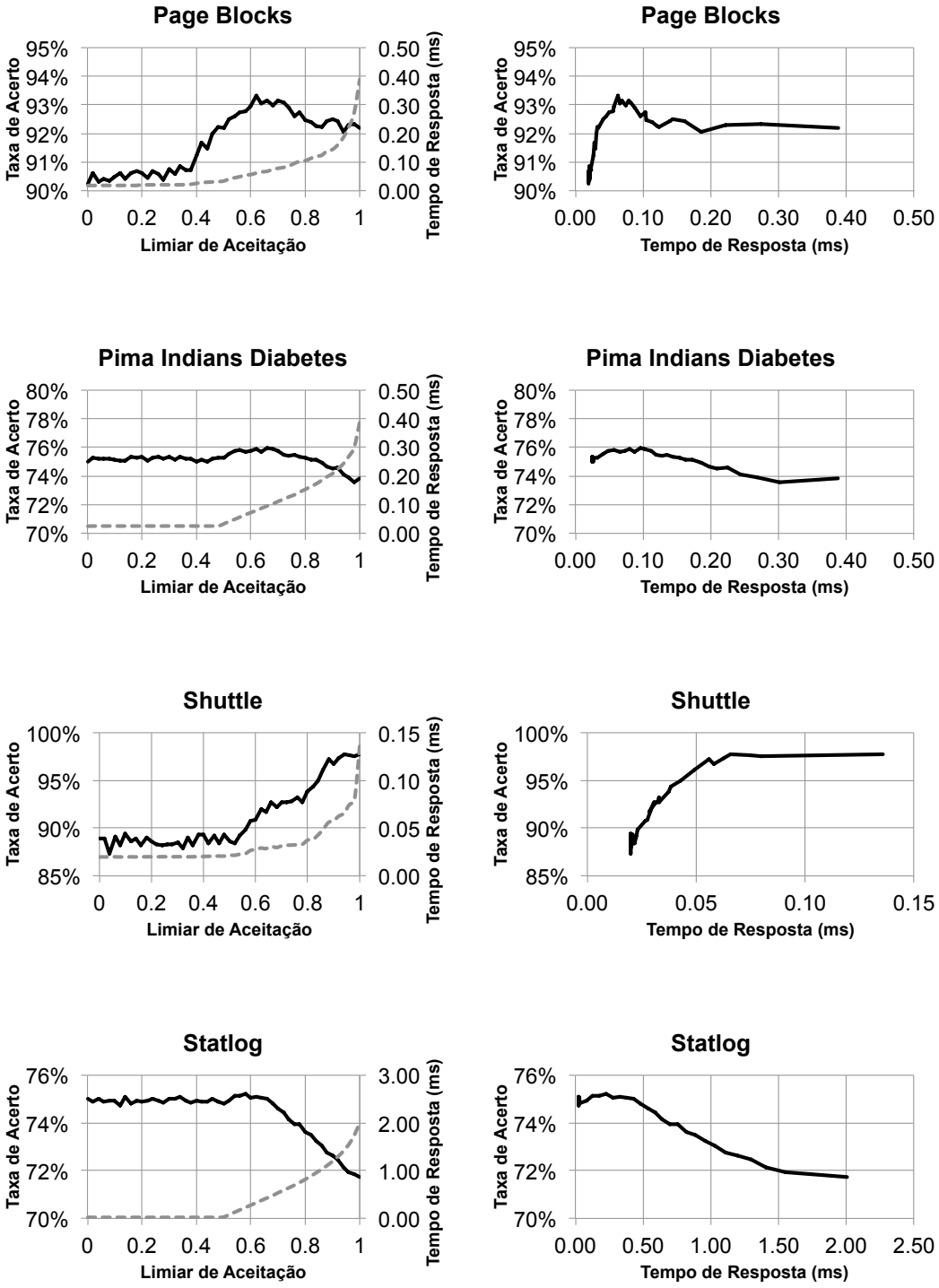


Figura 27: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema experimental em função do limiar. À direita, a curva de desempenho (taxa de acerto em função do tempo de resposta). (cont.)

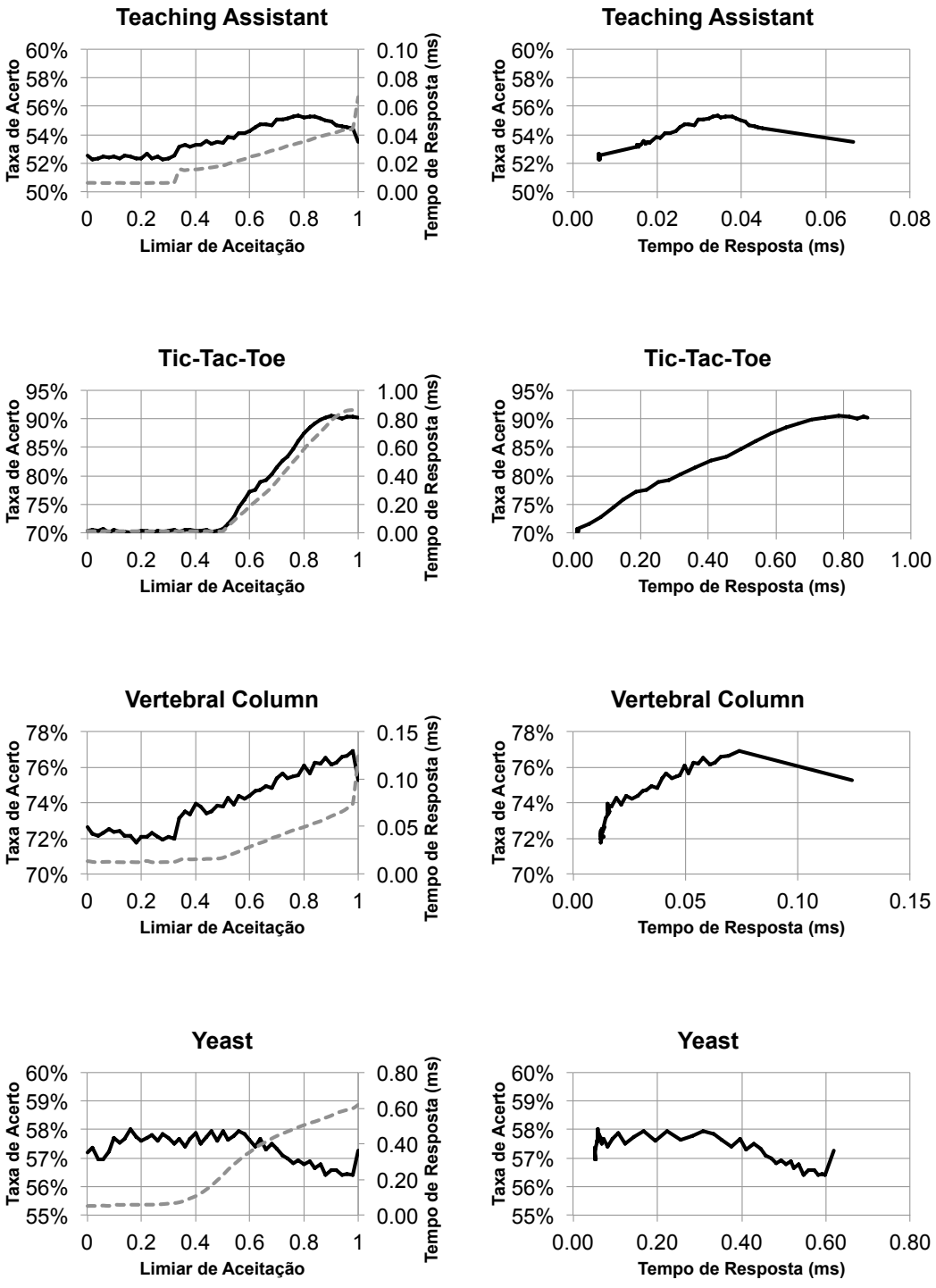


Figura 27: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema experimental em função do limiar. À direita, a curva de desempenho (taxa de acerto em função do tempo de resposta).

Quanto ao desempenho, verifica-se que a curva de desempenho mostra-se côncava voltada para baixo para a maior parte dos *datasets*, mesmo naqueles em que a taxa de acerto do *k-NN* é menor do que aquela apresentada pelo classificador de Bayes. Nos *datasets* que cumprem a premissa, a taxa de acerto apresenta um crescimento inicial mais acentuado, mas eventualmente satura após um determinado valor. Como exemplo, no *Car Evaluation*, a saturação ocorre em $0,50ms$, que corresponde ao limiar de $0,6$ (aprox.). Neste limiar crítico, é possível obter uma taxa de acerto equivalente ao do *k-NN*, mas utilizando apenas uma fração de seu tempo de resposta, representando um ganho de desempenho em relação ao dispositivo individual.

Em parte dos *datasets*, a taxa de acerto máxima ocorre em limiares intermediários, afastado dos valores extremos 0 e 1. Por exemplo, no *Teaching Assistant*, a taxa máxima de acerto de 55,5% ocorre no limiar $0,8$, e é superior àquelas obtidas por cada dispositivo isoladamente. *Blood Transfusion*, *Cardiotocography Class*, *Contraceptive Method*, *Credit Aproval*, *Page Blocks* e *Vertebral Column* também apresentam tal comportamento, e até mesmo o *Statlog*, que não atende as premissas iniciais, apresenta sua taxa máxima de acerto no limiar $0,6$.

Além disso, alguns *datasets* apresentam uma mudança mais acentuada na taxa de acerto e no tempo de resposta perto do limiar $1/p$. Neles, grande parte das instâncias utilizadas para validar o sistema recebem expectativas de certeza muito próximas deste valor em todas as alternativas. Isso faz com que a frequência de uso do *k-NN* se mostre muito sensível a pequenas variações no limiar, indicando que a quantidade de instâncias utilizadas para treinamento foi insuficiente para criar um modelo robusto para o classificador de Bayes, considerando o número de critérios e alternativas analisadas. Ocorre principalmente em *datasets* com critérios predominantemente numéricos.

- Como exemplo, o *Glass Identification*, que possui apenas 214 instâncias para 9 critérios e 7 alternativas, apresenta uma variação mais acentuada quando o limiar atinge $1/7$.

- Em comparação, *datasets* com uma proporção maior de números de instâncias por critério e por alternativa apresentam variações mais suaves - e.g., *Car Evaluation*, com 1728 instâncias para 6 critérios e 4 alternativas.

De forma similar, certos *datasets* também apresentam variações mais acentuadas conforme o limiar se aproxima de 1 - especialmente no tempo de resposta. Neste caso, uma grande proporção de instâncias recebem expectativas de certeza muito elevadas em determinadas alternativas, e que somente serão rejeitadas com limiares muito elevados. Também ocorrem em *datasets* predominantemente numéricos - e.g., *Blood Transfusion*.

A partir destes resultados, pode-se concluir que é possível realizar um controle sobre a relação de compromisso entre taxa de acerto e tempo de resposta através da condição descrita pela função dada em (3.1), utilizando limiares entre $1/p$ e 1. O levantamento da curva de desempenho do sistema permite escolher o valor do limiar que melhor se ajuste às necessidades do usuário. Como exemplo, caso um dos requisitos seja obter a máxima taxa de acerto no *Tic-Tac-Toe* gastando, em média, $0,6$ ms por instância, o limiar $0,8$ é um excelente candidato a ser utilizado. Entretanto, um cuidado deve ser tomado ao escolher limiares próximos de $1/p$ ou 1, já que tanto a taxa de acerto como o tempo de resposta são mais instáveis nestes pontos críticos.

Também pode ser ressaltado que, em alguns casos, a distribuição de carga informada não só permite controlar esta relação de compromisso, mas também permite obter uma taxa máxima de acerto superior àquelas obtidas

pelos dispositivos individuais, mediante a escolha de um limiar ótimo que se encontra, em geral, afastado dos valores extremos.

4.4.2 Sistema Experimental com 100 Intervalos Discretos

O experimento apresentado na seção 4.4.1 foi repetido uma segunda vez, mas utilizando uma discretização em 100 intervalos (a seção 4.3.2 apresenta os resultados individuais obtidos para cada dispositivo nestas condições). Tal experimento permite verificar se o método de discretização utilizado afeta a relação de compromisso entre a taxa de acerto e o tempo de resposta, comparando-o com os resultados obtidos com 10 intervalos.

Os gráficos da Figura 28 mostram os resultados obtidos através desta configuração. Comparando-os com os resultados do experimento anterior, percebe-se que a taxa de acerto e o tempo de resposta nos *datasets* puramente nominais (e.g., *Car Evaluation*) não sofrem alterações. Este comportamento já era esperado, pois a discretização afeta somente aqueles que possuem critérios numéricos.

Também fica evidenciado que as novas curvas levantadas, em geral, possuem descontinuidades mais fortes quando comparadas com aquelas apresentadas pelo mesmo sistema com 10 intervalos discretos. Tal característica dificulta o uso do limiar como controle de desempenho do sistema, visto que não é possível realizar ajustes finos nestas regiões: pequenas alterações no limiar causam grandes variações de desempenho.

A partir destes resultados, também se observa que a curva de desempenho é fortemente influenciada pelo desempenho individual de cada dispositivo. Na grande maioria dos *datasets*, o desempenho dos dispositivos foi pior com 100 intervalos de discretização, o que leva a uma curva de desempenho me-

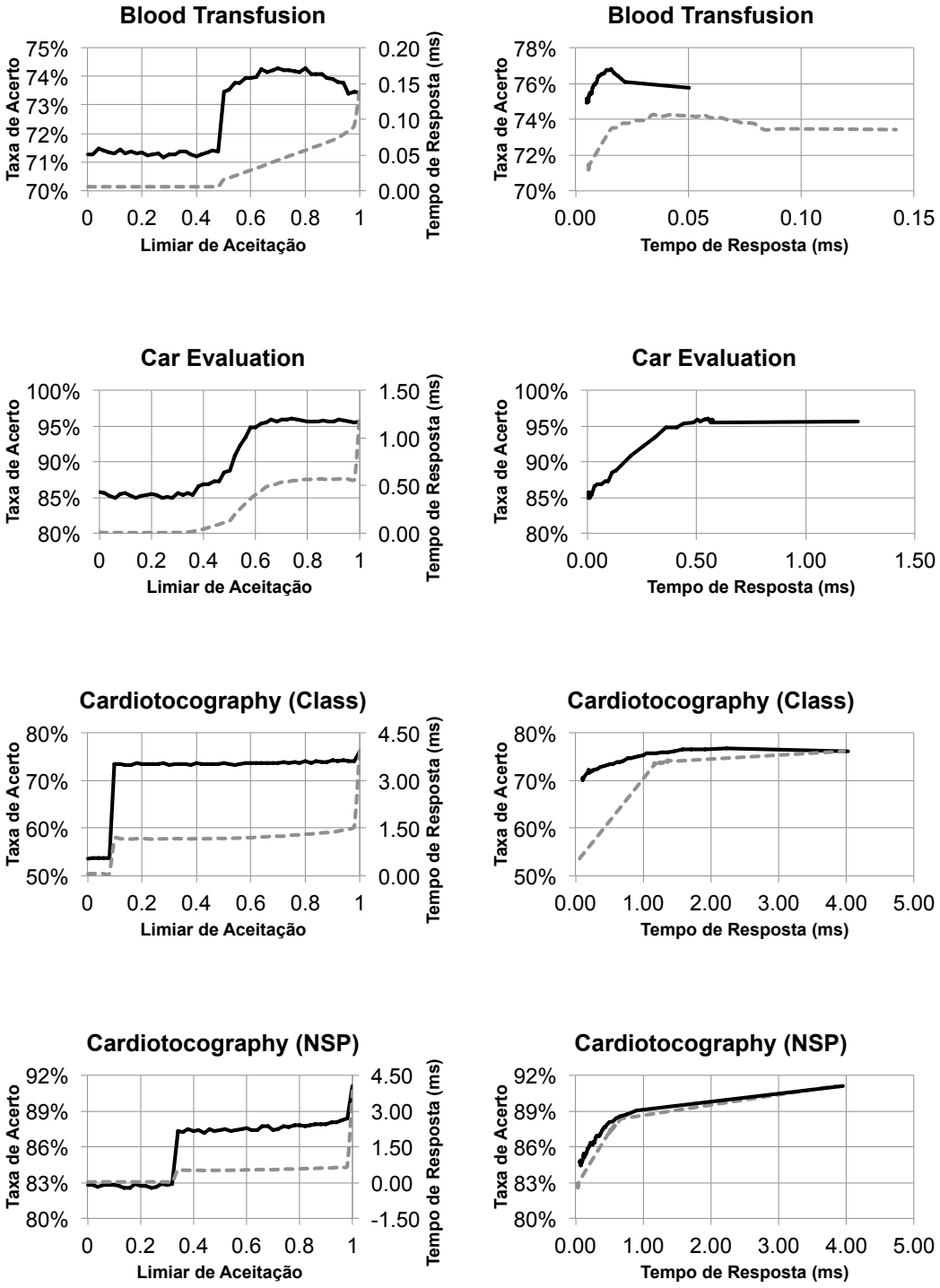


Figura 28: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema (discretizado em 100 intervalos) em função do limiar. À direita, a curva de desempenho do sistema com *EWD* com 10 intervalos (linha contínua) e 100 intervalos (linha tracejada). (cont.)

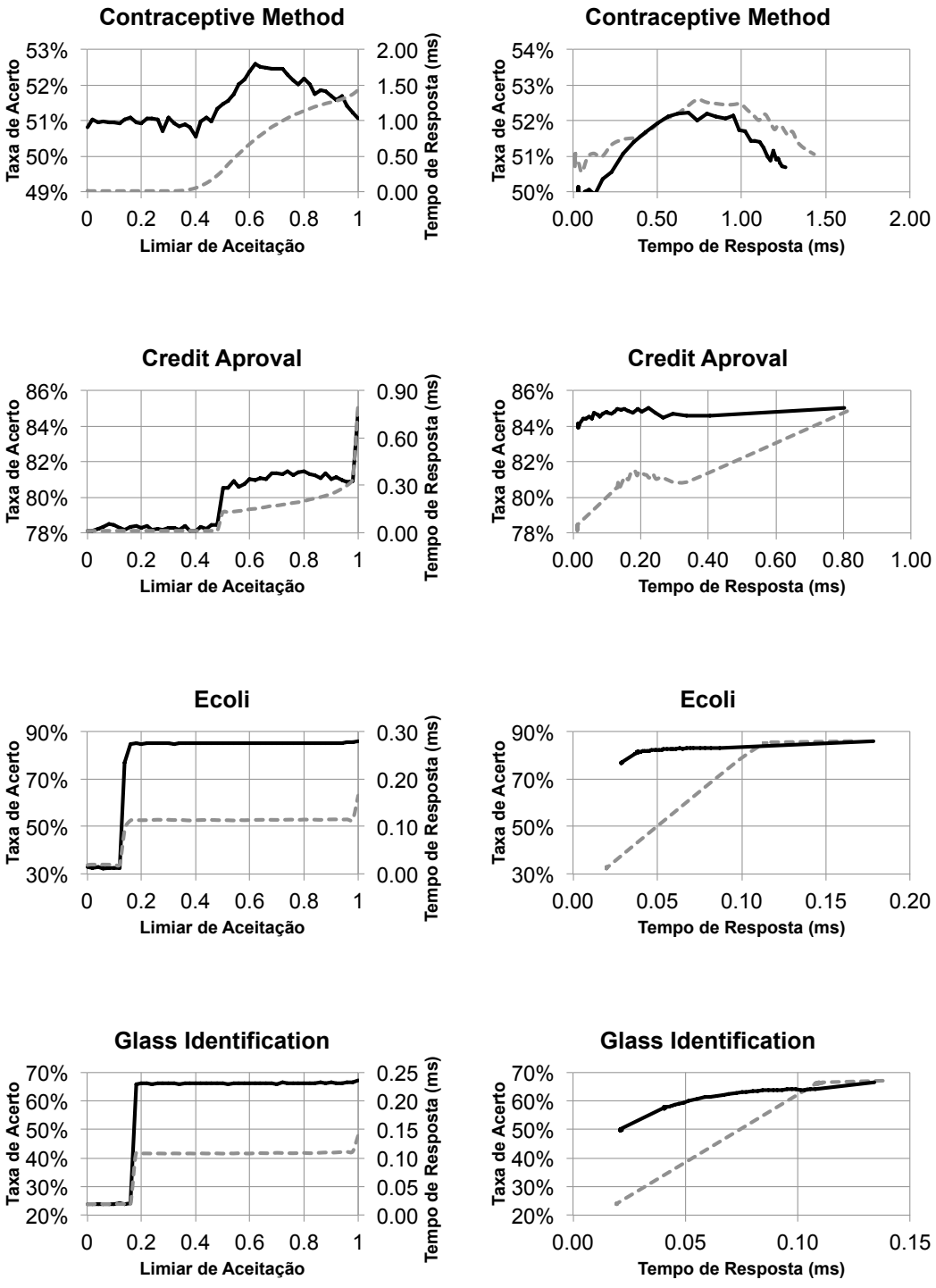


Figura 28: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema (discretizado em 100 intervalos) em função do limiar. À direita, a curva de desempenho do sistema com *EWD* com 10 intervalos (linha contínua) e 100 intervalos (linha tracejada). (cont.)

