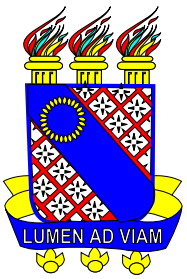


UNIVERSIDADE ESTADUAL DO CEARÁ

GILZAMIR FERREIRA GOMES

UMA ABORDAGEM MULTI-AGENTE PARA COMPETIÇÃO DE AGENTES NEGOCIADORES EM GERENCIAMENTO DE CADEIAS DE SUPRIMENTO

FORTALEZA, CEARÁ

2010

GILZAMIR FERREIRA GOMES

UMA ABORDAGEM MULTI-AGENTE PARA COMPETIÇÃO DE AGENTES NEGOCIADORES EM GERENCIAMENTO DE CADEIAS DE SUPRIMENTO

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Centro Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Gustavo Augusto Lima de Campos

FORTALEZA, CEARÁ

2010

G633a Gomes, Gilzamir Ferreira.

Uma Abordagem Multi-Agente para Competição de Agentes Negociadores em Gerenciamento de Cadeias de Suprimento / Gilzamir Ferreira Gomes. – Fortaleza, 2010.

119p.;il.

Orientador: Prof. Dr. Gustavo Augusto Lima de Campos

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

1. Sistemas Multi-Agente 2. Inteligência Artificial 3. Produtos Inteligentes I. Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

CDD:001.6

GILZAMIR FERREIRA GOMES

UMA ABORDAGEM MULTI-AGENTE PARA COMPETIÇÃO DE AGENTES NEGOCIADORES EM GERENCIAMENTO DE CADEIAS DE SUPRIMENTO

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Centro Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Gustavo Augusto Lima de Campos(Orientador)
Universidade Estadual do Ceará

Prof. Dr. Joaquim Celestino Júnior
Universidade Estadual do Ceará

Prof. Dr. Antonio Clecio Fontelles Thomaz
Universidade Estadual do Ceará

Prof. Dr. João Bosco da Mota Alves
Universidade Federal de Santa Catarina

AGRADECIMENTOS

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

Ao meu orientador, professor Gustavo, o meu muitíssimo obrigado pela paciência e confiança depositados.

Aos amigos, Enyo, Robson, Alex Martins, Fabiano, Lourival, Deladier, Walisson e a todos os amigos que conheci na UECE.

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

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

"Aventure-se, pois da mais insignificante pista surgiu toda riqueza que o homem já conheceu"
John Masefield

Resumo

Esta dissertação objetivou a concepção de uma abordagem computacional para o problema de gerenciamento de cadeias de suprimento no cenário *Trading Agent Compeition for Supply Chain* (TAC-SCM) quando perturbações no processo de fabricação são consideradas. Foi realizada uma investigação sobre a natureza centralizada e descentralizada do planejamento. Simulações realizadas têm demonstrado que a abordagem descentralizada baseada em produtos inteligentes é promissora em termos de robustez do sistema de controle e planejamento. Contudo, essa mesma abordagem tem obtido resultados pobres em termos de lucro. Dessa forma, é requerido mais investigação sobre como sistemas podem utilizar o melhor de ambas as abordagens: planejamento centralizado e descentralizado. Para se chegar a uma arquitetura multi-agente híbrida e robusta à estes problemas, foi investigado como a regulamentação social de mercado, os produtos inteligentes e aprendizagem de máquina poderiam ser combinados em um mecanismo de controle distribuído de recursos e responsabilidades. Os resultados mostraram que a solução desenvolvida é promissora quando combinada com técnicas adequadas de aprendizagem de máquina (ou mesmo boas heurísticas) e algum mecanismo de mercado controlando a distribuição de recursos para os elementos distribuídos do sistema.

Palavras-Chave: Sistemas Multi-Agente, Inteligência Artificial, Produtos Inteligentes

Abstract

This work aimed to design a computational approach to supply chain management problem in the scenario *Trading Agent Competition for Supply Chain* (TAC-SCM) when disturbances on manufacturing process are considered. Simulations have shown that the decentralized approach based on smart products is promising in terms of robustness of the system of control and planning. However, this same approach has achieved poor results in terms of profit. Thus, it is required more research into how systems can make the best of both approaches: centralized planning and decentralized planning. To achieve a hybrid multi-agent system and robust to these problems, we investigated how the social regulation of the market, intelligent products and machine learning could be combined as a mechanism of distributed control of resources and responsibilities. The result showed that the solution developed is promising when combined with techniques appropriate machine learning (or even good heuristics) and some market mechanism controlling the distribution of resources for distributed elements of the system.

