UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIA E TECNOLOGIA
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

LACONIBOT: UM AGENTE PARA ATUAR EM LEILÕES DO TIPO CDA DO TAC

ROBSON GONÇALVES FECHINE FEITOSA

FORTALEZA
2009

ROBSON GONÇALVES FECHINE FEITOSA

LACONIBOT: UM AGENTE PARA ATUAR EM LEILÕES DO TIPO CDA DO TAC

Dissertação submetida à Comissão Examinadora do Programa de Pós-Graduação Acadêmica em Ciência da Computação, da Universidade Estadual do Ceará, como requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Jerffeson Teixeira de Souza, Ph.D.
Co-orientador: Prof. Gustavo Augusto Lima de Campos, Dr.

FORTALEZA
2009

F311l Feitosa, Robson G. F.

LaconiBot: um agente para atuar em leilões do tipo CDA do TAC/ Robson Gonçalves Fechine Feitosa. __ Fortaleza, 2009.

100p. ; 2il.

Orientador: Prof. Ph.D Jerffeson Teixeira de Souza.

Dissertação (Mestrado Acadêmico em Ciência da Computação) – Universidade Estadual do Ceará, Centro de Ciência e Tecnologia.

1. Agentes inteligentes. 2. *Trading Agent Competition.* 3. *Continuous Double Auction.* 4. *Fuzzy.* 5. Algoritmos Genéticos. 6. Redes Neurais Artificiais. I. Universidade Estadual do Ceará, Centro de Ciência e Tecnologia.

CDD: 511.8

ROBSON GONÇALVES FECHINE FEITOSA

LACONIBOT: UM AGENTE PARA ATUAR EM LEILÕES DO TIPO CDA DO TAC

Dissertação submetida à Comissão Examinadora do Programa de Pós-Graduação Acadêmica em Ciência da Computação, da Universidade Estadual do Ceará, como requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 28/08/2009.

COMISSÃO EXAMINADORA

_____

Prof. Ph.D. Jerffeson Teixeira de Souza (Orientador)
Universidade Estadual do Ceará – UECE

_____

Prof. Dr. Gustavo Augusto Lima de Campos (Co-orientador)
Universidade Estadual do Ceará – UECE

_____

Prof. Ph.D. José Raimundo de Araújo Carvalho Júnior
Universidade Federal do Ceará – UFC

_____

Prof. Dr. Gerardo Valdisio Rodrigues Viana
Universidade Federal do Ceará – UFC

Aos meus pais Roberto e Mundinha, e aos meus irmãos Rafael e Roney, minha família, minha fortaleza, meu refúgio.

AGRADECIMENTOS

A Deus, sempre e em primeiro lugar.

A minha família, em especial meus pais e irmãos, graças ao apoio deles foi possível concluir este trabalho.

Aos meus orientadores, professores Gustavo e Jerffeson, o meu muitíssimo obrigado pela paciência e confiança depositados.

Aos professores Cidcley, Mariela e Jackson, por suas contribuições, não só durante a qualificação, mas sempre que surgiam novas dúvidas.

Aos amigos Amauri, Fabricio, Paulo de Tarso, Rafael e Renato, o meu muito obrigado pela ajuda valiosa.

Aos amigos conquistados nesse mestrado, em especial aos colegas da primeira turma: Alex, Carlos Cidade, Fabiano, Marcelo, Márcia, Mario, Tarciane, Vigno e Viviane.

A todos os amigos da UECE de uma maneira geral, Daladier, Dalmo, Gilzamir, Tales, Wagner, Wallison.

Aos amigos que, a duras penas, se privaram da minha presença, o meu muito obrigado pela paciência.

A todos os professores e funcionários do MACC, por proporcionarem um excelente ambiente de trabalho.

A todos que, de alguma forma, contribuíram para o engrandecimento desse trabalho. Perdão se não mencionei alguém, mas sintam-se agraciados.

A FUNCAP pelo apoio financeiro.

“panta ischuô en tô endunamounti me tsb=christô”
“Posso todas as coisas em Cristo que me fortalece.”
Filipenses, 4:13

RESUMO

Este trabalho descreveu e avaliou o agente LaconiBot em uma versão modificada do ambiente *Trading Agent Competition* (TAC) *Classic*. Tal agente se utiliza de técnicas de Inteligência Artificial voltadas a resolução de alguns problemas encontrados no processo de negociação *Continuous Double Auction* (CDA) do TAC. Foi utilizado um controlador *Fuzzy* para resolver o problema de determinação dos lances. Para selecionar quais leilões o agente deve participar efetuando lances, foi utilizado um Algoritmo Genético Multi-Objetivo. E, para prover uma estimativa mais apurada dos valores de transação dos lances, foi utilizada uma Rede Neural Artificial.

Palavras-chave: Agentes inteligentes. *Trading Agent Competition. Continuous Double Auction. Fuzzy.* Algoritmos Genéticos. Redes Neurais Artificiais.

ABSTRACT

This work described and evaluated the agent LaconiBot at a modified version of the Trading Agent Competition (TAC) Classic environment. This agent makes use of Artificial Intelligence techniques for resolution of some problems found in the process of TAC Continuous Double Auction (CDA) negotiation. In other words, a Fuzzy control was used to solve the problem of bid determination. A Genetic Multi-objective Algorithm was used to select which auctions the agent must announce effectuating bids. And, a Artificial Network Neural was used to provide a more refined estimate of the transaction values.

Key-words: Intelligent Agents. Trading Agent Competition. Continuous Double Auction. Fuzzy. Genetic Algorithms. Artificial Network Neural.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AG	Algoritmo Genético
CDA	*Continuous Double Auction*
IA	Inteligência Artificial
MLP	*Multilayer Perceptrons*
NYSE	*New York Stock Exchange*
RMSE	*Root Mean Squared Error*
RNA	*Rede Neural Artificial*
SCM	*Supply Chain Management*
SICS	*Swedish Institute of Computer*
TAC	*Trading Agent Competition*

SUMÁRIO

INTRODUÇÃO

A facilidade de acesso à informação proporcionada por meios de comunicação, como a Internet, contribuiu para o surgimento de novas formas de prestação de serviços. Para o setor comercial, não foi diferente. Novas formas de negociar foram possíveis graças à velocidade na transferência da informação; e, recursos, como iteratividade. Um tipo de negociação muito utilizado através da Internet são os leilões CDA ou *Continuous Double Auction*.

Nesse contexto, o presente capítulo visa descrever a motivação, a relevância e os desafios encontrados no processo de negociação CDA. Além disso, são levantados os principais problemas enfrentados para o desenvolvimento de um agente que atue de forma autônoma em ambientes CDA, e de forma mais específica no ambiente CDA do TAC (*Trading Agent Competition*); bem como os objetivos e a metodologia utilizada para se chegar às soluções de alguns dos problemas encontrados nesse tipo de negociação com o apoio de técnicas de Inteligência Artificial.

Motivação

Entende-se por *e-commerce*, ou comércio eletrônico, qualquer ambiente de negociação onde os participantes interagem eletronicamente, utilizando-se das tecnologias da informação e comunicação para condução do processo de negociação comercial. Esse tipo de ambiente despertou o interesse de pesquisadores no mundo inteiro por permitir uma automatização no seu processo de gerenciamento e execução. Através dessa nova forma de negociar também surgiram novas possibilidades para a condução do processo de negociação, quer seja no desenvolvimento de novas regras de mercado, quer seja automatizando as atividades de compra e venda.

Um mecanismo bastante utilizado nos ambientes de comércio eletrônico são os leilões. Segundo Klemperer (1999), um leilão é um mecanismo no qual os vendedores fazem ofertas e os compradores dão lances em seqüência. Devido sua aplicação prática e fundamentação teórica, os leilões adquiriram grande importância, tornando-se objeto de estudo através da Teoria dos Leilões. A infra-estrutura da Internet proporcionou uma

redução nos custos para a realização dos leilões, bem como motivou o aumento do número de compradores e vendedores envolvidos nesse processo de negociação.

De acordo com Klemperer (1999), existem vários tipos de leilões, e eles podem ser classificados em diretos e reversos. Os tipos de leilões mais conhecidos e estudados são: o Inglês, o Holandês, o de lance fechado e primeiro preço, o Vickrey, e o Duplo. Nos leilões Duplos, por exemplo, todos os participantes podem atuar tanto como compradores quanto como vendedores e os fechamentos ocorrem continuamente, enquanto durar o tempo do leilão, o que os caracterizam como leilões com fechamento contínuo. O presente trabalho tem foco neste tipo de leilão.

Segundo Vytelingum (2006), os leilões Duplos, também conhecidos como *Continuous Double Auction* (CDA), são atualmente os mecanismos de negociação mais comuns adotados em *e-commerce*. Basicamente, o funcionamento de um CDA ocorre da seguinte forma: durante um intervalo de tempo pré-fixado, os participantes submetem, de forma assíncrona, ofertas de venda de bens e lances de compra. No momento em que o valor de um lance de compra ficar maior ou igual a alguma oferta de venda, a transação é realizada.

Os leilões CDA são amplamente utilizados por instituições especializadas em compra e venda de títulos e mercadorias, e.g. mercado de câmbio internacional e a bolsa de valores. De acordo com Vytelingum (2006), a bolsa de valores *New York Stock Exchange* (NYSE) movimenta cerca de 12,4 trilhões de dólares em transações anuais, o que demonstra o grau de importância desse mercado para a economia mundial. Mercados como o NYSE negociam diversos bens em paralelo, através de leilões simultâneos e independentes.

Além dos problemas clássicos encontrados na negociação CDA, tais como determinação da quantidade e valor dos bens, os participantes desse mercado devem analisar simultaneamente a evolução das várias cotações, em detrimento de suas preferências, para escolher em quais leilões efetivamente participar. Contudo, resolver esse problema não é uma tarefa trivial, uma vez que este se caracteriza como um problema clássico de alocação, semelhante ao problema da mochila (HOROWITZ; SAHNI, 1974), possuindo, assim, alta ordem de complexidade.

Na literatura, existem trabalhos que abordam diversas soluções para os problemas encontrados por participantes em ambientes de negociação do tipo CDA. Vytelingum (2006) fez um apanhado sobre os principais aspectos comportamentais e estruturais do CDA.

Outras abordagens baseadas em estudos estatísticos também são utilizadas para resolver problemas encontrados em ambientes de negociação CDA, como, por exemplo, as cadeias de Markov (PARK et al., 2004). Abordagens baseadas em Inteligência Artificial também são bastante utilizadas, como lógica *Fuzzy* (GUPTA et al., 2008) (HE et al., 2003) e Redes Neurais Artificiais (KO; LI, 2008).

A Inteligência Artificial (IA) surgiu do objetivo de tornar máquinas capazes de executar tarefas tão bem quanto o próprio homem. A partir desse objetivo, cientistas propuseram teorias que fundamentaram as pesquisas em Inteligência Artificial. Segundo Russell & Norvig (1995), a IA teve início após a Segunda Guerra Mundial e, devido sua variedade em campos de aplicação, as definições de IA transitam por duas dimensões principais: a de pensamento e a de raciocínio. A primeira abrange a capacidade de resolução de problemas e tomada de decisão de sistemas artificiais, bem como aprendizado e tarefas afins; a segunda preocupa-se com a capacidade de máquinas ou entidades, como agentes, executarem tarefas que exijam algum tipo de raciocínio.

Na dimensão do raciocínio em IA, um agente é qualquer coisa que pode ser vista como uma entidade capaz de perceber o ambiente através de sensores e atuar no ambiente através de executores Russell & Norvig (1995). Para que agentes sejam classificados como inteligentes, eles devem possuir características de flexibilidade de ações, tais como reatividade, pró-atividade e sociabilidade (WEIS, 2000).

Agentes inteligentes estão cada vez mais presentes no nosso cotidiano, quer seja atuando em jogos eletrônicos, ou como classificadores de conteúdo na Web e em dispositivos de segurança biométricos, por exemplo. Contudo, uma aplicação em particular de agentes inteligentes tem despertado o interesse de pesquisadores ao redor do mundo: agentes negociadores atuando em comércio eletrônico, e-commerce.

O estudo realizado por Das et al. (2001) comprovou que agentes inteligentes podem superar o desempenho de humanos, durante ações de negociação. Estes agentes

podem ser empregados na monitoração simultânea de vários mercados, no processamento de grande volume de informação e na execução de cálculos complexos quase que instantaneamente.

Visando fomentar a pesquisa de alto nível em agentes negociadores, pesquisadores da Universidade de Michigan criaram o ambiente TAC (*Trading Agent Competition*) (WELLMAN; WURMAN, 1999). O TAC é composto por um fórum sobre o tema de agentes negociadores e uma série de torneios anuais. Ele ainda fornece uma infra-estrutura completa de software para o desenvolvimento e simulação de agentes negociadores, o que facilita a geração de conhecimento na área.

Surge como um desafio, o desenvolvimento de um agente que se utilize de técnicas de Inteligência Artificial para atuar em leilões simultâneos do tipo CDA. Haja vista a complexidade das tarefas de negociação, bem como a dificuldade em expressar as preferências de um especialista humano através de uma entidade de negociação autônoma, como um agente.

Problema

Desenvolver sistemas para atuar em ambientes complexos, como os leilões CDA, demandam uma análise bem estruturada dos problemas encontrados nos mesmos. Mais especificamente, quando os sistemas são entidades autônomas como agentes, pode-se notar que os agentes que participam de leilões CDA simultâneos, como os do TAC, enfrentam o problema de determinação dos lances em cada leilão (BOYAN; GREENWALD, 2001). Baseado em (WELLMAN et al., 2007), tal problema pode ser decomposto em outros subproblemas:

- Dado um conjunto de bens que o agente possua, como alocar esses bens de acordo com suas preferências de maneira a maximizar a utilidade do agente?
- Dado um conjunto de bens que o agente possua, quais bens devem ser comprados?
- Dado um conjunto de bens que o agente possua, alocados de maneira ótima, quais bens podem ser vendidos?

Somado ao problema de determinação dos lances, os agentes enfrentam outras dificuldades como a determinação de quando efetuar o lance e também o valor do mesmo.

O problema de determinação dos lances pode ser visto como um problema de alocação, ou seja, dado um conjunto de leilões que o agente pode participar, como alocar a participação do agente de acordo com suas preferências e de uma medida de avaliação de desempenho? É possível observar que a formalização deste problema de alocação equivale à formalização do problema da mochila, tratando-se de um problema NP-Completo (GAREY et al., 1979), onde os bens a serem comprados ou vendidos equivalem, respectivamente, aos itens a serem colocados ou retirados da mochila.

Para apoiar o processo de decisão de quando e por quanto efetuar um lance, surge outro problema. Para que o agente seja competitivo, o valor de incremento ou decremento do lance deve ser calculado tomando como base uma estimativa do preço de transação do leilão analisado. Assim, como calcular a estimativa do preço de transação dos leilões?

É fácil perceber que cada um desses problemas pode ser resolvido através da implementação de uma entidade autônoma de processamento, isto é, um agente. Entretanto, gerenciar tal sistema, onde os agentes executam tarefas de maneira colaborativa, demanda uma arquitetura que venha a facilitar a coordenação, bem como a atribuição de tarefas aos agentes. Dessa forma, a arquitetura de tal sistema constitui outro problema a ser tratado nesse trabalho.

Objetivos

Como objetivo geral, o presente trabalho visou conceber um agente de software para atuar no ambiente de negociação CDA do TAC. Este agente foi denominado LaconiBot. Como resultados da pesquisa, projeto e desenvolvimento necessários à esta concepção, mais especificamente, este trabalho visou: contribuir com a geração de conhecimento sobre agentes negociadores, permitindo que outros pesquisadores na área possam utilizar este conhecimento como fundamentação, aprimorando os resultados presentes e explorando os aspectos não abordados e indicados neste trabalho.

Metodologia

Considerando a etapa de pesquisa bibliográfica a respeito do assunto, realizada até a elaboração da proposta de dissertação e em outros momentos que foram necessários durante a concepção, este trabalho focou as abordagens para as soluções dos problemas encontrados nas negociações em leilões CDA; já que a arquitetura proposta para o agente LaconiBot permitiu que as tarefas fossem atribuídas a cada componente, de maneira que os mesmos possam, de maneira modular, resolver cada um dos problemas levantados.

Dessa forma, o LaconiBot utilizou-se de um componente baseado em utilidade que resolve o problema de determinação dos lances como um problema de otimização multi-objetivo com o uso de um algoritmo genético. Para resolver os problemas de quando, e por qual valor efetuar um lance, foi utilizado um componente baseado em regras, que se utiliza de um controlador nebuloso no seu processo de inferência. Para o cálculo da estimativa do preço de transação dos leilões, utilizou-se um componente que faz uso de uma rede neural com arquitetura MLP (*Multilayer Perceptrons)*.

Aplicar tal agente em mercados hipotéticos levaria a uma distorção dos resultados quando comparados com a realidade. Isso aconteceria uma vez que os mercados reais são constituídos de muitos participantes, onde o desempenho de um determinado participante influencia diretamente nas dos outros (WELLMAN et al., 2007). Dessa forma utilizou-se o ambiente de simulações TAC para avaliar o desempenho do LaconiBot, uma vez que o TAC disponibiliza uma variedade de agentes competitivos para atuar em seu ambiente, remetendo a uma simulação mais próxima da realidade.

Assim, de maneira geral, os passos realizados para o alcance dos objetivos podem ser divididos em 4 partes, a saber:

Parte I – Revisão Bibliográfica

Foi realizada uma pesquisa, através das mais diversas fontes literárias, sobre os temas: Inteligência Artificial, Engenharia de Software, Agentes Negociadores e o TAC. Posteriormente, iniciou-se um estudo sobre os mecanismos de negociação, mais especificamente, sobre o mecanismo CDA, bem como estratégias relacionadas aos

problemas encontrados nesse cenário. Tal pesquisa foi realizada visando um maior embasamento e, conseqüentemente, melhor entendimento do estado da arte do tema em estudo.

Parte II – Especificação da Arquitetura

Com o conhecimento do estado da arte relacionado ao tema explorado no atual trabalho, foi necessário delimitar o escopo do problema a ser atacado. Nessa fase, listaram-se os principais problemas encontrados no processo de negociação em leilões CDA, bem como as abordagens que seriam utilizadas nas soluções.

Posteriormente, foi levantada uma arquitetura para o agente. Ainda nessa fase, foram definidos os papéis adotados por cada componente do sistema, bem como as técnicas que seriam utilizadas para implementar as abordagens utilizadas nas soluções.

Parte III – Implementação e Validação

Para implementar o LaconiBot foi utilizada a linguagem de programação Java que, apesar de ser uma linguagem que segue o paradigma de orientação a objetos, é amplamente utilizada pelas comunidades acadêmica e empresarial para o desenvolvimento dos mais diversos sistemas. Um dos grandes benefícios da implementação com Java é a existência de diversas bibliotecas que facilitam sobremaneira o desenvolvimento das mais variadas funcionalidades inerentes à programação de sistemas complexos.

Para a validação da implementação do LaconiBot, foi utilizada uma versão modificada do ambiente TAC para que fosse analisado o desempenho das estratégias em leilões do tipo CDA. Os detalhes de como foi implementado o LaconiBot e como foram analisados os desempenhos de cada abordagem são descritos no decorrer do trabalho.

Parte IV – Escrita

O processo de escrita foi realizado após a implementação e avaliação do LaconiBot. A estratégia foi embasada na avaliação dos componentes criados, onde foi questionado se as soluções propostas possuíam relevância e mérito. Esta abordagem foi utilizada visando a organização e clareza na transmissão das idéias do presente trabalho.

Organização

O trabalho foi organizado em 4 capítulos. No capítulo 1 discorre-se sobre os leilões do tipo CDA, apresentando um breve embasamento sobre a Teoria dos Leilões e o protocolo de mercado dos leilões CDA voltado ao cenário utilizado como estudo de caso deste trabalho, o TAC *Classic*. Ainda no mesmo capítulo, apresenta-se uma pequena descrição das técnicas de Inteligência Artificial utilizadas na implementação do LaconiBot

No capítulo 2, são levantados alguns trabalhos relacionados aos problemas listados na introdução, além de serem descritas as estratégias de alguns agentes, utilizados como parâmetro para a analise de desempenho do LaconiBot.

O capítulo 3 descreve a arquitetura do agente e as abordagens utilizadas pelo mesmo para a determinação do valor dos lances, da estimativa do preço de transação e o processo de seleção dos leilões, respectivamente.

No capítulo 4, são discutidos os resultados obtidos com as abordagens do LaconiBot para o ambiente TAC, bem como o processo de avaliação dessas abordagens. Posteriormente, ao final do trabalho, são apresentadas algumas conclusões e trabalhos futuros.

1. FUNDAMENTAÇÃO TEÓRICA

O presente capítulo esboça alguns dos referenciais teóricos que fundamentaram a concepção do agente LaconiBot. Inicialmente, o ambiente TAC é descrito em maiores detalhes; em seguida, apresenta-se de forma resumida a teoria relacionada ao funcionamento dos leilões, dando ênfase aos leilões do tipo CDA no ambiente TAC. Para finalizar, as técnicas de Inteligência Artificial utilizadas no presente trabalho são descritas resumidamente.

1.1.O Ambiente *Trading Agent Competition*

O *Trading Agent Competition* é um fórum internacional destinado a promover pesquisas de alto nível para o problema dos agentes negociadores. As duas primeiras edições do TAC foram realizadas na cidade de Boston (EUA) em 2000 e 2001. A primeira edição contou com 16 participantes de 6 países da América do Norte, Europa e Ásia. A partir de 2002, as edições do TAC foram realizadas no SICS (*Swedish Institute of Computer Science*). E, ao longo dos anos, as regras do jogo foram refinadas (WELLMAN *et* al., 2007), proporcionando um aumento no nível dos participantes.

Três tipos de torneios são promovidos pelo fórum: TAC *Classic*; TAC SCM (*Supply Chain Management*), para o problema da Gerência de Cadeia de Suprimento; e TAC *Market Design* relacionado ao projeto de regras de mercados. Mais informações podem ser obtidas através do site do fórum (SICS, 2009).

TAC *Classic*

No TAC[1], um agente compete com outros sete competidores. Os agentes desempenham o papel de agências de viagens, que devem planejar pacotes turísticos para um conjunto de oito clientes, atendendo da melhor forma as preferências de cada um destes. Os clientes desejam partir da cidade *TACtown* com destino a cidade turística *Tampa*, permanecendo nesta ao menos um dia. Os pacotes turísticos são compostos por três itens: passagens aéreas (ida e volta), reservas em hotéis para todo o período de estadia e ingressos para eventos de entretenimento, como ilustrado na Figura 1.1.

[1] Para simplificação, a partir de agora o TAC Classic será denominado apenas TAC no presente trabalho.

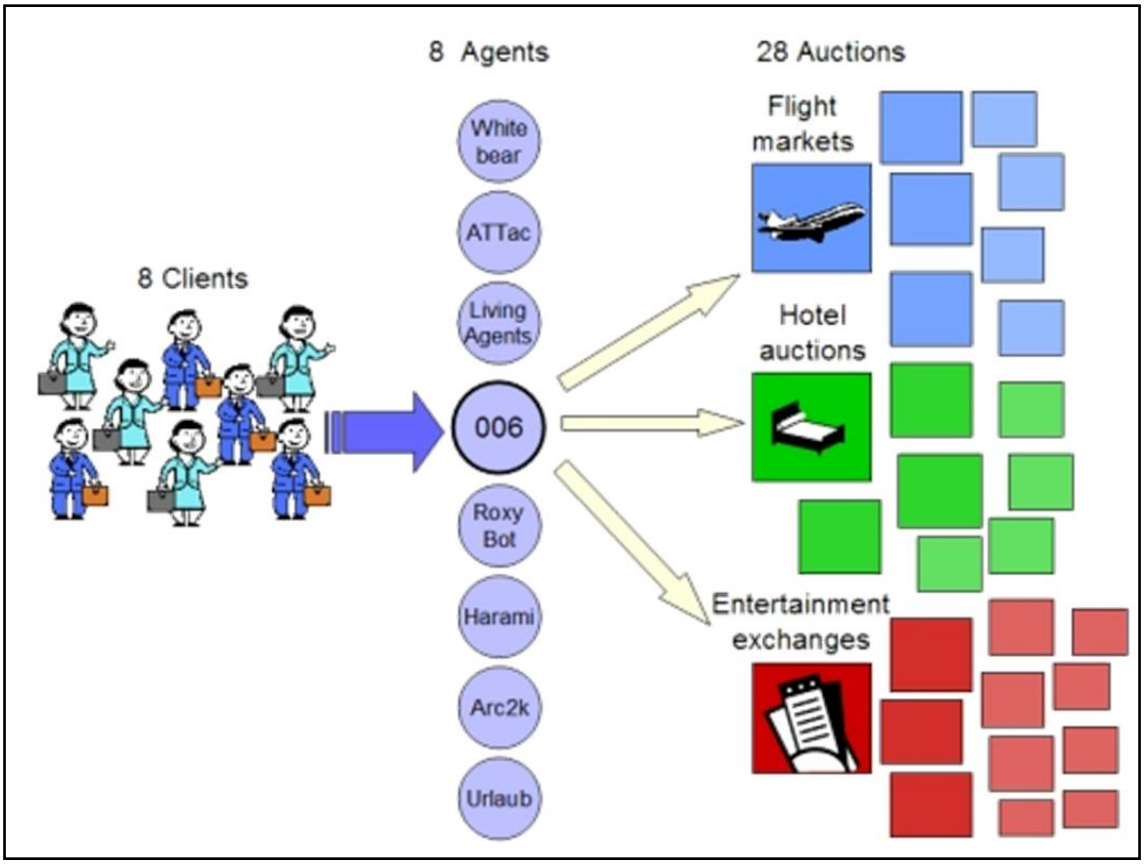

Figura 1.1. Cenário do TAC Classic. Fonte SICS (2009).