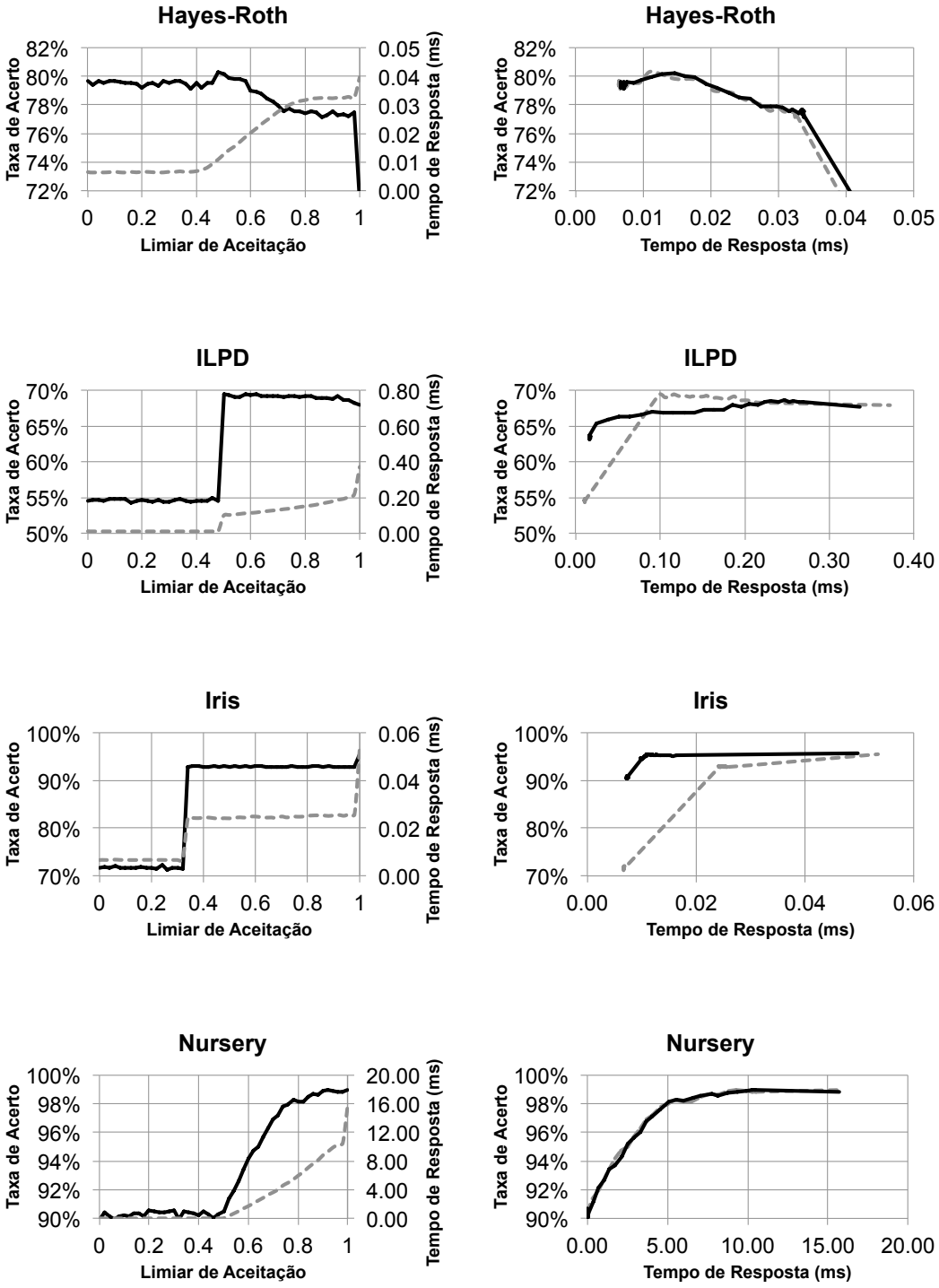


Figura 28: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema (discretizado em 100 intervalos) em função do limiar. À direita, a curva de desempenho do sistema com *EWD* com 10 intervalos (linha contínua) e 100 intervalos (linha tracejada). (cont.)

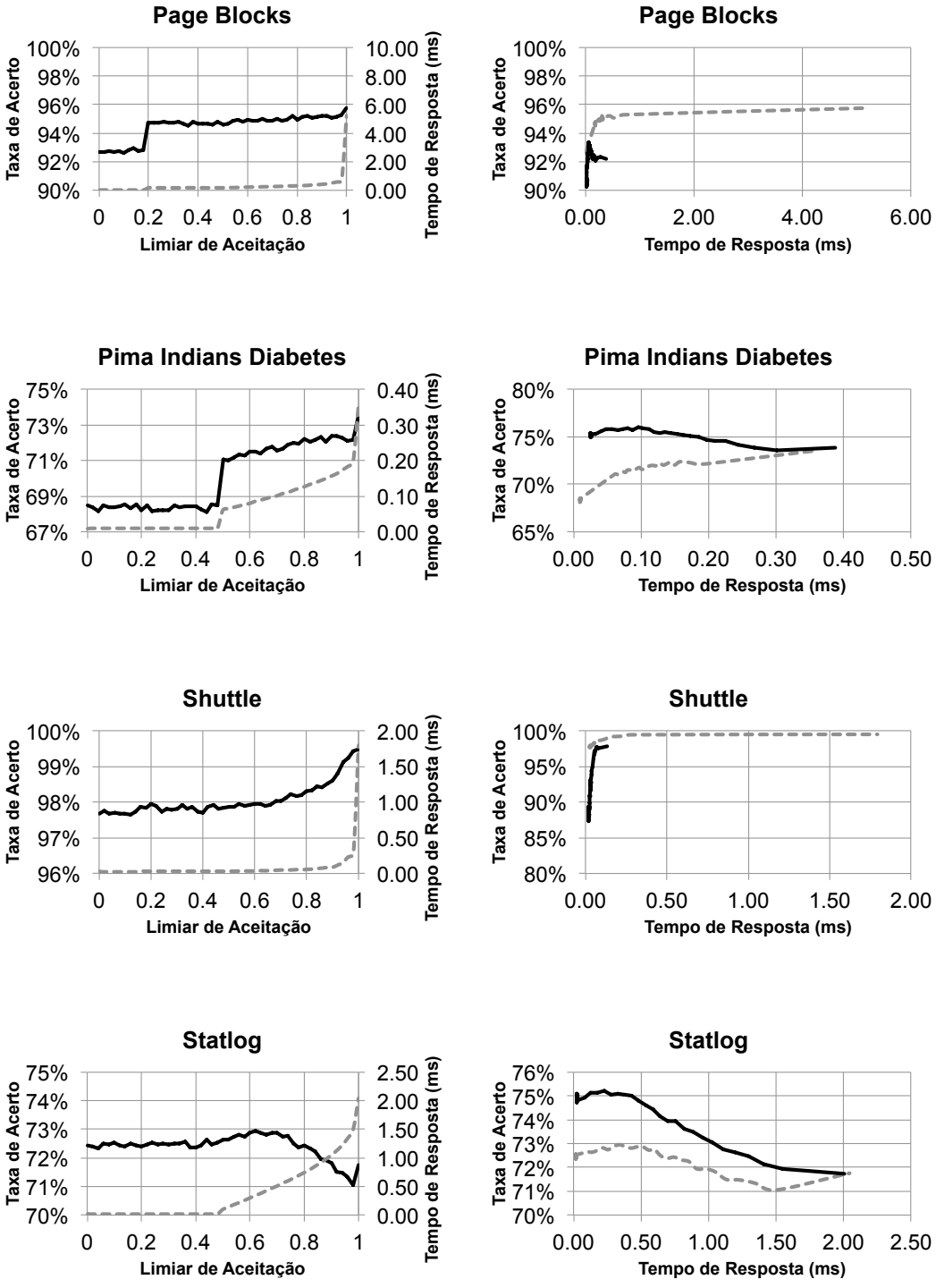


Figura 28: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema (discretizado em 100 intervalos) em função do limiar. À direita, a curva de desempenho do sistema com *EWD* com 10 intervalos (linha contínua) e 100 intervalos (linha tracejada). (cont.)

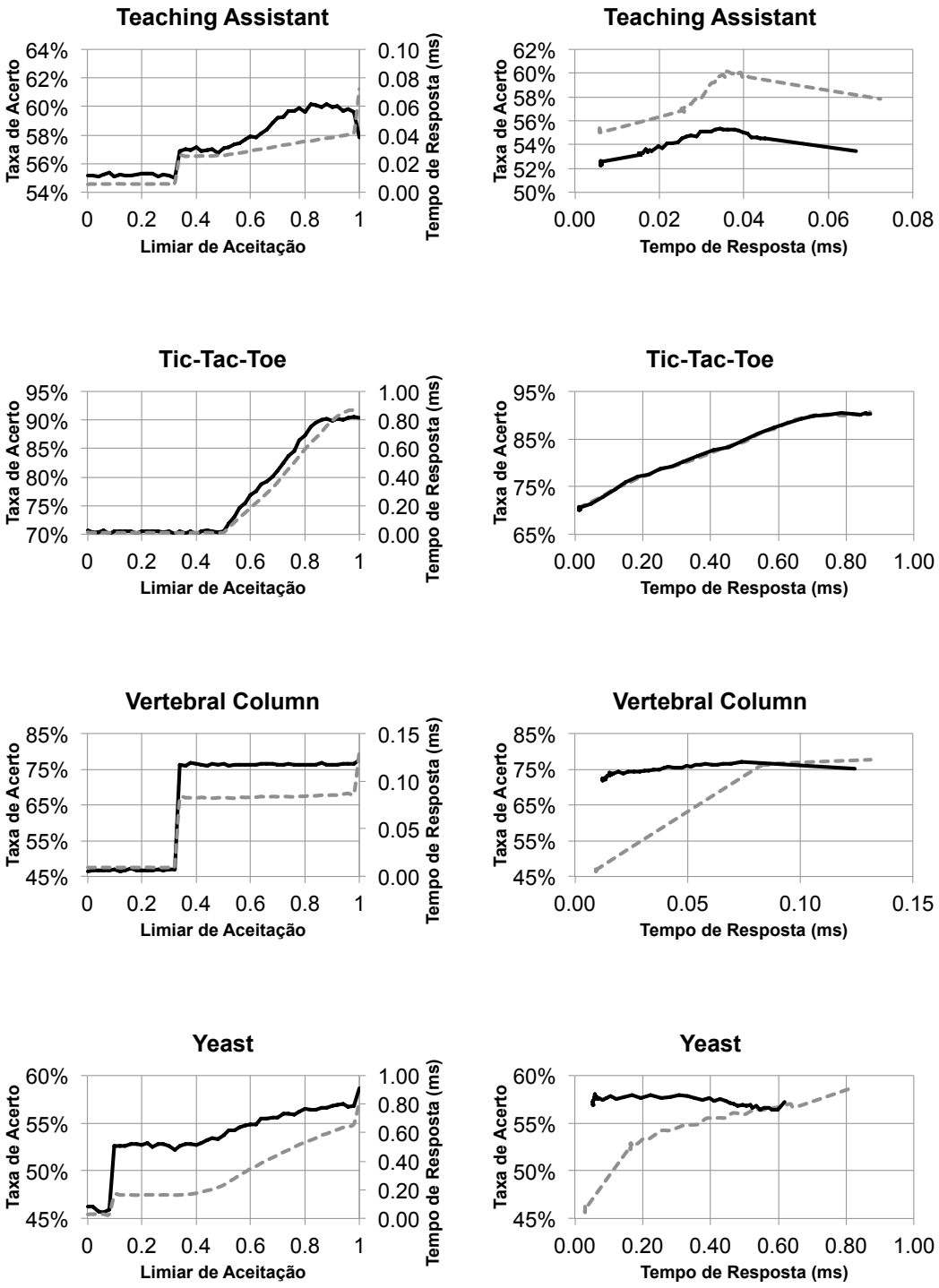


Figura 28: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema (discretizado em 100 intervalos) em função do limiar. À direita, a curva de desempenho do sistema com *EWD* com 10 intervalos (linha contínua) e 100 intervalos (linha tracejada).

nos favorável. Em comparação, o desempenho nos *datasets Page Blocks* e *Shuttle* é superior quando 100 intervalos são utilizados. Note que estes são os *datasets* com o maior número de instâncias, apesar de o tempo de resposta máximo observado ser muito superior ao máximo observado no experimento com 10 intervalos.

Este experimento explicita a necessidade de uma escolha adequada para o método de discretização e seus parâmetros, uma vez que uma escolha equivocada não só piora o desempenho individual de cada dispositivo, mas também prejudica o ajuste da relação de compromisso entre taxa de acerto e tempo de resposta de todo o sistema. Exemplos de métodos de discretização incluem o *Recursive Minimal Entropy Partitioning* e geração de clusters (DOUGHERTY; KOHAVI; SAHAMI, 1995), conforme apresentado previamente na seção 2.2.1.

4.4.3 Sistema Experimental com *TDAE*

O experimento da seção 4.4.1 foi novamente realizado, desta vez, inserindo uma *TDAE* de entrada estendida como dispositivo inicial da fila. A presença desta tabela permite que instâncias numéricas sejam potencialmente solucionadas sem a necessidade de utilizar os demais dispositivos, conforme mostrado na seção 4.3.3.

Deve ser ressaltado que, por natureza, uma tabela de decisão é um dispositivo cujo foco é solucionar problemas previstos em seu conjunto de regras, criado a partir de um treinamento. No entanto, o método de Monte-Carlo, empregado para validação da taxa de acerto, simula situações em que o dispositivo recebe problemas não previstos no conjunto de treinamento (vale lembrar que os subconjuntos de treinamento e de validação não possuem elementos em comum, a menos que o *dataset* apresente instâncias repetidas). Logo,

neste caso específico, a tabela de decisão não aparenta ser um dispositivo adequado. Os resultados obtidos pela análise preliminar sustentam esta hipótese:

- Quando comparado com o classificador de Bayes, o desempenho da *TDAE* é, na maior parte dos casos, inferior, tanto na taxa de acerto como no tempo de resposta;
- Apesar de apresentar tempo de resposta menor do que o *k-NN*, a taxa de acerto da *TDAE* é bastante inferior.

Uma forma de visualizar e comparar o desempenho de cada um dos dispositivos em um determinado *dataset* é através de um gráfico que plote a taxa de acerto em função do tempo de resposta. O desempenho de cada dispositivo é representado na forma de um ponto (t, h), referente ao tempo de resposta t e à taxa de acerto h apresentados. Através deste gráfico, é possível avaliar o desempenho da *TDAE* verificando se o ponto $(t_{observado}, h_{observado})$, referente aos parâmetros de desempenho da *TDAE*, está acima da reta que liga os pontos (t_{bayes}, h_{bayes}) e (t_{knn}, h_{knn}), refentes aos parâmetros de desempenho dos outros dois dispositivos. A Figura 29 ilustra um exemplo deste comparativo.

Para verificar se o ponto $(t_{observado}, h_{observado})$ está acima da referida reta, calcula-se a posição de um ponto de referência que apresente o mesmo tempo de resposta, mas que esteja sobre a reta. Sua taxa de acerto $h_{critico}$, denominada *valor crítico*, é calculada através de:

$$r = (t_{observado} - t_{bayes}) \,/\, (t_{knn} - t_{bayes}) \tag{4.1}$$

$$h_{critico} = \min\left((h_{knn} - h_{bayes}) \cdot r + h_{bayes}, 100\%\right) \tag{4.2}$$

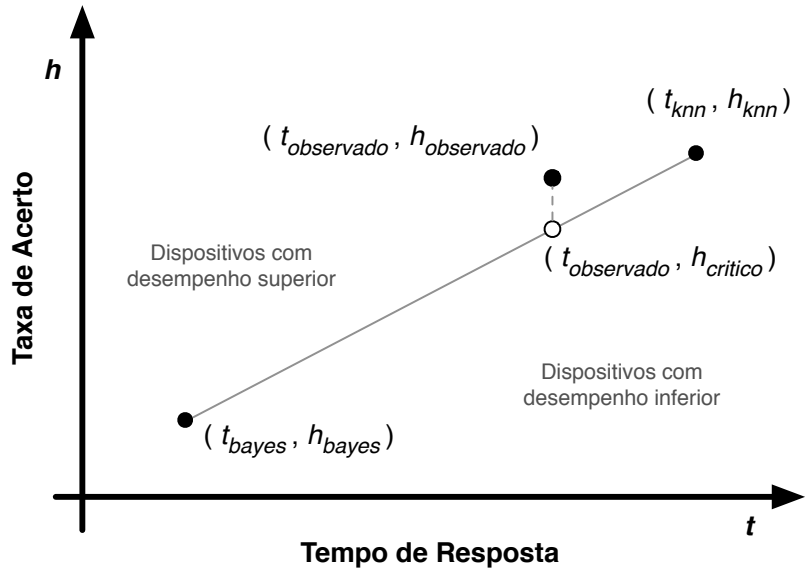


Figura 29: Cálculo da taxa de acerto crítica a partir dos parâmetros de desempenho dos classificador de Bayes e do método *k-NN*.

Para um determinado *dataset*, o uso da *TDAE* pode ser considerado inadequado se $h_{observado} < h_{critico}$, já que a tabela estaria aquém do desempenho dos demais dispositivos. Por outro lado, se $h_{observado} \geqslant h_{critico}$, o desempenho da *TDAE* estaria de acordo com o que é apresentado pelos demais dispositivos, tornando-o potencialmente adequado. A Figura 30 compara as taxas de acerto apresentadas pela *TDAE* em cada *dataset* com seus respectivos valores críticos. Dela, é verificado que a taxa de acerto observada é superior ao valor crítico apenas no *Shuttle*, o que potencialmente justifica o uso da tabela de decisão neste *dataset* em particular. Nos demais, a tabela não aparenta ser adequada.