Keywords: Multi-Agent System, Artificial Intelligence, Intelligent Products

Lista de Figuras

Lista de Tabelas

LISTA DE SIGLAS

AC	Agente Central
ACC	Agente de Compra
AE	Agente de Entrega
AP	Agente de Produção
ATC	Agente de Tipo de Componente
ATP	Agente de Tipo de Produto
ATO	Assembler to Order
AV	Agente de Vendas
IP	Intelligent Product
PC	Personal Computer
RFQ	Request for Quotes
RMSE	Root Mean Squared Error
RNA	Rede Neural Artificial
SCM	Supply Chain Management
SMA	Sistema Multi-Agente
TAC	Trading Agent Competition

Sumário

1 INTRODUÇÃO

Motivação

O crescimento e popularização dos sistemas de informação e uso intensivo da tecnologia em diversos setores da economia têm possibilitado novas formas de se fazer negócio: complexos mecanismos de mercado executando sobre a Internet, gerenciamento de cadeias de suprimento utilizando tecnologias como auto-identificação (RFID e código de barras, por exemplo), GPS, processamento e armazenamento distribuído da informação. A necessidade do uso de sistemas inteligentes para tratar a crescente quantidade de informações produzidas nesse cenário tecnológico tem motivado a busca de novas soluções para gerenciamento de cadeias de suprimento. Mais especificamente, a possibilidade de controlar e acompanhar automaticamente o movimento de mercadorias tem dado uma nova perspectiva em termos de abordagens descentralizadas para gerenciamento de cadeias de suprimento.

Quando se fala em cadeia de suprimentos, refere-se a companhias e atividades de empresas necessárias para projetar, produzir, entregar e utilizar produtos ou serviços (HUGOS, 2003). Outra forma de definir cadeia de suprimento é especificando seus estágios: uma cadeia de suprimento consiste de todos os estágios envolvidos, direta ou indiretamente, em satisfazer uma solicitação de cliente. Essa definição não apenas inclui o fabricante e o fornecedor, mas também transportadores, armazéns, varejistas, atacadistas e os próprios consumidores(CHOPRA; MEINDL, 2001). Os participantes dessa cadeia formam elos por onde trafegam mercadorias, dinheiro e informações. O sucesso das empresas depende de suas cadeias de suprimento, pois é por meio destas que conseguem o que é necessário para seus negócios e para obterem sucesso.

Gerenciar uma cadeia de suprimento consiste em influenciar o comportamento da cadeia para se obter um resultado desejado. Para se conseguir este resultado, é realizado o planejamento e a coordenação de atividades de organizações ao longo da cadeia, desde a obtenção da matéria-prima até a entrega de bens finalizados (COLLINS et al., 2006). Assim, na atual economia globalizada, o gerenciamento eficaz da cadeia de suprimentos é vital para a competitividade de empresas, pois isso impacta diretamente em sua habilidade para se adequar às mudanças de demanda no mercado de uma maneira eficaz e em tempo hábil (COLLINS et al., 2006).

O gerenciamento eficaz da cadeia de suprimento requer melhorias simultâneas em ambos: níveis de serviços de clientes e eficiência interna de operações das companhias na cadeia de suprimento. Serviço de cliente significa basicamente altas taxas de pedidos satisfeitos, alta taxa de entrega em tempo (sem atrasos) e baixa taxa de produtos voltados pelo cliente por qualquer

razão. Eficiência interna de organizações em uma cadeia de suprimentos significa que estas organizações obtêm uma taxa de retorno atrativa em seus investimentos em estoque e outros bens e que elas encontram formas de minimizar seus gastos operacionais e de vendas.