Os clientes têm preferências por determinados dias de partida e chegada (dentre os cinco possíveis) e são escolhidos aleatoriamente, de forma que a probabilidade dos pares de combinações possíveis seja a mesma. Além disso, oferecem bônus se ficam no melhor hotel; e, possuem interesse variável de acordo com o tipo de entretenimento, oferecendo um bônus que varia uniformemente no intervalo entre $0 e $200. Um exemplo de preferências de clientes pode ser visto na Tabela 1.1, onde IAD e IDD representam respectivamente os dias de partida e chegada ideais; HB representa o bônus para estadia no melhor hotel; e AW, AP e UM representam o tipo de entretenimento (Luta de Crocodilos, Parque de Diversão e Museu, respectivamente).

Tabela 1.1. Exemplo de preferências dos clientes.

Cliente	IAD	IDD	HB	AP	AW	MU
1	2	5	73	175	34	24
2	1	3	125	113	124	57
3	4	5	73	1557	12	177
4	1	2	102	50	67	49
5	1	3	75	12	135	111
6	2	4	856	197	8	59
7	1	5	90	56	197	162
8	1	3	50	79	92	136

Para adquirir os itens que formam os pacotes, os agentes devem participar de leilões conforme descrito a seguir.

Leilões de passagens aéreas: leilões simples e contínuos com único vendedor, a TACAIR (única companhia aérea que fornece vôos de ida e volta entre *TACtown* e *Tampa*). Para efeito de simplificação, considera-se ilimitado o número de assentos por vôo. O preço das passagens é ajustado segundo uma função estocástica através de perturbações periódicas.

Leilões de hotéis: existem dois hotéis em Tampa, o *Tampa Towers* e o *Shoreline Shanty*. O primeiro possui a preferência dos clientes, sendo, conseqüentemente, mais caro. Os quartos são negociados através de leilões do tipo inglês, ascendente, multi-item, 16º preço, isto é, ao encerramento de cada leilão, 16 reservas são negociadas com os agentes que ofereceram as 16 maiores ofertas, considerando como preço de venda a menor dentre estas ofertas, ou seja, o 16º maior preço. Existe um total de oito leilões, um para cada combinação de hotel/dia, que se encerram de forma aleatória a cada minuto. Apenas os hotéis podem vender reservas para quartos.

No TAC, existem três tipos de ingressos de entretenimento: Luta de Crocodilos, Parque de Diversão e Museu. Os agentes estão livres para negociar esses ingressos entre si através de leilões do tipo *Continuous Double Auction*. Logo, eles podem atuar como

compradores, ou vendedores. Os ingressos são negociados em 12 leilões: um para cada combinação tipo x dia[2].

A cada dia, está disponível um total de 8 ingressos para cada evento, totalizando 96 ingressos[3] negociados entre os 8 agentes. Cada agente recebe um conjunto inicial de 12 ingressos, que são distribuídos da seguinte forma: quatro ingressos de um evento para o dia 1 ou 4; dois ingressos de um evento (diferente do anterior) para o dia 1 ou 4; quatro ingressos de um evento para o dia 2 ou 3; dois ingressos de um evento (diferente do anterior) para o dia 2 ou 3.

Todos os leilões do TAC funcionam de acordo com o protocolo de alto-nível, cabendo a cada leilão definir quando e sob que condições uma transação ocorrerá. Assim, o agente submete uma oferta para um dado leilão; e, o leilão atualiza a cotação, indicando aos participantes os novos preços.

Ao final de cada partida, é calculada a pontuação de cada agente de acordo com uma função *utilidade*. A função *utilidade* mensura o quão próximo o pacote turístico criado pelo agente se aproximou do desejado pelo cliente. Baseado em (Stone e Greenwald, 2000) ela pode ser expressa da seguinte forma:

$$utilidade = 1000 - penalidadeViagem + bonusHotel + bonusIngresso \quad (1.1)$$

onde,

$$penalidadeViagem = 100 * (|IAD - AD| + |IDD - PD|) \quad (1.2)$$

$$bonusHotel = TT? * HB \quad (2.1.3)$$

$$bonusIngresso = AW? * AW + AP? * AP + MU? * MU \quad (1.3)$$

E, conforme descrito para a Tabela 1.1, IAD e IDD são os dias de partida e chegada ideiais. AD e PD são os dias de partida e chegada comprados pelo agente, e $TT?$, $AW?$, $AP?$ e $UM?$ são valores 0 ou 1 que indicam, respectivamente, se o cliente

[2] Um leilão para a negociação de ingressos do tipo AW no dia 1, outro para o mesmo tipo no dia 2, e assim sucessivamente.

[3] Sendo 8 ingressos para cada um dos 3 tipos, em cada um dos 4 dias, todos os dias exceto o 5º dia quando não ocorrem eventos; ou seja: 8 x 3 x 4 = 96.

será hospedado no *Tampa Towers*, se o cliente possuirá ingressos para únicos tipos de entretenimentos possíveis: Luta de Crocodilos (AW), Parque de Diversão (AP) e Museu (MU).

A pontuação final dos agentes é dada por:

$$pontuação = utilidade - custo - penalidade \quad (1.4)$$

onde, $custo$ refere-se ao valor de aquisição dos bens do pacote e $penalidade$ é uma punição de $200 para balanço negativo de ingressos de entretenimento, isto é, uma punição para cada ingresso que o agente vendeu sem possuí-lo

1.2. Leilões CDA

Segundo Klemperer (1999) a Teoria dos Leilões surgiu devido à aplicação prática e fundamentação teórica dos leilões. Atualmente, o mecanismo de leilão é bastante utilizado nos ambientes de comércio eletrônico; e, a infra-estrutura da Internet proporcionou uma redução nos custos para sua realização, conseqüentemente, aumentando o número de participantes que se utilizam dessa forma de negociação.

Dentre os tipos de leilões mais conhecidos estão: o Inglês, onde seu funcionamento se dá através de lances (abertos ou secretos) sempre com valores ascendentes e apenas o melhor lance é apresentado (KLEMPERER, 1999). O Holandês funciona como o contrário do leilão inglês, ou seja, o leiloeiro decrementa o preço do bem até algum comprador enviar um lance. No de lance fechado e primeiro preço, os lances são enviados em envelopes lacrados; todos os envelopes são abertos em conjunto e o bem será vendido pelo maior lance. O Vickrey diferencia-se do anterior pelo fato de em vez da primeira é a segunda oferta que arremata o bem. O Duplo, também conhecido como *Continuous Double Auction* (CDA) é o foco do atual trabalho e descrito com mais detalhes a seguir.

Em um CDA, os participantes submetem ofertas de compra e venda de bens no leilão, e caso uma oferta de compra combine com uma de venda, a transação é realizada. Contudo, para o total entendimento desse processo é necessário que se conheça o

mecanismo que define a natureza dos lances em válidos ou não. Tal mecanismo é descrito no protocolo de mercado (*market protocol*).

O *market protocol* é o protocolo que define o conjunto de regras para o processo de negociação utilizado em um mercado. Nele são descritas as regras de visibilidade e acesso das informações disponíveis entre os participantes, bem como a validade de uma transação e quando ela ocorre, em termos de valores para preço e quantidade (VYTELINGUM, 2006). Pode-se definir o mecanismo CDA por meio de um descritor e um protocolo de funcionamento como descrito a seguir.

A cada período de tempo, os agentes recebem o valor de cotação do preço de venda a_0, para cada leilão de ingressos de entretenimento; para um agente de venda submeter um lance de valor a, ele deve seguir algumas restrições. Para um agente de compra efetuar um lance de valor b, sendo b_0 o valor de cotação do preço de compra do bem g, ele também deve seguir o mesmo protocolo.

Segundo Oliveira (2008), o descritor de um CDA genérico voltado para o CDA do TAC, cenário de estudo do presente trabalho, pode ser descrito da seguinte forma:

$$P_{CDA} = < g, B, S, V_b, C_s, \Delta_{price}, t_{round} > \quad (1.5)$$

onde,

- g é o item em leilão.
- $B = \{b_1, b_2, \ldots, b_n\}$ é o conjunto finito de agentes compradores.
- $S = \{s_1, s_2, \ldots, s_m\}$ é o conjunto finito de agentes vendedores.
- $V_b = (V_1, \ldots, V_n)$, onde V_i $(v_{i1}, v_{i2}, \ldots, v_{in_i})$ é o vetor avaliação do item g para o agente b_i. Onde n_i é o número de unidades que b_i necessita, e v_{ij} é o valor de avaliação do item para j-ésima aquisição.
- $C_s = (C_1, \ldots, C_m)$, onde C_i $(c_{i1}, c_{i2}, \ldots, c_{im_i})$ é o vetor de custo do item g para o agente s_i. Onde m_i é o número de unidades que s_i deseja vender, e c_{ij} é o custo da j-ésima unidade.
- Δ_{price} é o menor valor de incremento/decremento de preços. Toda variação de ofertas devem ser através de múltiplos de $\Delta_{price.}$

- t_{round} determina a condição de parada do CDA, por exemplo, se não houver uma nova oferta de compra e venda durante um período t_{round} o CDA termina.
- Relacionando este descritor ao CDA do TAC temos alguns fatores importantes:
- g pertencente ao produto cartesiano $\{AW, AU, MU\} \times \{Dia\ 1, Dia\ 2, Dia\ 3, Dia\ 4\}$.

Todos os agentes pertencem a B e a S[4], portanto, para efeitos de simplificação, consideramos que B = S, V_b e C_s são determinados a partir das preferências dos clientes, $\Delta_{price} = 0.01$, representando uma variação em centavos, $t_{round} = 9$ minutos, definindo o tempo total do jogo, não o tempo de inatividade. Baseado em Oliveira (2008), o protocolo de funcionamento do CDA, também voltado ao cenário do TAC, é descrito de maneira mais detalhada, conforme apresentado a seguir:

1. O CDA é iniciado, $a_o = 200$ e $b_o = 0$, $t_{round} = 0$ minutos.
2. A um determinado período de tempo Δ_t o agente recebe informações sobre as cotações de compra e venda de um bem g:
   a. Quando uma nova oferta de venda ***a*** é submetida:
      i. Se $a \geq a_o$ então a fica armazenado no servidor, mas o valor de cotação continua sendo a_o;
      ii. Se $b_o < a < a_o$ então a_o é atualizado para a;
      iii. Se $a \leq b_o$, uma transação é efetivada e b_o retorna para o valor de sua última atualização;
   b. Quando uma nova oferta de compra ***b*** é submetida:
      i. Se $b \leq b_o$ então b fica armazenado no servidor, mas o valor de cotação continua sendo a_o;
      ii. Se $b_o < b < a_o$ então b_o é atualizado para b;
      iii. Se $b \geq a_o$, uma transação é efetivada e a_o retorna para o valor de sua última atualização.
3. O passo anterior se repete até $t_{round} = 9$ minutos.

Ao iniciar o CDA, a maioria dos agentes que possuem bens a serem vendidos inicia os lances de venda a \$200, e os de compra a \$0. Posteriormente, novos lances são

[4] Mesmo que este não possua o item em leilão, a dinâmica do jogo permite que ele efetue a venda do item e o adquira apenas em um momento posterior. Caso o leilão seja encerrado e o agente não possua o item vendido, ele estará sujeito a uma punição.

enviados ao servidor TAC, onde são repassadas a todos os agentes as atualizações das cotações. Cabendo ao servidor decidir os passos 2.a e 2.b., até o prazo de fechamento dos leilões, ou seja, 9 minutos de jogo.

1.3.Agentes de Software

De acordo com Russell & Norvig (1995) “um agente é tudo o que pode ser considerado capaz de perceber o ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores”. Fazendo uma analogia com um agente humano, os olhos, ouvidos e outros órgãos seriam os sensores e mãos, pernas, boca e outras partes do corpo seriam os atuadores, vide Figura 1.2.

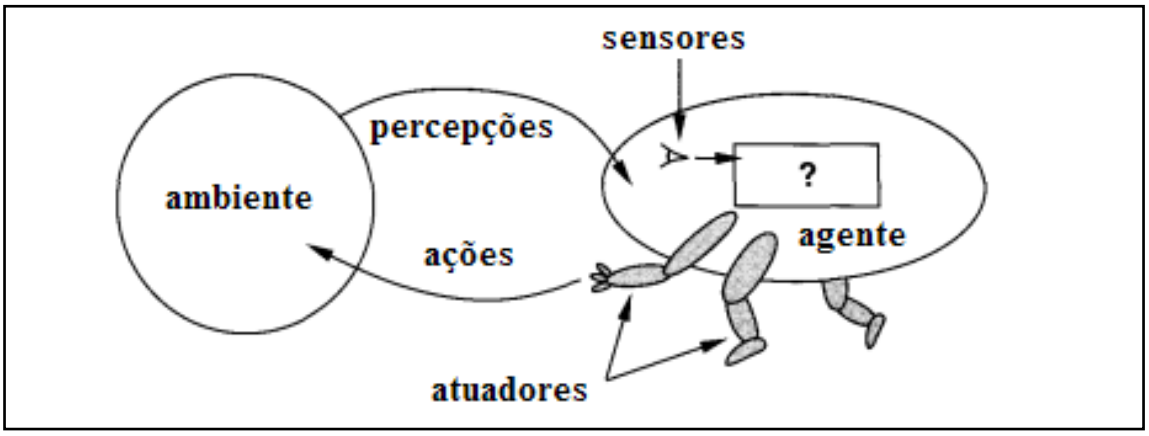

Figura 1.2. Modelo de Agente visualizado por Russell & Norvig (1995).

Segundo Weiss (2000) um agente que possui características de flexibilidade de ações tais como reatividade, pró-atividade e sociabilidade, pode ser classificado como um agente inteligente. Já para Russell & Norvig (1995) um agente, para ser dito racional, por definição, deve para cada seqüência de percepções possíveis selecionar uma ação que se espera que venha maximizar sua medida de desempenho, dada a evidência fornecida pela mesma seqüência e por qualquer conhecimento interno ao agente.

Conforme discutido por Néri (2005), não existe um consenso quanto à definição do que são agentes de software, contudo, de forma geral, eles podem ser considerados

programas aos quais são delegadas tarefas, entendendo-se delegação como uma transferência de tomada de decisão.

A disciplina Arquitetura de Software nasceu através do esforço de pesquisadores em Engenharia de Software para apoiar o desenvolvimento estruturado de software. Ela funciona como uma ponte entre requisitos e código, além de prover uma descrição abstrata dos sistemas de software, expondo certas propriedades enquanto escondem outras (BASS et al., 1996) (SHAW; GARLAN, 1996).

Russell & Norvig (1995) consideram que um agente é formado por uma arquitetura mais um programa agente. O programa agente consiste em uma implementação em linguagem de programação do mapeamento entre o conjunto de percepções possíveis e o conjunto de ações possíveis para o agente. A arquitetura do agente é o dispositivo computacional responsável por disponibilizar as percepções do ambiente que chegam pelos sensores para o programa, de executar o mesmo e de alimentar as ações selecionadas para os atuadores do agente. Além disto, especificou as estruturas de cinco tipos de agentes, adequados a diferentes propriedades dos ambientes de realização de tarefas dos agentes. A seguir são detalhadas as estruturas de dois agentes, cujos módulos contribuíram para a concepção do agente proposto neste trabalho.

Os Agentes reativos simples são agentes que reagem diretamente a estímulos que chegam do ambiente. Os Agentes reativos baseados em modelos, ou estados são agentes que mantêm a informação sobre o ambiente para controlar aspectos do mundo que não estão evidentes na percepção atual. A Figura 1.3 ilustra a estrutura de um agente reativo baseado em estados.

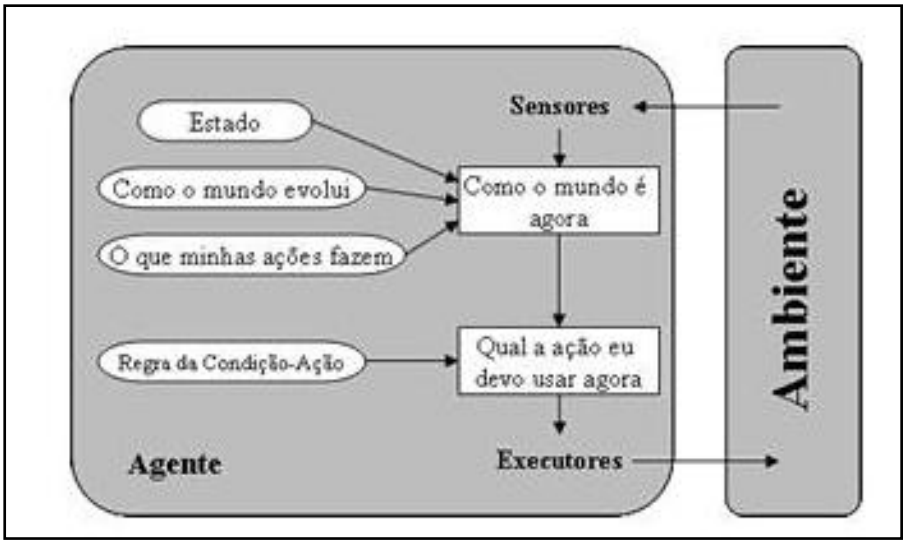

Figura 1.3. Modelo arquitetural de um agente baseado em estados. Fonte Russell & Norvig (1995).

O agente DealerBot, implementado por Oliveira (2008), utiliza o modelo arquitetural de agente baseado em estados. Nele, as regras condição-ação são regras Fuzzy, e seu estado interno é guardado em suas bases de conhecimento, como a *EntertainmentKnowledge*. O estado vai sendo atualizado à medida que as cotações de compra e venda vão se alterando e as negociações vão ocorrendo. Todos os elementos que fazem parte da condição nas regras são armazenados no estado interno em um tempo t corrente.

Os Agentes baseados em objetivos são agentes que armazenam informações sobre o ambiente e agem para alcançar seus objetivos de utilidade para maximizar seu grau de "felicidade". Algumas vezes tal objetivo pode ser alcançado simplesmente com uma ação, outras vezes podem ser necessárias longas seqüências de ações.

Os Agentes orientados por utilidade (Figura 1.4) se preocupam com o grau de felicidade alcançado. Isto facilita a tomada de decisões quando há objetivos conflitantes, e que somente um deles pode ser alcançado ou quando há vários objetivos mas nenhum pode ser alcançado de maneira precisa. Assim, a utilidade irá auxiliar o agente na busca do que indicar um melhor grau de felicidade.

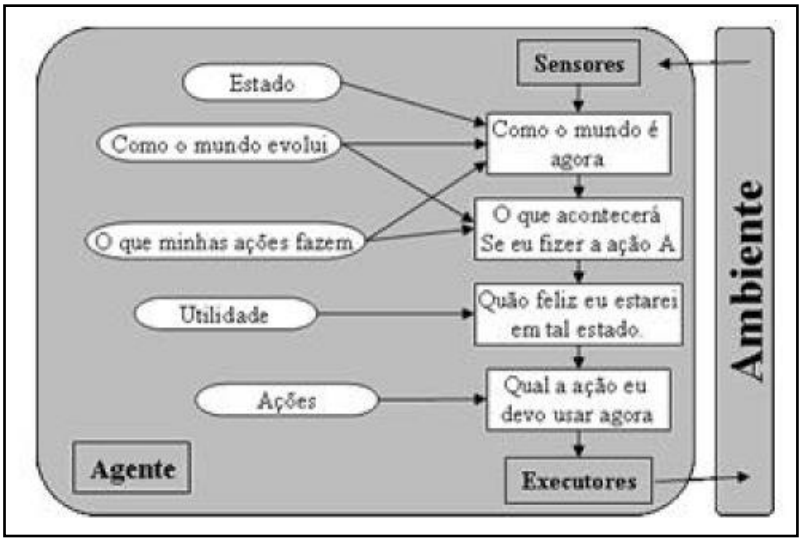

Figura 1.4. Modelo arquitetural de um agente baseado em utilidade. Fonte Russell e Norvig (1995).

O modelo de agente baseado em utilidade tem grande relevância para esse trabalho, uma vez que o módulo Alocador faz uso de uma função utilidade para auxiliar a busca do LaconiBot sobre qual leilão participar. Dessa forma, as ações do agente são orientadas no sentido de escolher em qual leilão participar com o objetivo de aumentar seu grau de "felicidade".

1.4.Técnicas de Inteligência Artificial

Para apoiar a solução dos diversos problemas listados no capítulo Introdução o agente foi concretizado por meio de três técnicas de Inteligência Computacional bastante utilizadas, a saber: os Sistemas Fuzzy, as Redes Neurais e os Algoritmos Genéticos.

1.4.1. Sistemas *Fuzzy*

Em meados da década de 60, Lotfi Zadeh, professor da Universidade da Califórnia em Berkeley, propôs a lógica nebulosa (ZADEH, 1965), ou lógica Fuzzy, com o objetivo de tratar o julgamento subjetivo inerente ao homem. Ele observou que as pessoas são capazes de chegar a conclusões, mesmo não conseguindo explicar bem o processo de formação das regras utilizadas no seu processo de inferência. Por exemplo,

é possível se chegar à conclusão que uma pessoa tem aproximadamente 30 anos, apenas olhando para ela.

Conjuntos Nebulosos e Lógica Fuzzy

A lógica Fuzzy é uma generalização da lógica clássica proposta por Aristóteles e formalizada matematicamente por Boole, em 1854, com a lógica Booleana. A grande diferença entre ambas está na classificação dos elementos em um conjunto, enquanto a lógica Booleana utiliza apenas dois possíveis valores (0 ou 1). Ou seja, com a lógica Fuzzy é possível classificar os elementos por meio do seu grau de pertinência ao conjunto, podendo assumir qualquer valor real no intervalo [0,1] incorporando, assim, a possível imprecisão na classificação desses elementos.

A Teoria dos Conjuntos Nebulosos criada por Zadeh é um meio de especificar o quanto um elemento satisfaz um conceito impreciso. Por exemplo, para representar o conceito "*Alto*", pode-se afirmar que uma pessoa com altura de 1,80 metro é alta? A resposta de muitas pessoas poderia ser "mais ou menos" em vez de "sim" ou "não" (RUSSELL; NORVIG, 1995). Utilizando a teoria clássica dos conjuntos, o conjunto alto poderia ser descrito como "o conjunto das pessoas com altura superior ou igual a 1,90 metro". Mas o que dizer das pessoas com 1,89 metro: elas pentenceriam ao conjunto alto?

Contudo, o termo lingüístico *Alto* implica em uma classificação imprecisa, pois se pode considerar uma pessoa com 1,90m alta, e outra com 1,89m um pouco menos alta, porém ainda alta. Dessa forma, o conjunto das pessoas altas remete a imprecisão capaz de ser representada por conjuntos nebulosos.

"Um conjunto nebuloso (*Fuzzy set*) é uma classe de objetos com um intervalo contínuo de vizinhança" (Zadeh, 1965). Conjuntos nebulosos são uma extensão do conceito de conjunto clássico. Seus elementos possuem um grau de pertinência no conjunto. Isto o difere do conjunto clássico, onde a participação dos elementos é sempre binária: 1 ou 0 (participação total ou nenhuma participação, respectivamente). A Figura 1.5 (a) mostra como exemplo o conjunto *Alto* representado por um conjunto clássico. A Figura 1.5 (b) mostra o mesmo conjunto representado por um conjunto nebuloso.