A Figura 31 apresenta a taxa de acerto e o tempo de resposta de cada *dataset* em função do limiar, e compara seu desempenho com aquele obtido através do sistema experimental original. Naqueles que apresentam baixa taxa de acerto para a *TDAE* (e.g., *Cardiotocography Class*, cuja taxa de acerto individual é 6,6873%, conforme mostra a Tabela 12), a curva de desempenho

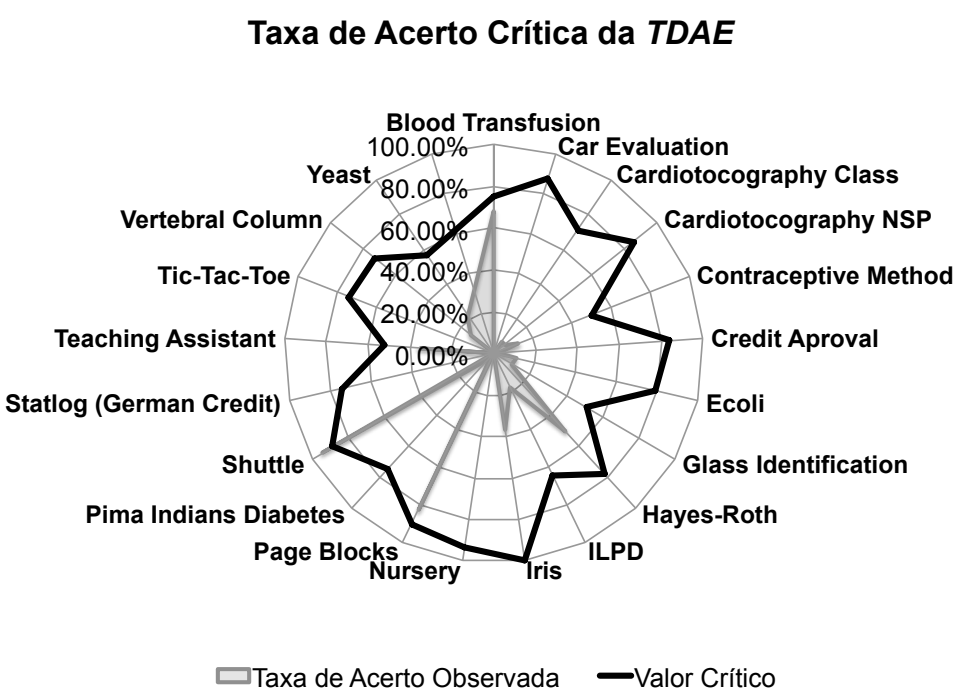


Figura 30: Comparativo entre as taxas de acerto apresentada pela *TDAE* em cada *dataset* e seus respectivos valores críticos.

sofre uma translação no eixo horizontal, sem sofrer grandes alterações em sua forma. Este comportamento mostra que, nestes casos, a utilização da *TDAE* fez apenas aumentar o tempo de resposta, sem alterar significativamente a taxa de acerto. Em especial, *datasets* estritamente nominais (e.g., *Car Evaluation*) possuem taxa de acerto nula na *TDAE*, e, por consequência, também apresentam a curva de desempenho transladada.

Contudo, em alguns *datasets*, o ganho observado é positivo em determinadas situações. Como exemplo, a curva de desempenho apresentada no *Shuttle* favorece o uso da *TDAE* para tempos de resposta menores - note que, neste *dataset*, o desempenho individual da *TDAE* foi superior ao valor crítico previamente estimado, justificando seu uso. Também observa-se um ganho no *Teaching Assistant*, que, por apresentar instâncias repetidas, possibilita que os conjuntos de validação e de treinamento tenham elementos em comum - situação favorável ao uso de tabelas de decisão. Vale ressaltar que

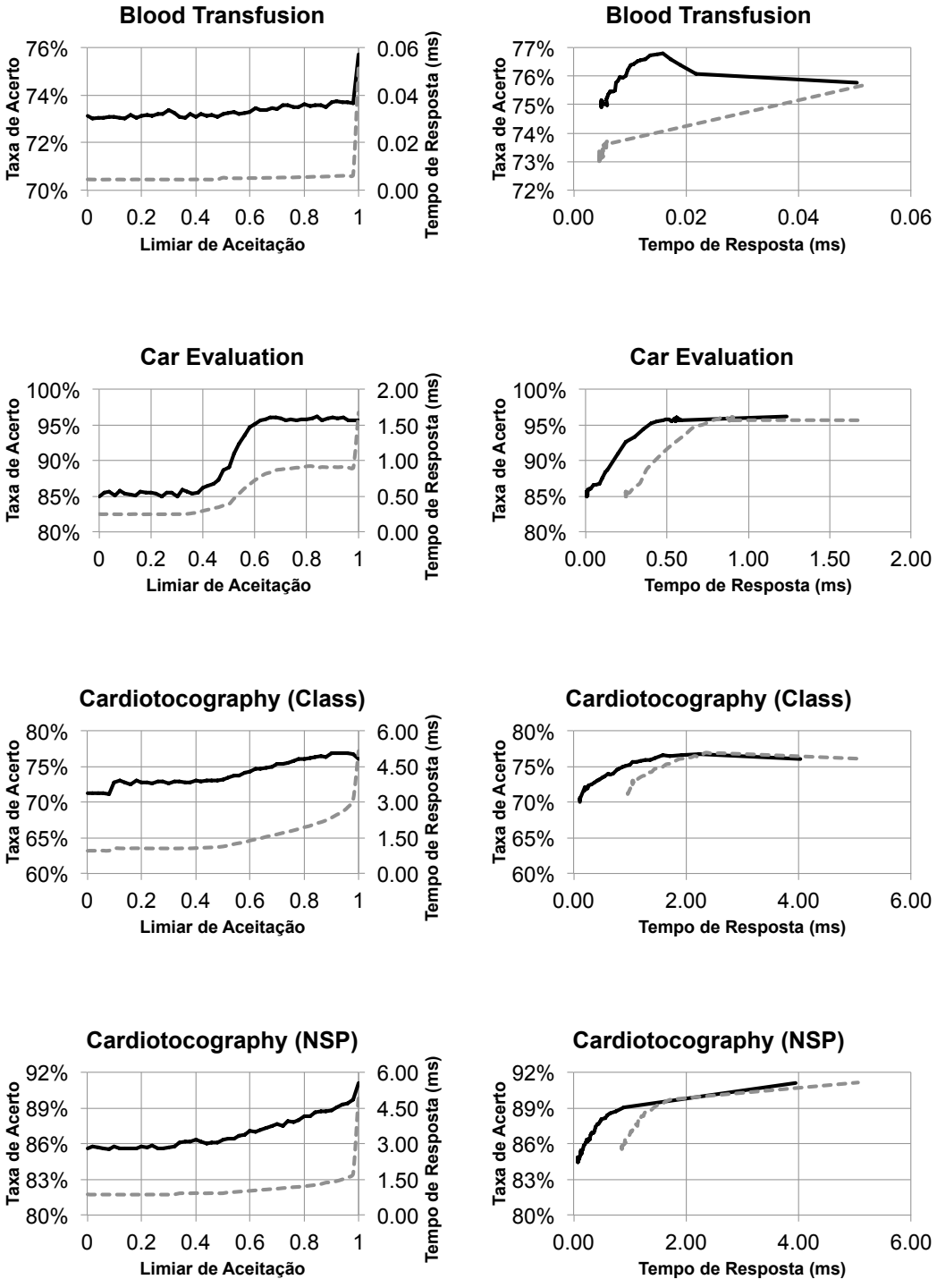


Figura 31: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema com *TDAE* em função do limiar. À direita, a curva de desempenho do sistema experimental original (linha contínua) e com o *TDAE* (linha tracejada). (cont.)

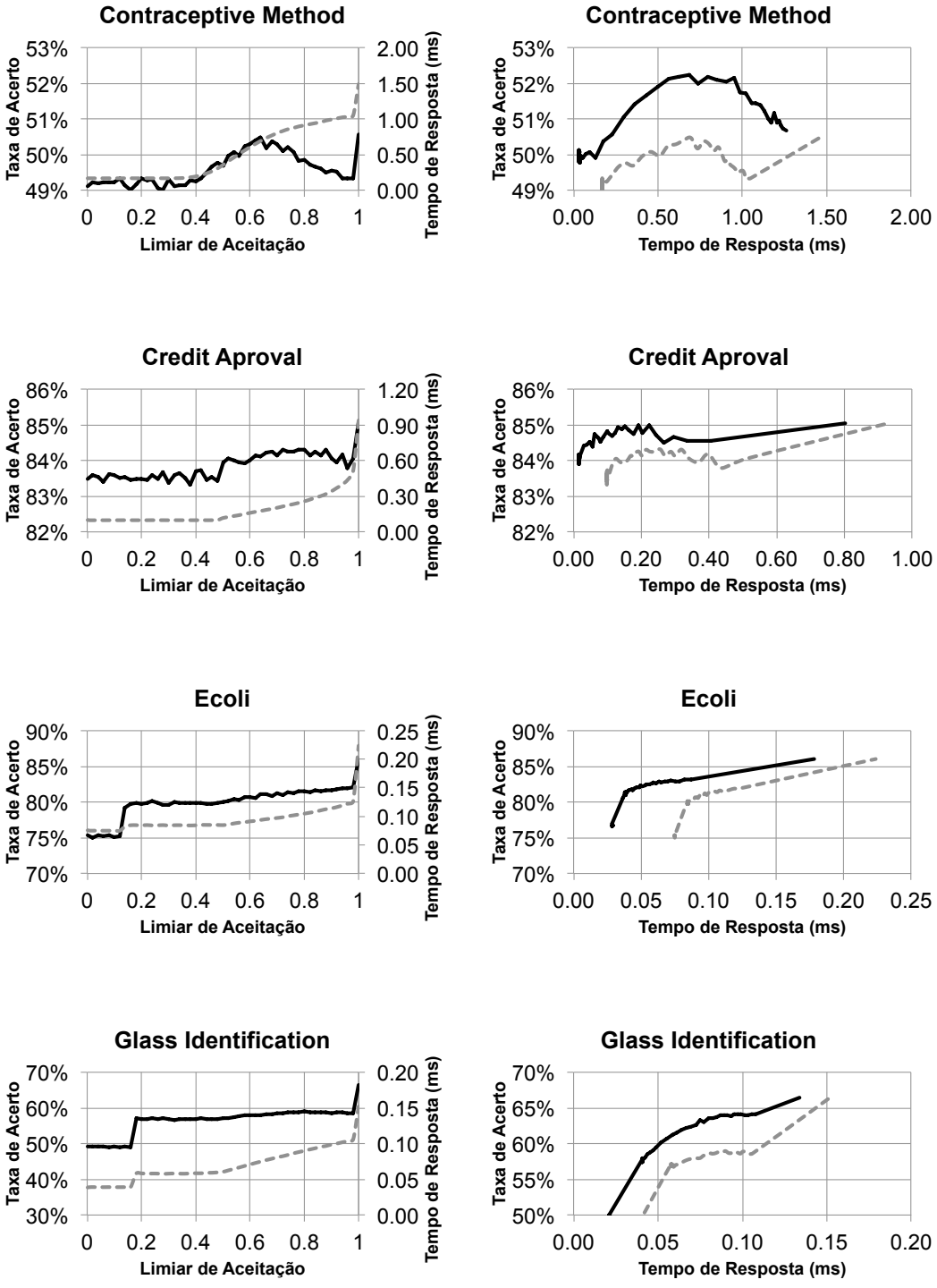


Figura 31: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema com *TDAE* em função do limiar. À direita, a curva de desempenho do sistema experimental original (linha contínua) e com o *TDAE* (linha tracejada). (cont.)

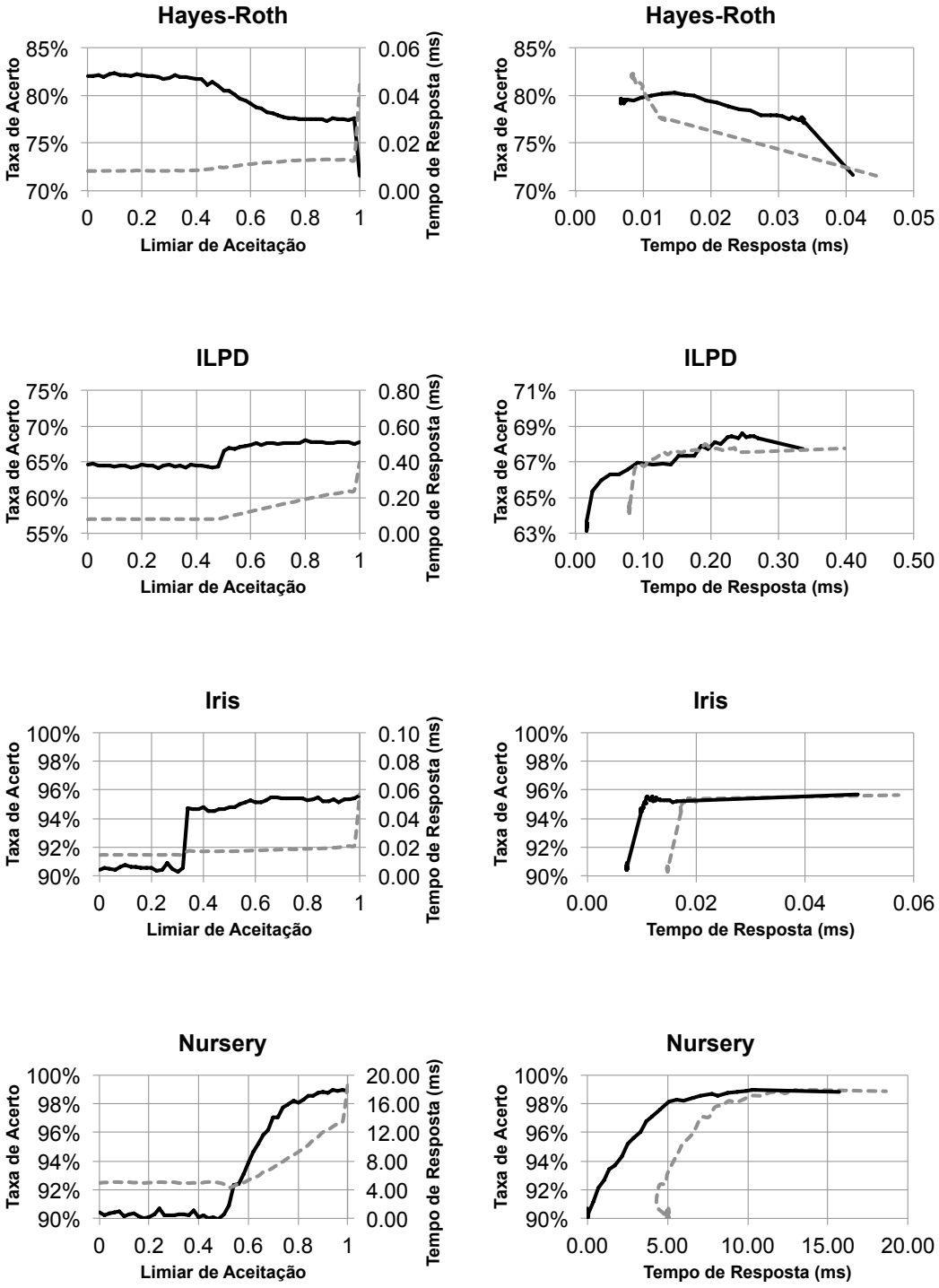


Figura 31: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema com *TDAE* em função do limiar. À direita, a curva de desempenho do sistema experimental original (linha contínua) e com o *TDAE* (linha tracejada). (cont.)

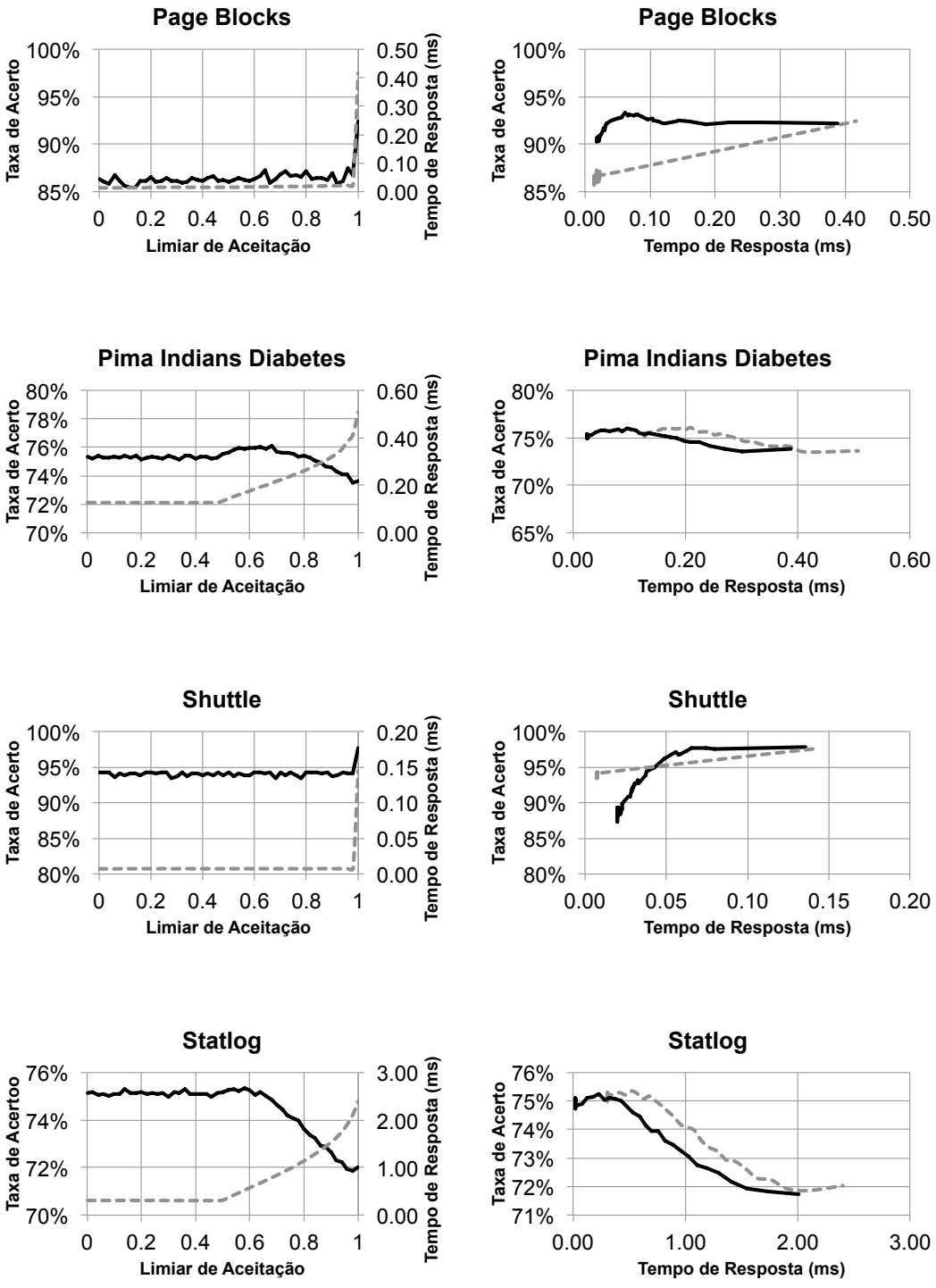


Figura 31: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema com *TDAE* em função do limiar. À direita, a curva de desempenho do sistema experimental original (linha contínua) e com o *TDAE* (linha tracejada). (cont.)