Portanto, o desafio do gerenciamento da cadeia de suprimento está justamente no planejamento e coordenação de atividades interdependentes, mantendo bons níveis de serviços aos clientes e, ao mesmo tempo, melhorando a eficiência interna de organizações. Tomadas individualmente, diferentes requisitos da cadeia de suprimentos tem necessidades conflitantes: o requisito de manter altos níveis de serviço de clientes requer a manutenção de altos níveis de estoque, mas o requisito para operar eficientemente requer reduzir o nível de estoque (HUGOS, 2003). Ainda segundo Hugos (2003), quando estes requisitos são vistos juntos como parte de um grande cenário é que caminhos podem ser encontrados para efetivamente balancear suas diferentes demandas. Por isso, o gerenciamento de cadeia de suprimento visualiza a cadeia de suprimento e as organizações que a compõe como uma única entidade. Uma fábrica em uma cadeia de suprimentos tipicamente possui recursos para a produção de bens que são vendidos a varejistas ou consumidores finais. Os recursos podem ser obtidos de fontes externas, como a matéria-prima utilizada na produção de bens, ou disponível internamente, como a capacidade de produção das linhas de montagem de uma fábrica. A determinação de como estes recursos são utilizados internamente no processo de produção de bens para suprir a demanda externa é o que define o planejamento de uma fábrica na cadeia de suprimentos. Assim, há tipicamente dois tipos de abordagens de planejamento da produção: centralizada e descentralizada. No planejamento centralizado, estratégias de otimização são utilizadas para se determinar a alocação ótima de recursos, cabendo a um planejador central a tarefa de alocação. Já no planejamento descentralizado, a utilização de recursos é resolvida como um problema de coordenação de entidades auto-gerenciáveis que alocam e utilizam estes recursos localmente.

Para conseguir um alto nível de serviço aos clientes e eficiência interna de organizações, o planejamento centralizado tem sido consolidado ao longo de décadas de aplicação, principalmente devido à sua eficiência em minimizar o efeito chicote e otimização no uso de recursos (MEYER; WORTMANN, 2009). Também é mostrado que o planejamento centralizado tem suas deficiências na prática, causadas principalmente por pequenas perturbações que ocorrem no processo de fabricação e no transporte de mercadorias. Um exemplo de tais perturbações ocorre quando componentes que foram planejados para serem utilizados na produção são avariados. Neste caso, um componente similar precisa ser obtido de qualquer fornecedor para que o pedido continue com o plano original. Geralmente, estes tipos de perturbações não são mesmo conhecidos pelos planejadores centrais, mas são geralmente resolvidos em um nível local.

Meyer e Wortmann (2009) mostraram que uma estratégia de controle e planejamento mais robusta a estes tipos de problemas pode ser obtida a partir de uma função de planejamento descentralizada, inspirada no conceito de produtos inteligentes. Contudo, também foi observado que o controle descentralizado de desordens locais não obteve bons resultados quanto se mede o lucro total obtido. Uma questão levantada foi: uma solução utilizando o melhor de ambas as abordagens de planejamento, resulta em um gerenciamento eficaz da cadeia de suprimentos? Uma das principais motivações deste trabalho foi buscar uma resposta para esta questão.

Objetivos

Conceber uma abordagem multi-agente para competir no ambiente do *Trading Agent Competition for Supply Chain*(TAC-SCM) que seja robusta a diferentes métricas de gerenciamento de cadeias de suprimento. Outros objetivos foram: estabelecer uma abordagem que possa ser explorada em pesquisas futuras no contexto da competição de agentes negociadores e avaliar diferentes soluções de negociação em gerenciamento de cadeias de suprimento, de modo a identificar pontos fracos e pontos fortes de cada solução.

Metodologia

Para realizar os objetivos estabelecidos, uma abordagem multi-agente baseada em produtos inteligentes e regulamentação social foi concebida com base no trabalho de (MEYER; WORTMANN, 2009) e inspirado no conceito de regulamentação social (SHERMAN, 2001). O embasamento teórico está fundamentado em livros, artigos, periódicos, conferências, dissertações de mestrado e teses de doutorado. Portanto, com base no levantamento teórico realizado, a abordagem foi concebida com regulamentação social para estabelecer um mecanismo de coordenação das decisões de diferentes agentes e no uso de produtos inteligentes como inspiração para a concepção de entidades auto-gerenciáveis, mais especificamente, agentes que atuam como pedidos auto-gerenciáveis de clientes. Para mostrar a flexibilidade da abordagem, diferentes versões foram implementadas suportando diferentes técnicas de aprendizagem de máquina e heurística simples.

As técnicas de aprendizagem de máquina foram utilizadas para a geração de hipóteses para predição de preços de oferta de outros competidores em leilões de venda. Os modelos gerados foram treinados com base em logs de simulações geradas contra abordagens que apresentaram os melhores desempenhos em competições recentes do TAC-SCM. Em uma das versões, no lugar de hipóteses aprendidas com base em logs de simulações passadas, foi utilizada uma heurística simples para predição de preços.