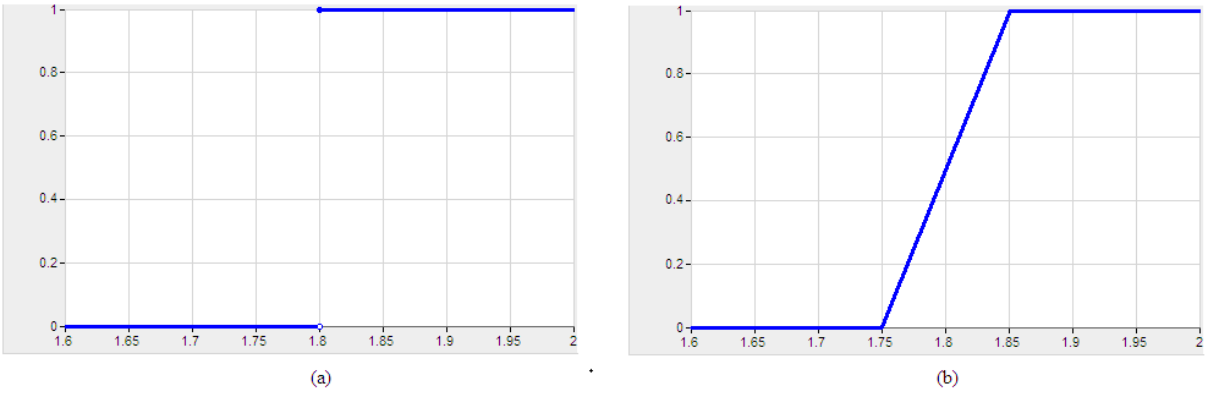

Figura 1.5. Alto (a) Representação clássica (b) Representação nebulosa.

Formalmente, um conjunto nebuloso A do universo de discurso X é definido por uma função de pertinência $\mu_A: X \rightarrow [0,1]$, que associa cada elemento de X com um número real no intervalo [0,1]. Seja um elemento genérico de X denotado por x, $\mu_A(x)$ define o grau com que x pertence ao conjunto A (ZADEH, 1965). Em outras palavras, a função de pertinência $\mu_A(x)$ indica o grau de compatibilidade entre x e o conceito expresso por A (SANDRI; CORREA, 1999), onde:

$\mu_A(x) = 1$ indica que *x* é completamente compatível com *A*;

$\mu_A(x) = 0$ indica que *x* é completamente incompatível com *A*;

$0 < \mu_A(x) < 1$ indica que *x* é parcialmente compatível com *A*, com grau $\mu_A(x)$.

As operações de complemento ($\bar{A}$), união ($A \cup B$) e interseção ($A \cap B$) são originalmente definidas como:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$
$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$
$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

No entanto, estas definições são apenas uma dentre várias possibilidades. A Tabela 1.2 ilustra as definições de união e interseção mais utilizadas.

Tabela 1.2. Operadores de união e interseção mais utilizados.

Nome	$\mu_{A\cap B}(x)$	$\mu_{A\cup B}(x)$
Probabilista	$\mu_A(x)\cdot\mu_B(x)$	$\mu_A(x)+\mu_B(x)-\mu_A(x)\cdot\mu_B(x)$
Lukasiewicz	$\max(\mu_A(x)+\mu_B(x)-1,0)$	$\min(\mu_A(x)+\mu_B(x),1)$
Weber	$\begin{cases}\mu_A(x), se\ \mu_B(x)=1\\ \mu_B(x), se\ \mu_A(x)=1\\ 0, senão\end{cases}$	$\begin{cases}\mu_A(x), se\ \mu_B(x)=0\\ \mu_B(x), se\ \mu_A(x)=0\\ 0, senão\end{cases}$

Fonte: Sandri & Correa (1999)

Outro operador importante é o operador de implicação. Estes operadores são usados para modelar regras de inferência do tipo se-então entre conjuntos nebulosos. As principais definições para estes operadores são mostradas na Tabela 1.3.

Tabela 1.3. Operadores de implicação mais utilizados.

Nome	**Implicação ($\mu_{A\to B}(x,w)$)**
Kleen-Diemes	$max\big(1-\mu_A(x),\mu_B(w)\big)$
Lukasiewicz	$min(1-\mu_A(x)+\mu_B(w),1)$
Rescher-Gaines "Sharp"	$\begin{cases}1, & se\ \mu_A(x)\le\mu_B(w)\\ 0, & senão\end{cases}$
Brower-Gödel	$\begin{cases}1, & se\ \mu_A(x)\le\mu_B(w)\\ \mu_B(w), & senão\end{cases}$
Goguen	$\begin{cases}min(\mu_A(x)/\mu_B(w)), & se\ \mu_A(x)\neq\mu_B(w)\\ 1, & senão\end{cases}$
Reichenbach "Estocástica"	$1-\mu_A(x)+\mu_A(x)\mu_B(w)$
Zadeh-Wilmott	$max\left(1-\mu_A(x),min\big(\mu_A(x),\mu_B(w)\big)\right)$
Mamdani	$min\big(\mu_A(x),\mu_B(w)\big)$
Larsen	$\mu_A(x)\mu_B(w)$

Fonte: Sandri & Correa (1999)

Seguindo a continuidade natural de aplicabilidade dos conjuntos, Zadeh desenvolveu um método de raciocínio lógico envolvendo os conjuntos Fuzzy, o qual denominou Lógica Nebulosa ou Lógica Fuzzy. A lógica Fuzzy é uma variação da lógica clássica onde são admitidos valores lógicos intermediários aos clássicos valores verdadeiro e falso. Mais especificamente, a lógica Fuzzy é um mecanismo de raciocínio

sobre expressões lógicas onde as proposições envolvem os graus de pertinência de elementos do universo de discurso em conjuntos nebulosos (RUSSELL; NORVIG, 1995). A correspondência entre lógica e conjuntos nebulosos, assemelha-se à que ocorrem entre lógica e conjuntos clássicos.

Números Fuzzy

Os números Fuzzy foram criados com o objetivo de trazer incerteza aos números reais, visando contornar a exatidão, e conseqüente limitação, ao se trabalhar com os mesmos através da lógica Fuzzy. Para isso, um número Fuzzy é definido como um conjunto nebuloso com as seguintes características (LEE, 2004): i) definido em $\mathbb{R}$; (ii) normalizado e convexo; (iii) possui função de pertinência contínua.

Uma representação mais comum para números Fuzzy é o número Fuzzy triangular. Ele é formalmente definido como uma tripla (HE et. al, 2003): $\alpha = (m, \theta, \chi)$ onde m é o centro, θ e χ são os intervalos direito e esquerdo ao centro, respectivamente. Essa representação é interpretada como a função de pertinência ilustrada na Figura 1.6.

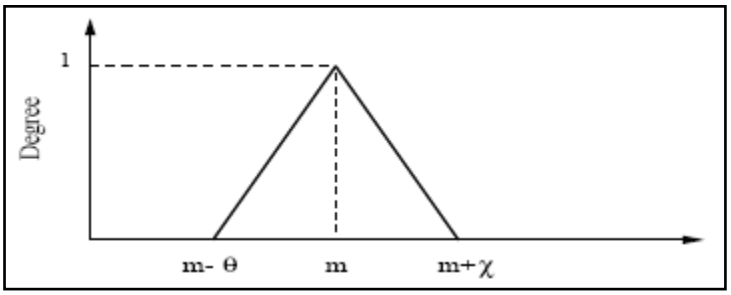

Figura1.6. Representação de um número Fuzzy triangular. Fonte: He et al. (2003).

Seja $\alpha_1 = (m_1, \theta_1, \chi_1)$ e $\alpha_2 = (m_2, \theta_2, \chi_2)$ números Fuzzy triangulares e $k \in \mathbb{R}$, as seguintes operações são validas sobre este conjunto (HE; LEUNG; JENNINGS, 2003, p. 3):

$$\alpha_1 + \alpha_2 = (m_1 + m_2, \theta_1 + \theta_2, \chi_1 + \chi_2)$$
$$\alpha_1 - \alpha_2 = (m_1 - m_2, \theta_1 + \theta_2, \chi_1 + \chi_2)$$
$$\alpha_1 \times \alpha_2 = (m_1 m_2, m_1\theta_2 + m_2\theta_1 - \theta_1\theta_2, m_1\chi_2 + m_2\chi_1 + \chi_1\chi_2)$$
$$k \times \alpha_1 = (km_1, k\theta_1, k\chi_1)$$

Métodos de Inferência

Os métodos de inferência existentes se diferem em alguns aspectos como: à forma de representação dos termos na premissa; à representação das ações de controle; e, aos operadores utilizados para execução do método (SANDRI;CORREA, 1999).

Método de Inferência de Sugeno

No método de Takagi-Sugeno, ou simplesmente Sugeno, cada conclusão é dada através de uma função estritamente monotônica, usualmente diferente para cada regra se-então. Para cada uma dessas regras, obtém-se um único valor para sua variável de controle. Finalmente, uma ação global é calculada através de uma média ponderada dos valores individuais obtidos, onde o peso é o próprio grau de compatibilidade da premissa da regra e das entradas (SANDRI; CORREA, 1999).

Por exemplo, considerando o bloco de regras se-então (HE et al., 2003), a seguir:

R_1: SE x é A_1 E y é B_1 ENTÃO $z_1 = c_1$
R_2: SE x é A_2 E y é B_2 ENTÃO $z_2 = c_2$
⋮
R_n: SE x é A_n E y é B_n ENTÃO $z_n = c_n$

Fato: x é x_0 E y é y_0
_____
Conseqüência: z_0

onde A_1, ..., A_n e B_1, ..., B_n são conjuntos nebulosos e z_1,..., z_n são números reais. Seja α_i o grau de compatibilidade da regra R_i, e o operador E calculado pela função min, temos:

$$\alpha_i = \min\left(\mu_{A_i}(x), \mu_{B_i}(y)\right) \quad (1.6)$$

O resultado da combinação das regras de inferência é dado pela fórmula:

$$z_0 = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i} \quad (1.7)$$

He et al. (2003) estenderam a fórmula anterior para a situação onde os valores reais z_i foram substituídos por números Fuzzy triangulares $\tilde{z}_i$. Segundos os autores, essa

modificação evitou a dificuldade de definir uma boa ação de controle a partir de valores reais fixos.

Controladores Nebulosos

Dos estudos e projetos de E. H. Mamdani surge o conceito de controlador nebuloso. O objetivo era tornar prática a teoria da lógica Fuzzy de Zadeh.

> "Ao contrário dos controladores convencionais, em que o algoritmo de controle é descrito analiticamente por equações algébricas ou diferenciais, através de um modelo matemático, um controlador nebuloso utiliza-se de regras lógicas no algoritmo de controle, com a intenção de descrever em uma rotina a experiência humana, intuição e heurística para controlar um processo" (SANDRI; CORREA, 1999).

Controladores Fuzzy são especialmente indicados em duas situações: (i) quando o processo é significativamente complexo para ser analisado por técnicas convencionais; e (ii) quando as informações devem ser interpretadas de forma qualitativa, inexata ou incerta (LEE, 2004).

A estrutura básica de um controlador nebuloso está ilustrada na Figura 1.7. Ele pode ser dividido em quatro partes fundamentais: interface de *fuzzificação*, base de conhecimento, módulo de inferência e interface de *defuzzificação*.

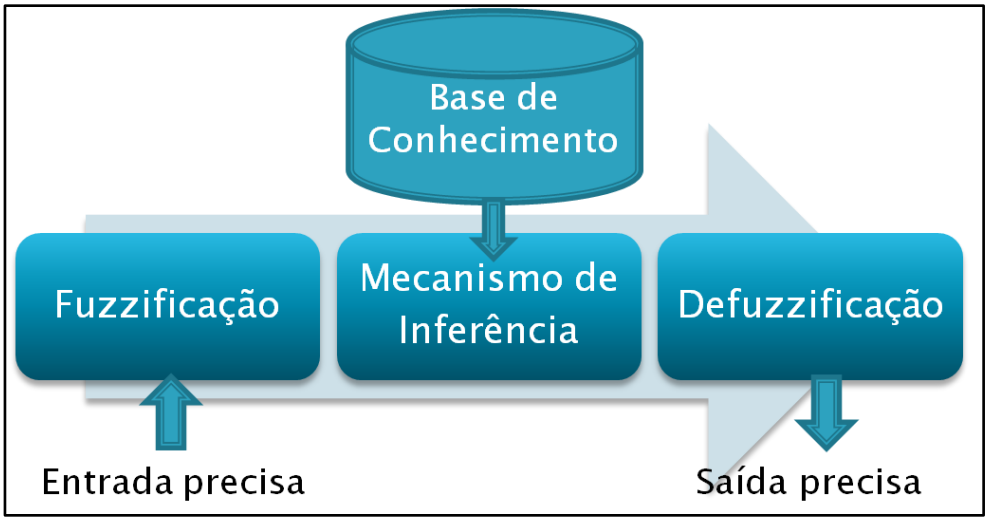

Figura 1.7. Estrutura básica de um controlador nebuloso.

A interface de *fuzzificação* transforma os valores numéricos vindos do processo em valores Fuzzy (lingüísticos). A base de conhecimento armazena informações sobre dados (normalização dos universos de discurso, definições das funções de pertinência, etc.) e regras (estruturas do tipo se-então) de forma a orientar a estratégia de controle e suas metas. O módulo de inferência utiliza os dados e regras da base de conhecimento, juntamente com os dados vindos da interface de *fuzzificação*, para inferir a ação de controle a ser tomada, simulando uma decisão humana com base em conceitos Fuzzy. Por fim, a interface de *defuzzificação* traduz a saída do módulo de inferência para uma ação ordinária aplicável ao processo.

Os controladores nebulosos são tipicamente usados para controle de características físicas, químicas e numéricas em geral, como controle de temperatura e pressão, de corrente elétrica, de movimento de máquinas e de quantificadores financeiros (como fluxo de caixa), entre outros (MUNAKATA, 2007). Dessa forma, esse tipo de controlador adquiriu uma vasta gama de aplicações, indo do uso em reatores nucleares ao uso em máquinas de lavar domésticas, mostrando empiricamente sua eficiência para tratar a incerteza envolvida nestes problemas.

1.4.2. Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) foram inspiradas nos neurônios biológicos. A idéia para a sua concepção foi simular o comportamento neural de organismos inteligentes através de um modelo matemático. Conforme ilustrado na Figura 1.8, um neurônio biológico é formado por um corpo celular ou soma que contém o núcleo da célula, dendritos, através dos quais impulsos elétricos são recebidos, e um axônio, por onde os impulsos elétricos são enviados.

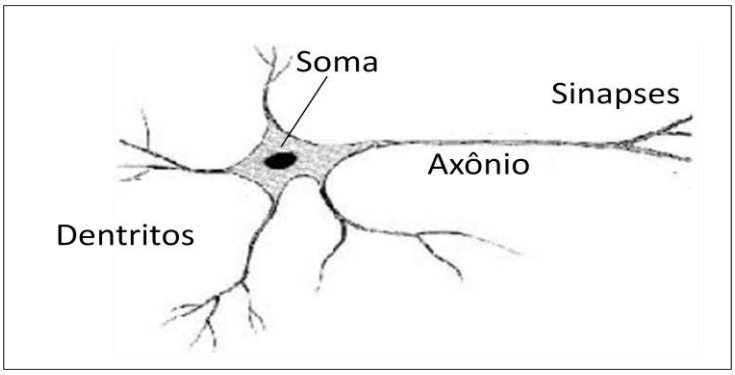

Figura 1.8. Neurônio biológico.

Segundo Kohonen (1988), "Redes Neurais Artificiais são redes massivamente paralelas interconectadas de elementos e suas organizações hierárquicas que, são voltadas para interagir com objetos do mundo real do mesmo modo que um sistema nervoso biológico faz". Elas têm se mostrado um modelo eficaz na criação de programas de computador que possam simular os circuitos neurais biológicos, auxiliando a solução de diversos problemas como: a capacidade de tomada decisões em situações inesperadas baseada em um conhecimento adquirido por experiência de situações semelhantes, a possibilidade de auto-organização e o aprendizado de novas tarefas.

Características Gerais

Em meados de 1940, foi desenvolvido um modelo matemático utilizado para simular o neurônio biológico conhecido como neurônio de McCulloch-Pitts (HAYKIN, 2001), conforme ilustrado na Figura 1.9.

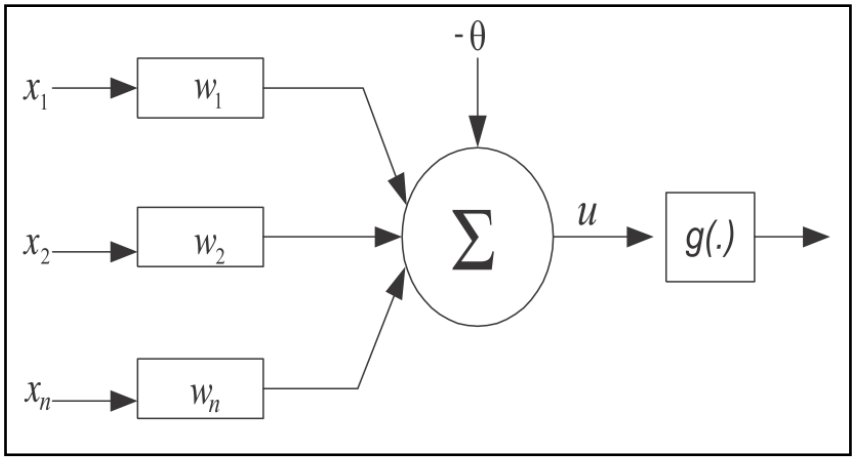

Figura 1.9. O neurônio artificial. Fonte: (SOARES, 2008).

Nesse modelo, os neurônios possuem em cada uma de suas entradas (x_1, x_2, x_n) pesos a elas associados (w_1, w_2, w_n), que são utilizados para armazenar o conhecimento adquirido na etapa de aprendizagem. O θ é o limiar de ativação, definindo se a saída será disparada. Já o *u* é o combinador linear formado pela função:

$$u = \sum_{i=1}^{N} w_i \times x_i - \theta \quad (1.8)$$

O $g(.)$ é a função de ativação da saída do combinador linear que processa o conjunto de entradas recebidas e as transforma em estados de ativação. A Figura 1.10 ilustra as principais funções de ativação utilizadas em RNAs.

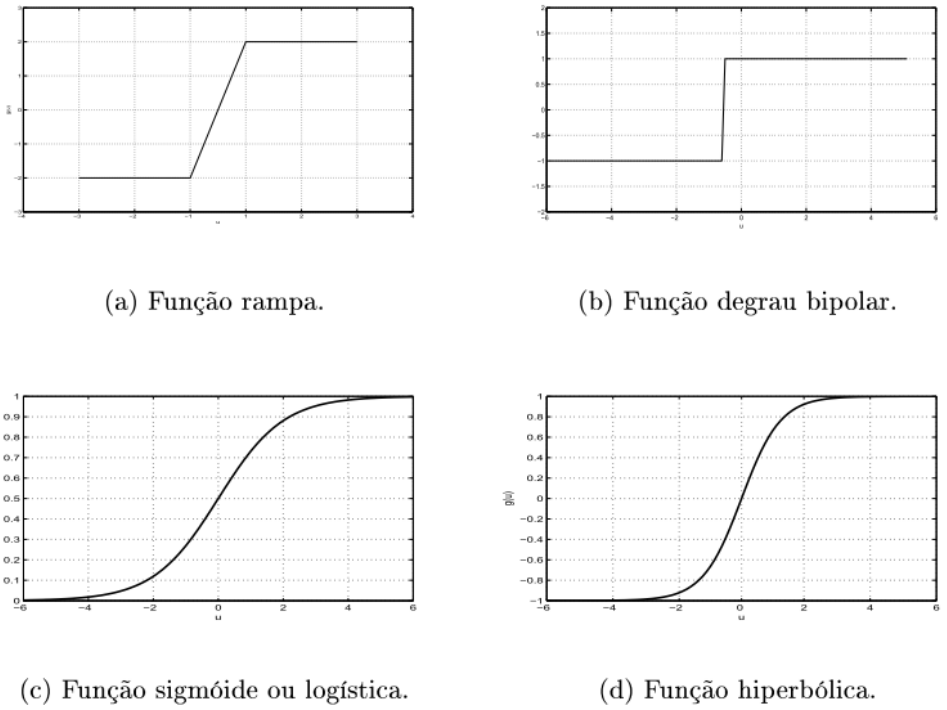

Figura 1.10. Principais funções de ativação utilizadas em RNAs. Fonte (SOARES, 2008).

As RNAs são compostas por neurônios artificiais conectados via sinapse. Nos neurônios artificiais são executadas operações baseadas em uma função de transferência e geralmente cálculos de média ponderada. Algumas RNAs são formadas por mais de uma camada de neurônios. Um tipo bastante conhecido de RNA multi-camada é a MLP (*Multilayer Perceptrons)*. As MLPs são formadas por uma camada de entrada, uma camada de saída e n camadas intermediárias; onde seus neurônios são completamente conectados (*Fully Connected*).

Para que uma RNA possa produzir resultados satisfatórios é necessário que ela seja treinada a reconhecer os valores que recebe. Esse treinamento pode ser realizado com a ajuda de um treinador ou não. Quando existe a presença de um "professor", onde a RNA é avaliada quanto aos resultados por meio de um conjunto de exemplos conhecidos, diz-se que o aprendizado é supervisionado. Já o aprendizado não-supervisionado ocorre quando a rede é treinada de forma a aprender de acordo com o ambiente e com a experiência histórica de padrões semelhantes. Os algoritmos de aprendizado não-supervisionados mais comuns são o algoritmo de Hopfield (1982) e os mapas de Kohonen (1988).

Redes *Perceptron*

Um dos primeiros modelos de redes neurais que utilizou o neurônio de McCulloch-Pitts foi o *perceptron.* Essa rede, originalmente com uma estrutura de três camadas, é formada por uma camada de entrada que é responsável pelos sinais recebidos para todas as camadas de processamento; e os neurônios compostos por uma unidade de função que calcula a soma ponderada dos sinais vindos da camada de entrada e uma função de transferência, conforme ilustrado na Figura 1.9.

Segundo Kasabov (1996), a primeira camada de um *perceptron* é a camada de *buffer*, onde os dados são armazenados. A camada de entrada é conectada totalmente aos elementos da segunda camada ou camada de características. Nessa camada, cada neurônio atua combinando as informações vindas dos elementos da camada de entrada. Da mesma forma, a camada interna é totalmente conectada a camada de saída, também chamada camada de *perceptrons*. A conexão entre a camada de buffer e a camada de características é feita com a presença de pesos. Dessa forma, o treinamento de um *perceptron* ocorre conforme ilustrado na Tabela 1.4.

Tabela 1.4. Passos para o treinamento de um perceptron.

1. Arbitrar os pesos existentes nas conexões entre a camada de entrada e a camada de características (2ª camada).
2. Executar um padrão de treinamento à camada de entrada e calcular a soma ponderada de acordo com os valores advindos das conexões.
3. Aplicar a função de transferência ao resultado da soma ponderada.
4. Se:
    a. A saída estiver correta, voltar ao passo 2.
    b. A saída estiver incorreta e for 0, adicionar a cada peso das conexões o sinal de entrada relativo.
    c. A saída estiver incorreta e for 1, subtrair de cada peso das conexões o sinal de entrada relativo
5. Voltar ao passo 2.

Algoritmo de retro propagação

A aprendizagem de uma RNA *perceptron* multi-camadas (MLP) é feita de acordo com alguma regra ou método de treinamento de seus neurônios. Um método largamente utilizado em redes MLP é o de retro propagação (*backpropagation)* (RUMELHART, 1986). O algoritmo de retro propagação é um método de aprendizado supervisionado que consiste em treinar a rede a partir de uma comparação dos resultados obtidos com os dados desejados, propagando os ajustes nos neurônios desde a última camada até a camada de entrada da rede.

O algoritmo de retro propagação é composto de duas etapas:

- Na primeira etapa, etapa de propagação, após a apresentação de um padrão à rede, todas as unidades (neurônios) processam, camada por camada, o resultado até a camada de saída. Nesse passo, não ocorre alteração nos pesos sinápticos, apenas é calculada a diferença entre a resposta esperada e a saída produzida conforme a equação 1.9.

$$E = \frac{1}{2} \sum_{n} \sum_{i=1}^{p} (x_i - \widehat{x_i})^2 \quad (1.9)$$

onde, *E* é a função da medida de erro total da rede, *n* é o numero de padrões apresentados, *p* é o numero de neurônios na saída, x_i é o valor desejado e $\widehat{x_i}$ é o valor gerado pela rede.

- Após a geração do resultado, inicia-se a segunda etapa que consiste na observação do erro comparando os resultados obtidos com os resultados desejados. Em seguida, os erros são propagados a partir da camada da saída até a camada de entrada de modo a ajustar os pesos de acordo com uma regra de correção definido na equação 1.10.

$$\Delta w_{ij} = -\eta \times \left(\frac{\delta E}{\delta w_{ij}} \right) \quad (1.10)$$

onde, i e j são os índices para o neurônio i da camada posterior, e neurônio j da camada anterior. A taxa de aprendizado é dada por η, ou a velocidade do passo em direção a minimização do erro *E*.