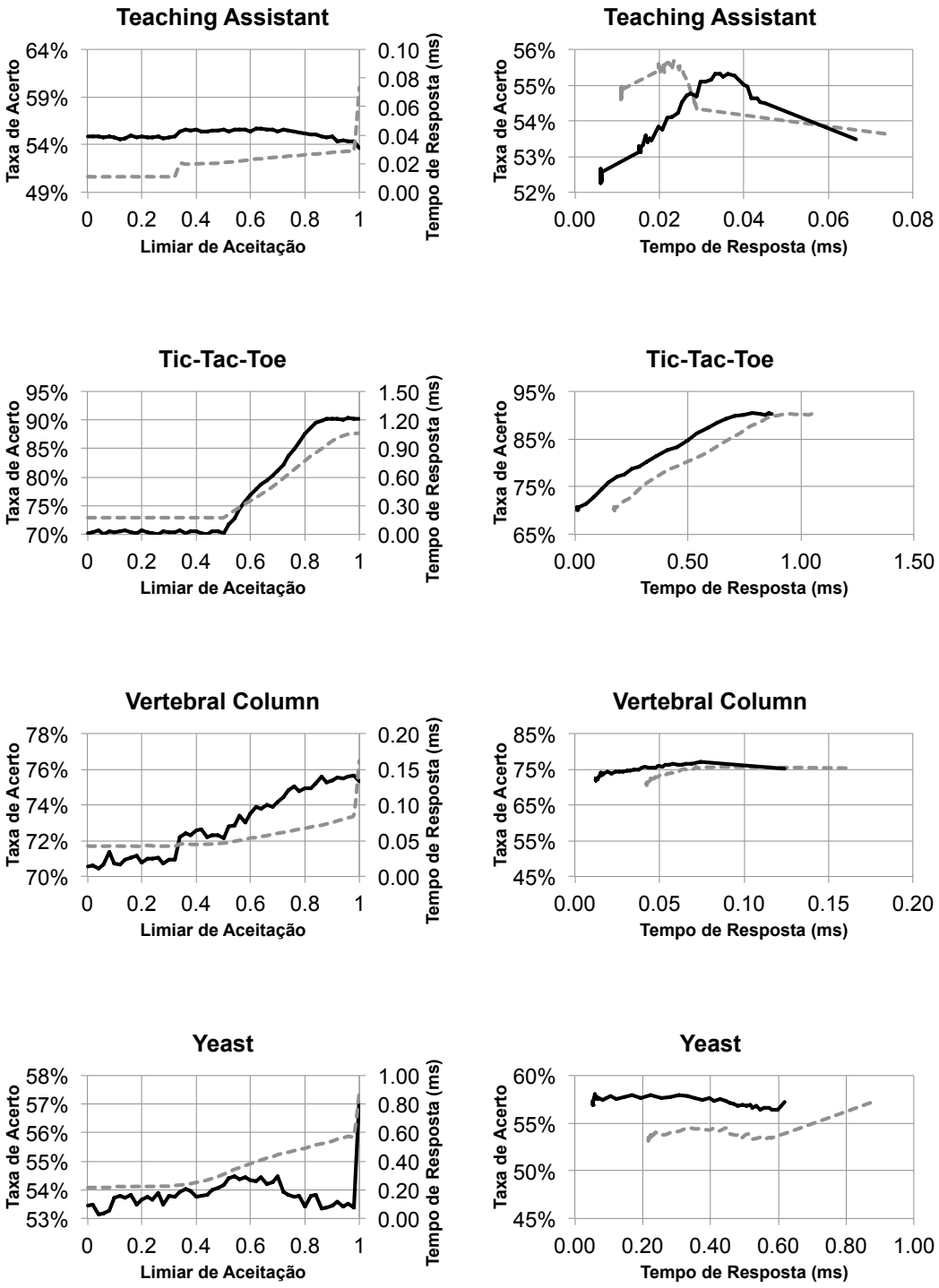


Figura 31: À esquerda, o gráfico de taxa de acerto (linha contínua) e tempo de resposta (linha tracejada) do sistema com *TDAE* em função do limiar. À direita, a curva de desempenho do sistema experimental original (linha contínua) e com o *TDAE* (linha tracejada).

este é o único *dataset* que apresenta instâncias repetidas.

De forma análoga aos experimentos anteriores, também pode ser observado que o tempo máximo de resposta verificado no limiar 1 é igual à somatória dos tempos de respostas de cada dispositivo utilizado, no caso, do classificador de Bayes, do *k-NN* e da *TDAE*.

Estes resultados demonstram a importância da escolha dos dispositivos que devem compor o sistema de tomada de decisão: mesmo utilizando a adaptatividade para controlar a fila de dispositivos e o aprendizado, o grau de sucesso do sistema também depende do desempenho individual dos dispositivos em cada problema específico. Neste experimento, como os conjuntos de instâncias utilizadas para treinamento e validação eram disjuntos, o uso da *TDAE* foi prejudicial na maior parte dos *datasets*, mas mostrou-se adequado naqueles que apresentavam características favoráveis ao uso de uma tabela de decisão.

4.5 Comparação com um Sistema com Distribuição de Carga Não Informada

A fim de criar uma base de comparação para o sistema proposto, foi levantada a curva de desempenho de um sistema que realiza uma distribuição de carga não informada, que escolhe, aleatoriamente, um dispositivo para solucionar cada uma das instâncias. Diferentemente do sistema experimental, tal distribuição de carga não utiliza informações a respeito de cada instância para decidir qual dispositivo utilizar, logo, espera-se que seu desempenho seja inferior.

Considerando os dispositivos testados previamente, este experimento separa o conjunto de testes a ser utilizado em dois subconjuntos: o primeiro

deles, com $(100 - r)\,\%$ das instâncias de testes, é validado com o classificador de Bayes, enquanto o segundo, com $r\%$, é validado com *k-NN*.

Por conseguinte, para levantar a curva de desempenho com esta distribuição de carga, a taxa de acerto e o tempo de resposta são medidos para cada valor de r, partindo de 0% até 100% em passos de 2%. Estas medidas são extraídas a partir da média obtida em 100 testes realizados de forma independentes. Em cada teste, são utilizados 70% das instâncias para treinamento, utilizando os demais 30% para o conjunto de testes. Deste último conjunto, cria-se um subconjunto de instâncias que serão solucionadas com *k-NN* e um outro para o classificador de Bayes, de acordo com o valor de r.

Considerando-se que a escolha é aleatória, espera-se que, em cada valor de r, sejam obtidas as seguintes estimativas para a taxa de acerto e tempo de resposta:

$$h\,(r) = (1 - r)\,h_{bayes} + (r)\,h_{kNN} = r\,(h_{kNN} - h_{bayes}) + h_{bayes} \tag{4.3}$$

$$t\,(r) = (1 - r)\,t_{bayes} + (r)\,t_{kNN} = r\,(t_{kNN} - t_{bayes}) + t_{bayes} \tag{4.4}$$

onde h_{bayes} e h_{kNN} são constantes representando as taxas de acerto individuais de cada dispositivo em um determinado *dataset*, enquanto t_{bayes} e t_{kNN} representam os seus respectivos tempos de resposta. Manipulando as funções acima para definir a taxa de acerto em função do tempo de resposta, chega-se na seguinte função de grau 1:

$$h\,(t) = \frac{t - t_{bayes}}{t_{kNN} - t_{bayes}}\,(h_{kNN} - h_{bayes}) + h_{bayes} \tag{4.5}$$

Logo, espera-se que a curva de desempenho seja, aproximadamente, um segmento de reta que liga os pontos (t_{bayes}, h_{bayes}) e (t_{kNN}, h_{kNN}).

O resultado deste experimento está ilustrado na Figura 32, comparando o desempenho deste sistema com o sistema proposto na seção 4.1 nos *datasets* testados previamente. Conforme esperado, as curvas são aproximadamente segmentos de retas que, no geral, apresentam custo-benefício pior do que aquele apresentado pelo sistema experimental proposto. Apenas o *Statlog* apresenta um desempenho favorável à escolha aleatória na maior parte do tempo.

Deve-se notar que, em cada *dataset*, as duas curvas de desempenho começam em um mesmo ponto (t_{bayes}, h_{bayes}), mas acabam em pontos finais distintos, apesar de possuírem a mesma taxa de acerto h_{kNN}. Isso ocorre pois o sistema experimental, no pior caso, faz uso dos dois dispositivos (quando o limiar é igual a 1), consumindo $t_{bayes} + t_{kNN}$ para ser executado, enquanto a seleção aleatória, no pior caso, consome apenas t_{kNN}. Isso faz com que o desempenho da seleção aleatória seja favorável nestas regiões em determinados *datasets* (e.g., *Ecoli* e *Glass Identification*). Todavia, observa-se que utilizar o sistema híbrido neste ponto de operação faz pouco sentido, visto que no limiar 1 apenas o último dispositivo é efetivamente utilizado, ainda que sejam executados todos os demais. Entretanto, este comportamento poderia ser alterado se, ao invés do Algoritmo 7, uma outra técnica de decisão fosse utilizada para solucionar instâncias cujas decisões obtidas pelos dispositivos não satisfazem a condição de saída da fila (e.g., um sistema de votação entre os dispositivos) - um possível tema para trabalho futuro.

Estes resultados consolidam a ideia de que a distribuição de carga através do limiar se mostra mais eficiente do que uma simples escolha aleatória, obtendo, na maior parte dos casos, taxas de acerto maiores para um mesmo tempo de resposta. Consequentemente, tais resultados respaldam o uso das função auxiliares introduzidas pela *TDAE* como forma colaborativa de tomada

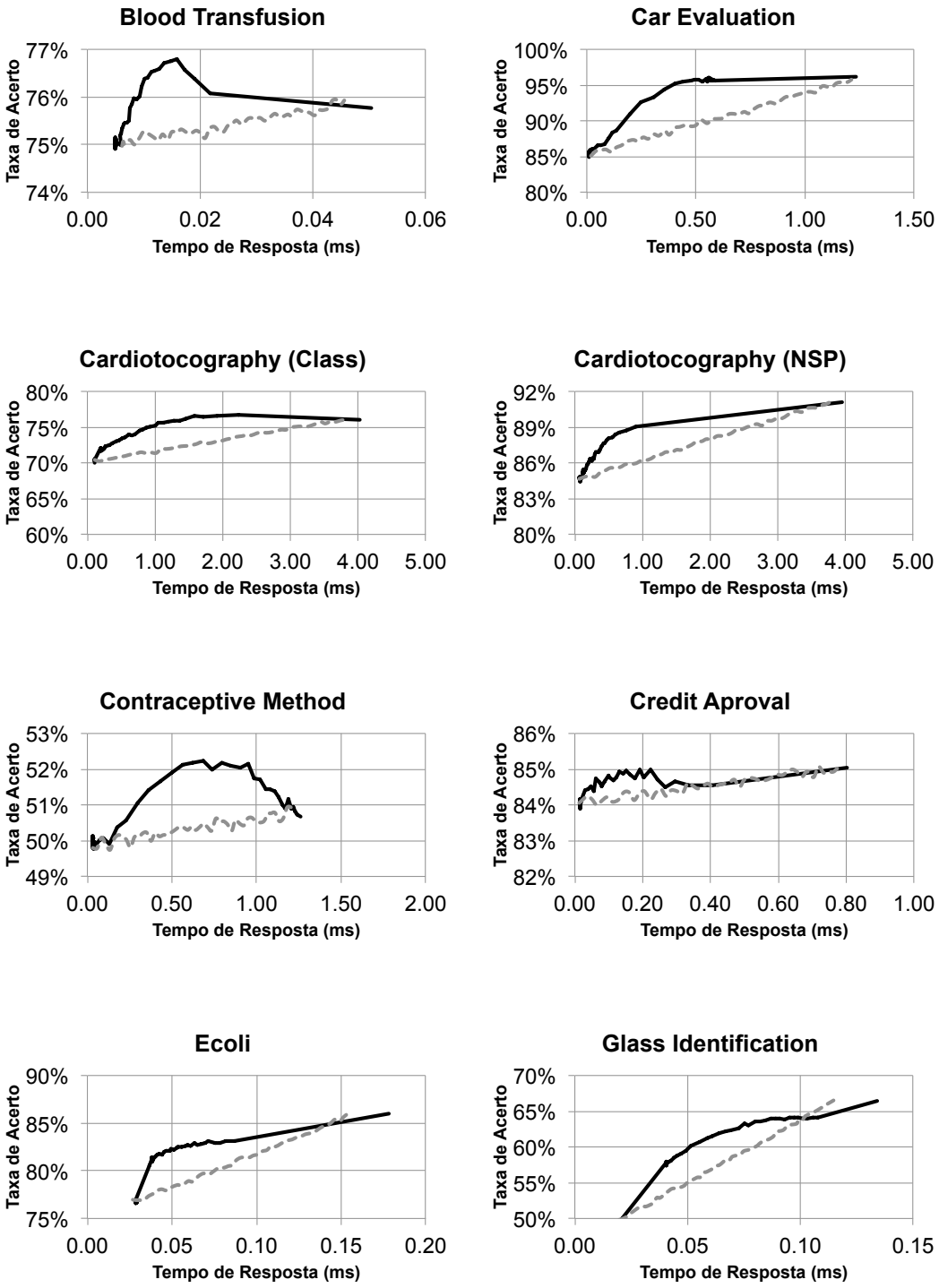


Figura 32: Comparação de desempenho do sistema experimental (linha contínua) com o sistema de escolha aleatória (linha tracejada). (cont.)

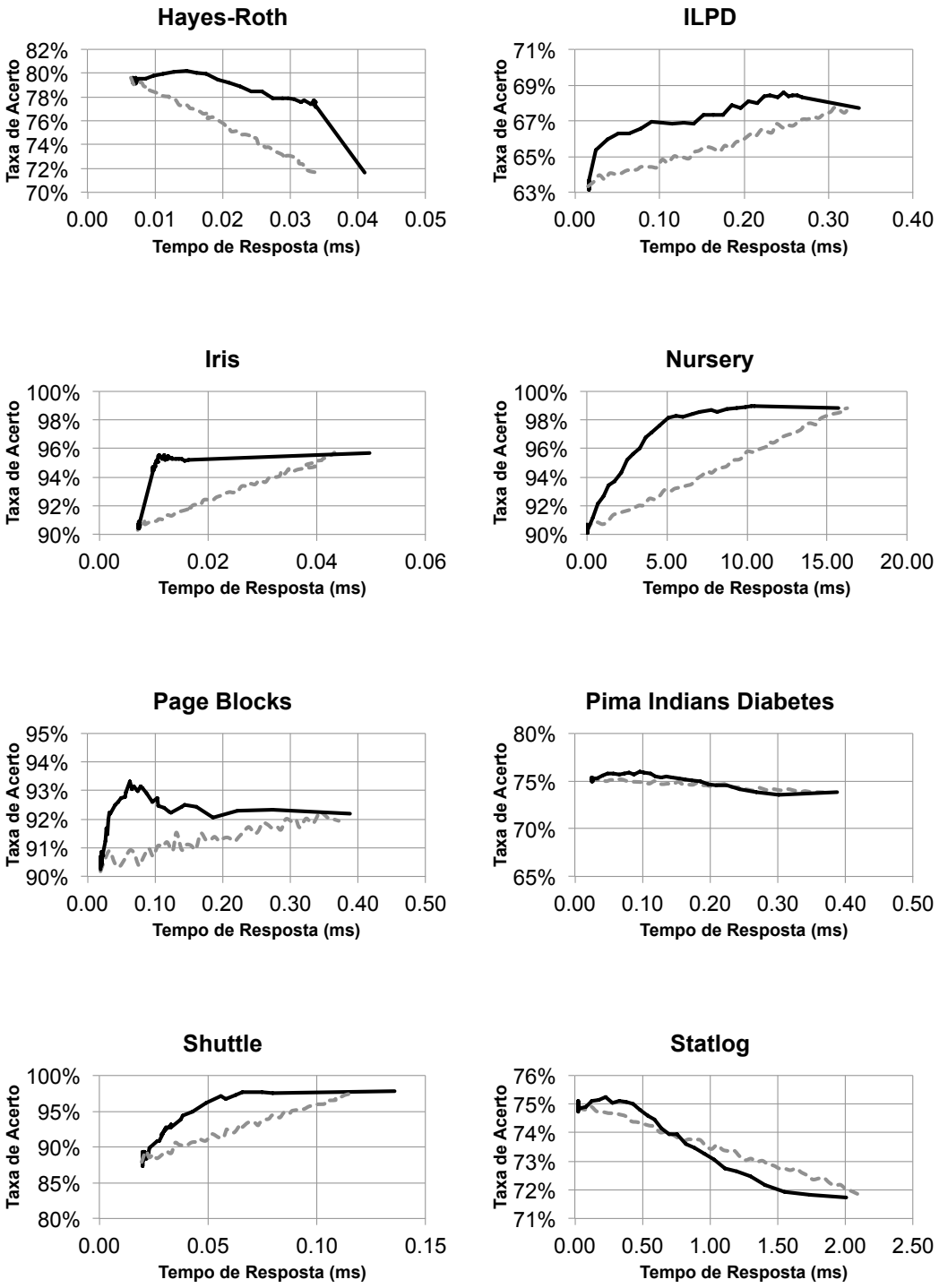


Figura 32: Comparação de desempenho do sistema experimental (linha contínua) com o sistema de escolha aleatória (linha tracejada). (cont.)

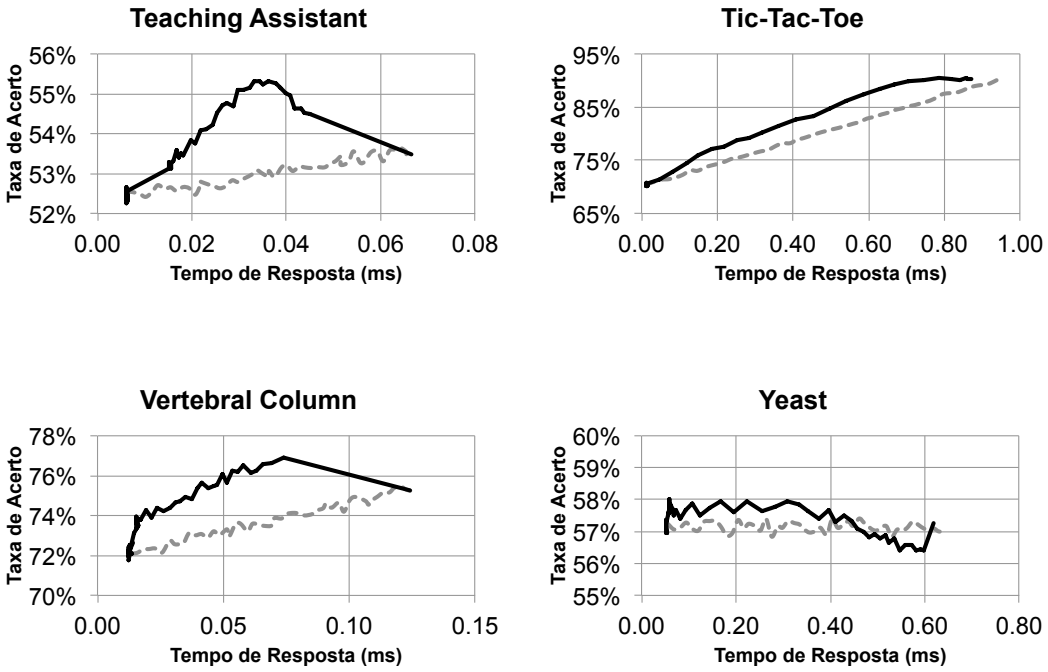


Figura 32: Comparação de desempenho do sistema experimental (linha contínua) com o sistema de escolha aleatória (linha tracejada).

de decisão, permitindo que múltiplos dispositivos possam operar em conjunto, obtendo um desempenho que não seria possível em uma distribuição de carga não informada.

4.6 Aprendizado Incremental

Finalmente, foi realizado um experimento visando medir a evolução da taxa de acerto e do tempo de resposta à medida que o sistema recebe realimentações durante sua execução. Para esta finalidade, é definido um sistema experimental baseado na configuração dada na seção 4.1, mas com as seguintes particularidades:

- O valor do limiar de saída da fila é fixado em $0,7$.
- O valor utilizado para a credibilidade do viés de repetição é $cr_{repetition} =$

$0,01$, ao contrário do valor nulo utilizado nos experimentos sobre o compromisso entre taxa de acerto e tempo de resposta. Contudo, este experimento também não conta com um bloco de auto-análise.

Esta configuração é utilizada em duas análises:

1. Primeiramente, é verificado como o viés de repetição influencia a evolução do desempenho do sistema.
2. Em um segundo caso, é analisado como realimentações recebidas por fontes externas influenciam a evolução de desempenho do sistema.

As seções seguintes mostram como estas duas análises foram realizadas, e os resultados obtidos em cada uma.

4.6.1 Viés de Repetição

Conforme descrito previamente na seção 3.1, o viés de repetição é uma forma de realimentação que reforça a credibilidade das soluções obtidas pelos próprios dispositivos, permitindo que cada um deles aprenda com as soluções obtidas com o auxílio dos demais - em particular, dispositivos mais rápidos podem aprender com os dispositivos mais precisos. Por isso, espera-se que a frequência com que suas soluções são aceitas aumente progressivamente com o uso, reduzindo a utilização dos dispositivos mais lentos, e, consequentemente, reduzindo o tempo médio de resposta do sistema.