A abordagem multi-agente concebida (doravante denominada UECEGRUNN) foi implementado sobre um arquitetura mostrada na Figura 1.1. Foi utilizada a linguagem de programação Java para codificação. Para alguns algoritmos de aprendizagem de máquina, foram utilizadas implementações do *Framework* Weka. Assim, no caso da predição de preços em vendas, diferentes técnicas de predição e classificação, como redes neurais artificiais e métodos de *boosting*, foram utilizadas.

Três versões do sistema concebido foram avaliadas, cada versão com uma técnica de aprendizagem de máquina diferente e uma versão com uma heurística simples desenvolvida. As diferentes versões foram executadas em um ambiente TAC-SCM modificado para gerar perturbações no processo de produção, como descrito por Meyer e Wortmann (2009).

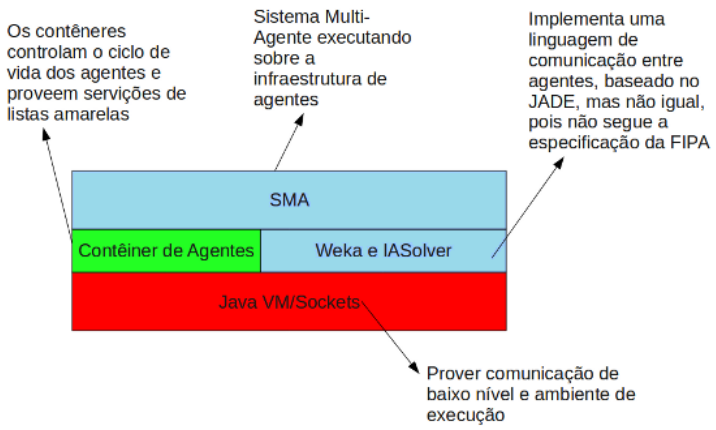


Figura 1.1: Infraestrutura de execução do SMA UECEGRUNN.

A avaliação da abordagem concebida ocorreu em três etapas. Na primeira etapa, as técnicas de aprendizagem de máquina e uma heurística simples foram avaliadas e comparadas no contexto da predição do preço de oferta de competidores. Nesta etapa diferentes medidas de desempenho foram utilizadas, como erro quadrático médio e matriz de confusão. Na segunda etapa, cada uma das versões com diferentes estratégias de predição de preço de oferta de competidores foi analisada em simulações contra um conjunto competitivo de abordagens com os melhores resultados nas últimas competições do TAC-SCM. Para cada estratégia foram realizadas 84 simulações, envolvendo três cenários com perturbações locais de produção, no caso, a probabilidade de componentes utilizados no processo de produção sofrerem avaria e não poderem ser utilizados: 0%, 5% e 10%. Na terceira etapa, o desempenho do sistema UECEGRUNN foi comparado com o desempenho do sistema GRUNN (MEYER; WORTMANN, 2009).

Organização

O restante deste trabalho é organizado como segue.

O Capítulo 2 aborda o gerenciamento de cadeias de suprimento no contexto da competição de agentes negociadores.

O Capítulo 3 aborda soluções consolidadas nos campeonatos do TAC-SCM e que serviram de embasamento para a concepção das estratégias de aprendizagem de máquina e heurísticas utilizadas no desenvolvimento do trabalho.

O Capítulo 4 aborda os fundamentos de Inteligência Artificial, Aprendizagem de Máquina e regulamentação social.

O Capítulo 5 aborda a arquitetura do sistema multi-agente concebido, as estratégias e as heurísticas utilizadas.

O Capítulo 6 apresenta e discute os resultados obtidos.

Finalmente, O Capítulo 7 apresenta as conclusões sobre o trabalho desenvolvido e indicações de trabalhos futuros.

2 Gerenciamento de Cadeias de Suprimento em Competições de Agentes Negociadores

Competição de agentes negociadores tem sido utilizada para investigar soluções autônomas em diferentes cenários de negociação. Para isso, foram concebidos diferentes cenários, como a venda de pacotes de viagens em leilões simultâneos e o gerenciamento de cadeias de suprimento. Simuladores e *frameworks* para o desenvolvimento de agentes para estes cenários tem sido desenvolvidos e disponibilizados, motivando a participação de pesquisadores de várias partes do mundo na busca de soluções autônomas para diferentes cenários de competição de agentes negociadores.