1.4.3. Algoritmos Genéticos

Algoritmos Genéticos (AGs) foram inicialmente propostos por John Holland (1975). Eles são uma estratégia para a resolução de problemas combinatórios complexos através de uma busca adaptativa. A idéia por trás dos AGs é fazer uma analogia com a evolução biológica, por meio da seleção natural de Darwin. Nestes algoritmos, uma população de indivíduos (soluções potenciais) sofre uma série de transformações unárias (mutação) e de ordem mais alta (cruzamento). Estes indivíduos competem entre si pela sobrevivência: um esquema de seleção, que favorece os indivíduos mais aptos seleciona os que sofrerão transformações, dando origem à próxima geração. Depois de algumas gerações, o algoritmo usualmente converge e o melhor indivíduo representa uma solução próxima da ótima.

Os AGs empregam uma estratégia de busca paralela e estruturada, porém aleatória, que é voltada em direção ao reforço da busca de pontos de maior aptidão. Ou seja, pontos nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos). E, apesar de aleatórias, a busca guarda informações históricas para encontrar novos pontos onde são esperados melhores desempenhos. Isto é feito através de processos iterativos, onde cada iteração é chamada de geração.

Durante cada iteração, os princípios de seleção e reprodução são aplicados a uma população de candidatos que pode variar dependendo da complexidade do problema e dos recursos computacionais disponíveis. Através da seleção, se determina quais indivíduos conseguirão se reproduzir, gerando um número determinado de descendentes para a próxima geração, com uma probabilidade determinada pelo seu índice de aptidão. Logo, os indivíduos com maior adaptação relativa têm maiores chances de se reproduzir e permanecer de alguma forma nas futuras soluções.

Como ilustra o fluxograma da Figura 1.11. O primeiro passo em um algoritmo genético é a representação dos problemas através de soluções genotípicas, utilizando vetores de tamanho finito em um alfabeto finito. Nela, os indivíduos são representados

por vetores binários, onde cada elemento de um vetor denota a presença (1) ou ausência (0) de uma determinada característica: o seu genótipo. Os elementos podem ser combinados formando as características reais do indivíduo, ou o seu fenótipo. Esta representação é independente do problema, pois uma vez encontrada a representação em vetores binários, as operações padrão podem ser utilizadas, facilitando o seu emprego em várias classes de problemas.

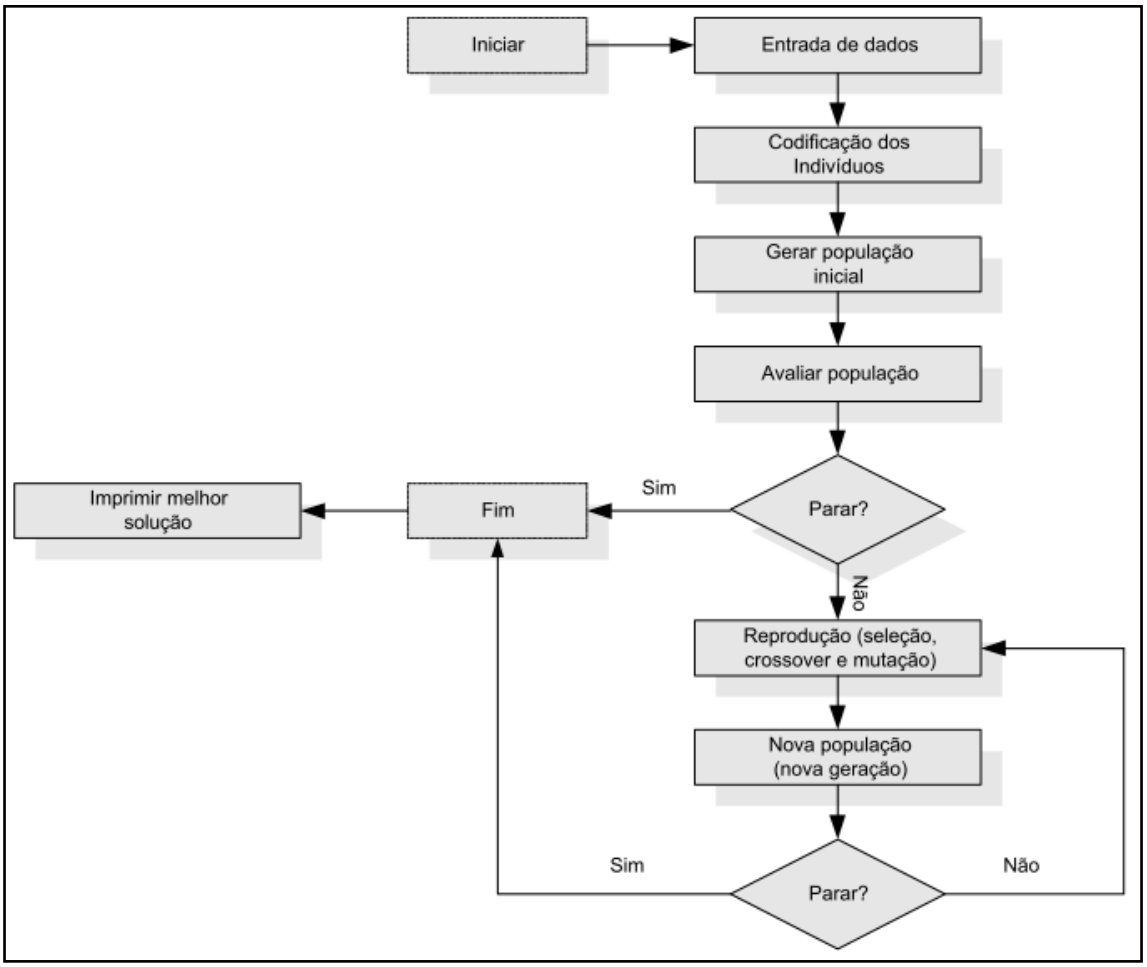

Figura 1.11. Fluxograma de um Algoritmo Genético. Fonte: (RODRIGUES et al., 2004).

Um método de seleção utilizado em AGs é o Método da Roleta. Nele, os indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta. Dessa forma, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão. Assim, os indivíduos com alta aptidão ocupam uma porção maior da roleta. Após a roleta girar um determinado número de vezes, dependendo do tamanho da população, os indivíduos sorteados são

escolhidos para participar da próxima geração. E, a forma de avaliar o grau de aptidão de uma solução é dada por uma função denominada *fitness*.

Indivíduo S_i	Aptidão $f(S_i)$	Aptidão Relativa
S_1 10110	2.23	0.14
S_2 11000	7.27	0.47
S_3 11110	1.05	0.07
S_4 01001	3.35	0.21
S_5 00110	1.69	0.11

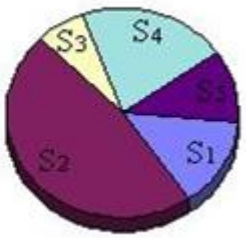

Figura 1.12. Indivíduos de uma população e a sua correspondente roleta de seleção. Fonte: (NORONHA et al., 2001).

Operadores Genéticos

Segundo Noronha et al. (2001) "O princípio básico dos operadores genéticos é transformar a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório". Em termos biológicos, os operadores genéticos são necessários para que a população se diversifique, mantendo as características de adaptação adquiridas pelas gerações anteriores.

A alteração de um ou mais componentes de um gene, é denominada operação de mutação. A mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será zero. Ela também contorna o problema de mínimos locais, alterando levemente a direção da busca. A Figura 1.13 ilustra um exemplo de mutação.

Antes da Mutação:	1 1 (1) 0 0
Depois da Mutação:	1 1 0 0 0

Figura 1.13. Exemplo de mutação. Fonte: (NORONHA, 2001).

Outro operador utilizado é o cruzamento. Através dele é possível recombinar características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso é aplicado com probabilidade maior que a taxa de mutação.

O operador de cruzamento pode ser utilizado de diversas formas. Por exemplo: no cruzamento *um-ponto* é escolhido um ponto onde a partir deste, as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto em um dos pais são ligadas às informações posteriores a este ponto no outro pai, conforme ilustrado na Figura 1.14. O *multi-pontos* é uma generalização da abordagem *um-ponto* onde ocorre a troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados. Já o *uniforme* não utiliza pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

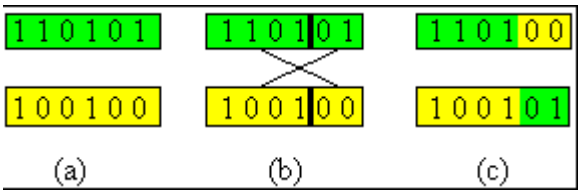

Figura 1.14. Crossover de um ponto. (a) dois indivíduos são escolhidos.(b) um ponto (4) de crossover é escolhido. (c) são recombinadas as características, gerando dois novos indivíduos.

Parâmetros Genéticos

O tamanho da população afeta diretamente no desempenho do AG, em termos de qualidade das soluções. Uma população pequena fornece uma pequena cobertura do espaço de busca do problema. Já uma grande população geralmente fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais. Entretanto, elas demandam maiores recursos computacionais, em termos de eficiência computacional (tempo de processamento).

Quanto maior for a taxa de cruzamento, mais rápido serão introduzidas novas estruturas na população. Contudo, estruturas com boas aptidões poderão ser retiradas mais rapidamente, ou seja, a maior parte da população será substituída, mas com valores muito altos pode ocorrer perda de estruturas de alta aptidão. Todavia, um valor baixo para a taxa de cruzamento o algoritmo pode se tornar muito lento.

Como mencionado, a taxa de mutação previne a estagnação da busca, além de possibilitar que se chegue em qualquer ponto do espaço de busca. Contudo, se essa taxa

for muito alta a busca se torna essencialmente aleatória, já que o intervalo de geração controla a porcentagem da população que será substituída durante a próxima geração. Quanto maior seu valor, maior é a parte da população será substituída, podendo ocorrer perda de estruturas de alta aptidão. Assim, a taxa de mutação é definida em termos relativamente baixos.

Aplicações

As aplicações que mais se beneficiam dos AGs são os problemas de otimização, em que se busca, se não uma solução ótima, ao menos uma solução suficientemente adequada para o problema (GREFENSTETTE, 1986). Segundo Sandri & Correia (1999), os AGs também têm sido usados em muitas aplicações ligadas a controle, em particular, envolvendo o desenvolvimento de controladores nebulosos (KARR, 1991).

Um sistema com bom desempenho em um ambiente dinâmico, geralmente exige soluções adaptativas. Sistemas adaptativos tentam resolver problemas acumulando conhecimento sobre o problema e utilizando estas informações para gerar soluções aceitáveis. Estes problemas, tipicamente, se encontram nas áreas de configuração de sistemas complexos, alocação de tarefas, seleção de rotas, e outros problemas de otimização e aprendizado de máquina.

Também podem citar-se alguns exemplos de sistemas adaptativos: Controle de Sistemas Dinâmicos; Indução e Otimização de Bases de Regras; Encontrar Novas Topologias Conexionistas: Engenharia de Sistemas Neurais Artificiais; Modelagem de Estruturas Neurais Biológicas; Simulação de Modelos Biológicos: Comportamento; Evolução; Evolução Interativa de Imagens; Composição Musical.

1.5.Considerações Finais

O presente capítulo forneceu uma pequena descrição sobre a fundamentação teórica do presente trabalho. Espera-se que, com a leitura do mesmo, seja possível compreender sem maiores dificuldades as abordagens detalhadas nos capítulos posteriores.

2. TRABALHOS RELACIONADOS

O presente capítulo visa dissertar sobre trabalhos relacionados ao processo de negociação em leilões CDA. Para isso, foram revisados alguns trabalhos presentes na literatura. Inicialmente, descrevem-se alguns dos principais trabalhos que se relacionam aos mesmos problemas levantados durante o capítulo de introdução; posteriormente, detalham-se algumas abordagens relacionadas ao processo de negociação CDA, com uma seção específica para listar algumas abordagens de agentes TAC.

2.1.Visão Geral

Park et al. (2004) levantam uma série de questionamentos, visando o projeto e implementação de agentes para CDA. Eles descrevem uma estratégia denominada *P-Strategy*, que utiliza a técnica de cadeias de Markov para simular o comportamento de outros agentes e estimar o andamento do processo de negociação. Outras pesquisas como (DAS et al., 2001) relatam a grande importância que o *double auction* adquiriu nos últimos anos.

He et al. (2003) descrevem o uso de um controlador *Fuzzy*, com mecanismo de inferência Sugeno na decisão dos valores de novos lances. Para isso, foram utilizados como entrada do controlador a estimativa de um preço de equilíbrio, bem como os valores de cotação. A calibração de alguns parâmetros do controlador é feita dinamicamente, de acordo com a análise da estimativa de demanda e suprimento no mercado, mais detalhes de tal abordagem podem ser encontrados na seção 2.5.

Vytelingum (2006) fez um apanhado sobre os aspectos comportamentais e estruturais do CDA. Ele avaliou estratégias de CDA bem conhecidas como: *Kaplan*, *Zero-Intelligence* e *Gjerstad-Dickhaut* e propôs uma estratégia denominada *Adaptive-Aggressiveness Bidding Strategy* que também se utiliza de um controlador nebuloso no seu processo de inferência para o cálculo de novos lances, sendo uma extensão do controlador descrito em He et al. (2003).

A previsão do preço das ações, opções, preço de venda de energia elétrica, dentre outros bens negociados em mercados CDA podem ser modelados com a

utilização de técnicas estatísticas, de sistemas inteligentes ou híbridos, que envolvem as duas anteriores (MELO, 2008). Sistemas nebulosos (PETRIDIS; KEHAGIAS, 1997) e RNAs (SOARES, A. S., 2008; MENEZES JÚNIOR, J. M. P., 2006) são algumas das técnicas empregadas em sistemas inteligentes, ou híbridos, para apoiar o processo de previsão em mercados CDA.

Outras abordagens utilizam técnicas de otimização para a seleção de carteira de investimento, dentre elas, Ko & Li (2008) empregaram redes neurais; Gupta et al. (2008) empregaram lógica Fuzzy; e Lin & Liu (2008) empregaram um algoritmo genético para o mesmo problema.

Segundo Ribeiro et al. (2007), "pesquisadores utilizam-se de uma variedade de modelos de otimização multiperíodo, incluindo otimização estocástica, para desenvolver um modelo dinâmico de alocação ótima de ativos. Entretanto, a complexidade destes modelos e o grande esforço computacional que demandam, acabam por desestimular seu uso na prática dos profissionais da área financeira". Ele, ainda, descreve a escolha da carteira ótima de investimento segundo um modelo que representa um problema de otimização estocástica. E, para resolver o problema de seleção dos leilões foram utilizados técnicas de geração de árvores de cenários na modelagem de incertezas, transformando o problema em um problema determinístico.

2.2.Agentes TAC

A estimativa de preços de transação nos leilões do TAC foi bastante pesquisada. Várias estratégias foram aplicadas a tal problema, por exemplo: alguns agentes incluindo Alta, Aster, CUHK (STONE; GREENWALD, 2005) fizeram uso de uma estratégia baseada na média histórica de valores de um determinado leilão para estimar preços. Técnicas de aprendizagem de máquina, i.e. *Machine Learning*, também foram utilizadas, como ilustra o trabalho de Stone et al. (2002) através da implementação do agente Attac.Esse agente possuía um modelo de preços dinâmicos, construído a partir de conhecimento empírico para calcular os lances a serem executados. Outra técnica bastante empregada foi a utilização de Redes Neurais Artificiais (RNA), conforme ilustrado no trabalho de Putchala et al. (2002) com a implementação do agente Kavayah.

Nos trabalhos relacionados a agentes TAC, observa-se o uso de diversas técnicas para a seleção de leilões dependentes, que vão desde programação inteira à busca A* (OLIVEIRA; SI, 2006). Contudo, não se observa nenhuma abordagem utilizada voltada exclusivamente para a seleção de leilões do tipo CDA, haja vista, que as estratégias dos mesmos estavam intrinsecamente relacionadas à dependência entre os leilões.

Também vale a pena citar o trabalho de Sardinha et al (2005), que descreve o *LearnAgents*, um Sistema Multi-Agentes para atuar em todos os leilões do TAC; e, apesar de não estar voltado especificamente para os leilões CDA, a arquitetura *blackboard* implementada serviu como inspiração para a arquitetura do LaconiBot, uma vez que os problemas levantados por cada agente *LearnAgents* são semelhantes ao problemas enfrentados pelo LaconiBot.

2.3.Agente DummyAgent

Apesar do nome, o DummyAgent possui uma estratégia reativa simples e interessante. Logo no início do jogo, ele marca os leilões de passagens aéreas e entretenimento que deseja comprar, mas apenas nas datas pretendidas por seus clientes.

Os ingressos que ele possui, mas não foram marcados para compra, são vendidos por um valor inicial de $200; e têm seu valor reduzido de forma linear com o tempo até o mínimo de $120. Já para os ingressos que foram marcados para compra, ele faz lances iniciais por $50 e incrementa esse valor durante a partida, de maneira linear com o tempo até $60.

2.4.Agente SICS02

Assim como o DummyAgent, o SICS02 possui uma estratégia simples de negociação dos ingressos de entretenimento. Após o retorno das alocações de quais leilões participar, feita através de uma heurística de busca *branch-and-bound*, o agente envia lances de compra que iniciam com o valor de $50, e posteriormente são escolhidos novos valores aleatoriamente dado o intervalo ilustrado no gráfico da Figura 2.1.

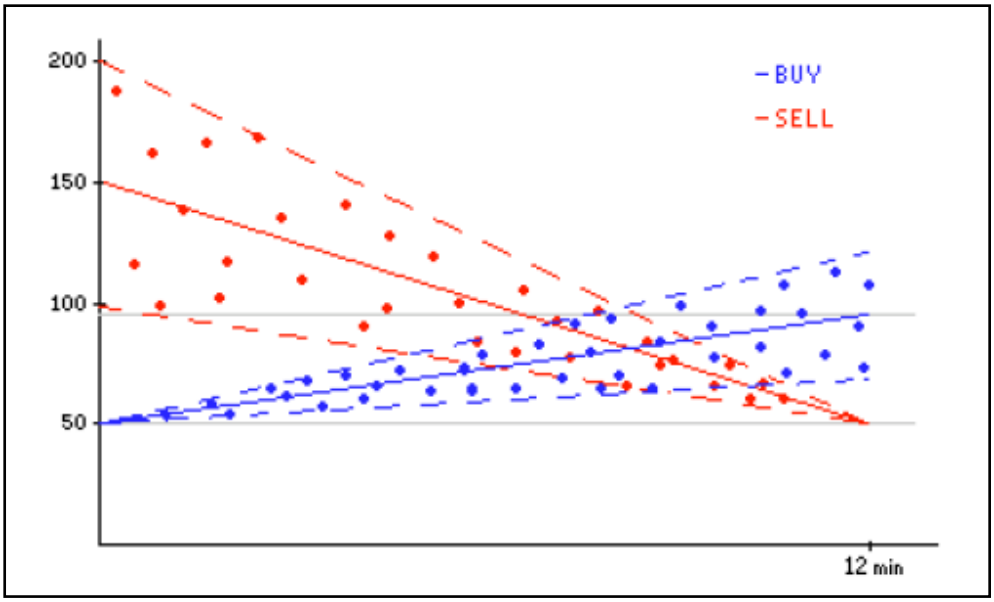

Figura 2.1. Gráficos dos intervalos de valores para compra e venda de ingressos para o agente SICS02.

O agente procede de forma similar para os ingressos que ele quer comprar, onde o valor inicial pode ser qualquer valor do intervalo entre $100 e $200, e para lances posteriores são escolhidos novos valores aleatoriamente dado o intervalo ilustrado no gráfico da figura 2.1.

2.5.Agente UTTA06

Analisando a evolução dos preços nos leilões CDA do TAC, como a ilustrado no gráfico da Figura 2.2, Tabarzad et. al (2006) observaram um comportamento estocástico de alta complexidade e de difícil modelagem. Eles também observaram que, na medida em que os leilões chegavam ao fim, a concretização de uma transação era menor e menos rentável.

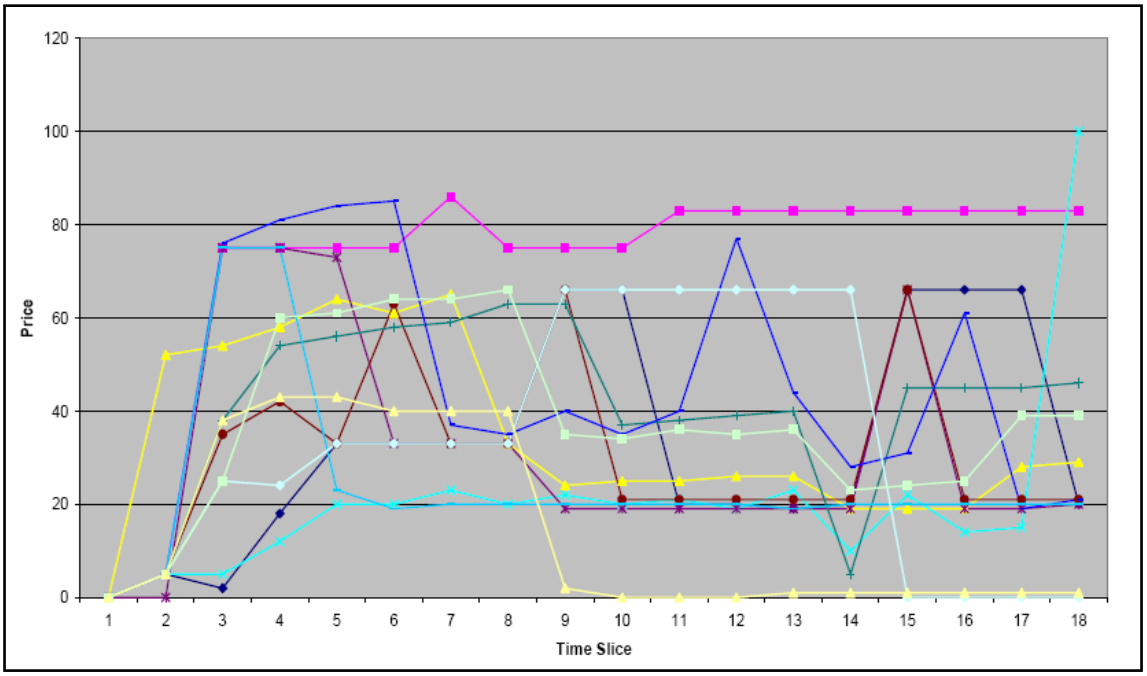

Figura 2.2. Exemplo da evolução da cotação de compra dos 12 leilões durante uma partida.

A partir dessa análise os autores chegaram a duas funções, uma para a estimativa do preço de venda (2.1), e outra (2.2) para a estimativa do preço de compra:

$$P_{t_{current}} = P_{min} + \frac{t_{close} - t_{current}}{t_{close} - t_{start}} \times P_{profit} \quad (2.1)$$

$$P_{t_{current}} = P_{max} - \frac{t_{close} - t_{current}}{t_{close} - t_{start}} \times P_{profit} \quad (2.2)$$

onde, P_{min} é o menor preço do bem; P_{max} é o valor máximo de preço aceitável; t_{close} é o tempo de fechamento do leilão; t_{start} é o tempo de inicio do leilão; $t_{current}$ é o tempo corrente; P_{profit} é o lucro máximo obtido pela venda. Logo, por exemplo, para um lance de compra no intevalo $[P_{profit} - \delta, \infty]$, com δ sendo um pequeno decréscimo a P_{profit} o bem será vendido.

2.6. Agente Mertacor

Kehagias et al. (2006) descreveram a estratégia *A Long-Term Profit Seeking Strategy* adotada pelo agente Mertacor; ele foi muito bem classificado no torneio TAC Classic de 2005; e, o foco da mesma foi direcionado ao mercado CDA do TAC. Ela se baseia na hipótese que vendedores em CDA procuram negociar seus recursos dentro de

uma margem de lucro M, aceitando diferentes preços para efetuar uma transação, ou seja, caso um preço R esteja longe da margem de lucro M de um vendedor, é enviado um novo lance de venda para atrair outros compradores.

Para o Mertacor o interessante não é lucrar em apenas uma transação, levando ao fechamento de poucas transações, e sim conseguir uma média positiva de lucro no conjunto de todas as transações que ele participar.

Utilizar apenas a cotação de um bem como critério para a compra ou a venda de um bem pode levar a um desempenho insatisfatório para o agente. Por exemplo, se um cliente está disposto a pagar $120 como bônus, então comprar o mesmo por $100 parece um excelente negócio, o lucro seria 120-100=20; contudo, para um vendedor que teria um custo de $40, a venda do mesmo por $100 é um negócio ainda melhor, com um lucro de 100-40=60; e como 60 >> 20 o lucro do vendedor seria muito maior que o lucro do comprador.