Este experimento visa verificar esta influência do viés de desempenho sobre a evolução do sistema. Para isso, o seguinte procedimento é adotado:

1. Treinar o sistema com 70% das amostras, selecionadas de forma arbitrária.

2. Iterar processo abaixo 2000 vezes:

   (a) Embaralhar conjunto de testes, alterando a ordem de suas instâncias;

   (b) Para cada instância $(x, a_{instancia})$ desta lista embaralhada:

      i. Executar o sistema com o conjunto x (Algoritmo 8);

      ii. Verificar se a alternativa retornada $a_{retorno}$ é igual à alternativa $a_{instancia}$ esperada pela instância;

   (c) Medir a taxa de acerto e tempo de resposta.

Note que não há um novo treinamento entre as iterações, diferentemente do procedimento adotado nos experimentos anteriores: o sistema mantém-se o mesmo entre as iterações, mas retém as novas informações recebidas pelas realimentações feitas através do viés de repetição, conforme descrito pelo Algoritmo 8. Dessa maneira, espera-se que o dispositivo mais rápido (o classificador de Bayes) consiga aprender à medida que o *k-NN* soluciona problemas, passando a solucionar estes problemas futuramente. Portanto, espera-se que o tempo de resposta seja reduzido à medida que o número de iterações aumenta.

Como a evolução do sistema depende do conjunto de treinamento inicial, o processo acima é repetido 10 vezes por *dataset*, para extrair a média da taxa de acerto e do tempo de resposta em cada iteração.

A evolução da taxa de acerto e do tempo de resposta em cada *dataset* estão ilustrados na Figura 33. A partir destes gráficos observa-se que na maior parte destes *datasets*, o tempo de resposta cai à medida que as iterações são feitas. Este resultado indica que as soluções obtidas através do classificador de Bayes estão sendo aceitas com maior frequência à medida que as

realimentações oriundas do viés de repetição são geradas. Entretanto, a proporcionalidade da queda do tempo de resposta não é a mesma em cada um deles: como exemplo, a queda observada no *Cardiotocography (Class)* foi de aproximadamente 50%, enquanto no *Iris*, esta diferença é mínima.

Também se observa que a taxa de acerto, apesar de mostrar uma pequena tendência de queda, mantém-se estável na maioria dos *datasets*, quando comparado aos ganhos obtidos no tempo de resposta. Esta queda é mais acentuada no *Cardiotocography (NSP)* (3% de queda no período observado), apesar de ser proporcionalmente pequena em comparação com a queda de mais de 50% no tempo de resposta. Todavia, para casos em que a taxa de acerto é crítica, pode ser mais vantajoso evitar o uso do viés de repetição.

Em uma análise mais pontual, nota-se que no *Nursery*, a queda mais acentuada do tempo de resposta ocorre em um período em que a taxa de acerto se matinha estável. Porém, a partir da 500ª iteração, o uso do viés de repetição causa uma deterioração mais acentuada na taxa de acerto, indesejável em sistemas em que o custo do erro é alto. Neste caso, pode ser vantajoso desabilitar o viés de repetição ao chegar neste ponto crítico.

Deve ser ressaltado que o classificador de Bayes, neste exemplo, realiza seu aprendizado incremental exclusivamente através das soluções obtidas pelo *k-NN*, sejam elas corretas ou não. Assim, existe o risco de deterioração da taxa de acerto do classificador de Bayes à medida que novas realimentações são geradas, já que este dispositivo tende a ganhar mais confiança em informações ruidosas, aumentando a expectativa de certeza de suas soluções, e consequentemente, a frequência com que elas são aceitas.

Estes resultados reforçam a ideia de que dispositivos mais rápidos podem aprender com as soluções obtidas por dispositivos mais lentos, sem que a taxa de acerto se degrade significativamente, e, portanto, também res-

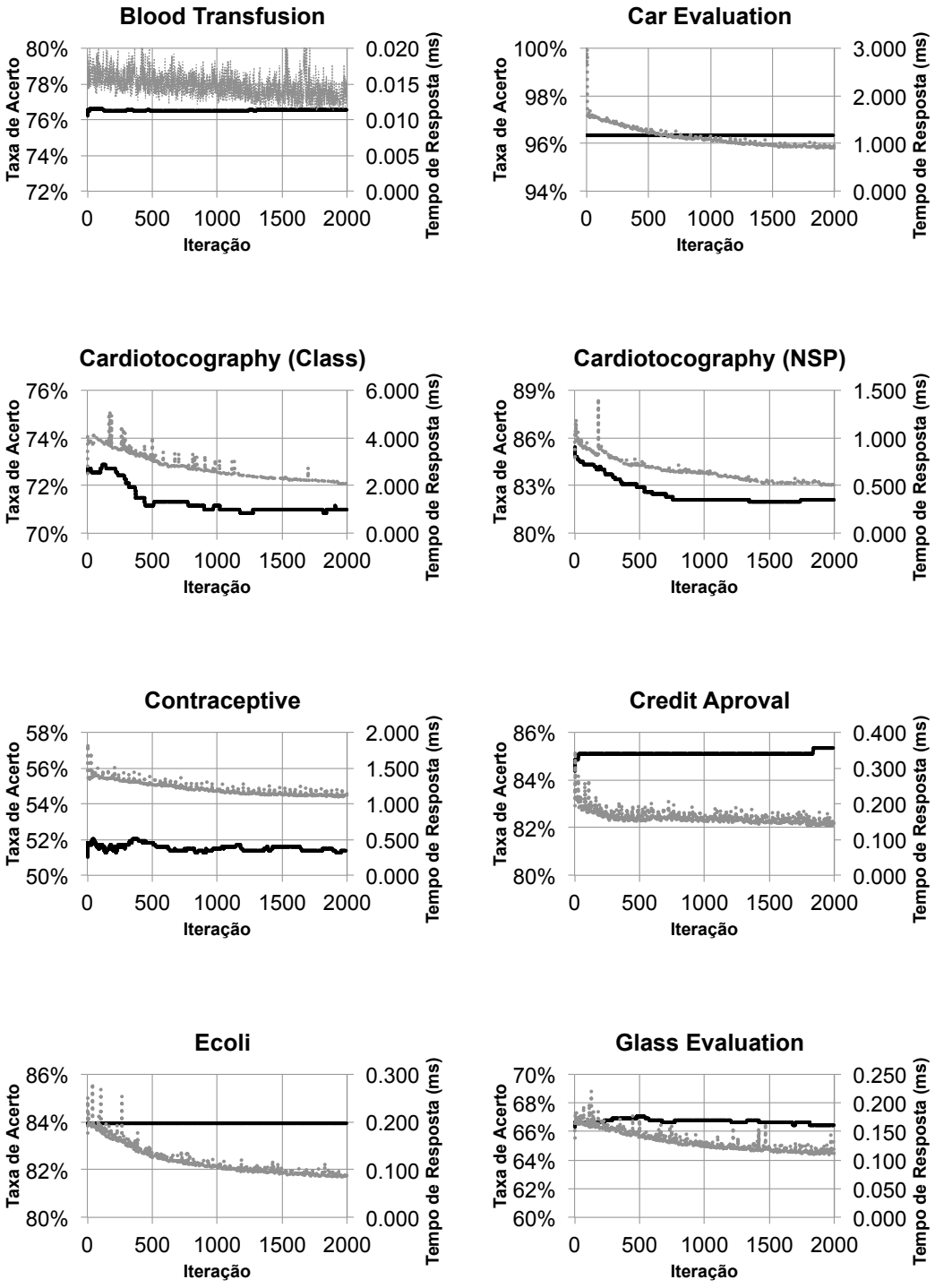


Figura 33: Evolução da taxa de acerto (linha preta) e do tempo de resposta (linha cinza) conforme os dispositivos recebem realimentações originadas no viés de repetição (cont.)

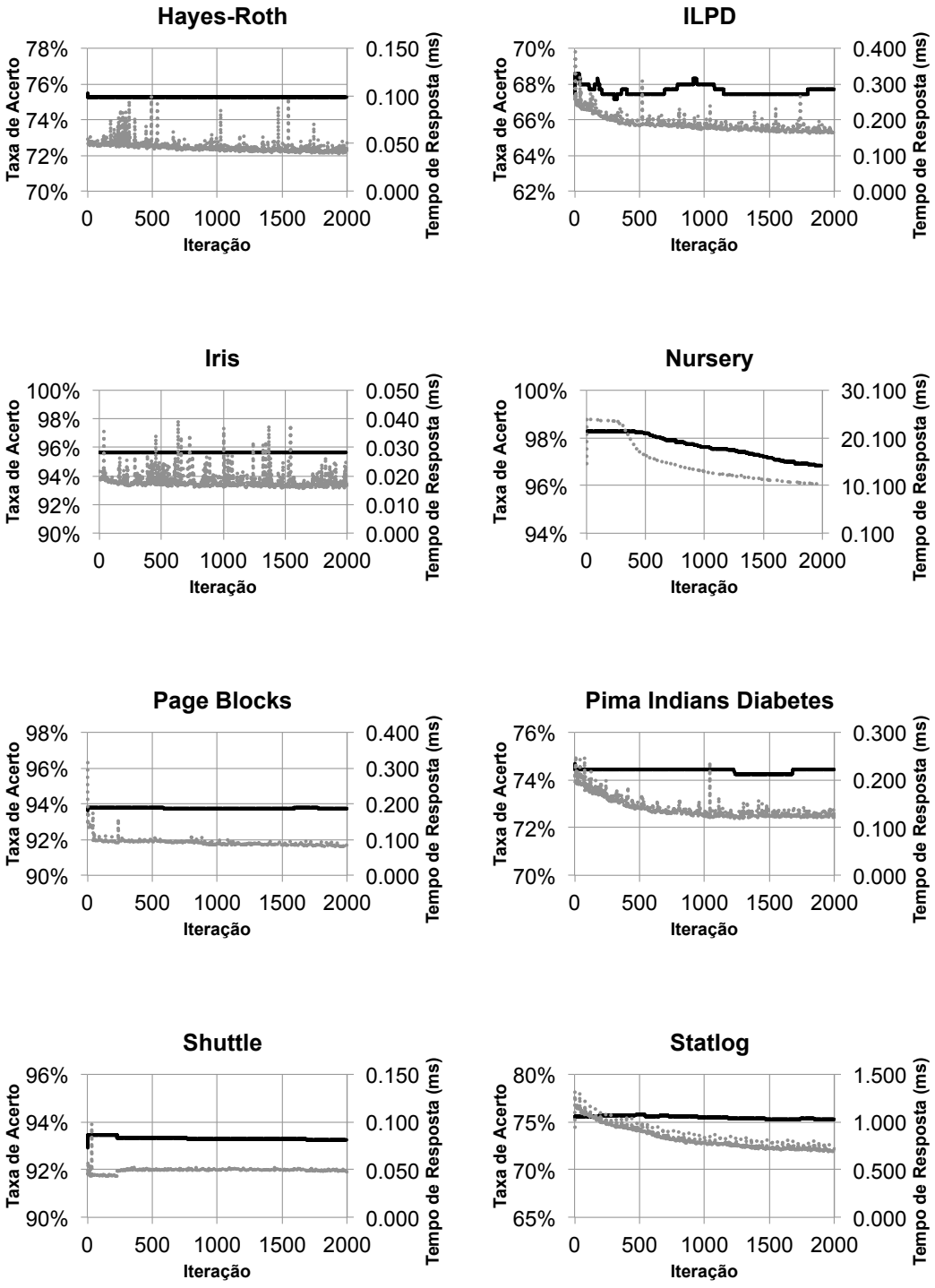


Figura 33: Evolução da taxa de acerto (linha preta) e do tempo de resposta (linha cinza) conforme os dispositivos recebem realimentações originadas no viés de repetição (cont.)

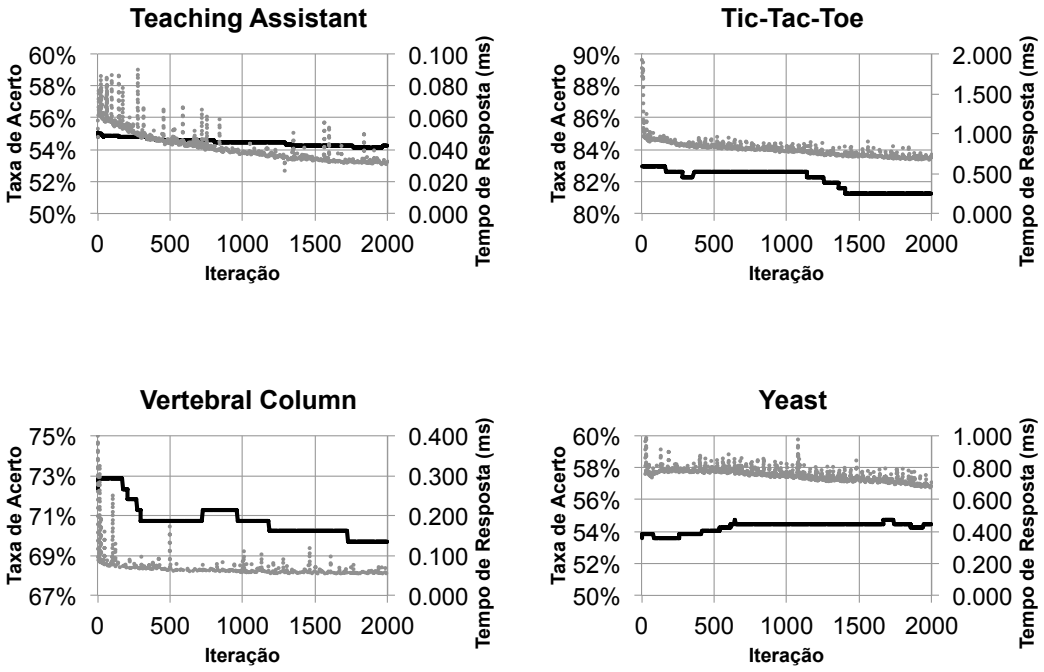


Figura 33: Evolução da taxa de acerto (linha preta) e do tempo de resposta (linha cinza) conforme os dispositivos recebem realimentações originadas no viés de repetição.

paldam o modelo adaptativo inicialmente proposto por Tchemra (TCHEMRA, 2009), estendendo-o para além da tabela de decisão adaptativa. Entretanto, o uso do viés de repetição por um período prolongado se mostra potencialmente prejudicial à taxa de acerto, podendo ser necessário desligá-lo antes que esta queda exceda um limite aceitável, ou até mesmo evitar seu uso, caso o custo do erro seja elevado.

4.6.2 Realimentações Externas

Por fim, para complementar a análise de realimentações, faz-se a verificação de como realimentações originadas através de fontes externas influenciam a evolução da taxa de acerto e do tempo de resposta. Este tipo de realimentação ocorre quando a solução retornada pelo sistema não é aquela esperada pela instância rotulada do *dataset*. Deste experimento, espera-se

que a taxa de acerto cresça à medida que novas informações são recebidas, eventualmente saturando para um valor-limite imposto pelas limitações de cada dispositivo. Enquanto isso, pressupondo que um volume maior de instâncias passe a ser solucionado através de dispositivos mais rápidos, espera-se que o tempo de resposta eventualmente passe a cair, ou que, ao menos, se mantenha estável.

Para esta finalidade, os seguintes passos são utilizados:

1. Construir o sistema experimental com a base de conhecimento inicialmente vazia, i.e., conjunto de treinamento é vazio.
2. Iterar processo abaixo 2000 vezes:
   (a) Embaralhar conjunto de testes, alterando a ordem das instâncias do *dataset*;
   (b) Para cada instância $(x, a_{instancia})$ desta lista embaralhada:
      i. Executar o sistema com o conjunto x;
      ii. Verificar se a alternativa retornada $a_{retorno}$ é igual à alternativa $a_{instancia}$ esperada pela instância;
      iii. Se as alternativas forem diferentes:
         A. Gerar um número real aleatório rnd entre 0 e 1;
         B. Se $rnd < 0,01$, a banca de testes envia duas realimentações:
            - Uma negativa, com $(x, a_{retorno}, v = 0, \beta_-, cr = 1)$, e;
            - Uma positiva, com $(x, a_{instancia}, v = 1, \beta_+, cr = 1)$.
   (c) Mede a taxa de acerto e tempo de resposta.

Assim como no experimento da seção 4.6.1, não há um novo treinamento em cada iteração. Contudo, após o processamento da instância, compara-se

a alternativa retornada com a da instância rotulada. Caso não sejam idênticas, existe 1% de probabilidade de a banca de testes informar este erro ao sistema através de uma realimentação negativa (ou seja, informa que a alternativa retornada não era correta) e uma positiva (informa qual devia ser a alternativa esperada). A credibilidade desta realimentação é $cr = 1$, ordens de grandeza maior que a credibilidade $cr_{repetition}$ do viés de repetição.

Também deve ser ressaltado que o conjunto de treinamento deste sistema é vazio. Assim, sua taxa de acerto e tempo de resposta serão dependentes exclusivamente das informações recebidas através das realimentações. Como a evolução do sistema depende das instâncias que, aleatoriamente, são selecionadas para serem realimentadas, o processo acima é repetido 10 vezes por *dataset*, para extrair a média da taxa de acerto e do tempo de resposta em cada iteração.

A Figura 34 mostra as taxas de acerto e tempo de respostas de cada *dataset* em cada iteração. Conforme esperado, a taxa de acerto cresce rapidamente nas primeiras iterações, em que as realimentações fornecem um conjunto inicial de informações para um sistema que, no princípio, não tinha conhecimento algum sobre como solucionar problemas, e, portanto, era altamente propenso a erros.

Este crescimento em algum momento satura, e, em determinados casos, se deteriora à medida que novas realimentações são recebidas. Esta queda ocorre devido ao aumento da frequência com que as soluções do classificador de Bayes (na maior parte dos *datasets*, mais sujeito a erros) são aceitas. O *Cardiotocography (NSP)* representa o caso mais crítico desta queda, com uma perda de 13% entre a 300ª e a 2000ª iteração.

Entretanto, o aumento da frequência com que as soluções do classificador de Bayes são aceitas também impulsiona uma queda no tempo de resposta,

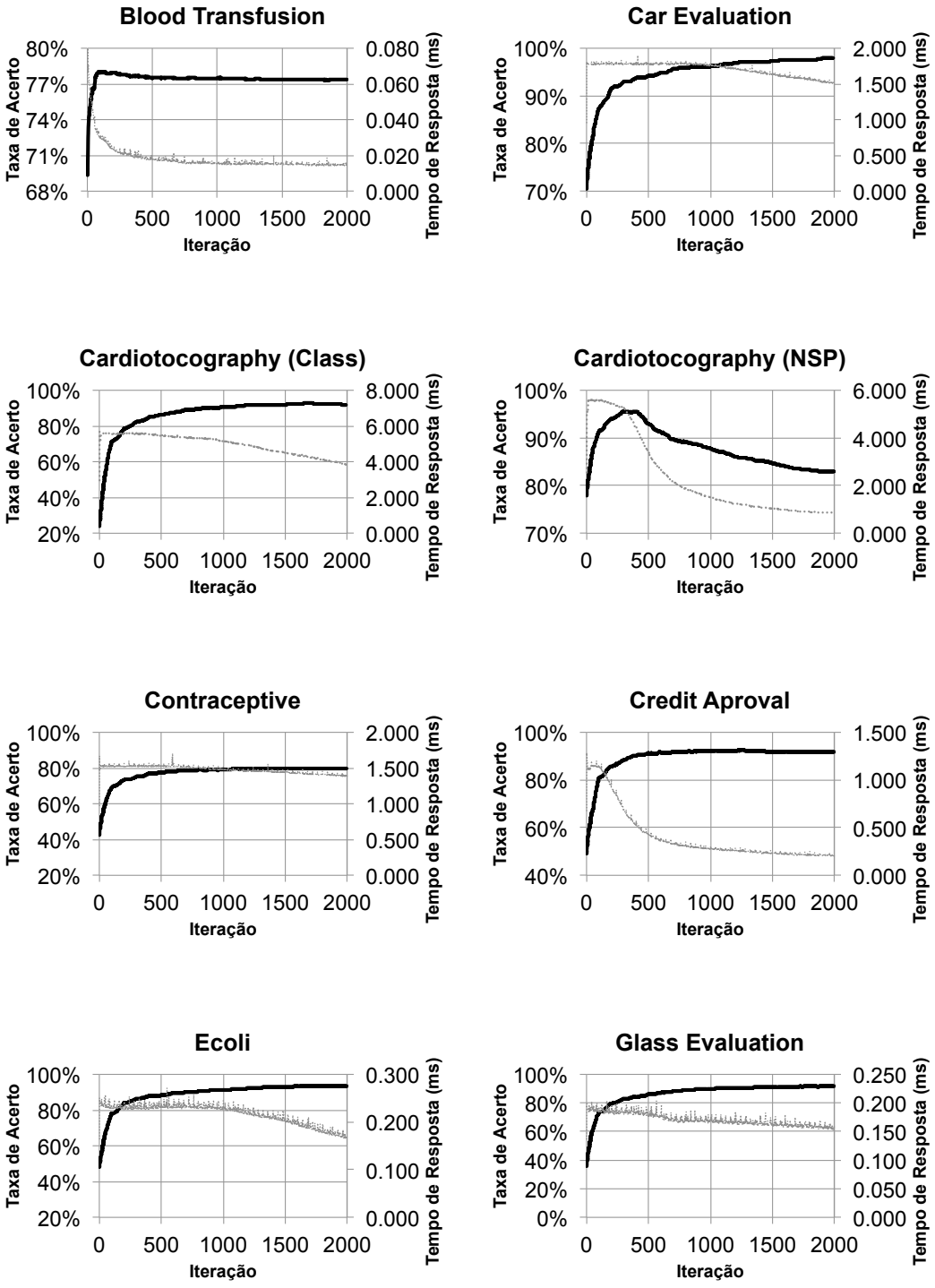


Figura 34: Evolução da taxa de acerto (linha preta) e do tempo de resposta (linha cinza) conforme o sistema aprende com as realimentações recebidas. (cont.)