O objetivo deste capítulo é apresentar a problemática do gerenciamento de cadeias de suprimento no contexto do cenário TAC-SCM (*Trading Agent Competition for Supply Chain*). Para isso, na Seção 2.1 é apresentada uma visão geral do cenário de agentes negociadores no TAC-SCM. Na Seção 2.2, o papel dos agentes no cenário TAC-SCM é descrito em função das tarefas que os agentes devem realizar. Na Seção 2.3, as configurações possíveis de computadores pessoais (PCs) e os componentes necessários para a montagem destes PCs são apresentados. Na Seção 2.4, os fornecedores simulados são descritos em função das tarefas que realizam no ambiente do TAC-SCM. Na Seção 2.5, os clientes simulados são descritos em termos das tarefas que realizam no ambiente do TAC-SCM. Na Seção 2.6, o cenário do TAC-SCM modificado para suportar perturbações locais no processo de produção é apresentado. Finalmente, na Seção 2.7 é apresentada uma visão geral do problema de lidar com perturbações locais no processo de produção no contexto do TAC-SCM.

2.1 O Cenário de Competição de Agentes Negociadores para Gerenciamento de Cadeias de Suprimento

Trading Agent Competition (TAC) é um fórum internacional projetado para promover e encorajar pesquisas de alta qualidade para o problema de negociação entre agentes. Para isso, vários cenários de negociação são propostos. A primeira competição ocorreu em 2002 e foi baseada no cenário de "agentes de viagem", ou TAC Clássico. Desde 2003 a competição também tem um cenário de cadeias de suprimento, ou TAC-SCM, desenvolvido por pesquisadores do laboratório de *e-Supplay Chain Management* da Universidade *Carnegie Mellon* e do Instituto de Ciência da Computação Sueco (SICS – *Swedish Institute of Computer Science*).

O TAC-SCM é um cenário de montagem de computadores pessoais (PC – Personal

Computer) baseado na aquisição de componentes, fabricação de PCs e venda para clientes. Neste cenário, seis agentes "montadores de PC" (ou agentes) competem por pedidos de clientes e pela aquisição de vários tipos de componentes. Portanto, o TAC-SCM é um cenário que simula a problemática do gerenciamento de cadeias de suprimento e prover um ambiente de simulação onde diferentes soluções podem ser propostas.

O cenário do TAC-SCM foi projetado para suportar muitos dos desafios envolvidos em práticas dinâmicas para gerenciamento de cadeias de suprimento, enquanto mantém as regras do jogo simples o bastante para atrair um grande número de competidores para submissão de soluções.

Collins et al. (2006) realizam uma descrição detalhada da competição TAC-SCM. Essa competição ocorre ao longo de 220 dias simulados, cada dia simulado corresponde a 15 segundos. Ao longo dos 220 dias simulados, seis fábricas de computadores pessoais (PCs) competem pela venda de PCs para clientes e pela obtenção de componentes para a montagem de PCs. Ou seja, PCs são montados a partir de componentes como placa-mãe, memória principal, processador e disco rígido. Estes componentes são disponibilizados em diferentes versões (por exemplo, diferentes CPUs, diferentes placas-mãe, etc.) por vários fornecedores. As montadoras contam com uma fábrica com uma linha de montagem para PCs e um armazém para estocagem de componentes e PCs finalizados. A capacidade da fábrica é expressa em ciclos de produção. O problema das montadoras pode ser divido nos seguintes subproblemas: venda para cliente, obtenção de componentes, gerenciamento da produção e gerenciamento da entrega de produtos finalizados.

Dessa forma, os agentes são limitados pela capacidade de suas linhas de montagem e procuram por componentes de um conjunto de oito fornecedores. Cada componente está disponível em diferentes modelos. A cada dia, clientes lançam pedidos de orçamento (*Request For Quotes* – RFQs) aos agentes e selecionam ofertas enviadas pelos agentes de acordo com a data de entrega prometida e o preço da oferta. A demanda de clientes vem na forma de RFQs enviadas diariamente requisitando diferentes tipos de PCs, cada tipo sendo composto por uma combinação diferente de componentes.

O jogo TAC-SCM começa quando um ou mais agentes se conectam a um servidor de jogo. O servidor simula os fornecedores e clientes, prover um banco para serviços bancários, serviços de produção e estocagem para cada agente. O jogo continua por um número pré-definido de dias e, quando termina, o agente com maior saldo em conta bancária é declarado vencedor.