Para calcular a estimativa de lucro do Mertacor são levados em consideração alguns aspectos como uma avaliação V do ingresso tomando as preferências dos clientes e o potencial de lucro (*profit*), baseado nas cotações de compra e venda e no estado do agente como comprador ou vendedor. O cálculo da estimativa do lucro pode ser dado por *profit* = V - 2Pb, no caso de um agente de venda; *profit* = Pa – 2V no caso de um agente de compra, onde Pb é o preço de cotação de compra e Pa é o preço de cotação de venda. Além disso, a estratégia de venda do Mertacor é implementada segundo o procedimento MertacorSellStrategy ilustrado no algoritmo a seguir.

```
PROCEDURE MertacorSellStrategy
 1:  FOR each entertainment ticket in possession
 2:      Assign a pre-specified value to target;
 3:      mean ← getMeanValue();
 4:      M ← A*target + B*mean;
 5:      IF M ≤ (1/2)*target OR M ≥ (3/2)*target THEN
 6:        M ← relocateM();
 7:      END IF
 8:      V ← calcVal();
 9:      IF V < Vo THEN V ← Vo; END IF
10       profit ← Pa – 2*V;
11       Mt ← w(t)*M;
12       Ra ← M;
13       Rb ← 3*M/2;
14       Rc ← rand(M/2, M);
15       IF profit ≥ Mt-Ra THEN sellTicket();
16       ELSE IF profit ≥ Mt-Rb THEN ask (Mt-Ra+2*V);
17       ELSE ask (Mt - Rc +2*V);
18       END IF
19  END FOR
```

A variável *mean* guarda o valor da média dos lucros das últimas transações realizadas. A variável *target* é um valor no intervalo [5,10]; já A e B são os pesos que foram ajustados para o TAC, com os valores A=0.7 e B=0.3, ou seja, o *target* influencia em 70% do peso. Na linha 6, relocateM() garante o valor M no intervalo [min {target, mean}, max {target, mean}]. A variável V recebe a avaliação do ingresso de acordo com as preferências dos clientes passados como parâmetro para um modelo de Programação Linear. A variável V_o garante que esse valor não será inferior a $40. A variável Mt limita o intevalo do lucro estimado considerando a função dependente do tempo ilustrada na Figura 2.3. As variáveis Ra, Rb e Rc remetem ao interesse de compradores em relação a lucro estimado em M.

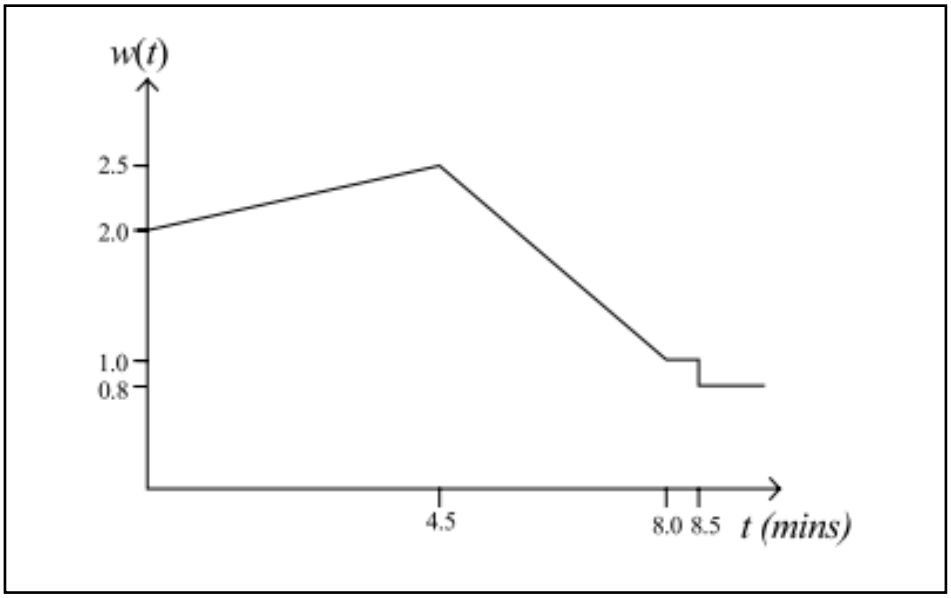

Figura 2.3. Função w(t) em relação ao tempo em minutos.

2.7.Agente DealerBot

Oliveira (2008) implementou o agente DealerBot, utilizando uma adaptação da abordagem descrita por He et al. (2003) para o processo de negociação CDA do TAC, com o objetivo de participar de múltiplos leilões CDA. Ele ainda realizou um estudo comparativo entre os mecanismos de inferência Sugeno e Mandami, utilizados no controlador nebuloso do agente. Entretanto, não foi utilizado nenhum mecanismo de seleção para a escolha de qual leilão CDA participar, o que fazia com que o agente participasse de todos os 12 leilões, inclusive, os que ele não tinha interesse.

He et al. (2003) fizeram uso de um controlador *Fuzzy* para automatizar o processo de negociação em um ambiente de negociação CDA. Os autores levantaram um conjunto de regras nebulosas e utilizaram o mecanismo de inferência Sugeno para elaborar uma estratégia para a tomada de decisão durante a negociação em CDA, denominada *FL-Strategy*.

A estratégia baseia-se na utilização da lógica *Fuzzy* (ZADEH, 1965) para análise e criação dos lances negociados em CDA. Basicamente, a *FL-Strategy* pode ser vista como um controlador *Fuzzy*, que recebe como entrada os preços das cotações de compra e venda do bem negociado, além de um preço de referência, detalhado a seguir. E como

saída, o controlador decide a viabilidade em efetuar ou ajustar um lance, de compra ou venda, baseado no seu conjunto de regras nebulosas, e no seu mecanismo de inferência.

O preço de referência P_r é o valor da mediana no conjunto ordenado $H = \{p_{(1)}, p_{(2)}, \dots, p_{(n)}\}$, onde n é a quantidade de transações salvas como histórico e $p_{(i)}$ é o preço da i-ésima transação em um CDA passado. A relação entre P_r, a_0 (cotação de venda) e b_0 (cotação de compra) remete a três situações: (i) $P_r \leq b_0 < a_0$; (ii) $b_0 < a_0 \leq P_r$; e (iii) $b_0 \leq P_r \leq a_0$.

Nos casos (i) e (ii), não é necessário executar muitos cálculos. Por exemplo, se o agente está atuando como um comprador, o bem analisado é comprado pelo valor a_0 quando a cotação recai sobre o caso (ii), e a cotação satisfaz a regra *Muito Menor* (Figura 2.4); caso contrário, um novo lance é enviado com o valor $(b_0 + \beta_{b,1}, \theta, \chi)$. No caso (i), ignora-se o lance se a cotação satisfaz a regra *Muito Maior* (Figura 2.4); e, caso contrário, o lance de compra é atualizado com um novo valor, incrementado de é $(b_0 + \beta_{b,2}, \theta, \chi)$.

$$\beta_{b,1}\beta_{b,2}A_1A_2P_1P_2\gamma_{b,1}\gamma_{b,2}A_1(b_0) \geq \gamma_{b,1}b_0P_rA_2(b_0) \geq \gamma_{b,2}b_0$$

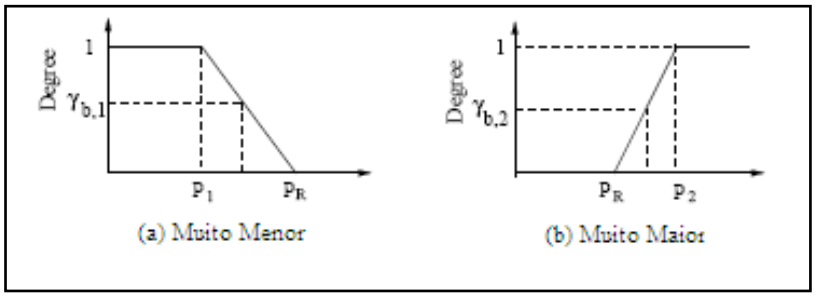

Figura 2.4. Representação dos termos lingüísticos Muito Menor e Muito Maior. Adaptado de: He et al (2003).

Os valores $\beta_{b,1}$ e $\beta_{b,2}$ definem quanto deverá ser o acréscimo em cada caso (em geral, maior no primeiro e menor no segundo). Os termos lingüísticos *Muito Menor* e *Muito Maior* são representados por conjuntos *Fuzzy* A_1 e A_2, cujas funções de pertinência são ilustradas na Figura 2.4. Os parâmetros P_1 e P_2 são decididos pela experiência e intuição humana de acordo com o intervalo de valor e avaliação do item. Por fim, os valores $\gamma_{b,1}$ e $\gamma_{b,2}$ indicam que quando $A_1(b_0) \geq \gamma_{b,1}$, b_0 será considerado *Muito Menor* que P_r; e que quando $A_2(b_0) \geq \gamma_{b,2}$, b_0 será considerado *Muito Maior*.

Já o caso (iii) recai em um conjunto de regras que faz uso do mecanismo de inferência Sugeno ou Mandami que irá estimar um valor para um valor para z com o novo lance:

SE (a_0 está distante OU médio a P_r) E (b_0 está distante de P_r)
ENTÃO novo lance é $(b_0 + \lambda_{b,1}, \theta, \chi)$

SE (a_0 está distante OU médio a P_r) E (b_0 está médio a P_r)
ENTÃO novo lance é $(b_0 + \lambda_{b,2}, \theta, \chi)$

SE (a_0 está distante OU médio a P_r) E (b_0 está próximo de P_r)
ENTÃO novo lance é $(b_0 + \lambda_{b,3}, \theta, \chi)$

SE (a_0 está próximo de P_r)
ENTÃO novo lance é $(P_r - \lambda_{b,4}, \theta, \chi)$

Por exemplo, estando a_0 e b_0 distantes de P_r, é possível elevar o preço de compra, contudo se a_0 estiver distante de P_r, e b_0 estiver próximo, deve-se decrementar o preço de compra, evitando que o mesmo ultrapasse o preço de referência. A proximidade entre P_r, a_0 e b_0 é medida com o uso das variáveis lingüísticas *Fuzzy*: *Próximo*, *Médio* e *Distante* (Figura 2.5).

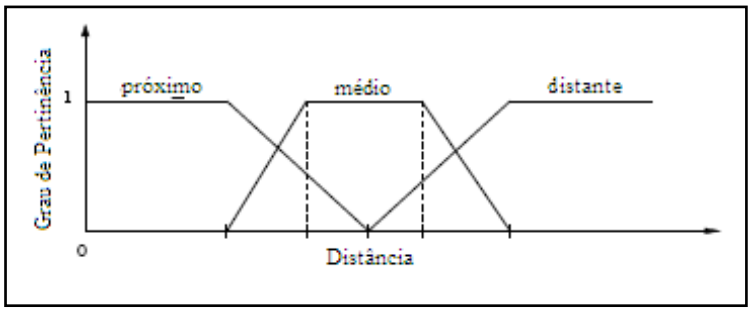

Figura 2.5. Conjuntos nebulosos Próximo, Médio e Distante. Adaptado de: He et al. (2003).

Antes de retornar o valor calculado pela estratégia, o novo lance passa por mais um crivo, ou seja, busca-se o melhor valor no conjunto decisão DS_b, de acordo com:

$$Lance = \begin{cases} a_0 & se\ a_0 \in DS_b, \\ \arg \max\limits_{b \in DS_b} \{z_b(b)\} & caso\ contrário. \end{cases}$$

onde $DS_b = \{b | b_0 < b \leq \min(a_0, v_{item}) \ e \ z_b(b) \geq \pi_b\}$; v_{item} é o valor que o item avaliado pelo agente; e π_b é o grau mínimo com que b participa de z_b.

A estratégia do agente sob o ponto de vista de venda é análoga àquela adotada na compra. As regras para a seleção do novo lance podem ser descritas como:

Quando $P_r \leq b_0 < a_0$:
SE b_0 é Muito_Maior que P_r
ENTÃO aceita b_0
SENÃO novo lance é $(a_0 - \beta_{s,1}, \theta, \chi)$

Quando $b_0 < a_0 \leq P_r$:

SE a_0 é Muito_Menor que P_r
ENTÃO sem novo lance
SENÃO novo lance é $(a_0 + \beta_{s,2}, \theta, \chi)$

Quando $b_0 \leq P_r \leq a_0$:

SE (b_0 está Distante OU Médio a P_r) E (a_0 está Distante de P_r)
ENTÃO novo lance é $(a_0 - \lambda_{s,1}, \theta, \chi)$

SE (b_0 está Distante OU Médio a P_r) E (a_0 está Médio a P_r)
ENTÃO novo lance é $(a_0 - \lambda_{s,2}, \theta, \chi)$

SE (b_0 está Distante OU Médio a P_r) E (a_0 está Próximo de P_r)
ENTÃO novo lance é $(a_0 - \lambda_{s,3}, \theta, \chi)$

SE (b_0 está Próximo de P_r)
ENTÃO novo lance é $(P_r + \lambda_{s,4}, \theta, \chi)$

A busca pelo valor real do lance é expressa por:

$$\text{lance} = \begin{cases} b_0 & \text{se } b_0 \in DS_s, \\ \arg\max_{a \in DS_s} \{z_s(b)\} & \text{caso contrário.} \end{cases}$$

com DS_s definido como

$$DS_s = \{a \mid \max(b_0, c_{item}) \leq a < a_0 \text{ e } z_s(a) \geq \pi_s\}$$

onde c_{item} designa o custo do item para o agente e π_b determina o grau mínimo com que b deve participar de z_b.

2.8.Considerações Finais

Este capítulo apresentou uma revisão bibliográfica de algumas das principais contribuições encontradas na literatura sobre a temática abordada neste trabalho. Foram descritos trabalhos que de alguma forma se relacionam com o presente trabalho, quer seja na resolução de problemas semelhantes, quer seja na resolução dos mesmos problemas aqui atacados. Também é possível notar a variedade de estratégias adotadas ao processo de negociação CDA, com implementações simples como as dos agentes DummyAgent e SICS, até abordagens mais complexas como a FL-Strategy descrita por He et al. (2003) que utilizaram de um controlador nebuloso no seu processo de negociação.

3. O AGENTE LACONIBOT

O presente capítulo disserta sobre o agente LaconiBot. Primeiramente descreve-se a arquitetura do agente LaconiBot; posteriormente, detalham-se as abordagens utilizadas em seus componentes para a resolução de cada um dos principais problemas observados no processo de negociação CDA do TAC.

3.1.Arquitetura do LaconiBot

Uma interessante arquitetura de agente foi desenvolvida por Oliveira (2008). Ela é uma extensão da arquitetura de agentes baseada em estados (Figura 1.3), pois se observa a escolha de uma ação referente a um conjunto de regras pré-esbelecidas; além de um registro do estado do ambiente nas bases de conhecimentos associadas a cada uma das possíveis ações. A Figura 3.1 apresenta o modelo arquitetural empregado na implementação do agente DealerBot para participar de leilões CDA do TAC.

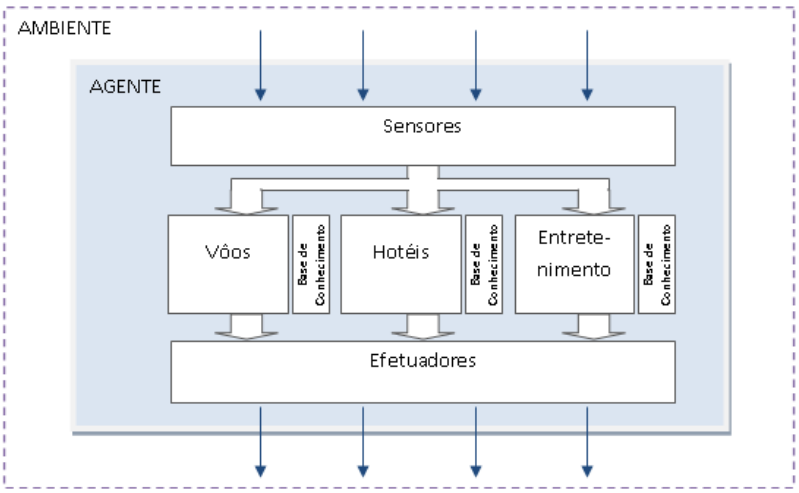

Figura 3.1. Modelo arquitetural do agente DealerBot.

Motivado pelo desempenho obtido por uma arquitetura desenvolvida no Laboratório de Computação Natural e Inteligente (LACONI), no Curso de Computação da Universidade Estadual (UECE), este trabalho aprofundou os estudos sobre agentes adequados para múltiplos CDA, gerando uma nova arquitetura, tanto no que diz respeito à presença de novos componentes, quanto à utilização de outras heurísticas e técnicas de inteligência artificial como, por exemplo, Redes Neurais e Algoritmos Genéticos Multi-

Objetivos, que foram empregadas nas concretizações dos novos componentes, propostos para resolver a problemática descrição na Introdução.

Esta nova arquitetura deu origem ao agente LaconiBot. A Figura 3.2 apresenta seus principais subsistemas e componentes (módulos processadores de informação), bem como as informações que são processadas por cada subsistema. Este trabalho é uma contribuição ao desempenho do subsistema de leilões voltados para a compra e venda de ingressos para entretenimento, conforme protocolo definido no CDA da competição TAC.

Uma das principais diferenças está na utilização de uma Base de Conhecimento Compartilhada, ou BCC, que funciona como uma memória *blackboard*, onde todos os módulos podem acessá-la. Assim, é possível compartilhar em uma mesma estrutura de dados, tanto o estado externo do ambiente, quanto o estado interno do agente. Como, por exemplo: os valores das cotações atualizados e outros parâmetros utilizados nas estratégias de negociação, o retorno da solução apresentada pelo componente Alocador e a estimativa de preço calculada pelo Previsor conforme descrito a seguir.

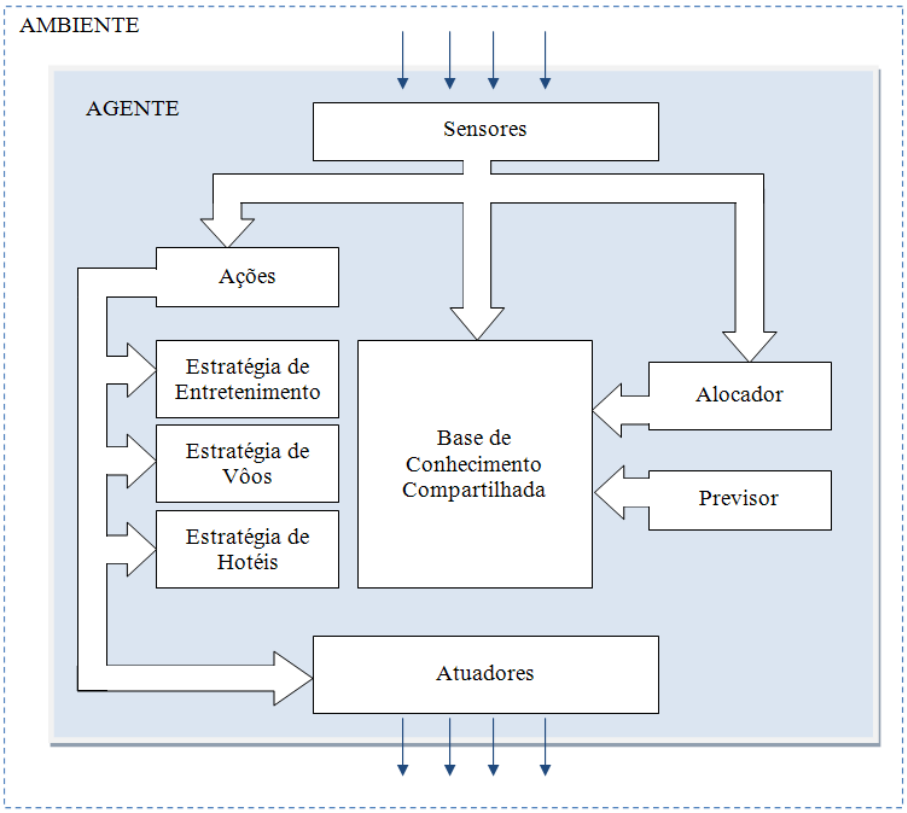

Figura 3.2. Modelo arquitetural do agente LaconiBot.

Mais especificamente, o subsistema voltado para entretenimento supõe que:

(1) por meio dos Sensores o agente recebe informações do Ambiente que são os valores de cotação para compra, e venda; além do preço e quantidade de itens as quais ele efetuou transações. As atualizações ocorrem de 30 em 30 segundos para todos os 12 leilões.
(2) logo depois o estado interno do agente é atualizado, onde os valores percebidos pelos Sensores são armazenados na BCC.
(3) são executadas as ações do Previsor e Alocador. O Previsor verifica os dados de cotação e o tempo decorrido de jogo da BCC, em seguida estima os valores de transação sobre a informação passada como parâmetro por meio da sua RNA. Já o Alocador, verifica os dados da BCC e atualiza as alocações dos itens a serem negociados pela Estratégia de Entretenimento. O resultado das alocações fica armazenado na BCC.
(4) de acordo com o estado atualizado das alocações na BCC, são executadas as ações da Estratégia de Entretenimento.
(5) por meio de Atuadores o agente envia as ações selecionadas pela Estratégia de Entretenimento para o ambiente.

3.2.Lógica do Componente Negociador

Um dos grandes problemas encontrados na automatização das tarefas de negociação em CDA diz respeito a quando enviar um lance e por qual valor. Tal problema influencia bastante no desempenho da negociação. Por exemplo, anunciar um lance de venda para um bem, no momento em que seu preço de cotação está desvalorizado pode remeter a uma venda com pouca rentabilidade. Outra situação remete ao problema de incremento ou decremento do lance, como: dado um lance de compra que não gerou uma transação, qual deve ser o valor de incremento ou decremento do lance para que seja concretizada a transação com a maior rentabilidade possível? E como saber se o bem está ou não desvalorizado?

A abordagem proposta por He et al. (2003) foi apresentada com detalhes na mesma seção, uma vez que serviu como base tanto para o processo de negociação do agente DealerBot quanto para o agente LaconiBot. Para ilustrar a viabilidade de tal abordagem, no capitulo 4 de Avaliação, mostra-se o desempenho da mesma através da

execução do agente DealerBot. Além disso, tal agente serviu como controle para análise do desempenho das outras estratégias do LaconiBot, pois é através da comparação do DealerBot com o LaconiBot que se pode verificar o real desempenho após a incorporação dos outros componentes, conforme descrito nas seções seguintes.

3.3.Lógica do Componente Previsor

Conforme descrito na seção 2.5, a estratégia de entretenimento necessita de parâmetros para determinar um novo lance, são eles: a cotação do preço de compra de um item em leilão, a cotação de venda para o mesmo item e o preço de referência. Como o nome sugere, o preço de referência serve para apoiar a tomada de decisão do negociador, fornecendo um valor referencial, para calcular o desvio do preço da cotação em relação ao valor de transação do item. Dessa forma, quanto mais apurada for a estimativa do valor de transação, mais apurada será a tomada de decisão do controlador, remetendo a uma melhor performance do agente.

Dentro do contexto das técnicas de Inteligência Computacional, as Redes Neurais Artificiais, ou RNA, são bastante empregadas na concepção de modelos de previsão baseada em séries temporais. Estes resultados influenciaram a concepção do componente Previsor do agente LaconiBot, ou seja, para prever o preço de transação em leilões CDA, considerando dados históricos dos leilões em que o agente esteja participando ou de leilões que ocorreram em competições passadas. Assim, nesta primeira etapa do desenvolvimento do Previsor, fez-se opção pela Rede Neural Perceptron Multi-Camadas (MLP) com algoritmo de aprendizagem Back-propagation.

Para implementação da RNA proposta, foi utilizado o *framework* Joone (*Java Object Oriented Neural Engine*). Ele foi escolhido por uma série de razões: ele foi escrito na linguagem de programação Java, é distribuído sob a licença LGPL, possui boa documentação, possui interface bem definida para o desenvolvimento da RNA, fornece diversas classes implementadas que facilitam sobremaneira o treinamento e a execução da rede neural, e, além disso, tem a aprovação da comunidade acadêmica, sendo utilizado em diversos trabalhos científicos.

3.3.1. A Rede Neural Artificial

A implementação de uma RNA envolve vários passos, que vão desde a escolha da topologia da rede até a obtenção do conjunto de variáveis significativas para a resolução do problema, ou seja, o pré-processamento dos dados bem como o processo de treinamento e validação. Não existe um método considerado padrão para a implementação de uma RNA, portanto, todos os passos utilizados para a implementação da mesma são descritos a seguir.

Existem várias topologias de RNA, no entanto, por questão de simplificação optou-se pela utilização de uma arquitetura multicamadas. Geralmente, o número de neurônios da camada de entrada é igual ao numero de atributos que se deseja avaliar e a quantidade de neurônios da camada de saída é o número de respostas que se tem para o padrão a ser aprendido (TAFNER et al.,1996). Para o atual trabalho, foi escolhida uma MLP formada por três camadas. A primeira camada foi composta por três neurônios, a camada interna com quatro neurônios para que apresentasse um bom nível de generalização, e, por fim, a camada de saída com apenas um neurônio para representar o valor de saída da rede.