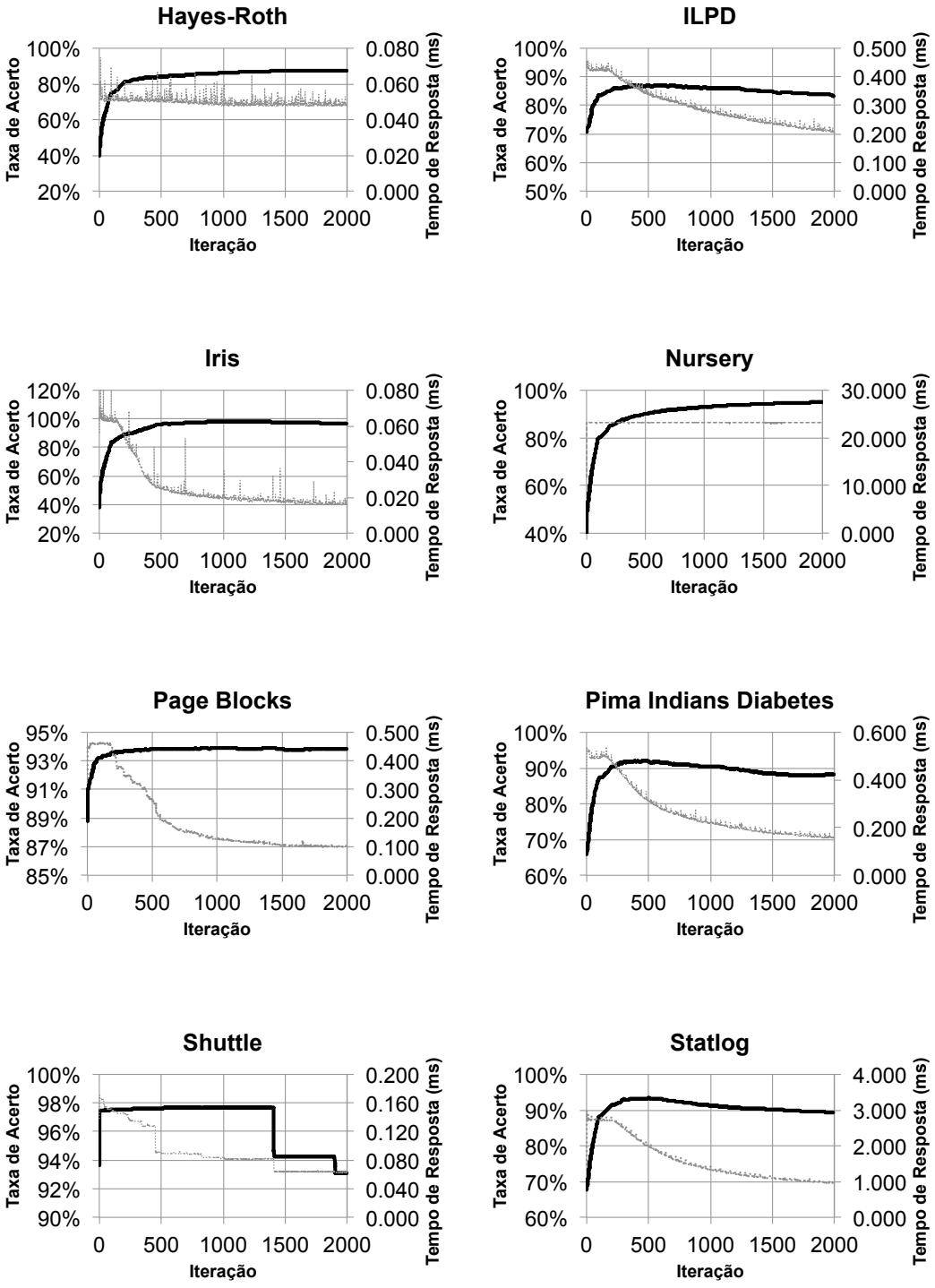


Figura 34: Evolução da taxa de acerto (linha preta) e do tempo de resposta (linha cinza) conforme o sistema aprende com as realimentações recebidas. (cont.)

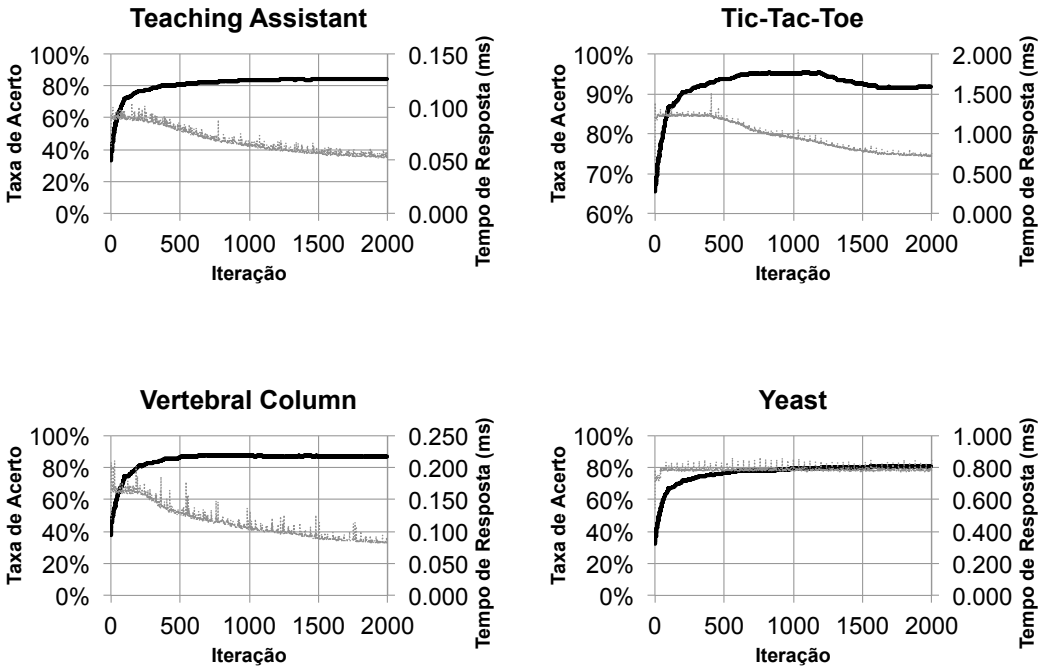


Figura 34: Evolução da taxa de acerto (linha preta) e do tempo de resposta (linha cinza) conforme o sistema aprende com as realimentações recebidas.

uma vez que tal dispositivo mostra-se mais rápido que o *k-NN* em todos os testes realizados. O local em que esta queda é observada varia em cada um dos *datasets*, podendo ocorrer logo nas primeiras iterações (e.g., *Blood Transfusion*) ou tardiamente (e.g., *Ecoli*). Em determinados casos, sequer há uma queda no período observado (e.g., *Yeast*), apesar de apresentarem um ganho na taxa de acerto.

Estes resultados indicam que pode mostrar-se oportuno estender o modelo adaptativo híbrido proposto inicialmente por Tchemra de modo que possa comportar dispositivos de naturezas diferentes, permitindo não só um aprendizado entre dispositivos (através do viés de repetição), mas também através de agentes externos - no caso, simulado pela banca de testes. Entretanto, assim como no experimento da seção 4.6.1, um período prolongado de aprendizado pode ser prejudicial à taxa de acerto do sistema, ainda que seja vantajoso do

ponto de vista do tempo de resposta. Um compromisso deve ser estabelecido para determinar um limite ao aprendizado e evitar que este cenário ocorra.

5 CONCLUSÃO

Neste trabalho está apresentada a formulação de um modelo-base para o desenvolvimento de sistemas multi-dispositivos de apoio à decisão. A aplicação de técnicas adaptativas permite que múltiplos dispositivos possam operar em conjunto, realizando uma distribuição de carga com base na dificuldade encontrada pelos dispositivos em solucionar, individualmente, um determinado problema. Tal dificuldade é estimada através da expectativa de certeza que cada dispositivo atribui às possíveis alternativas a serem adotadas como solução.

O uso das técnicas adaptativas também permite que cada dispositivo aprenda, de forma incremental, com as soluções obtidas através dos demais dispositivos - em particular, permite que dispositivos mais rápidos aprendam com as soluções obtidas através daqueles de maior taxa de acerto (e, em geral, mais lentos), aumentando a frequência com que suas soluções são aceitas, e, consequentemente, diminuindo o tempo médio de resposta à medida que novos problemas são solucionados.

Através deste método de aprendizado, é também possível realimentar os sistemas de apoio à decisão a partir de fontes externas, possibilitando sua manutenção por parte de operadores humanos, ou então, com a ajuda de algum agente inteligente, que seja capaz de avaliar a qualidade das soluções encontradas.

A base conceitual deste trabalho foi originada pela combinação do modelo híbrido da tabela de decisão adaptativa proposta por *Tchemra* (TCHEMRA, 2009) e de conceitos de aprendizagem de máquina mediante incertezas. O tratamento de incertezas é utilizado para que o processo de aprendizagem de cada dispositivo pondere a credibilidade das novas informações recebidas de acordo com a fonte de origem - e.g., outros dispositivos, agentes inteligentes e operadores humanos.

Este trabalho apresenta tanto contribuições teóricas no campo de técnicas adaptativas, como aplicações práticas para a construção de sistemas híbridos de tomada de decisão, conforme apresentado a seguir.

5.1 Contribuições

Para a área de tecnologia adaptativa, este trabalho mostra que o modelo híbrido de tomada de decisão proposto por Tchemra (TCHEMRA, 2009) pode ser estendido para além das tabelas de decisão, englobando outros tipos de dispositivos adaptativos - não necessariamente restritos àqueles guiados por regras. Esta característica viabiliza a transferência de conhecimento entre dispositivos quaisquer, permitindo que cada um deles aprenda com as soluções determinadas pelos demais. O uso desta forma de aprendizado, contudo, requer que os dispositivos tenham capacidade de agregar informações de forma incremental ao longo de sua execução.

De forma complementar, o uso de um modelo de confiança com valores-verdade numéricos ao invés de verdades absolutas (booleanas) permite que o aprendizado possa ocorrer de forma suave, ponderando as informações não só pelo nível de verdade que cada uma possui, mas também pelo quão confiáveis elas são. Logo, novas informações têm influência menor do que aquelas

já utilizadas e validadas há mais tempo, mas se tornam mais confiáveis à medida que vão sendo mais utilizadas. Como este modelo de confiança não está associado a um dispositivo em particular, qualquer método de tomada de decisão capaz de lidar com verdades não absolutas (e.g., lógica difusa ou *fuzzy*) também pode se aproveitar deste formalismo.

Outra contribuição é o uso de um critério para regular a relação de compromisso entre taxa de acerto e tempo de resposta dos sistemas, sob a premissa de que os dispositivos mais lentos devem apresentar taxas de acerto maiores. Experimentos com dados de testes reais mostram que, na maior parte dos casos, as taxas de acerto destes sistemas para um mesmo tempo de resposta são maiores daquelas obtidas por uma simples distribuição de carga não informada. Este é um resultado importante, pois viabiliza uma distribuição mais eficiente dos problemas, restringindo o uso dos dispositivos mais lentos para aqueles mais difíceis.

No âmbito prático, o formalismo proposto neste trabalho permite que novos sistemas de apoio à decisão sejam construídos através do reuso de conceitos previamente estudados, incorporando técnicas aprendizagem de máquina para tratar informações mediante incertezas - um aspecto que não havia sido tratado na *TDAE* original. Assim, novos dispositivos podem ser implementados e integrados ao modelo-base, adequando-os para calcular a expectativa de certeza de cada alternativa ao invés de retornar uma única solução.

Outra contribuição prática consiste na disponibilização de um banco de testes para levantar as curvas de taxa de acerto e de tempo de resposta em função de um limiar, que controle a distribuição de carga. Estas medidas permitem não só escolher um valor ótimo para este limiar, que maximize a taxa de acerto mediante uma limitação do tempo de resposta, mas também comparar os desempenhos obtidos por estes sistemas com os obtidos em

outras técnicas de tomada de decisão.

5.2 Trabalho Futuro

A partir dos resultados alcançados neste trabalho, outros aspectos relevantes ao processo de tomada de decisão podem ser investigados para verificar como influenciam a taxa de acerto e o tempo de resposta, bem como propor formas de minimizar seus efeitos negativos.

Um aspecto não explorado nesta dissertação é a detecção e o tratamento dos desvios de conceito. Ainda que o modelo de confiança proposto não utilize verdades absolutas, o tratamento de conceitos variáveis no tempo requer não apenas um aprendizado incremental, mas também a capacidade de detectar informações obsoletas, que deterioram a qualidade das soluções, e removê-las do sistema. Esta forma de aprendizado é dito *decremental*, e exige que os dispositivos consigam remover informações uma-a-uma - aspecto que, embora ainda pouco explorado no campo de aprendizado de máquina, é considerado no âmbito dos métodos adaptativos, que operam essencialmente através de incrementos, tanto na incorporação como na remoção de informações essenciais às decisões por eles tomadas.

De maneira complementar, outra sugestão de continuidade deste trabalho é o desenvolvimento de um módulo capaz de analisar situações em que o conjunto corrente de possíveis soluções não atende às necessidades do problema por conter muitos casos não solucionados. Como estes casos não possuem um gabarito para verificar qual seriam suas soluções, métodos de aprendizado não supervisionados podem ser utilizados para encontrar padrões nestes casos (e.g., clustering), e inferir se há a necessidade de uma nova alternativa. Tanto a análise de desvio de conceito como a análise de casos sem solução

podem ser vistos como blocos realimentadores, conforme proposto na seção 3.1.

Também pode ser destacado um estudo de novas estratégias de colaboração entre dispositivos, comparando-as com a fila de dispositivos ordenada por tempo resposta, utilizada neste trabalho. Ainda que esta fila permita controlar a relação de compromisso entre taxa de acerto e tempo de resposta, ela não é adaptável, muito menos configurável. Algum tipo de inteligência poderia ser injetada neste processo, usando para isso a adaptatividade para efetuar uma seleção dinâmica do dispositivo mais apropriado para cada problema a ser solucionado, sem precisar executar os dispositivos de forma cega. Como exemplo, a base de conhecimento poderia memorizar não só as decisões tomadas, mas também qual a frequência com que cada dispositivo teve sua solução aceita. Esta informação poderia ser utilizada para definir um mecanismo adaptativo que escolhe o dispositivo com melhor índice de aceitação para cada problema a ser solucionado.

Dentre os aspectos com potencial de aprofundamento, destaca-se uma investigação dos efeitos que um treinamento inicial pode ter sobre a evolução da taxa de acerto e do tempo de resposta. Variáveis tais como a quantidade de instâncias de treinamento e a credibilidade das realimentações recebidas (carga inicial, viés de repetição e do usuário) podem ser estudadas individualmente para verificar como elas se relacionam e influem na forma como um dispositivo aprende à medida que recebe novas realimentações.

Outra possibilidade é explorar outros critérios de aceitação de uma solução. Neste trabalho, foi utilizada uma função limiar, que simplesmente verifica se a expectativa de certeza de uma solução é superior a um valor de referência. Todavia, outras informações poderiam ser levadas em consideração para definir tal critério, como exemplo, uma comparação entre os elementos do ve-

tor da expectativa de certeza. Também deve ser notado que o critério adotado neste trabalho opera apenas na expectativa de certeza do dispositivo corrente - uma possível alternativa seria agregar todos os resultados dos dispositivos previamente utilizados, e verificar se eles têm alguma alternativa em comum.

De forma similar, outro ponto a ser explorado é a ação a ser tomada quando nenhum dispositivo obtém uma solução aceita pelo critério adotado. Ao invés de retornar a solução obtida pelo último dispositivo, outras ações poderiam ser tomadas, como exemplo, utilizar um sistema de votação entre os dispositivos.

Finalmente, sugere-se a adequação de outros tipos de dispositivos e verificar como cada um deles influencia a taxa de acerto e o tempo de resposta dos sistemas de tomada de decisão. Algoritmos que apresentem capacidade de inserir e remover informações de forma incremental, e.g., *ITI* (*Incremental Tree Inducer*) (UTGOFF; BERKMAN; CLOUSE, 1997), têm potencial para estudo futuro, uma vez que estariam de acordo não só com as especificações necessárias para o atual modelo de desenvolvimento proposto, mas também para um suposto formalismo que trate desvios de conceito.

REFERÊNCIAS

AAMODT, A.; PLAZA, E. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, IOS Press, v. 7, p. 39–59, March 1994. ISSN 0921-7126.

ACKLAM, P. J. *An algorithm for computing the inverse normal cumulative distribution function*. 2003. Acesso em: 27 de março de 2013. Disponível em: <http://home.online.no/~pjacklam/notes/invnorm/>.

AGASANDJAN, G. A. Automata with a variable structure. In: *Doklady Akademii Nauk SSSR*. [S.l.: s.n.], 1967. v. 174, p. 529–530.

AHA, D. W.; KIBLER, D. Instance-based learning algorithms. In: *Machine Learning*. [S.l.: s.n.], 1991. p. 37–66.

AKAHO, S. Conditionally independent component extraction for naive bayes inference. In: *Proceedings of the International Conference on Artificial Neural Networks*. [S.l.]: Springer-Verlag, 2001. (ICANN '01), p. 535–540. ISBN 3-540-42486-5.

ALFENAS, D. A.; PEREIRA-BARRETTO, M. R. Adaptatividade em robôs sociáveis: uma proposta de um gerenciador de diálogos. In: *6º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2012.

ALFENAS, D. A. et al. Sistemas de markov adaptativos: Formulação e plataforma de desenvolvimento. In: *6º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2012.

ANDERSON, J. R. *The architecture of cognition*. Cambridge, MA, USA: Harvard University Press, 1983.

ANKERST, M. et al. Optics: ordering points to identify the clustering structure. *ACM SIGMOD international conference on Management of data*, ACM, New York, NY, USA, v. 28, p. 49–60, June 1999. ISSN 0163-5808.

ARNOTT, D. Decision support systems evolution: framework, case study and research agenda. *European Journal of Information Systems*, v. 13, n. 4, p. 247–259, December 2004.

ARNOTT, D.; PERVAN, G. A critical analysis of decision support systems research. *Journal of Information Technology*, v. 20, n. 2, p. 67–87, June 2005.

BABENKO, B.; YANG, M.-H.; BELONGIE, S. A family of online boosting algorithms. In: *Online Learning for Computer Vision Workshop*. [S.l.: s.n.], 2009. p. 1346–1353.