O jogo representa uma ampla variedade de situações de cadeias de suprimento. É um desafio que requer múltiplos agentes competirem de forma concorrente em múltiplos mercados (mercados para diferentes componentes no lado dos fornecedores e mercado para diferentes produtos no lado dos clientes) com interdependência e informações incompletas. Isso permite diferentes estratégias de agentes (por exemplo, especializar-se em um determinado tipo de produto, armazenar componentes que estão em baixa no mercado). Os agentes devem demonstrar suas habilidades ao reagir a variações na demanda de clientes e disponibilidade de fornecedores, assim como se adaptarem às estratégias adotadas por outros competidores. Nas próximas

seções, as regras do jogo e os papéis dos agentes competidores serão apresentados.

2.2 Agentes

Os agentes que competem no TAC-SCM são responsáveis pelas seguintes tarefas:

- Negociar contratos de suprimentos (componentes para a montagem de PCs). Para isso, os agentes enviam pedidos de orçamento (RFQs – Request for Quotes) de componentes para fornecedores.
- Enviar ofertas aos clientes em resposta aos pedidos de orçamento enviados pelos clientes.
- Gerenciar diariamente as atividades de uma linha de montagem de PCs.
- Entregar pedidos completos aos clientes.

Cada uma das tarefas listadas acima pode ser decomposta em duas ou mais sub-tarefas. Por exemplo, a tarefa de oferecer produtos aos clientes pode ser precedida da seleção clientes e seguida da decisão do valor da oferta. As quatro tarefas listadas e suas sub-tarefas são realizadas de forma autônoma pelos agentes, sem intervenção humana. A Figura 2.1 ilustra os eventos diários ocorridos no TAC-SCM. No início de cada dia, cada agente recebe:

- **Dos Clientes**
  - Requisições de orçamentos (RFQs – *Request for Quotes*) para PCs. Estas requisições ou solicitações representam a demanda diárias por PCs.
  - Pedidos em resposta às ofertas enviadas aos clientes no dia anterior.
  - Penalidades e notificação de cancelamento de pedidos devido aos atrasos nas entregas.
- **Dos Fornecedores**
  - Orçamento/Oferta de componentes em resposta às RFQs que o agente enviou no dia anterior.
  - Entrega de suprimentos para satisfazer pedidos anteriores. Os suprimentos (componentes) que podem ser usados para produção no dia seguinte ao da entrega.
- **Do Banco**
  - Saldo na conta do agente.
- **Da Fábrica**
  - Relatório do estoque dando a quantidade de componentes e PCs finalizados disponíveis.

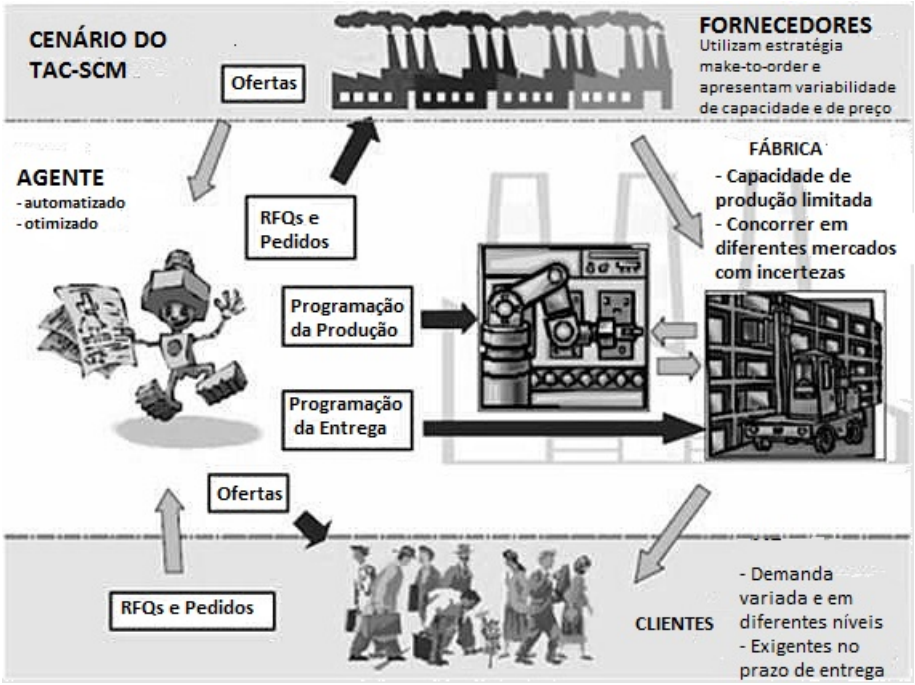


Figura 2.1: Eventos diários ocorridos durante alguns dias no *TAC-SCM*, onde os agentes negociam com clientes e fornecedores, produzem e entregam computadores pessoais (COLLINS et al., 2006).