Alguns trabalhos como o de Melo (2008) relatam o ganho no desempenho de uma RNA, a partir de uma escolha mais refinada de sua topologia. Para a escolha da topologia da RNA utilizada neste trabalho, foi realizada uma busca por tentativa e erro, até se chegar ao modelo ilustrado na Figura 3.3.

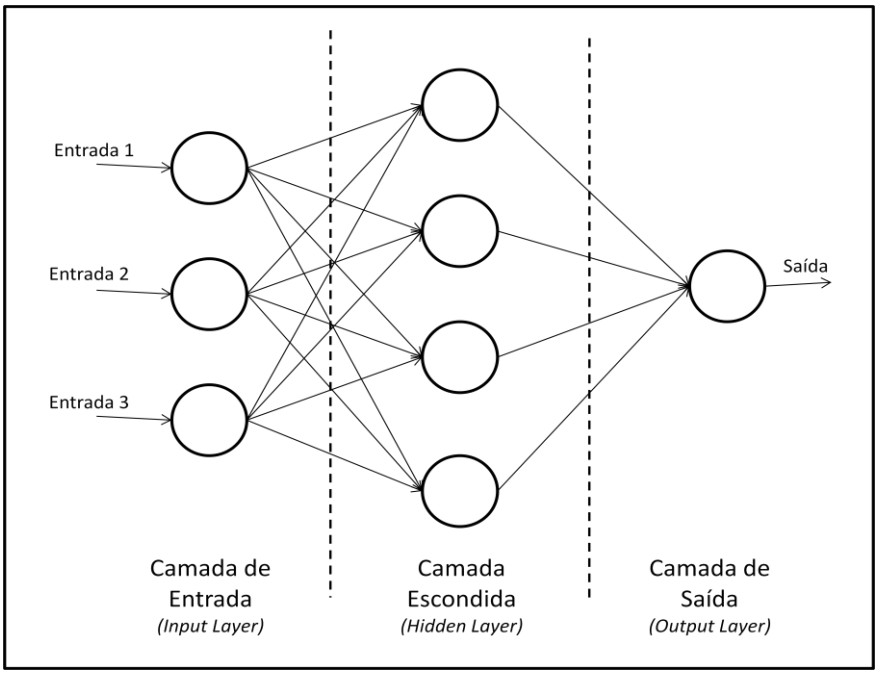

Figura 3.3. Topologia da RNA utilizada no componente Previsor.

A seguir, as próximas subseções descrevem mais especificamente as outras etapas necessárias à concretização do componente Previsor.

Processo de aprendizado da RNA

Optou-se pelo aprendizado supervisionado da RNA, o que necessitou um conjunto de treinamento capaz de representar tanto a informação passada como entrada para a rede neural, quanto a informação esperada na saída. Outro aspecto necessário foi um conjunto de dados para testar e validar a RNA treinada. Dessa forma, o fluxograma da Figura 3.4 ilustra todo o processo aprendizado da rede neural aqui detalhada.

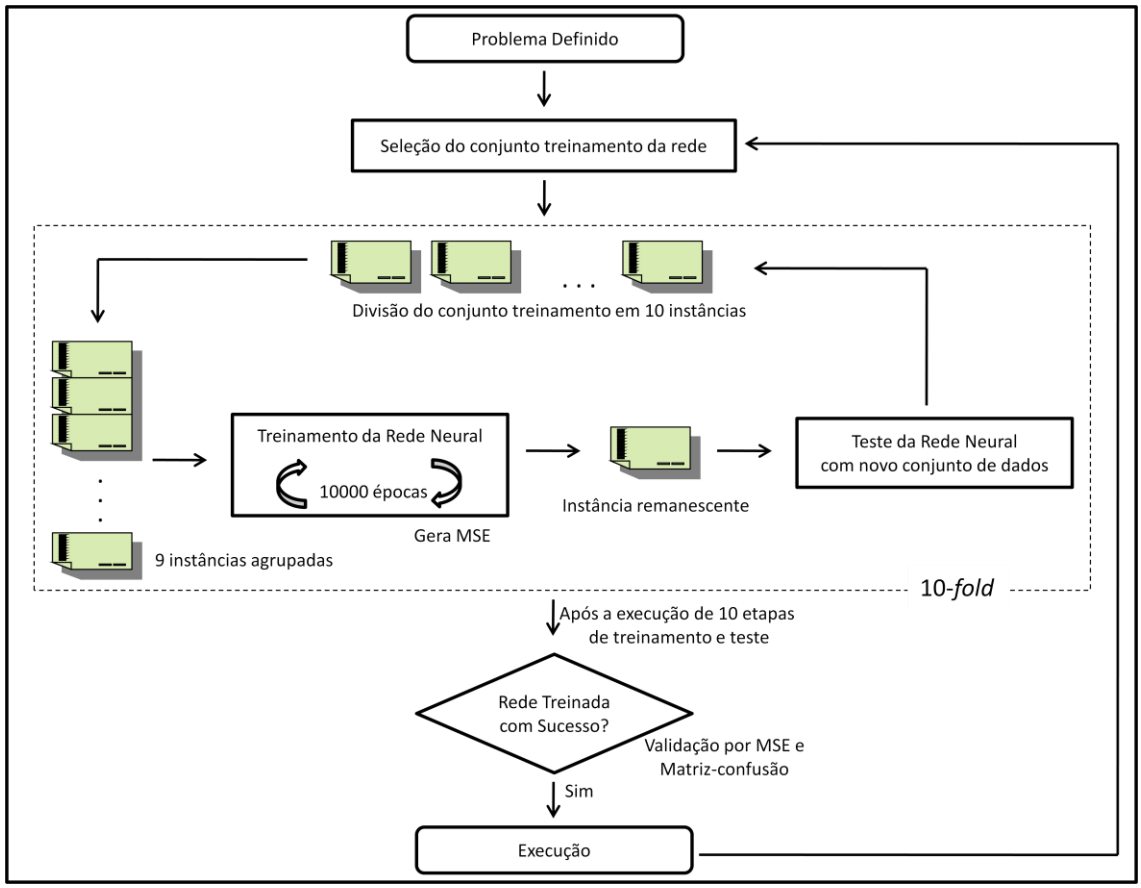

Figura 3.4. Fluxograma ilustrando o processo de aprendizado da RNA.

As próximas subseções descrevem os principais detalhes do aprendizado da RNA no Previsor que foram especificados na Figura 3.4.

Conjunto de Treinamento

Conforme descrito no capítulo 4 (Avaliação), o TAC fornece uma infra-estrutura de software completa para o desenvolvimento, execução e análise do desempenho de agentes. Dessa forma, utilizou-se o servidor TAC e alguns agentes disponibilizados no site do TAC para a geração dos dados que iriam compor o conjunto de dados de treinamento.

Outra preocupação na geração dos dados de treinamento foi a montagem de um cenário que refletisse o real ambiente onde a RNA seria utilizada. Logo, utilizou-se o mesmo cenário descrito nas seções 4.3 e 4.4.1 para o agente DealerBot. Uma vez que utilizando um cenário onde só existissem *DummyAgent*, a RNA poderia não retornar bons resultados quando inserida em um cenário real.

Após uma análise detalhada do problema, foi possível abstrair as informações mais relevantes para compor os atributos da base de dados de treinamento da RNA. Dessa forma, selecionou-se o **preço de cotação de compra do bem**; o **preço de cotação de venda do bem**; o **tempo decorrido de jogo**; e o **valor de transação**. Conforme descrito de maneira mais formal através da tupla *T* a seguir.

$$T = < C_v, C_c, t, P_t >$$

onde:

C_v = preço de cotação de venda de um item em leilão no intervalo de tempo Δt.

C_c = cotação do preço de compra de um item em leilão no intervalo de tempo Δt.

t = tempo decorrido de jogo.

P_t = preço de transação em um dado instante de tempo t.

Após a execução de 70 jogos e o pré-processamento dos dados, descrito a seguir, foi gerado um conjunto com o total de 1450 exemplos no formato de 4-úplas. Não foi realizada nenhuma separação quanto aos tipos de ingresso, ou seja, em cada jogo, à medida que os logs eram analisados, eram acrescentadas novas linhas em um arquivo de texto para se formar o conjunto de treinamento.

Pré-Processamento dos Dados de Entrada

Após a seleção dos atributos e a geração dos dados para a formação do conjunto de treinamento, outro passo importante na mineração desses dados é o pré-processamento. Se forem apresentadas quantidades significativas de dados com ruído, faltosos, inconsistentes ou não normalizados, a RNA poderá remeter a um desempenho insatisfatório. Portanto, o pré-processamento dos dados ocorre com a preparação dos valores de exemplo que serão apresentados à rede de modo que ela possa generalizar o conjunto de treinamento. Foram realizados os seguintes procedimentos:

- Como a quantidade de lances que remetem a uma transação é bem menor que a quantidade de lances que não remetem a transações, foram coletados apenas exemplos onde $C_v > 0$, $C_c > 0$ e $P_t > 0$.

- Para os valores referentes à cotação de compra e venda de algum bem, assim como o preço de transação, se observou que: os preços não podem ser menores que zero, pois não existem preços negativos; e o preço máximo oferecido para os eventos de entretenimento não ultrapassam 200 unidades monetárias. Dessa forma todos os valores de cotação foram divididos por 200.

- O cálculo da unidade de tempo foi feito baseado na informação do tempo total de partida. Como cada partida dura 9 minutos ou 540 segundos, os valores de tempo foram divididos por 540 para a obtenção do dado normalizado.

Por exemplo, analisando o leilão de ingressos do tipo parque de diversões para o terceiro dia. Se no tempo 308 segundos de andamento do leilão a cotação para compra for $66 e a cotação para venda fosse $98 e foi realizada uma transação por $79, o conjunto treinamento geraria a seguinte tupla T = <98.0, 66.0, 307, 79.0>. Após o pré-processamento os valores apresentados ficam iguais a T = <0.49, 0.33, 0.57, 0.39>.

Treinamento, teste e validação da RNA

Segundo Witten et al. (2005), o procedimento de validação cruzada, ou *cross-validation,* considera a troca de papéis entre os conjuntos de exemplos de treinamento e de testes. Dessa forma, a rede deve ser treinada com exemplos de teste e deve ser testada com exemplos de treinamento reduzindo assim o efeito de representações desiguais nos treinamentos e nos testes. Essa metodologia emprega que o mesmo conjunto de amostras deve ser utilizado para as duas fases mencionadas. Witten ainda afirma que é melhor utilizar mais da metade dos dados para o treinamento em detrimento à parte de testes. Na técnica de *cross-validation* deve ser decidido um número fixo de partes (*folds*) de dados.

Durante a separação das partes de treinamento e testes deve ser assegurado que cada parte contenha uma correta proporção de dados de mesma classe do conjunto total. Denominam-se de dados de mesma classe os dados que representam um padrão igual dentro do conjunto. Esse procedimento é chamado de estratificação. Em resumo, a estratificação busca a garantia contra representações desiguais dos conjuntos de treinamento e de testes (WITTEN et al., 2005).

No presente trabalho, a separação do conjunto de exemplos foi feita em 10 partes ou instâncias. Após a separação 90% dos dados, ou 9/10 das partes foram utilizadas para treinamento e o restante, 10%, para testes. Esse conjunto de técnicas utilizadas para o treinamento e teste da rede neural é similar à técnica de validação cruzada estratificada n-*fold*, nesse caso 10-*fold*.

O treinamento da rede, com a utilização da estratégia descrita, foi realizado de acordo com as seguintes etapas:

1. Com o conjunto treinamento (entrada e resposta desejada) já construído, fez-se uma separação do mesmo em 10 partes preocupando-se com a igual representação das classes de dados em cada uma das partes.
2. As partes foram reagrupadas em 10 subconjuntos diferentes, cada um com 90% do total de exemplos. Note que para cada subconjunto criado, uma parte fica sem participação no conjunto treinamento, garantindo que todas as instâncias permaneçam separadas pelo menos uma vez. A parte restante correspondente aos 10% não utilizados foi separados para testes.
3. Após a separação dos conjuntos foram realizadas 10 baterias de treinamentos. Ao fim de cada treino o conjunto remanescente foi testado.
4. O treinamento da rede ocorre através da apresentação repetitiva de um conjunto exemplo denominada época[5]. O número de épocas deve ser escolhido com o objetivo de que o resultado da rede possa convergir a um erro mínimo desejado. No treinamento da RNA, a cada 1000 épocas a ordem dos valores contidos no conjunto treinamento é alterada de maneira aleatória para minimizar alguma eventual tendência no processo de aprendizagem.
5. Por fim, avaliou-se a eficiência das previsões obtidas na rede utilizando as métricas de *Root Mean Squared Error* (RMSE), Matriz-confusão e erro médio, conforme descrito no capítulo 4 de Avaliação.

Metodologia de Aprendizagem e Testes

Com a utilização da estratégia de validação cruzada 10-*fold*, os 1450 exemplos do conjunto de exemplos foram subdivididos em 10 instâncias de 145 exemplos. De modo a garantir a presença de todos os momentos da negociação em cada uma das

[5] Época (*epoch*) é o nome dado a uma apresentação completa do conjunto treinamento à rede neural.

instâncias, o conjunto de exemplos é ordenado e depois é feita a divisão. A divisão é feita de modo a distribuir as linhas seqüencialmente para cada uma das instâncias, logo, o conjunto ordenado é percorrido de cima a baixo e cada exemplo lido é extraído para uma parte. Por exemplo, se os valores contidos na linha quatro foram para a parte 4, o próximo exemplo irá para a parte 5, e assim sucessivamente. Este processo resulta em instâncias ordenadas e divididas de forma igualitária.

Seguindo a metodologia, as 10 partes de exemplos são agrupadas em 10 novos conjuntos de 1305 padrões (90%) para o treinamento, e um padrão restante (10%) para testes. São executadas 10 baterias de treinamentos e ao fim de cada uma é executado um teste. Em cada etapa o treinamento foi executado com um total de 10000 épocas. A cada 1000 épocas os valores dentro de cada conjunto são recolocados de forma aleatória visando uma maior generalização no treinamento da rede.

3.4.Lógica do Componente Alocador

O problema de seleção dos leilões de entretenimento que um agente TAC deve participar consiste na escolha dos ingressos de entretenimento que devem ser negociados (comprados ou vendidos) com o objetivo de maximizar o lucro do agente. Para isso, os agentes TAC precisam analisar sua carteira de clientes e os preços de cotação para compra e venda.

Uma estratégia de seleção de leilões é necessária, pois sem tal abordagem a estratégia de negociação do agente LaconiBot teria que analisar a evolução da cotação para todos os leilões, inclusive para aqueles que não remeteria a um aumento no lucro do agente. Haja vista que, no caso do DealerBot bastava a cotação de um bem satisfizer as condições do conjunto de regras nebulosas do agente (vide seção 2.2.5), para o bem ser imediatamente comprado ou vendido, mesmo que não fosse vantajoso comprar ou vender o bem

Com a utilização da estratégia de seleção, o agente irá efetuar lances apenas nos leilões que ele espera que incrementarão seu lucro. Além disso, em um cenário real, como em mercados onde existem muitos bens, não é viável para um participante de CDA efetuar lances em todos os bens disponíveis no mercado, por exemplo, no mercado da bolsa de valores de Nova Iorque (NYSE) existem 29 papéis de ações apenas no setor de telefonia móvel NYSE (2009).

A seleção dos melhores leilões para participar vai depender muito de uma função que associe um valor de utilidade a cada leilão no conjunto de leilões factíveis, definido pelas restrições que têm origem nos preços de cotação atuais nos leilões, nos requisitos dos clientes e nos recursos que estão disponíveis para o agente. Mais especificamente, a seleção dos leilões que o agente espera que sejam rentáveis, vai depender da formulação do problema de seleção como um problema de otimização e da estratégia de busca empregada para resolvê-lo. As próximas seções focam a especificação destes dois aspectos implícitos no componente Alocador do agente LaconiBot.

3.4.1. Formalização do Problema de Seleção de Leilões CDA

Um problema de alocação pode ser definido formalmente como um problema de otimização onde se deve maximizar ou minimizar um, ou mais objetivos, sujeito a uma série de restrições. O problema de seleção dos leilões de entretenimento que um agente TAC deve participar consiste na escolha dos ingressos de entretenimento que devem ser comprados ou vendidos, com o objetivo de maximizar o lucro do agente. Para isso, os agentes TAC precisam analisar sua carteira de clientes, e os preços de cotação para compra e venda.

No que diz respeito à formulação da função utilidade, o modelo considera três objetivos, que podem conflitar entre si: **$\mathbf{Ganho_k}$, Inventário** e **$\mathbf{Despesa\ Operacional_k}$**. Os termos empregados na descrição dos objetivos são empregados com mais freqüência no setor industrial. Neste caso, tiveram inspiração na Teoria das Restrições, defendida por Eliyahu M. Goldratt e apresentada na forma de um romance (GOLDRATT; COX, 1995), onde os acontecimentos se desenrolam, em maior parte, dentro de uma fábrica.

O primeiro objetivo, **$\mathbf{Ganho_k}$**, representa o dinheiro ganho pelo agente devido à alocação do ingresso $\mathbf{l_{ij}^k}$ para um cliente $\mathbf{C_k}$ da sua carteira, ou seja:

$$Ganho_{\mathrm{k}} = 1000 + \sum_{\mathrm{i} \in Nd^{k}} \sum_{\mathrm{j} \in \mathrm{J^k}} \mathrm{l_{ij}^k} * \mathrm{Prêmio_{ij}^k} \quad (3.1)$$

onde, dado um conjunto de ingressos do agente $\mathbf{T_{ij}}$, $\mathbf{l_{ij}^k}$ guarda quais ingressos devem ser alocados ao cliente $\mathbf{C_k}$, assumindo os valores **0** ou **1** que indicam, respectivamente, se o

ingresso foi ou não alocado ao cliente $\mathbf{C_k}$; e, Prêmio_{ij}^{k} é o valor pago pelo cliente $\mathbf{C_k}$, dada a alocação **ij**. O conjunto $\mathbf{Nd^k}$ guarda o número de dias disponíveis para montagem de pacotes de entretenimento para um cliente $\mathbf{C_k}$; e $\mathbf{J^k}$ é o conjunto de entretenimentos disponíveis para um cliente $\mathbf{C_k}$.

O segundo objetivo é o **Inventário**, que representa a estimativa do dinheiro retido nas mãos do agente devido aos ingressos **ij** não alocados aos clientes, ou seja:

$$Inventário = \sum_{i=1}^{4} \sum_{j=1}^{3} T_{ij} * l_{ij}^{A} * Pr_{ij} \quad (3.2)$$

onde $\mathbf{l_{ij}^A}$ indica se o agente deve incrementar (**1**), ou não (**0**), o número de ingressos **ij** para venda. Para apoiar o processo de seleção, foi utilizada uma matriz $\mathbf{Pr_{ij}}$ que guarda a estimativa dos preços de transação para os leilões **ij**. onde **i** representa o dia do entretenimento (assumindo valores inteiros no intervalo de 1 a 4); e, **j** o tipo de entretenimento (assumindo valores inteiros no intervalo de 1 a 3, onde 1 equivale ao tipo AW, 2 ao tipo AP e 3 ao tipo MU). Os valores nessa matriz são os valores estimados pela RNA, contudo, enquanto o valor da RNA não for atualizado, utiliza-se o valor da mediana de vetores ordenados para cada leilão, com transações passadas.

O terceiro objetivo é a $\mathbf{Despesa\ Operacional_k}$, que representa a estimativa do dinheiro que o agente gastará para comprar ingressos para o cliente $\mathbf{C_k}$, ou seja.

$$DespesaOperacional_k = \sum_{i \in Nd^k} \sum_{j \in J^k} l_{ij}^{k} * Pr_{ij} \quad (3.3)$$

Assim, considerando os objetivos definidos, o problema de otimização multi objetivo foi formulado da seguinte maneira:

maximizar $\sum_{\mathbf{k=1}}^{\mathbf{8}} \mathbf{Ganho_k}$;
minimizar $\mathbf{Inventário}$
minimizar $\sum_{\mathbf{k=1}}^{\mathbf{8}} \mathbf{Despesa\ Operacional_k}$
s. a.:

$$\forall \mathbf{k} \in \mathbf{Clientes}: \forall \mathbf{i} \in \mathbf{Nd^k}: \sum_{\mathbf{j} \in \mathbf{J^k}} \mathbf{l_{ij}^k} \leq \mathbf{1} \quad (3.4)$$

$$\forall \mathbf{k} \in \mathbf{Clientes}: \forall \mathbf{j} \in \mathbf{J^k}: \sum_{\mathbf{i} \in \mathbf{Nd^k}} \mathbf{l_{ij}^k} \leq \mathbf{1} \quad (3.5)$$

$$\forall \mathbf{k} \in \mathbf{Clientes}: \forall \mathbf{i} \in \mathbf{Nd^k}: \forall \mathbf{j} \in \mathbf{J^k}: \mathbf{l_{ij}^k} * \mathbf{b_o^{ij}} \leq \text{máximo que o agente pode pagar na compra do ingresso } \mathbf{ij} \quad (3.6)$$

$$\forall \mathbf{i} \in \mathbf{Nd}: \forall \mathbf{j} \in \mathbf{Entretenimentos}: \mathbf{T_{ij}} * \mathbf{l_{ij}^A} * \mathbf{a_o^{ij}} \geq \text{mínimo que o agente pode receber na venda do(s) ingresso(s) } \mathbf{ij} \quad (3.7)$$

$$\sum_{\mathbf{i=1}}^{\mathbf{Nd}} \sum_{\mathbf{j=1}}^{\mathbf{Ne}} \mathbf{T_{ij}} * \mathbf{l_{ij}^A} \leq \sum_{\mathbf{k=1}}^{\mathbf{Nc}} \sum_{\mathbf{i} \in \mathbf{Nd^k}} \sum_{\mathbf{j} \in \mathbf{J^k}} \mathbf{l_{ij}^k} \quad (3.8)$$

onde **Clientes** é a carteira de clientes; $\mathbf{Nd^k}$ equivale ao conjunto de dias que o cliente $\mathbf{C_k}$ tem para entretenimento; **Entretenimento = {1,2,3}** são os tipos de entretenimento disponíveis para os agentes; **Ne** é o número de tipos de entretenimentos e **Nc** número de clientes; $\mathbf{a_o^{ij}}$ é o preço de cotação de venda no leilão do tipo **ij;** $\mathbf{b_o^{ij}}$ é preço de cotação de compra no leilão do tipo **ij**.

A primeira restrição (3.4) garante que serão alocados ingressos apenas nos dias **j** em que o cliente **k** ficará hospedado. A segunda restrição (3.5) garante que o agente não irá alocar mais de um ingresso do tipo **i** no dia **j** para o cliente $\mathbf{C_k}$. A terceira restrição (3.6) garante que não seja pago um valor maior que a cotação do preço de compra. A quarta restrição (3.7) garante que somente serão alocados para venda, os ingressos cujo preço de cotação de venda for maior que o bônus do cliente. A quinta restrição (3.8) garante que são alocados para venda, apenas os ingressos que o agente possua.

3.4.2. Busca de uma Solução com Algoritmo Genético

Para resolver o problema de otimização formulado no Alocador, foi empregada uma abordagem fundamentada em uma metaheurística evolucionária, isto é, um algoritmo genético. Foram levados em consideração diversos pontos para essa escolha, tais como, facilidade de utilização, simplicidade na representação das soluções, complexidade e qualidade das soluções geradas. Neste trabalho, foi utilizado o *Nondominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002), que é um algoritmo genético desenvolvido para resolver problemas de otimização multiobjetivo, semelhante ao problema do Alocador.

O algoritmo NSGA-II tem complexidade $O(MN^2)$ onde M é o número de objetivos e N é o tamanho da população. Além de elitismo, o algoritmo utiliza um operador de diversificação não-paramétrico que calcula a distância entre soluções e escolhe as entre as existentes aquelas que estão mais dispersas pelo *Pareto-front*, fazendo com que não apenas as melhores soluções permaneçam na próxima geração, mas também soluções diversificadas e "distantes" umas das outras.

Para a implementação deste algoritmo foi utilizado o *framework* de metaheurísticas aplicadas a resolução de problemas de otimização multiobjetivo JMetal (DURILLO et al., 2006). Ele foi escrito na linguagem de programação Java e possui um conjunto extenso de metaheurísticas já implementadas, bem como uma interface bem definida para a utilização das mesmas além de fornecer diversas estruturas de dados para a implementação da representação do problema que se quer resolver. Dessa forma, o trabalho para a implementação do algoritmo genético foi voltado à criação da representação de soluções para o problema atacado.