BAKKEN, B. E. *Learning and Transfer of Understanding in Dynamic Decision Environments*. Tese (Doutorado) — Massachusetts Institute of Technology, 1993.

BALAKRISHNAN, J. D.; RATCLIFF, R. Testing models of decision making using confidence ratings in classification. *J Exp Psychol Hum Percept Perform*, v. 22, n. 3, p. 615–633, June 1996. ISSN 0096-1523.

BASSETO, B. A.; NETO, J. J. A stochastic musical composer based on adaptative algorithms. In: *Proceedings of the 6th Brazilian Symposium on Computer Music - SBC&M99*. Rio de Janeiro: [s.n.], 1999. p. 105–113.

BECHARA, A. The role of emotion in decision-making: Evidence from neurological patients with orbitofrontal damage. *Brain and Cognition*, v. 55, n. 1, p. 30–40, 2003.

BECKMANN, N. et al. The r*-tree: an efficient and robust access method for points and rectangles. In: *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 1990. (SIGMOD '90), p. 322–331. ISBN 0-89791-365-5.

BELL, D. E.; RAIFFA, H.; TVERSKY, A. Descriptive, normative, and prescriptive interactions in decision making. In: *Decision making: descriptive, normative, and prescriptive interactions*. [S.l.]: Cambridge University Press, 1988. p. 9–30.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, ACM, New York, NY, USA, v. 18, p. 509–517, September 1975. ISSN 0001-0782.

BOULLÉ, M. A robust method for partitioning the values of categorical attributes. In: *Extraction et Gestion des Connaissances*. [S.l.]: Cépaduès-Éditions, 2004. (Revue des Nouvelles Technologies de l'Information), p. 173–184. ISBN 2-85428-633-2.

BROWN, L. D.; CAI, T. T.; DASGUPTA, A. Interval estimation for a binomial proportion. *Statistical Science*, v. 16, p. 101–133, 2001.

CAMARGO, R.; RAUNHEITTE, L. Convolução adaptativa. In: *5º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2011. p. 6–11.

CARUANA, R.; NICULESCU-MIZIL, A. An empirical comparison of supervised learning algorithms. In: *Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM, 2006. p. 161–168. ISBN 1-59593-383-2.

CASTRO JR., A. A.; NETO, J. J.; PISTORI, H. Determinismo em autômatos de estados finitos adaptativos. *Revista IEEE América Latina*, v. 5, p. 515–521, Novembro 2007. ISSN 1548-0992.

CEREDA, P. R. M.; NETO, J. J. Mineração adaptativa de dados: Aplicação à identificação de indivíduos. In: *6º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2012.

CHEN, Y. A monte carlo study on impacts of the size of subsample catch on estimation of fish stock parameters. *Fisheries Research*, v. 26, n. 3, p. 207–223, 1995.

CHITTKA, L.; SKORUPSKI, P.; RAINE, N. E. Speed-accuracy tradeoffs in animal decision making. *Trends in Ecology & Evolution*, v. 24, n. 7, p. 400–407, July 2009.

CHRISTIANSEN, H. Recognition of generative languages. In: *on Programs as data objects*. New York, NY, USA: Springer-Verlag New York, Inc., 1985. p. 63–81. ISBN 0-387-16446-4.

____. A survey of adaptable grammars. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 25, n. 11, p. 35–44, November 1990. ISSN 0362-1340.

CHUA, E. A. et al. Dissociating confidence and accuracy: Functional magnetic resonance imaging shows origins of the subjective memory experience. *Journal of Cognitive Neuroscience*, Massachusetts Institute of Technology Press, v. 16, n. 7, p. 1131–1142, 2004.

CLEARY, J. G.; TRIGG, L. E. K*: An instance-based learner using an entropic distance measure. In: *In Proceedings of the 12th International Conference on Machine Learning*. [S.l.]: Morgan Kaufmann, 1995. p. 108–114.

CORTES, C.; VAPNIK, V. Support-vector networks. In: *Machine Learning*. [S.l.: s.n.], 1995. v. 20, n. 3, p. 273–297.

COSTA, E. R.; HIRAKAWA, A. R.; NETO, J. J. An adaptive alternative for syntactic pattern recognition. In: *In Proceeding of 3rd International Symposium on Robotics and Automation, ISRA*. Toluca, Mexico: [s.n.], 2002. p. 409–413.

CRIVELARO, F. J. B. C. V.; ROCHA, R. L. de Azevedo da. Proposta de modelo adaptativo para geração de contextos na recomendação de locais. In: *6º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2012.

CROZIER, R.; RANYARD, R. Cognitive process models and explanations of decision making. In: *Decision Making: Cognitive Models and Explanations*. Saffron Walden, UK: Routledge, 1997. p. 5–20.

DING, D.; LI, Q.; YANG, J. Multimedia data integration and navigation through mediaview: Implementation, evolution and utilization. In: LEE, Y. et al. (Ed.). *Database Systems for Advanced Applications*. [S.l.]: Springer Berlin / Heidelberg, 2004, (Lecture Notes in Computer Science, v. 2973). p. 263–271.

DOUGHERTY, J.; KOHAVI, R.; SAHAMI, M. Supervised and unsupervised discretization of continuous features. In: *Machine Learning: Proceedings of the Twelfth International Conference*. [S.l.]: Morgan Kaufmann, 1995. p. 194–202.

EINHORN, H. J.; HOGARTH, R. M. Confidence in judgment: Persistence of the illusion of validity. *Psychological Review*, v. 85, n. 5, p. 395–416, September 1978.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 1996. p. 226–231.

EVANGELISTA, P. F.; EMBRECHTS, M. J.; SZYMANSKI, B. K. Taming the curse of dimensionality in kernels and novelty detection. In: *Applied Soft Computing Technologies: The Challenge of Complexity*. [S.l.]: Springer Verlag, 2006. p. 431–444.

FRANK, A.; ASUNCION, A. *UCI Machine Learning Repository*. 2011. Acesso em: 27 de março de 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

FRANK, H.; ALTHOEN, S. C. *Statistics: Concepts and Applications Workbook*. [S.l.]: Cambridge University Press, 1994. ISBN 978-0521445542.

GENTNER, D. Structure mapping - a theoretical framework for analogy. *Cognitive Science*, v. 7, p. 155–170, 1983.

GÖKER, M. H.; ROTH-BERGHOFER, T. Development and utilization of a case-based help-desk support system in a corporate environment. In: *Case-Based Reasoning and Development*. [S.l.: s.n.], 1999. p. 132–146.

GRIFFIN, D.; TVERSKY, A. The weighing of evidence and the determinants of confidence. *Cognitive Psychology*, v. 24, n. 3, p. 411–435, July 1992.

HENDRICKX, I.; BOSCH, A. van den. Hybrid algorithms with instant-based classification. In: *Machine learning - ECML 2005*. [S.l.]: Springer, 2005. p. 158–169.

HIRAKAWA, A. R.; SARAIVA, A. M.; CUGNASCA, C. E. Autômatos adaptativos aplicados em automação e robótica. *Revista IEEE América Latina*, v. 5, n. 7, p. 539–543, November 2007. ISSN 1548-0992.

HOLTON, G. A. *Monte Carlo Method*. 2004. Acesso em: 27 de março de 2013. Disponível em: <http://www.riskglossary.com/link/monte_carlo_method.htm>.

HUGHES, M. L.; SHANK, R. M.; STEIN, E. S. *Decision Tables*. Wayne, Pennsylvania: MDI Publications, 1968.

JESUS, L. de et al. Adaptools 2.0: Aspectos de implementação e utilização. *Revista IEEE América Latina*, v. 5, n. 7, p. 527–532, November 2007. ISSN 1548-0992.

KAHNEMAN, D.; TVERSK, A. Prospect theory: An analysis of decision under risk. *Econometrica*, v. 47, n. 2, p. 263–292, March 1979.

KEEN, P. G. W. *Decision Support Systems: An Organizational Perspective*. [S.l.]: Addison-Wesley, 1978. ISBN 0201036673.

KELLER, J. M.; GRAY, M. R.; GIVENS JR., J. A. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 15, p. 580–585, 1985.

KIBRIYA, A. M.; FRANK, E. An empirical comparison of exact nearest neighbour algorithms. In: *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer-Verlag, 2007. (PKDD 2007), p. 140–151. ISBN 978-3-540-74975-2.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the 14th international joint conference on Artificial intelligence*. [S.l.]: Morgan Kaufmann Publishers Inc., 1995. (IJCAI'95, v. 2), p. 1137–1143. ISBN 1-55860-363-8.

KOLODNER, J. L. Improving human decision making through case-based decision aiding. *AI Mag.*, American Association for Artificial Intelligence, Menlo Park, CA, USA, v. 12, p. 52–68, April 1991. ISSN 0738-4602.

KOTSIANTIS, S.; KANELLOPOULOS, D. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, v. 32, n. 1, p. 47–58, 2006.

KUNCHEVA, L. I. *Using Control Charts for Detecting Concept Change in Streaming Data*. [S.l.], 2009.

LANE, D. M. *Percentiles*. 2010. Acesso em: 27 de março de 2013. Disponível em: <http://cnx.org/content/m10805/latest/>.

LEAKE, D. B.; KINLEY, A.; WILSON, D. Learning to improve case adaptation by introspective reasoning and cbr. In: *Proceedings of the First International Conference on Case-Based Reasoning*. [S.l.]: Springer-Verlag, 1995. p. 229–240.

LI, C.; LI, H. A survey of distance metrics for nominal attributes. *Journal of Software*, v. 5, n. 11, 2010.

LIU, W.; CHAWLA, S. 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*. Berlin, Heidelberg: Springer-Verlag, 2011. (PAKDD'11, v. 2), p. 345–356. ISBN 978-3-642-20846-1.

MARKOVITCH, S.; SCOTT, P. The role of forgetting in learning. In: *Proceedings of The Fifth International Conference on Machine Learning*. [S.l.: s.n.], 1988. p. 459–465.

MENEZES, C. E. D. de; NETO, J. J. Um método híbrido para a construção de etiquetadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos. In: *Anais da Conferencia Iberoamericana en Sistemas, Cibernética e Informática*. Orlando, Florida: [s.n.], 2002. p. 19–21.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw Hill, 1997. ISBN 0-07-042807-7.

MULLEN, J. D. *Decision Making: Its Logic and Practice*. [S.l.]: Rowman & Littlefield Publishers, 1990. ISSN 08476-76196.

MYERSON, R. B. *Game Theory: Analysis of Conflic*. [S.l.]: Harvard University Press, 1991. ISBN 0674341163.

NETO, J. J. Uma solução adaptativa para reconhecedores sintáticos. In: *Anais EPUSP - Engenharia de Eletricidade*. [S.l.: s.n.], 1988. (B, v. 1), p. 645–657.

NETO, J. J. *Contribuições à metodologia de construção de compiladores*. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, 1993.

____. 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*. [S.l.: s.n.], 2001. (Springer-Verlag), p. 234–250.

____. Um levantamento da evolução da adaptatividade e da tecnologia adaptativa. *Revista IEEE América Latina*, v. 5, n. 7, p. 496–505, November 2007. ISSN 1548-0992.

NETO, J. J.; MAGALHÃES, M. E. S. Reconhecedores sintáticos - uma alternativa didática para uso em cursos de engenharia. In: *XIV Congresso Nacional de Informática*. [S.l.: s.n.], 1981. p. 171–181.

OUYANG, D.; LI, D.; LI, Q. Cross-validation and non-parametric k nearest-neighbour estimation. *Econometrics Journal*, v. 9, n. 3, p. 448–471, 2006.

PARSLOE, E.; WRAY, M. J. *Coaching and Mentoring: Practical Methods to Improve Learning*. [S.l.]: Kogan Page Publishers, 2000. ISBN 0749431180.

PISTORI, H. *Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações*. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, 2003.

PISTORI, H.; NETO, J. J.; PEREIRA, M. C. Adaptive non-deterministic decision trees: General formulation and case study. *INFOCOMP Journal of Computer Science*, 2006.

PLOUS, S. *The Psychology of Judgment and Decision Making*. 1st. ed. [S.l.]: McGraw-Hill, 1993. ISBN 978-0070504776.

POUNDSTONE, W. *Prisoner's Dilemma: John Von Neumann, Game Theory and the Puzzle of the Bomb*. 1st. ed. New York, NY, USA: Doubleday, 1992. ISBN 0385415672.

QUINLAN, J. R. Induction of decision trees. *Machine Learning*, v. 1, p. 81–106, March 1986.

____. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, v. 4, p. 77–90, March 1996.

RAJARAMAN, V. *Analysis and Design of Information Systems*. 2nd. ed. [S.l.]: PHI Learning Pvt. Ltd, 2004.

ROCHA, R. L. de Azevedo da; NETO, J. J. Autômato adaptativo, limites e complexidade em comparação com máquina de turing. In: *Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000*. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia: [s.n.], 2000. p. 33–48.

SAATY, T. L.; VARGAS, L. G. *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*. [S.l.]: Springer, 2000.

SABALIAUSKAS, J. A.; ROCHA, R. L. de Azevedo da. Projeto e implementação de uma linguagem de programação para a execução de algoritmos adaptativos. In: *5º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2011. p. 91–96.

SAFFARI, A. et al. On-line random forests. In: *3rd IEEE ICCV Workshop on On-line Computer Vision*. [S.l.: s.n.], 2009. p. 1393–1400.

SALOMAA, A. On finite automata with a time-variant structure. *Information and Control*, v. 13, p. 85–98, 1968.

SANFEY, A. G. et al. The neural basis of economic decision-making in the ultimatum game. *Science*, IOS Press, v. 300, p. 1755–1758, June 2003.

SANTIBAñEZ, M. R. F.; NETO, J. J. Um sistema para ensino da tecnologia adaptativa. In: *5º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2011. p. 16–24.

SCHANK, R. C. *Dynamic memory; a theory of reminding and learning in computers and people*. [S.l.]: Cambridge University Press, 1982.

SCHOONJANS, F.; BACQUER, D. D.; SCHMID, P. Estimation of population percentiles. *Epidemiology*, v. 22, n. 5, p. 750–751, 2011. ISSN 1044-3983.

SCHULTZKIE, L. *Percentiles and More Quartiles*. 1998. Acesso em: 27 de março de 2013. Disponível em: <http://regentsprep.org/REgents/math-/ALGEBRA/AD6/quartiles.htm>.

SHAW, W. T. Sampling student?s t distribution ? use of the inverse cumulative distribution function. *Journal of Computational Finance*, v. 9, n. 4, p. 37–73, 2006.

SHEPARD, D. A two-dimensional interpolation function for irregularly-spaced data. In: *Proceedings of the 1968 23rd ACM national conference*. New York, NY, USA: ACM, 1968. (ACM '68), p. 517–524.

SILVA, S. R. B. da. *Software adaptativo: método de projeto, representação gráfica e implementação de linguagem de programação*. Dissertação (Mestrado) — Escola Politécnica, Universidade de São Paulo, São Paulo, 2011.

SMYTH, B.; CUNNINGHAM, P. Déjà vu: A hierarchical case-based reasoning system for software design. In: *European Conference on Artificial Intelligence*. [S.l.: s.n.], 1992. p. 587–589.

SMYTH, B.; KEANE, M. T. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Montreal, Quebec: [s.n.], 1995. p. 377–382.

SOL, H. G.; TAKKENBERG, C. A. T.; ROBBÉ, P. F. de V. Expert systems and artificial intelligence in decision support systems. In: *In Proceedings of the Second Mini Euroconference*. [S.l.]: Springer, 1985. p. 17–20.

SPRAGUE, R. H.; CARLSON, E. D. *Building Effective Decision Support Systems*. [S.l.]: Prentice Hall Professional Technical Reference, 1982. ISBN 0130862150.

STANGE, R. L.; NETO, J. J. Aprendizagem incremental usando tabelas decisão adaptativas. In: *5º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2011. p. 35–42.

TCHEMRA, A. H. *Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério*. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, 2009.

UTGOFF, P. E.; BERKMAN, N. C.; CLOUSE, J. A. Decision tree induction based on efficient tree restructuring. *Machine Learning*, Kluwer Academic Publishers, Hingham, MA, USA, v. 29, p. 5–44, October 1997. ISSN 0885-6125.

VALLE, L. E. C. D.; ROCHA, R. L. de Azevedo da. Adaptatividade em imagens de satélite para previsão do tempo. In: *5º Workshop de Tecnologia Adaptativa*. [S.l.: s.n.], 2011. p. 72–78.

VOULGARIS, Z.; MAGOULAS, G. D. Extensions of the k nearest neighbour methods for classification problems. In: *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*. Anaheim, CA, USA: ACTA Press, 2008. (AIA '08), p. 23–28. ISBN 978-0-88986-710-9.

WEBB, G. I.; BOUGHTON, J. R.; WANG, Z. Not so naive bayes: Aggregating one-dependence estimators. In: *Machine Learning*. [S.l.: s.n.], 2005. p. 5–24.

WESTBROOK, J. I.; GOSLING, A. S.; COIERA, E. W. The impact of an online evidence system on confidence in decision making in a controlled setting. *Medical Decision Making*, v. 25, n. 2, p. 178–185, 2005.

WITTEN, I. H.; FRANK, E. *Data mining: practical machine learning tools and techniques*. San Francisco, CA, USA: Morgan Kaufmann, 2005. ISBN 0120884070.

WU, X. et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, Springer-Verlag New York, Inc., New York, NY, USA, v. 14, n. 1, p. 1–37, dez. 2007. ISSN 0219-1377.

YANG, Y.; WEBB, G. I. A comparative study of discretization methods for naive-bayes classifiers. In: *In Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop*. [S.l.: s.n.], 2002. p. 159–173.

ZHANG, Y. et al. Efficient rank based knn query processing over uncertain data. In: *Data Engineering (ICDE)*. [S.l.]: IEEE, 2010. p. 28–39. ISBN 978-1-4244-5444-0.

APÊNDICE A – PERCENTIL DA DISTRIBUIÇÃO NORMAL

Em estatística, *percentil* é uma medida que estima a frequência com que as observações de uma variável fica abaixo de um valor de referência (SCHULTZKIE, 1998). O percentil é frequentemente utilizado em diversos ramos de pesquisa como forma de avaliar a posição de uma observação em relação às demais. Como exemplo, avaliações acadêmicas utilizam o percentil para ranquear cada indivíduo de acordo com sua nota.

Formalmente, não há uma definição universalmente aceita para o significado de percentil (LANE, 2010; SCHOONJANS; BACQUER; SCHMID, 2011). Abaixo, seguem duas interpretações comumente utilizadas:

- O k-ésimo percentil de uma amostra é o menor valor z que é *maior* do que k por cento das observações em uma determinada variável.
- O k-ésimo percentil de uma amostra é o menor valor z que é *maior ou igual* do que k por cento das observações em uma determinada variável.

Apesar de sutis, estas diferenças podem levar a resultados discrepantes, especialmente para amostras pequenas. No entanto, estes resultados devem convergir à medida que o número de observações aumenta (SCHOONJANS; BACQUER; SCHMID, 2011). Para padronização, as fórmulas a serem apresentadas adiante utilizam a primeira interpretação como referência.