Cada agente deve tomar várias decisões sobre o processo de negociação e sobre o gerenciamento da cadeia de suprimentos. No TAC-SCM, um agente deve decidir durante o transcorrer de um dia:

1. Quais RFQs de clientes atender, se alguma. Para isso, o agente lança ofertas aos clientes.
2. Que componentes tentar obter. Para isso, o agente deve enviar aos fornecedores RFQs de componentes.
3. Que ofertas feitas pelos fornecedores aceitar, se alguma. Para isso, o agente deve enviar pedidos aos fornecedores.
4. Como alocar componentes disponíveis em estoque e a capacidade da fábrica para a produção de PCs.
5. Quais produtos enviar aos clientes. Para isso, os agentes devem enviar diariamente uma programação de entrega para suas respectivas fábricas.

Quanto à produção, os agentes contam com uma fábrica composta de uma célula de montagem capaz de produzir qualquer tipo de PC a partir dos tipos de componentes disponíveis. A célula de montagem do agente tem uma capacidade de produção diária fixada. A medida da capacidade é dada em ciclos de produção. Cada tipo de PC requer um número especificado de ciclos de produção para ser produzido. A produção e o envio de pedidos podem ser controlados diariamente por meio do envio da programação da produção e da entrega.

Os agentes recebem mensagens diárias de suas fábricas especificando a quantidade de PCs e componentes em estoque. O custo de manter PCs e componentes em estoque é definido no início do jogo, como descrito por Collins et al. (2006).

Os agentes possuem uma conta em um banco (simulado pelo servidor do TAC-SCM) e iniciam o jogo com saldo zero na conta. Dinheiro é adicionado na conta quando um cliente paga o envio de um produto. Dinheiro é deduzido da conta quando agentes recebem componentes de fornecedores. Um agente é penalizado quando falha a entrega de componentes e, com isso, tem dinheiro deduzido de sua conta. Agentes podem ter saldo negativo. Quando o saldo do agente é negativo, ele tem de pagar juros diariamente pelo saldo negativo. Quando o balanço do agente é positivo, é pago ao agente uma taxa de juros diariamente. A regras de atualização do saldo do agente são detalhadas por (COLLINS et al., 2006).

2.3 Produtos e Componentes

Os produtos fabricados são computadores pessoais (PCs). Cada modelo de PC é construído de quatro tipos de componentes: CPUs, placas-mãe, memórias e disco rígidos. A Tabela 2.1 mostra o catálogo de componentes para um jogo típico, com informações sobre cada componente, seus preços base e respectivos fornecedores. O preço base de cada componente é usado para o cálculo do preço que o componente será vendido e da variação do preço de reserva do componente. A relação de PCs e o catálogo de componentes são enviados a todos os agentes no início do jogo.

Tabela 2.1: Catálogo de Componentes e Fornecedores (COLLINS et al., 2006).

Componente	Preço Base	Fornecedor	Descrição
100	1000	Pintel	Pintel CPU, 2.0 Ghz
101	1500	Pintel	Pintel CPU, 5.0 Ghz
110	1000	IMD	IMD CPU, 2.0 Ghz
111	1500	IMD	IMD CPU, 5.0 Ghz
200	250	Basus, Macrostar	placa-mãe Pintel
210	250	Basus, Macrostar	placa-mãe IMD
300	100	MEC, Queenmax	Memory, 1 GB
301	200	MEC, Queemax	Memory, 2 GB
400	300	Watergate, Mintor	Hard Disk, 300 GB
401	400	Watergate, Mintor	Hard Disk, 500 GB

Há um total de 10 componentes diferentes, os quais podem ser combinados em 16 diferentes configurações de PC, todos apresentados na Tabela 2.2. CPUs e placas-mãe estão disponíveis em duas famílias de produtos diferentes: Pintel e IMD. Uma CPU Pintel apenas trabalha com placa-mãe Pintel enquanto uma CPU IMD pode ser usada apenas com uma placa-mãe IMD. CPUs estão disponíveis em duas velocidades, 2.0 e $5.0GHz$, memórias em tamanhos $1GB$ ou $2GB$, e discos em tamanhos $300GB$ e $500GB$.