Para o problema formulado neste trabalho, cada solução (gene) foi codificada como sendo um vetor ordenado, onde um conjunto de quatro alelos representa um cliente, e cada alelo representa um possível dia de viagem para o cliente. Cada alelo pode assumir um valor no intervalo de inteiros entre 0 e 3, ou seja: 0 se não for alocado ingresso; 1 se for alocado 1 ingresso do tipo AW no dia dado pela posição do alelo no conjunto de cada cliente; 2 se for alocado 1 ingresso do tipo AM; e 3 caso seja alocado 1 ingresso do tipo MU.

Os valores assumidos pelos alelos equivalem ao valor do índice **j**, da estrutura de três dimensões $\mathbf{l_{ij}^{k}}$, descrita na formalização do problema. Ou seja, quando $\mathbf{l_{ij}^{k}}$ é igual a 0, o alelo de posição $\left(\mathbf{i} + \left((\mathbf{k} - 1) \times 4\right)\right)$ recebe o valor 0; e, quando $\mathbf{l_{ij}^{k}}$ é igual a 1, o alelo de posição $\left(\mathbf{i} + \left((\mathbf{k} - 1) \times 4\right)\right)$ recebe o valor de **j**.

A Figura 3.5 apresenta parte de um gene descrito de acordo com o esquema elaborado. Nela o cliente C1 chega à cidade no dia 1 e retorna no dia 4, portanto são alocados ingressos do tipo 1 e tipo 2 para os dias 1 a 3, respectivamente. Já o cliente C2 permanece os 5 dias na cidade, mas como cada cliente não pode possuir mais de um ingresso de cada tipo, não foi alocado nenhum ingresso no dia 3. E o cliente C8 chega

no dia 2 e retorna no dia 5, contudo, não foi alocado nenhum ingresso no dia 2 para o cliente C8.

Cliente	C1				C2					C8			
Dia	1	2	3	4	1	2	3	4	.....	1	2	3	4
Alelo	1	0	2	0	2	1	0	3		0	0	2	3

Figura 3.5. Exemplo de gene de uma solução válida.

Desta forma, as operações de mutação e reprodução podem ser executadas de uma maneira simples pelo algoritmo genético utilizado. A mutação troca os valores de um alelo qualquer por outro valor dentro dos possíveis [0..3], respeitando as restrições; e, a reprodução se faz através de operações simples que dividem os genes em dois pedaços e os combinam em ordem aleatória.

3.5.Considerações Finais

O presente capítulo ilustrou em riqueza de detalhes o modelo e implementação das soluções que se utilizaram de técnicas de inteligência artificial, visando resolver os problemas listados na Introdução.

4. AVALIAÇÃO

O presente capítulo ilustra o processo de avaliação do agente LaconiBot. Nas seções iniciais descrevem-se alguns detalhes importantes sobre o ambiente de avaliação, bem como a metodologia de avaliação. Posteriormente, são analisados os desempenho na incorporação dos componentes descritos no capitulo anterior, tomando como controle, o agente DealerBot. Ao final do capítulo, é apresentado um resumo dos resultados, bem como a discussão dos mesmos.

4.1.Arquitetura do ambiente de execução disponível

A comunidade que organiza as competições TAC desenvolveu uma infra-estrutura de software completa para o desenvolvimento e execução de agentes negociadores. Ela é constituída de um servidor, um *framework* para implementação de novos agentes, e um *toolkit* para a consulta dos dados de log armazenados no servidor. O servidor é o ambiente no qual os agentes vão competir uns com os outros; e, assim como os outros itens dessa infra-estrutura, é desenvolvido em Java.

O *framework* é denominado *Agentware*. Através dele é possível implementar um novo agente para o cenário do TAC. Utilizando o mesmo, todo agente TAC deve herdar de uma classe denominada *AgentImpl*. Esta classe implementa toda a interface entre o agente e o ambiente. O *framework* também disponibiliza um agente exemplo denominado *DummyAgent*. Para implementar a arquitetura proposta na Figura 1.4 foi criada uma estrutura de classes conforme ilustrado na Figura 4.1.

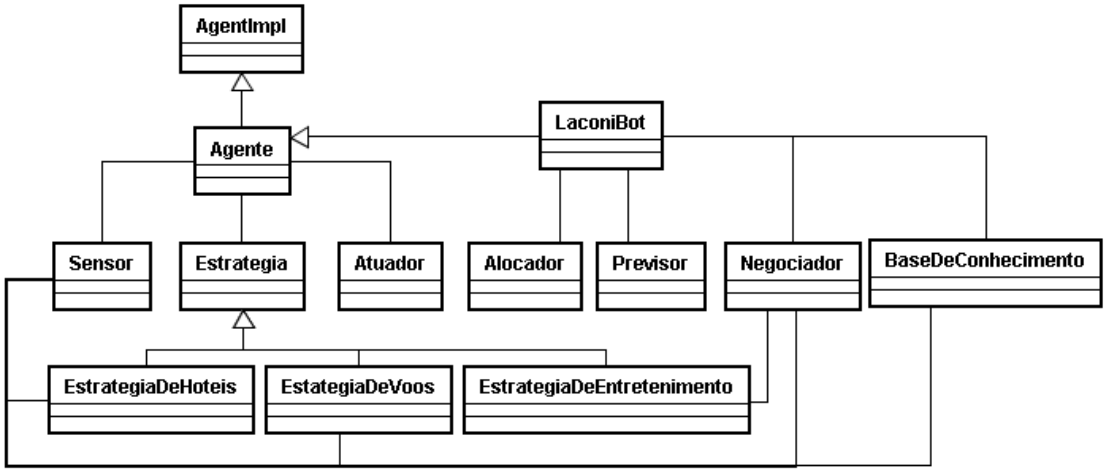

Figura 4.1. Diagrama com as principais classes do agente LaconiBot.

Como observado no diagrama de classes da Figura 4.1, a classe LaconiBot é uma especialização da classe Agente, a qual vem a ser a implementação do modelo arquitetural de agentes baseado em estados. A classe Agente define um agente TAC composto por: um sensor, classe Sensor; três estratégias, nas classes EstrategiaDeVoos, EstrategiaDeHoteis e EstrategiaDeEntretenimento; e um atuador, classe Atuador. À classe Sensor, são delegadas todas as percepções do ambiente, sendo ela a responsável por classificar a informação percebida e por notificar as estratégias sobre o novo estado do ambiente. Além disso, ela inicializa a classe Alocador e, baseado no retorno da estratégia de alocação, ela instancia um objeto EstrategiaDeEntretenimento referente ao leilão determinado na alocação.

As estratégias para o mercado de vôos e hotéis são idênticas ao do agente exemplo *DummyAgent*, uma vez que o objetivo do trabalho se concentra no mercado de entretenimento. Ao final, repassam-se as ações à classe Atuador, onde segundo o modelo, ela é a única classe capaz de atuar junto ao ambiente TAC.

4.2.Adaptações no ambiente de execução disponível

Para validar as soluções apresentadas neste trabalho, foi necessário executar o agente LaconiBot utilizando uma versão modificada do servidor TAC, uma vez que o objetivo do trabalho é analisar o desempenho do agente apenas em leilões do tipo CDA.

Nessa modificação, foi realizado um isolamento dos leilões de entretenimento dos demais leilões, conforme descrito a seguir.

Em Oliveira (2008) foi explicado, de forma resumida, quais modificações eram necessárias para isolar os leilões CDA do TAC dos demais leilões. De maneira mais detalhada, podem-se listar as seguintes modificações no código-fonte do servidor TAC:

1 - Na classe *ClassicMarket* foi modificado o método *setupClients*, no qual o prêmio pago pelo cliente por conseguir ficar no melhor hotel. Esse prêmio foi configurado para 0 (zero), fazendo com que não haja diferença de ganho entre alocar o cliente no melhor hotel ou não.

2 - Na classe *OnesideContinuousAuction2* foi modificada a forma como o preço das passagens se comporta. O valor inicial das mesmas foi configurado para iniciar em 0 (zero) e a variação do preço foi cancelada, fazendo com que as passagens continuem com o mesmo preço durante todo o leilão.

3 - Na classe *EngAscAuction* foi modificado o número de quartos disponíveis em cada hotel, fazendo todos os 64 clientes (8 agentes x 8 clientes/agente) se acomodarem em qualquer hotel e em qualquer dia. Esta mudança possibilitou o excedente de quartos, portanto o preço de compra dos mesmos sempre é igual a 0 (zero).

4.3.Metodologia de Experimentos

Avaliar cada uma das abordagens aqui descritas exige um método para a execução dos experimentos voltado a cada um dos subproblemas apresentados. O método de avaliação de desempenho do agente LaconiBot foi o mesmo que foi empregado para avaliar o agente DealerBot. Este método, desenvolvido por Oliveira (2008), dá origem a uma avaliação empírica apenas para os leilões do tipo CDA do TAC. Ele pode ser resumido em quatro passos principais:

1. Alterar o servidor TAC segundo as modificações detalhadas na seção 4.2.
2. Selecionar um conjunto de agentes competitivos.
3. Iniciar a bateria de testes, configurando os agentes para participarem de um determinado número de partidas.
4. Ao final de cada bateria de testes:

a. Analisar os logs do servidor para recuperar a quantidade de itens comprados e vendidos por cada agente;
b. Analisar os logs do servidor para recuperar a quantidade de lances enviados por cada agente;
c. Analisar a média da pontuação final, bem como a colocação de cada agente ao final de cada bateria de testes;

Ao todo, quatro agentes foram selecionados: *Mertacor*, *SICS02, UTTA06 e DummyAgent*. Destes, exceto o agente *DummyAgent*, todos foram finalistas da competição TAC *Classic* 2006. Além disso, *SICS02* e *Mertacor* também foram finalistas em 2005, ano em que este último foi o campeão do evento. Todos esses agentes estão disponíveis no site do torneio (http://www.sics.se/tac).

Contudo, para avaliar o impacto de cada uma das abordagens no desempenho final do agente LaconiBot, foi necessário incorporar ao método critérios de comparação entre tais abordagens. Dessa forma, alterou-se o 4º passo do método de Oliveira (2008), onde a análise dos *logs* foi realizada para recuperar a pontuação máxima ideal e a pontuação final, obtidas por cada agente ao final de cada partida. De acordo com Melo (2003), "A pontuação de cada agente é calculada baseada nas preferências dos clientes para pacotes montados pela agência e nas despesas líquidas do custo da viagem".

Dessa forma, a análise dos logs foi realizada para recuperar as seguintes informações:

a. Pontuação máxima ideal[6] de cada agente, ao final de cada partida.
b. Pontuação final[7] obtida por cada agente, ao final de cada partida.

As próximas seções exemplificam a utilização do método, bem como a forma de avaliação das estratégias, os resultados da avaliação de cada abordagem, e, posteriormente, do desempenho do agente contemplando todas as abordagens.

[6] A pontuação máxima, ou ótima, de cada agente pode ser calculada como o somatório do retorno máximo da função utilidade (2.1) para todos seus clientes, somada da venda dos bens sobressalentes pelo valor máximo de $200.0 e a compra de bens faltosos por $0.0.

[7] A pontuação final é dada pela função (2.2).

4.4.Análise dos Resultados

Foi necessário estabelecer um grupo de amostras de controle visando apoiar uma análise comparativa sobre o impacto no desempenho do agente LaconiBot devido à incorporação dos novos componentes da arquitetura do agente DealerBot. Dessa forma, tomou-se como grupo controle os experimentos realizados com o agente DealerBot, uma vez que tal agente possui a mesma estratégia base do processo de negociação do agente LaconiBot, com exceção dos componentes Previsor e Alocador.

4.4.1. Resultados para o agente DealerBot

A Tabela 4.1 e o gráfico da Figura 4.2 ilustram o desempenho do agente de controle (DealerBot). A anpalise dos mesmos indica que o agente obteve uma média de pontos igual a 9339,55 e uma média de 85,96% sobre o percentual do ótimo. A média de pontos do 1º colocado foi 9474,87 e a diferença de pontos entre o agente e o 1º colocado foi de 135,32 pontos, ou seja,1,43%.

Tabela 4.1. Pontuação e classificação do agente DealerBot.

Posição	10 jogos		20 jogos		40 jogos	
	Agente	**Pontuação**	**Agente**	**Pontuação**	**Agente**	**Pontuação**
1	**DealerBot**	9347.63	**Mertacor**	9518.15	**Mertacor**	9558.85
2	**Mertacor**	9316.15	**SICS02**	9368.68	**DealerBot**	9345.54
3	**UTTA06**	9234.74	**DealerBot**	9325.48	**UTTA06**	9237.42
4	**SICS02**	9169.22	**DummyAgent**	9255.28	**DummyAgent**	9205.06
5	**DummyAgent**	9143.40	**UTTA06**	9128.85	**SICS02**	9189.23

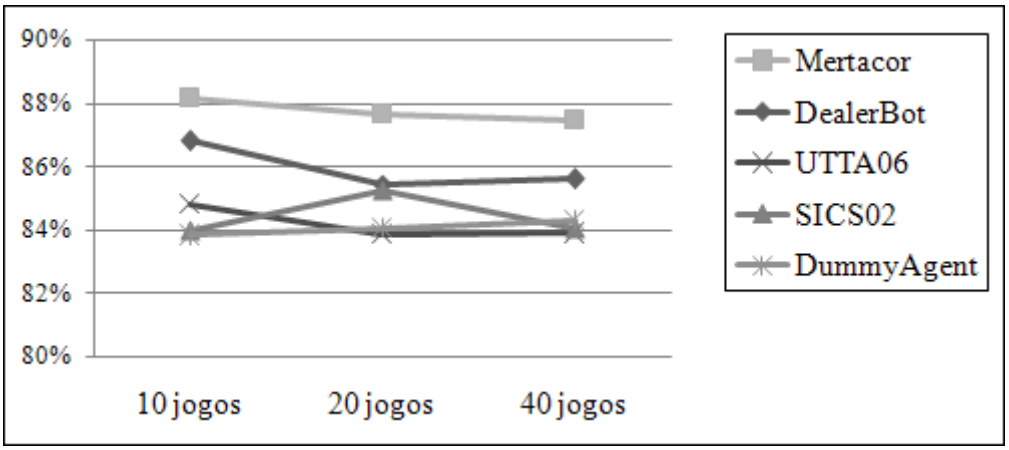

Figura 4.2. Percentual do ótimo para o agente DealerBot.

Através do gráfico acima, é possível ver que o agente campeão, o Mertacor, consegue manter a constância de um alto percentual sobre o ótimo dos pontos. O que reflete diretamente sobre sua pontuação, mostrando que sua estratégia de lucro a longo prazo garante o melhor desempenho quando comparado com os demais. Apesar de o agente DealerBot obter uma boa pontuação nos 10 primeiros jogos, chegando a superar o Mertacor, seu rendimento com relação ao percentual do ótimo não segue uma constância e não chega a ser tal alto quanto o do Mertacor.

4.4.2. Resultados para o agente LaconiBot sem o componente Alocador

Validação da RNA

Com o objetivo de validar o treinamento executado, duas formas de validação foram utilizadas. Primeiro foi avaliado o *Root Mean Square Error* (RMSE) na saída da rede, em seguida, é feita a análise por matriz confusão, e, por fim, foi calculado o erro médio obtido no treinamento da rede.

Root Mean Square Error (RMSE)

A rede implementada neste projeto foi treinada de acordo com a metodologia 10-*fold*, portanto 10 valores de erros foram gerados, um para cada *fold*. Dessa forma, para cada etapa do treinamento da rede é gerada uma estimativa de erro, ou RMSE global, e é necessário calcular a média dos valores de RMSE gerado. O RMSE para cada etapa é calculado de acordo com a seguinte equação:

$$RMSE = \sqrt{\frac{\sum_{k=1}^{N}(\frac{a_k - y_k}{a_k})^2}{N}} \quad (4.1)$$

onde N é o número de padrões, a_k representa o valor real , e y_k representa o valor previsto. Os valores gerados assim como a média global do RMSE são demonstrados na Tabela 4.2.

Tabela 4.2. Demonstra os valores de RMSE obtidos no treinamento da rede neural.

Etapa do Treinamento/Teste	Valor do RMSE	Valor * 100
1	0.03790378821188909	3.790
2	0.03792054262127483	3.792
3	0.03741238428226856	3.741
4	0.03837015819945445	3.837
5	0.0375940506948018	3.759
6	0.03772424081175014	3.772
7	0.037880241856337225	3.788
8	0.03692442292505494	3.692
9	0.037551458993737144	3.755
10	0.036275948189295366	3.627
Erro Global	**0.03755572367858635**	**3.755**

Como constatado na tabela 4.2, o erro gerado pelo treinamento ficou em torno de 3%. É importante salientar que somente a obtenção do RMSE não é o bastante para a validação correta em um treinamento de uma RNA. Dessa forma, foi utilizada outra métrica, ilustrado na próxima seção.

Matriz-Confusão

Uma matriz confusão tem por objetivo mostrar a quantidade de amostras classificadas corretamente para cada classe de dados, portanto medindo a eficiência do processo analisado. A matriz-confusão utilizada nesse trabalho foi adaptada para comparar os valores obtidos no treinamento e no teste da rede com os valores esperados.

Na construção da matriz-confusão foram adotados os intervalos percentuais de 0.26 a 0.30, 0.31 a 0.35, sucessivamente até 0.56 a 0.60. Tais intervalos equivalem aos valores de transação já normalizados. A justificativa da escolha desses valores se deve

ao fato de que esses são os intervalos em que as transações ocorrem com mais freqüência.

As matrizes foram calculadas para os conjuntos de treinamento e teste separadamente. Elas foram obtidas a partir da junção dos resultados obtidos em cada uma das etapas. As matrizes-confusão geradas para essas etapas correspondem aos valores obtidos pela rede treinada comparados aos valores esperados e estão demonstradas nas Tabelas 4.3 e 4.4. Pode-se perceber também a taxa de acerto da rede nos intervalos próximos ao que deveria ser estimado pelas matrizes demonstradas nas Tabelas 4.5 e 4.6.

Tabela 4.3. Matriz-Confusão de treinamento da rede com quantidade de acertos.

	[0.26-0.30]	**[0.31-0.35]**	**[0.36-0.40]**	**[0.41-0.45]**	**[0.46-0.50]**	**[0.51-0.55]**	**[0.56-0.60]**
[0.26-0.30]	**662**	93	0	0	0	2	0
[0.31-0.35]	438	**832**	248	61	41	16	9
[0.36-0.40]	43	941	**1241**	650	215	50	27
[0.41-0.45]	18	367	848	**2821**	415	134	83
[0.46-0.50]	8	8	3	383	**1677**	70	35
[0.51-0.55]	1	0	0	0	28	**139**	50
[0.56-0.60]	0	0	0	0	0	30	**57**

Tabela 4.4. Matriz-Confusão de treinamento da rede com porcentagem de acertos.

	[0.26-0.30]	**[0.31-0.35]**	**[0.36-0.40]**	**[0.41-0.45]**	**[0.46-0.50]**	**[0.51-0.55]**	**[0.56-0.60]**
[0.26-0.30]	**55.72%**	4.15%	0.00%	0.00%	0.00%	0.45%	0.00%
[0.31-0.35]	36.87%	**37.13%**	10.60%	1.56%	1.73%	3.63%	3.45%
[0.36-0.40]	3.62%	41.99%	**53.03%**	16.60%	9.05%	11.34%	10.34%
[0.41-0.45]	1.52%	16.38%	36.24%	**72.06%**	17.47%	30.39%	31.80%
[0.46-0.50]	0.67%	0.36%	0.13%	9.78%	**70.58%**	15.87%	13.41%
[0.51-0.55]	0.08%	0.00%	0.00%	0.00%	1.18%	**31.52%**	19.16%
[0.56-0.60]	0.00%	0.00%	0.00%	0.00%	0.00%	6.80%	**21.84%**

Tabela 4.5. Matriz-Confusão de teste da rede com quantidade de acertos.

	[0.26-0.30]	[0.31-0.35]	[0.36-0.40]	[0.41-0.45]	[0.46-0.50]	[0.51-0.55]	[0.56-0.60]
[0.26-0.30]	**68**	12	0	1	0	1	0
[0.31-0.35]	31	**91**	31	8	4	1	0
[0.36-0.40]	4	92	**137**	76	29	8	3
[0.41-0.45]	2	15	48	**300**	55	11	7
[0.46-0.50]	1	0	0	1	**139**	6	2
[0.51-0.55]	0	0	0	0	0	**15**	5
[0.56-0.60]	0	0	0	0	0	1	**5**

Tabela 4.6. Matriz-Confusão de teste da rede com porcentagem de acertos.

	[0.26-0.30]	[0.31-0.35]	[0.36-0.40]	[0.41-0.45]	[0.46-0.50]	[0.51-0.55]	[0.56-0.60]
[0.26-0.30]	**64.15%**	5.71%	0.00%	0.26%	0.00%	2.33%	0.00%
[0.31-0.35]	29.25%	**43.33%**	14.35%	2.07%	1.76%	2.33%	0.00%
[0.36-0.40]	3.77%	43.81%	**63.43%**	19.69%	12.78%	18.60%	13.64%
[0.41-0.45]	1.89%	7.14%	22.22%	**77.72%**	24.23%	25.58%	31.82%
[0.46-0.50]	0.94%	0.00%	0.00%	0.26%	**61.23%**	13.95%	9.09%
[0.51-0.55]	0.08%	0.00%	0.00%	0.00%	0.00%	**34.88%**	22.73%
[0.56-0.60]	0.00%	0.00%	0.00%	0.00%	0.00%	2.33%	**22.73%**

Nestas matrizes, os valores verticais representam os dados reais esperados e os valores horizontais, os dados estimados pela rede. Por exemplo, analisando a Tabela 4.6, no intervalo de 0.36 a 0.40, a rede teve um acerto de 1241 previsões, ou 53.03% do total de previsões para aquele intervalo. Para obter os dados reais de preço basta efetuar uma multiplicação do valor por 200 (o caminho inverso da normalização).

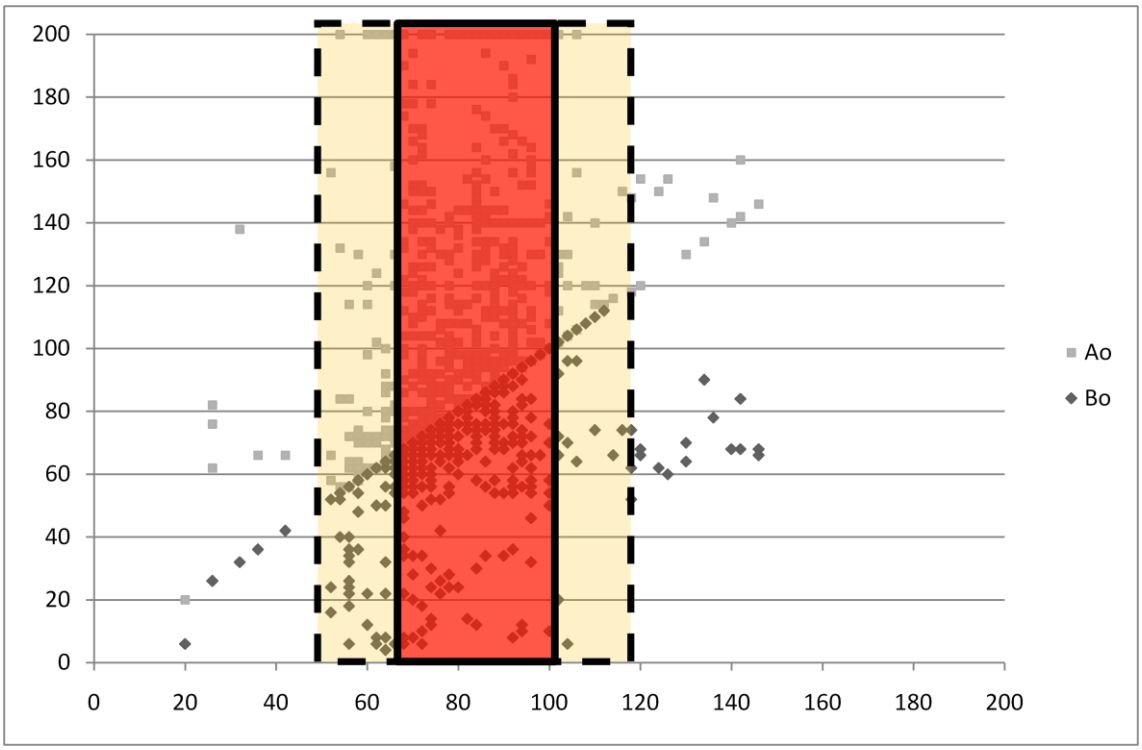

Figura 4.3. Gráfico dos Preços de Cotação para Compra (Bo) e Venda (Ao) X Preço de Transação.

O gráfico da Figura 4.3 ilustra a distribuição dos dados de cotação para compra, venda e preço de transação retirados do conjunto de treinamento. Neste gráfico estão representados os valores de cotação para compra (Bo) e venda (Ao) momentos antes da concretização de uma transação, cujo valor corresponde ao eixo das abscissas (horizontal). O retângulo de bordas listradas ilustra o limite dos dados analisados na Matriz-confusão (entre $52 e $120) e o retângulo de bordas lisas ilustra o limite dos dados com o maior número de instâncias, e também, onde ocorreu a melhor performance da RNA, apresentada na Matriz-confusão (entre $72 e $100).