Para o caso específico de uma distribuição normal padrão (i.e., distribuição normal que apresenta média 0 e desvio padrão 1), o valor do $(100 \cdot p)$-ésimo percentil z_p é calculado pela *função quantil* (i.e., inversa da função distribuição acumulada - FDA) da distribuição normal, representada por $\phi^{-1}(p)$. Esta função é definida como:

$$\phi^{-1}(p) = \sqrt{2}\,\mathrm{erf}^{-1}(2p-1) \tag{A.1}$$

onde $\mathrm{erf}(x)$ é a função erro:

$$\mathrm{erf}(x) = \frac{2}{\pi}\int_0^p e^{-t^2}\mathrm{d}t \tag{A.2}$$

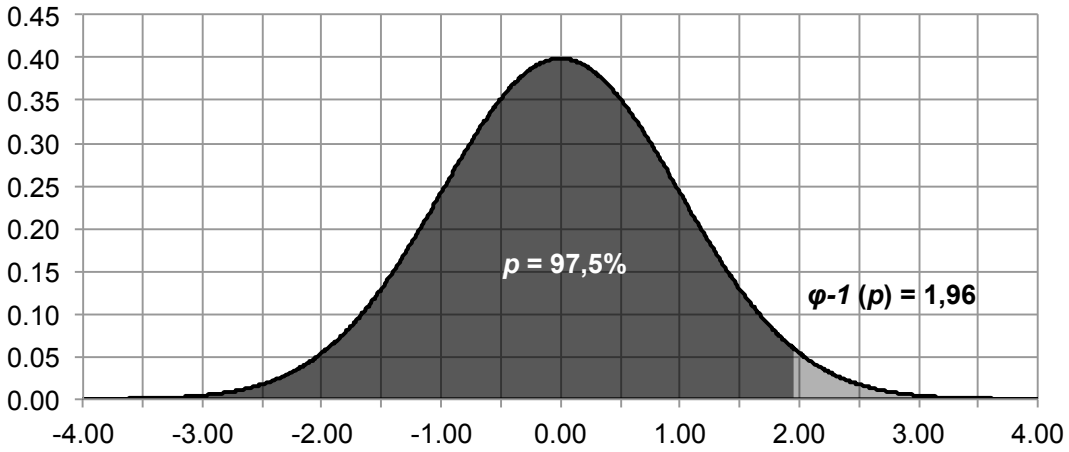


Figura 35: Curva da função densidade de probabilidade da distribuição normal padrão. Para $p = 0,975$, o valor do percentil correspondente é $z_p \approx 1,96$, i.e., $97,5\%$ das observações ficam abaixo de $1,96$.

A inversa da função erro utilizada por $\phi^{-1}(p)$ é dada por:

$$\mathrm{erf}^{-1}(z) = \sum_{k=0}^{\infty}\frac{c_k}{2k+1}\left(\frac{\sqrt{\pi}}{2}z\right)^{2k+1} \tag{A.3}$$

tal que $c_0 = 1$, e:

$$c_k = \sum_{m=0}^{k-1}\frac{c_m c_{k-1-m}}{(m+1)(2m+1)} \tag{A.4}$$

Observando estas funções, nota-se que não há uma forma fechada para a inversa da FDA da distribuição normal padrão. Na prática, utilizam-se aproxi-

mações (ACKLAM, 2003), ou então, uma tabela para consultar os valores mais comuns. O gráfico da Figura 36 ilustra o valor da função em todo o seu domínio. Note que a função tende ao infinito conforme p tende a 0 ou 1.

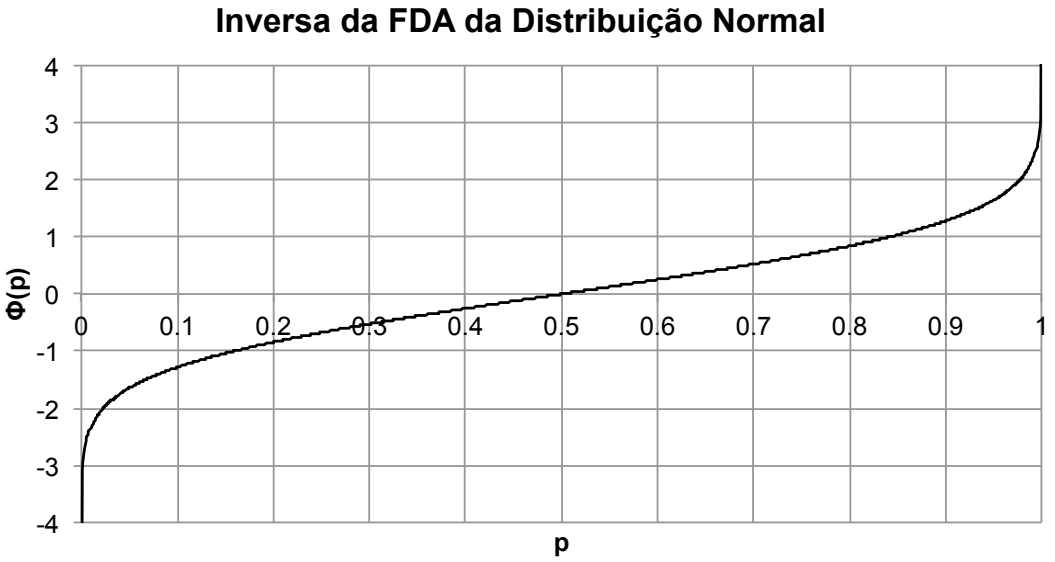


Figura 36: Curva da inversa da função distribuição acumulada da distribuição normal padrão.

Vale ressaltar que variáveis que seguem uma distribuição normal são contínuas, fazendo com que a probabilidade de um valor x qualquer assumir um valor específico z_i seja zero, i.e., $P\left(x = z_i\right) = 0$ (FRANK; ALTHOEN, 1994). Logo:

$$P\left(x \leqslant z_i\right) = P\left(x < z_i\right) \tag{A.5}$$

Esta propriedade das distribuições contínuas faz com que a definição do percentil para uma distribuição normal padrão seja independente da interpretação do percentil para o caso geral.

O cálculo do percentil de uma distribuição normal padrão é especialmente útil para determinar um intervalo $[-z, z]$ que apresente uma certa porcentagem das observações. A probabilidade de uma observação ficar *fora* deste

intervalo é representada por α. Portanto:

$$P\left(x < -z\right) + P\left(x > z\right) = \alpha \tag{A.6}$$

onde $P\left(s\right)$ é probabilidade da proposição s ser verdadeira. Como a distribuição normal padrão é simétrica em torno de zero (FRANK; ALTHOEN, 1994), $P\left(x < -z\right) = P\left(x > z\right)$. Logo:

$$P\left(x < -z\right) = P\left(x > z\right) = \alpha/2 \tag{A.7}$$

Utilizando o complemento destas probabilidades:

$$P\left(x \leqslant z\right) = 1 - P\left(x > z\right) = 1 - \alpha/2 \tag{A.8}$$

Assim, para descobrir valor de z do intervalo $[-z, z]$ em função de α, basta calcular o valor do $100{\cdot}(1 - \alpha/2)$-ésimo percentil da distribuição normal padrão. A Figura 37 ilustra um exemplo para $\alpha = 5\%$.

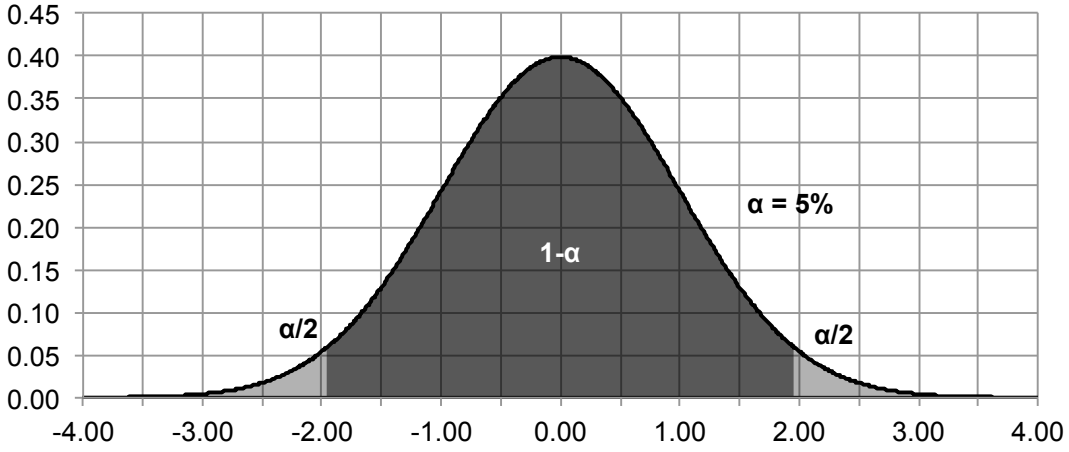


Figura 37: Curva da função densidade de probabilidade da distribuição normal padrão. Para $\alpha = 5\%$, tem-se um percentil $z_{1-\alpha/2} \approx 1,96$, i.e., 95% da área está no intervalo $[-1,96,\ +1,96]$.

APÊNDICE B – TESTE DE HIPÓTESE SOBRE A MÉDIA DE UMA AMOSTRA

A partir de uma média obtida experimentalmente através de uma amostra, uma análise comumente realizada é a comparação dela com algum valor de referência a fim de descobrir se a média verdadeira é, estatisticamente, maior ou menor do que esta referência, com algum grau de certeza. Tal análise permite verificar se um resultado experimental está dentro de uma faixa de valores esperada. Para esta finalidade, utiliza-se um *teste de hipótese*, capaz de rejeitar uma hipótese caso sejam encontradas evidências contrárias suficientes.

Todo teste de hipótese pode ser decomposto nos seguintes passos:

1. Definir uma hipótese nula H_0 e uma hipótese alternativa H_1;
2. Escolher o teste de estatística apropriado para buscar evidências que rejeitem a hipótese nula;
3. Derivar a distribuição do teste de estatística assumindo que H_0 seja verdadeira;
4. Calcular o valor t a partir do teste de estatística;
5. Escolher um nível de significância α apropriado;

6.A partir deste ponto, dois caminhos equivalentes podem ser seguidos:

(a)Verificar se t está na região crítica da distribuição: a hipótese nula é rejeitada se t estiver nesta região;

(b)Calcular o p-valor com base em t e comparar com α: a hipótese nula é rejeitada se $p < \alpha$.

A escolha das hipóteses é feita de acordo com o que se deseja provar. Em geral, a hipótese alternativa H_1 é aquela que se deseja provar, enquanto a hipótese nula H_0 é a negação e/ou complemento de H_1.

A Tabela 13 ilustra três possíveis testes de hipóteses que podem ser realizados acerca de uma média $\overline{x}$, buscando verificar se a média verdadeira μ deve ser diferente, menor ou maior que um valor de referência M. O número de caudas indica a posição da região crítica.

Teste	**Hipótese Nula H_0**	**Hipótese Alternativa H_1**	**Número de caudas**
1	$\mu = M$	$\mu \neq M$	2
2	$\mu \geqslant M$	$\mu < M$	1 (esquerda)
3	$\mu \leqslant M$	$\mu > M$	1 (direita)

Tabela 13: Hipóteses para análise da média de uma mostra com relação a uma referência M.

A escolha e a derivação do teste de estatística são feitas baseadas no que se deseja provar. Para a análise de média, uma possibilidade é fazer uso do *Teste t de Student*, que calcula uma estatística de teste t com base na média e no erro padrão observados. Mais especificamente:

$$t = \frac{\overline{x} - M}{SE_{\overline{x}}} \tag{B.1}$$

onde:

- $\overline{x}$ é a média da amostra;

- M é o valor de referência;

- $SE_{\overline{x}}$ é o erro padrão da amostra.

O erro padrão estimado a partir de uma amostra é calculado através da seguinte fórmula:

$$SE_{\overline{x}} = \frac{s}{\sqrt{n}} \tag{B.2}$$

onde n representa o desvio padrão da amostra, enquanto s é o desvio padrão medido experimentalmente.

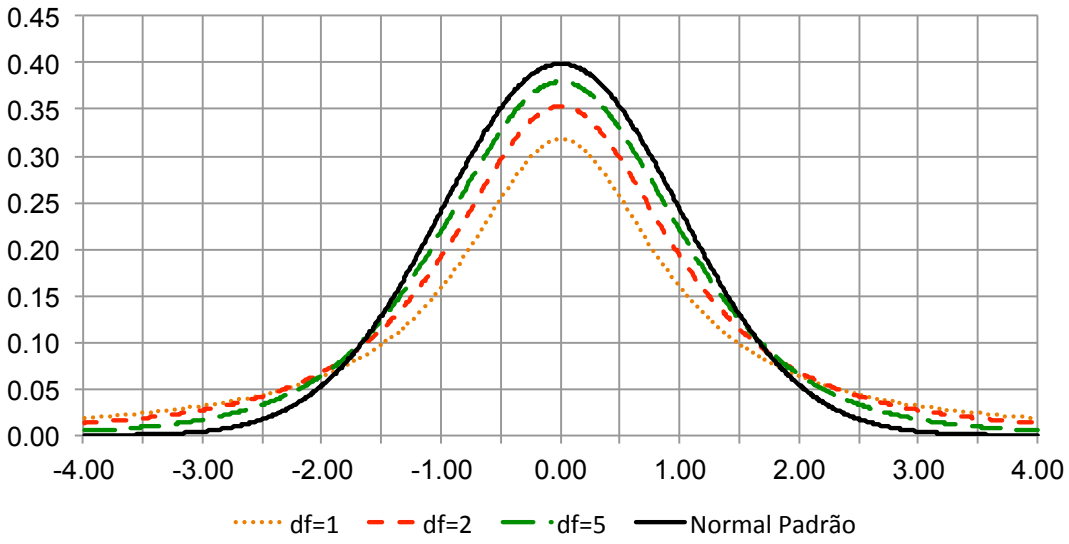


Figura 38: Função densidade de probabilidade da distribuição t de Student, de acordo com os graus de liberdade. No limite $df \to \infty$, aproxima-se da distribuição normal padrão.

Em cada tipo de teste, é importante saber qual distribuição a estatística de teste segue para que seja possível analisá-la. Neste caso, o valor t segue uma distribuição t de Student, com $df = n - 1$ graus de liberdade. *Grau de liberdade* é o número de valores independentes que podem variar em um teste estatístico, e é igual ao número de amostras menos o número de parâmetros avaliados - no caso, apenas um. A Figura 38 ilustra como os graus de liberdade afetam a forma da distribuição.

Em seguida, um nível de significância α apropriado deve ser escolhido (valores típicos incluem 5% e 1%). Este valor define a *região crítica* da distribuição em que a probabilidade de se encontrar um valor t amostrado aleatoriamente é α, assumindo que a hipótese nula era verdadeira. Classicamente, a determinação da região crítica para um valor α ocorre através de tabelas com valores pré-definidos para cada distribuição - no caso geral, os valores exatos não são computáveis, devendo ser estimados através de aproximações.

Para uma distribuição t de Student, a região crítica compreende as regiões extremas, também chamadas de *cauda*. A escolha das caudas a serem consideradas depende do teste de hipótese, conforme mostrado na Tabela 13. A Figura 39 ilustra a região crítica de uma distribuição t de Student com 9 graus de liberdade para $\alpha = 5\%$ e $\alpha = 1\%$ quando apenas a cauda direita é considerada.

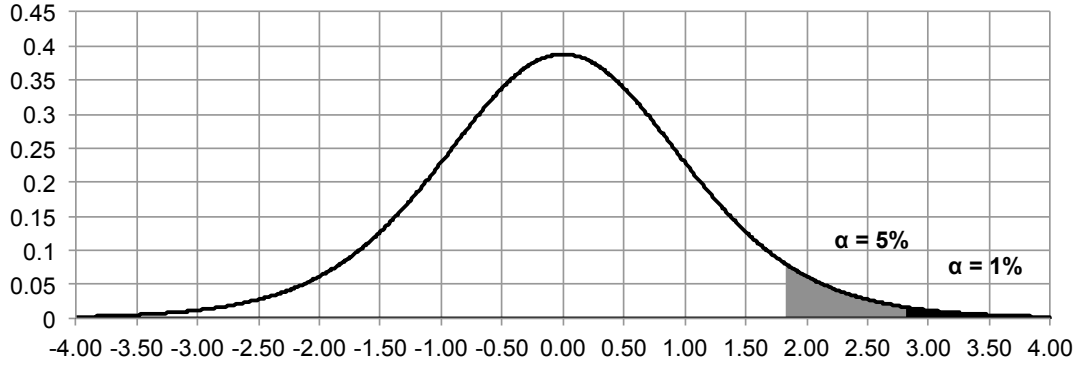


Figura 39: Região crítica de uma distribuição t de Student utilizando apenas a cauda direita, com 9 graus de liberdade, com $\alpha = 5\%$ e $\alpha = 1\%$.

Uma vez conhecida a região crítica e o valor t (calculado através da fórmula B.1), uma das seguintes conclusões pode ser tirada:

- Se o valor t estiver contido na região crítica, rejeita-se a hipótese nula H_0, pois existe uma probabilidade muito alta $(1 - \alpha)$ de que a hipótese não era verdadeira. Logo, H_1 deve ser verdadeira.

- Caso contrário, não podemos rejeitar a hipótese nula, pois não há evidências suficientes para derrubá-la.

Nota: independentemente da conclusão obtida, H_0 não pode ser provada neste tipo de teste. Mesmo que ela não seja rejeitada, só é possível afirmar que não foram encontradas evidências suficientes que permitam concluir que H_0 era falsa, o que não comprova que a afirmação contrária seja verdadeira.

Como exemplo, para um experimento com $n = 10$ amostras, média $\overline{x} = 101$ e desvio padrão $s = 1$, deseja-se verificar a hipótese de que $\mu > 100$ com significância $\alpha = 5\%$. Assim, são definidas duas hipóteses:

- H_0: $\mu \leqslant 100$
- H_1: $\mu > 100$

Através desses dados, calcula-se $SE_{\overline{x}} = 1/\sqrt{10} = 0,316$, $df = 10 - 1 = 9$ e $t = (101 - 100)/9 = 3,162$. Neste exemplo, a região crítica compreende apenas a cauda direita (conforme mostrado na Tabela 13). Consultando uma tabela, a região crítica para $\alpha = 5\%$ e $df = 9$ inicia-se em $1,833$. Como t está dentro da região crítica (i.e., $t > 1,833$), rejeita-se a hipótese nula. A Figura 39 ilustra a região crítica deste exemplo - note que t continua dentro da região crítica mesmo para $\alpha = 1\%$.

Com o advento de métodos computacionais eficientes, tornou-se possível realizar o caminho inverso: ao invés de verificar se o valor t está na região crítica determinado por α, pode-se determinar qual é a região crítica delimitada por t, e consequentemente, a probabilidade p (também chamada de *p-valor*) de uma estatística de teste cair nela, assumindo que hipótese nula era verdadeira. A partir desta probabilidade p, uma das seguintes conclusões podem ser obtidas:

- Se o valor-p é inferior a α, a hipótese nula H_0 é rejeitada, pois é pouco provável que a estatística de teste possa ser obtida a partir de amostra de uma população em que a hipótese é verdadeira.

- Caso contrário, não é possível rejeitar a hipótese nula.

Teoricamente, o valor-p pode ser calculando através da inversa da função densidade de probabilidade da distribuição utilizada, recebendo o valor t como entrada. Contudo, muitas destas funções não possuem formas fechadas, e requerem aproximações para serem calculadas - e.g., para a distribuição t de Student, Shaw define funções exatas para certos valores de df, mas, no caso geral, depende de funções aproximadas (SHAW, 2006). A maior parte dos softwares estatísticos contêm implementações próprias para estas aproximações - e.g., *T.DIST(x, df, cumulativo)* do *Microsoft Excel*, utilizado neste trabalho.

Para a distribuição t de Student, a função de cálculo do valor-p é identificado por $P\left(x \geqslant t, df\right)$ e $P\left(x \leqslant t, df\right)$, que calculam a probabilidade de um valor x obtido experimentalmente ser maior ou menor que um valor t para o número de graus de liberdade df dado. Vale ressaltar que, como t de Student é uma distribuição simétrica, $P\left(x \geqslant t, df\right) = P\left(x \leqslant -t, df\right)$.

Utilizando o exemplo anterior, o valor-p é igual a $P\left(x \geqslant 3,162,\ df = 9\right) = 0,00575$, inferior ao valor α estipulado - logo, rejeita-se a hipótese nula.

Em comparação, para um teste com duas caudas, as hipóteses seriam:

- H_0: $\mu = 100$

- H_1: $\mu \neq 100$

Neste teste, o valor-p é igual a $P\left(x \geqslant 3,162,\ df = 9\right) + P\left(x \leqslant -3,162,\ df = 9\right) = 2P\left(x \geqslant 3,162,\ df = 9\right)$, resultando em $0,01151$.

Para $\alpha = 5\%$, a hipótese nula seria rejeitada, enquanto, para $\alpha = 1\%$, não seria possível rejeitar a hipótese nula, uma vez que $0,01151 > 0,01$.