Avaliação do agente com a RNA treinada

A Tabela 4.7 e o gráfico da Figura 4.4 ilustram o desempenho do agente LaconiBot sem o componente Alocador. Analisando os mesmos nota-se que o agente obteve uma média de pontos igual a 9288,93 e média de 86,40% sobre o percentual do ótimo. A média de pontos do 1º colocado foi 9465,96 com diferença de pontos entre o agente e o 1º colocado de 177,03 pontos, ou 1,87%.

Tabela 4.7. Pontuação e classificação do agente LaconiBot sem o Alocador.

Posição	10 jogos		20 jogos		40 jogos	
	Agente	Pontuação	Agente	Pontuação	Agente	Pontuação
1	**Mertacor**	9460.05	**Mertacor**	9455.27	**Mertacor**	9482.58
2	**LaconiBot**	9310.86	**SICS02**	9296.70	**LaconiBot**	9289.96
3	**SICS02**	9299.19	**LaconiBot**	9265.98	**DummyAgent**	9248.99
4	**DummyAgent**	9203.32	**UTTA06**	9213.99	**UTTA06**	9195.72
5	**UTTA06**	9187.31	**DummyAgent**	9091.94	**SICS02**	9165.85

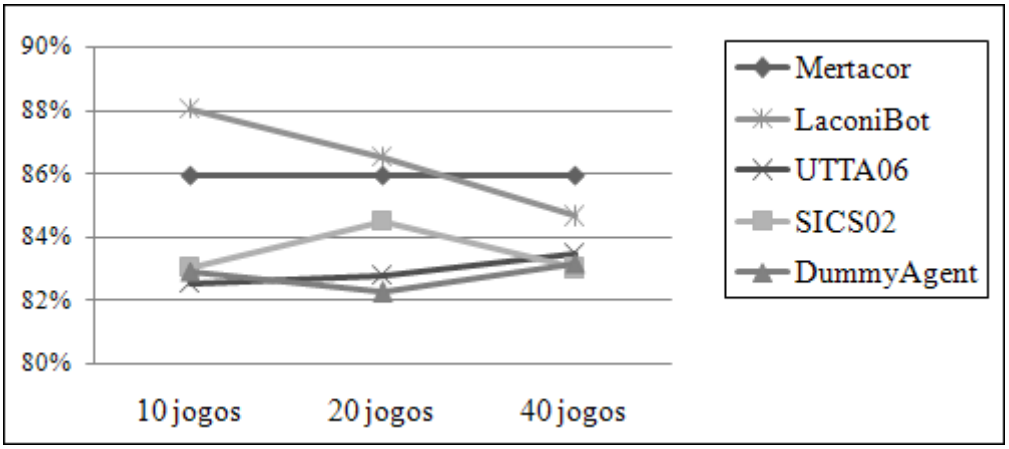

Figura 4.4. Percentual do ótimo para o agente LaconiBot sem o Alocador.

Analisando e o gráfico da Figura 4.4 é possível constatar que, mesmo ficando com o percentual do ótimo acima do agente campeão por 2 vezes, o LaconiBot não manteve uma constância, e acabou ficando com um desempenho pior que o agente DealerBot na média dos pontos (9288,93 contra 9339,55). Pode-se atribuir a piora no rendimento do agente ao fato da falta de calibração dos parâmetros do controlador nebuloso, pois com uma estimativa de preço mais apurada com o retorno da RNA, o agente acabava comprando mais bens que não iria utilizar, através das regras do controlador nebuloso, descritas na seção 2.2.5. Haja vista que não possuía nenhum mecanismo de seleção de leilões, logo ele participava de todos os leilões.

4.4.3. Resultados para o agente LaconiBot sem o componente Previsor

A Tabela 4.8 e o gráfico da Figura 4.5 ilustram o desempenho do agente LaconiBot sem o componente Previsor. Analisando os mesmos, nota-se que o agente obteve uma média de pontos igual a 9316,22 e média de 84,37% sobre o percentual do ótimo. A média de pontos do 1º colocado foi 9497,47 com diferença de pontos entre o agente e o 1º colocado de 181,25 pontos, ou 1,90%.

Tabela 4.8. Pontuação e classificação do agente LaconiBot sem o Previsor.

Posição	10 jogos		20 jogos		40 jogos	
	Agente	Pontuação	Agente	Pontuação	Agente	Pontuação
1	**Mertacor**	9460.43	**Mertacor**	9562.20	**Mertacor**	9469.79
2	**LaconiBot**	9372.86	**LaconiBot**	9321.54	**LaconiBot**	9254.27
3	**UTTA06**	9249.43	**UTTA06**	9298.85	**DummyAgent**	9222.63
4	**DummyAgent**	9230.95	**DummyAgent**	9233.11	**UTTA06**	9180.90
5	**SICS02**	8234.58	**SICS02**	8337.40	**SICS02**	8356.34

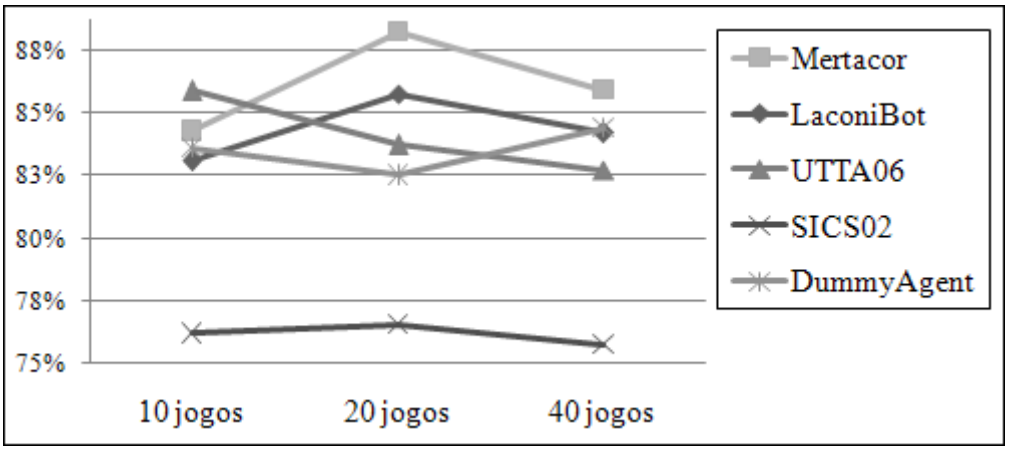

Figura 4.5. Percentual do ótimo para o agente LaconiBot sem o Previsor.

O gráfico da Figura 4.5 mostra que a estratégia de seleção de lances retornou uma curva semelhante para o percentual do ótimo, quando comparados os agentes LaconiBot e Mertacor. Contudo, o desempenho na pontuação do LaconiBot ficou inferior ao do agente controle, DealerBot. Pode-se explicar tal situação uma vez que o LaconiBot acabava participando de uma quantidade menor de leilões e não houve

mudança nos parâmetros do seu controlador, o que remetia a uma quantidade menor de itens negociados.

Avaliação dos objetivos da estratégia do Alocador

Considerando o conflito entre objetivos, como, por exemplo, entre Ganho e Despesa Operacional, visando analisar o impacto na mudança da maximização e minimização dos objetivos, foram testadas algumas combinações, que refletiam os seguintes cenários:

1. Fazer o agente comprar um número maior de itens do que vendê-los: maximização do *Ganho* e *Inventário*, minimização da *DespesaOperacional*;
2. Fazer o agente vender um número maior de itens do que comprá-los: maximização do *Ganho* e *DespesaOperacional*, minimização do *Inventário*;
3. Fazer o agente comprar e vender a maior quantidade de itens que pudesse: maximização de todos os objetivos;
4. Fazer o agente alocar todos os itens que ele já possuía, além de comprar e vender a menor quantidade de itens possível: maximização do *Ganho* e minimização da *DespesaOperacional* e do *Inventário.*

Como conseqüência, observou-se que a combinação utilizada no cenário 4 refletiu no melhor desempenho do agente, o que fez com que a mesma se tornasse a combinação adotada na avaliação da técnica.

4.4.4. Resultados para o agente LaconiBot com todos os componentes

A Tabela 4.9 e o gráfico da Figura 4.6 ilustram o desempenho do agente LaconiBot com todos os componentes presentes. Analisando os mesmos nota-se que o agente obteve uma média de pontos igual a 9389,69 e média de 84,19% sobre o percentual do ótimo. A média de pontos do 1º colocado foi 9489,56 com diferença de pontos entre o agente e o 1º colocado de 99,87 pontos, ou 1,05%.

Tabela 4.9. Pontuação e classificação da simulação LaconiBot completo.

Posição	10 jogos		20 jogos		40 jogos	
	Agente	Pontuação	Agente	Pontuação	Agente	Pontuação
1	**Mertacor**	9445.03	**Mertacor**	9539.59	**Mertacor**	9484.06
2	**LaconiBot**	9417.63	**LaconiBot**	9371.73	**LaconiBot**	9379.72
3	**UTTA06**	9324.26	**UTTA06**	9235.29	**UTTA06**	9274.00
4	**DummyAgent**	9260.46	**SICS02**	9198.99	**SICS02**	9230.51
5	**SICS02**	9245.65	**DummyAgent**	9191.72	**DummyAgent**	9198.55

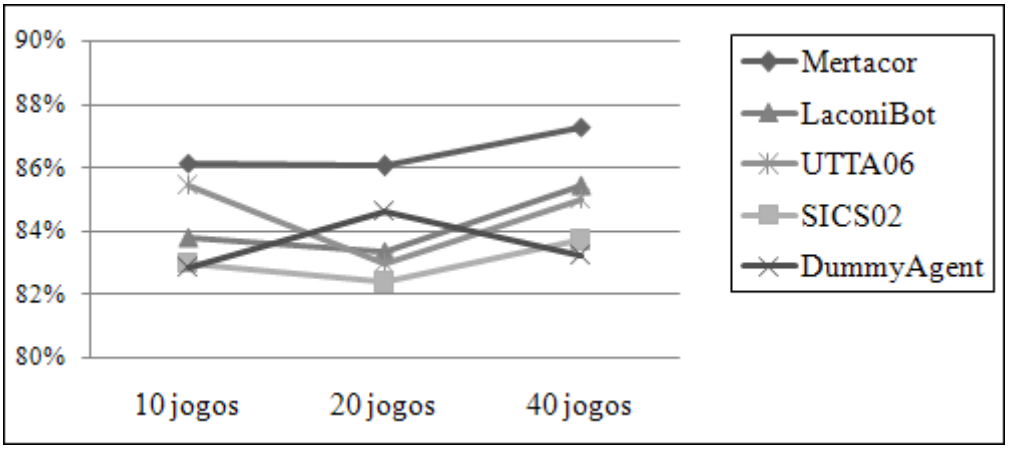

Figura 4.6. Percentual do ótimo para o agente LaconiBot completo.

Analisando o gráfico da Figura 4.6, novamente, percebe-se a semelhaça na curva do percentual do ótimo, quando comparados os agentes LaconiBot e Mertacor. Já o desempenho na pontuação do LaconiBot ficou superior ao do agente controle, DealerBot. Pode-se explicar tal situação uma vez que o LaconiBot estava participando somente dos leilões sobre os itens que ele tinha interesse em comprar ou vender; através do retorno do componente Alocador. Além disso, os lances enviados remetiam a uma estratégia um pouco mais agressiva, pois o retorno da RNA é mais precisa que a mediana de preços ordenados utilizada pelo DealerBot.

4.5. Resumo dos Resultados

A tabela as seguir, resume os principais dados necessários para avaliar o desempenho geral do LaconiBot, bem como sua evolução durante a incorporação dos seus componentes, quando comparado com o agente controle, o DealerBot.

Tabela 4.10. Resumo de todos os Resultados.

Agente	Média colocações	Média % do ótimo	Média Pont. final	Diferença % para 1º
DealerBot	2	85,96	9339,55	1,43
LaconiBot sem GA	2,3	86,40	9288,93	1,87
LaconiBot sem RNA	2	84,37	9316,22	1,90
LaconiBot Completo	2	84,19	9389,69	1,05

Como observado no resumo da tabela acima, o LaconiBot completo obteve um desempenho superior ao DealerBot em sua média de pontos. Além disso, ele conseguiu diminuir a diferença entre sua pontuação quando comparado com o agente campeão (o agente Mertacor). Apesar das estratégias incorporadas de maneira separada levarem a um desempenho insatisfatório, quando as mesmas trabalharam em conjunto percebeu-se uma melhora no desempenho do agente.

CONCLUSÕES E TRABALHOS FUTUROS

A pesquisa em comércio eletrônico, agentes negociadores e estratégias para negociação em CDA têm sido impulsionada por pesquisadores no mundo inteiro, gerando uma série de trabalhos que fazem uso das mais variadas técnicas estatísticas e de inteligência artificial. Como resultados dessas pesquisas surgiram ambientes como o TAC que facilitam sobremaneira o desenvolvimento de novos trabalhos voltados ao processo de negociação em comércio eletrônico.

Mais especificamente, este trabalho descreveu e avaliou o agente LaconiBot em uma versão modificada do ambiente TAC Classic. Tal agente se utiliza de técnicas de inteligência artificial voltadas a resolução de alguns problemas encontrados no processo de negociação CDA do TAC. Ou seja, para resolver o problema de determinação dos lances, foi utilizado um controlador Fuzzy. Para selecionar quais leilões o agente deve participar efetuando lances, foi utilizado um Algoritmo Genético Multi-Objetivo. E, para prover uma estimativa mais apurada dos valores de transação dos lances foi utilizada uma Rede Neural Artificial.

Para avaliar o agente foi necessário realizar algumas adaptações no cenário do ambiente TAC original, conforme discutido no capitulo de avaliação. Além disso, o ambiente TAC fornece apenas 2 parâmetros para análise do desempenho do agente, a saber: colocação e média da pontuação. Contudo, para uma análise mais detalhada foram utilizados mais alguns parâmetros como: média do percentual do ótimo obtido por cada agente, e percentual da diferença da média de pontos para o agente campeão.

Através da análise dos resultados constatou-se que a utilização das técnicas descritas neste trabalho proporcionaram um melhor desempenho do agente LaconiBot, quando comparado com o agente controle, o DealerBot implementado por Oliveira (2008). Apesar de permanecer na segunda colocação, o LaconiBot obteve um acréscimo tanto do percentual do ótimo quanto na média de pontos, e diminuiu a diferença da média de pontos para o agente campeão.

Além disso, a implementação do LaconiBot, por meio de sua arquitetura modular, bem como a descrição das abordagens aqui apresentadas podem ser úteis,

didaticamente, para o ensino de disciplinas como Engenharia de Software a Inteligência Artificial, servindo como ilustração da aplicação de teorias computacionais em um programa agente concreto construído para a resolução de problemas complexos.

Como trabalho futuro é possível observar que algumas modificações nos parâmetros do agente podem ser realizadas com o objetivo de resultarem em ganhos reais no seu desempenho, haja vista que muitos dos parâmetros de sua estratégia necessitam que seus valores sejam determinados pela intuição humana. Um modelo de adaptação automatizado, baseado no número de leilões selecionados pelo algoritmo genético, por exemplo, poderia fazer com que o agente se torne ou mais ou menos agressivo no cálculo de novos lances, influenciando diretamente em ganhos no seu desempenho.

Com relação a melhorias na RNA, indicam-se como trabalhos futuros a análise dos impactos na modificação da topologia da RNA, a implementação de outros modelos de RNA além do modelo MLP, e a utilização de uma RNA treinada para cada tipo de ingresso. Com relação à lógica do Alocador, pode ser analisado o impacto na utilização de outras metaheurísticas multi-objetivo além do algoritmo genético.

REFERÊNCIAS BIBLIOGRÁFICAS

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software architecture in practice. Addison-Wesley Professional, 2003.

BOYAN, J.; GREENWALD, A. Bid determination in simultaneous actions an agent architecture. 2001: 210-212.

DAS, R.; HANSON, J.E.; KEPHART; J.O.; Tesauro, G. Agent-human interactions in the continuous double auction. 17 (2001): 1169-1178.

DEB, K., PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6 (2002): 182-197.

DURILLO, J. J.; NEBRO, A. J.; LUNA, F.; DORRONSORO, B.; ALBA, B. jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics Departamento de Lenguajes y Ciencias de la Computacion, University of Malaga, ETSI Informática. *Campus de Teatinos, Tech. Rep. ITI-2006-10*, 2006.

GAREY, M. R. et al. Computers and Intractability: A Guide to the Theory of NP-completeness. Springer, 1979.

GOLDRATT, E.M.; COX, J. A meta um processo de melhoria contínua. Nobel, 2006.

GREFENSTETTE, J. J. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics* 16 (1986): 122-128.

GUPTA, P., MEHLAWAT, M. K.; SAXENA, A. Asset portfolio optimization using fuzzy mathematical programming. *Information Sciences* (Elsevier), 2007.

HAYKIN, S. Redes Neurais: princípios e prática. Porto Alegre: Bookman, 2001.

HE, M.; LEUNG, H.; JENNINGS, N.R. A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions. *IEEE Transactions on Knowledge and Data Engineering* (IEEE Computer Society), 2003: 1345-1363.

HOLLAND, J.H. Adaptation in natural and artificial systems. MIT Press Cambridge, MA, USA, 1975.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* (National Acad Sciences) 79 (1982): 2554-2558.

HOROWITZ, E.; SAHNI, S. Computing partitions with applications to the knapsack problem. *Journal of ACM*, 1974.

KARR, C.L. Design of an adaptive fuzzy logic controller using a genetic algorithm. *Morgan Kaufmann Publishers.* 1991. 450.

KASABOV, N.K. *Foundations of neural networks, fuzzy systems, and knowledge engineering.* The MIT press, 1996.

KEHAGIAS, D., TOULIS, P.; MITKAS, P. A Long-Term Profit Seeking Strategy for Continuous Double Auctions in a Trading Agent Competition. *Lecture Notes in Computer Science* (Springer) 3955 (2006): 116-126.

KLEMPLERER, P. Auction Theory: A Guide to the Literature. *Journal of Economic Surveys* 13 (1999): 227-286.

KO, PO-CHANG; LIN, PING-CHEN. Resource allocation neural network in portfolio selection. *Expert Systems with Applications* (Elsevier) 35 (2008): 330-337.

KOHONEN, T. An introduction to neural computing. *Neural Networks* 1 (1988): 3-16.

LEE, K.H. First course on fuzzy theory and applications. Springer, 2004.

LIN, C. C.; LIU, Y. T. Genetic algorithms for portfolio selection problems with minimum transaction lots. *European Journal of Operational Research* (Elsevier) 185 (2008): 393-404.

MARTELLO, S; TOTH, P. Knapsack problems: algorithms and computer implementations. JohnWiley & Sons, 1990.

MELO, G. S. Projeto Automático de Redes Neurais Artificiais para o Problema de Previsão em Séries Temporais. *Projeto Automático de Redes Neurais Artificiais para o Problema de Previsão em Séries Temporais.* 2008.

MELO, T. M. L. Estratégia multi-agente para leilões simultâneos de bens relacionados. Pontifícia Universidade Católica do Rio de Janeiro, 2003.

MENEZES JUNIOR, J. M. P. Redes Neurais Dinâmicas para Predição e Modelagem Não-Linear de Séries Temporais. Universidade Federal do Ceará, 2006.

MUNAKATA, T. Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More. Springer-Verlag New York Inc, 2007.

NÉRI, E.L. Agentes de Software: Delegando Decisões a Programas. *RAE eletrônica* (SciELO Brasil) 4 (2005).

NORONHA, T. F., ALOISE, D. J.; SILVA, M. M. Uma Abordagem sobre Estratégias Metaheurísticas. *Revista eletrônica de iniciação científica, http://www. sbc. org. br/reic* 1 (2001).

NYSE. Providers of mobile telephone services, including cellular, satellite and paging services. *NYSE. Providers of mobile telephone services, including cellular, satellite and paging services,* 2009. Disponível em: <http://www.nyse.com/about/listed /lc_ny_industry_7.html?ListedComp=All&supersector=23&sector=128&subsector= 1120128534197&start=1&startlist=1&item=1&prev=clicked&firsttime=done&default= 1>. Acesso em 29 jul. 2009.

OLIVEIRA, F.; SI, Y.W. Strategic Issues in Trading Agent Competition: TAC-Classic. 2006. 518-522.

OLIVEIRA, G. P. T. Uma Abordagem Fuzzy para o Mercado de Entretenimento da Trading Agent Competition. *Monografia. Fortaleza (CE): Universidade Estadual do Ceará/UECE.* 2008.

PARK, S., DURFEE, E.H.; BIRMINGHAM, W. P. Use of markov chains to design an agent bidding strategy for continuous double auctions. *Journal of Artificial Intelligence Research* 22 (2004): 175-214.

PETRIDIS, V.; KEHAGIAS, A.. Predictive modular fuzzy systems for time-series classification. *IEEE Transactions on Fuzzy Systems* 5 (1997): 381-397.

PUTCHALA, R.; MORRIS, V. K. R. S. S. kavayaH: A Trading Agent developed for TAC-02. *kavayaH: A Trading Agent developed for TAC-02.* 2002.

RIBEIRO, C. O.; SOSNOSKI, A. A. K. B. ; RUSSI, B. Otimização Multiperíodo de Carteiras de Investimento utilizando a técnica de Geração de Árvores de Cenários. *Encontro Nacional de Engenharia de Produção* (ENEGEP), 2007.

RODRIGUES, F.L.; LEITE, H.G., NASCIMENTOS, H., SOUZA, A.L.; SILVA, G.F. Metaheurística Algoritmo Genético para Solução de Problemas de Planejamento Florestal com Restrições de Integridade. *Sociedade de Investigações Florestais* (2004): 233-245.

RUMELHART, D.E.; MCCLELLAND, J.L. Parallel distributed processing: Explorations in the microstructure of cognition, *Vol. 1: Foundations.* MIT Press Cambridge, MA, USA, 1986.

RUSSELL, S.; NORVIG, P. Artificial Intelligence: a modern approach. Prentice Hall, 1995.

SANDRI, S.; CORREA, C. Lógica nebulosa. *ESCOLA DE REDES NEURAIS: CONSELHO NACIONAL DE REDES NEURAIS* (ITA) 5 (1999): 73-90.

SARDINHA, J. MAS-School e ASYNC: Um Método e um Framework para Construção de Agentes Inteligentes. tese de doutorado), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

SARDINHA, J.; MILIDIÚ, R.L.; PARANHOS, P.M.; CUNHA, P.M.; LUCENA, C. J. P. An Agent Based Architecture for Highly Competitive Electronic Markets. 2004. 16-18.

SHAW, M.; GARLAN, D. Software architecture: perspectives on an emerging discipline. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1996.

SICS. The Trading Agent Community. *SICS. The Trading Agent Community,* 2009. Disponível em: <http://www.sics.se/tac>. Acesso em 29 jul. 2009.

SOARES, A. S. Predição de séries temporais econômicas por meio de redes neurais artificiais e transformada wavelet: combinando modelo técnico e fundamentalista. Universidade de São Paulo, 2008.

STONE, P.; GREENWALD, A. The first international trading agent competition: Autonomous bidding agents. *Electronic Commerce Research* (Springer) 5: 229-265, 2000.

STONE, P.; SCHAPIRE, R.E.; CSIRIK, J.A.; LITTMAN, M.L.; MCALLESTER, D. ATTac-2001: A learning, autonomous bidding agent. *Lecture notes in computer science* (Springer), 2002: 143-160.

TABARZAD, M.A.; LUCAS, C.; Haghjoo, P. A Heuristic Price Prediction and Bidding Strategy for Internet Auctions. *IJCSNS* 6 (2006): 161.

TAFNER, M.A.; XEREZ, M.; RODRIGUES FILHO, I.W. Redes neurais artificiais: introdução e princípios de neurocomputação. *Blumenau: EKO* 202 (1996).

VYTELINGUM, P. The structure and behaviour of the Continuous Double Auction. *Tese de Doutorado. University of Southampton*, 2006.

WEISS, G. Multiagent systems: a modern approach to distributed artificial intelligence. The MIT Press, 2000.

WELLMAN, M.P.; GREENWALD, A.; STONE, P. Autonomous bidding agents: strategies and lessons from the trading agent competition. The MIT Press, 2007.

WELLMAN, M.P.; WURMAN, P.R. A trading agent competition for the research community. 1999.

WITTEN, I.H.; FRANK, E. Data mining: practical machine learning tools and techniques. Elsevier, 2005.

ZADEH, L. Fuzzy sets Fuzzy Systems and AI Reports and Letters, Morgan Kaufmann: 61, 129-136, 1965