Tabela 2.2: Relação de tipos de computadores pessoais (COLLINS et al., 2006)

SKU	Componentes	Cíclos de Produção	Segmento de Mercado
1	100, 200, 300, 400	4	Variação de baixo custo
2	100, 200, 300, 401	5	Variação de baixo custo
3	100, 200, 301, 400	5	Variação de médio custo
4	100, 200, 301, 401	6	Variação de médio custo
5	100, 200, 301, 400	5	Variação de médio custo
6	101, 200, 300, 401	6	Variação de alto custo
7	101, 200, 301, 400	6	Variação de alto custo
8	101, 200, 301, 401	7	Variação de alto custo
9	110, 210, 300, 400	4	Variação de baixo custo
10	110, 210, 300, 401	5	Variação de baixo custo
11	110, 210, 301, 400	5	Variação de baixo custo
12	110, 210, 301, 401	6	Variação de médio custo
13	111, 210, 300, 400	5	Variação de médio custo
14	111, 210, 300, 401	6	Variação de médio custo
15	111, 210, 301, 400	6	Variação de alto custo
16	111, 210, 301, 401	7	Variação de alto custo

Cada tipo de PC é identificado por um número inteiro denominado Unidade de Manutenção de Estoque (SKU - *Stock Keeping Unit*). A relação de PCs especifica, para cada tipo de PC, os componentes constituintes, o número de ciclos de montagem requerido e a qual segmento de mercado o PC pertence. Os dezesseis tipos de PC são classificados em três segmentos: um segmento de baixo custo (ou variação de baixo custo) com cinco produtos, um segmento de médio custo (ou variação de médio custo) com seis produtos e um segmento de alto custo (ou variação de alto custo), com cinco produtos.

2.4 Fornecedores

Um jogo padrão inclui oito fornecedores diferentes. Cada tipo de componente tem dois fornecedores, um para cada marca ou variação de componente. Assim, dois fornecedores fornecem todos os componentes de um determinado tipo. Por exemplo, dos fornecedores especializados em famílias de CPU, um (Pintel) é especializado em CPUs Pintel; o outro, o fornecedor IMD, em CPUs IMD. Placas-mãe são fornecidas por Basus e Macrostar, memória por MEC e Queenmax, e discos rígidos por Watergate e Mintor. A Tabela 2.1 mostra a lista de fornecedores e seus respectivos catálogos de produtos.

Cada fornecedor precisa realizar três tarefas a cada dia:

1. Gerenciar a capacidade de produção para satisfazer encomendas de pedidos.
2. Fazer ofertas aos agentes em resposta à RFQs de agentes e baseados na capacidade de produção futura projetada.

3. Enviar componentes para satisfazer encomendas de pedidos.

Os fornecedores possuem capacidade de produção limitada e foram modelados como entidades de maximização de lucro. Para decidirem para quais RFQs de agentes devem enviar ofertas, levam em conta o preço de oferta e a reputação dos agentes. Os agentes que mais rejeitarem ofertas de fornecedores terão uma reputação pior (representada por um valor real, quando menor este valor, pior a reputação) (COLLINS et al., 2006).

2.5 Clientes

A demanda de clientes é simulada por meio do envio de RFQs aos agentes, que devem realizar ofertas para satisfazer pedidos completos de clientes (data de vencimento e quantidade especificada). Se ou a data de vencimento ou o prazo de entrega não for satisfeitos, as respectivas ofertas são desconsideradas pelos clientes.

Cada RFQ (*Request For Quotes*) especifica um tipo de produto, uma quantidade q, data de vencimento, preço de reserva ρ, e uma penalidade x. Estes valores são gerados aleatoriamente dentro de intervalos pré-definidos ou gerados por meio de técnicas de caminho aleatório (COLLINS et al., 2006). No início de cada dia, os agentes recebem RFQs de clientes. Para realizar ofertas a uma RFQ, os agentes devem enviar uma oferta para o cliente contendo as seguintes informações: o preço, a quantidade ofertada e a data de vencimento da entrega. Os clientes apenas consideram ofertas que satisfazem três restrições: (1) a oferta promete entregar a quantidade completa especificada na RFQ, (2) a oferta promete entregar na data de vencimento especificada na RFQ e (3) o preço de oferta é menor ou igual ao preço de reserva especificado na RFQ. Todas as outras ofertas de agente são rejeitadas. Para cada RFQ, o cliente realiza os seguintes procedimentos: (1) coleta todas as ofertas que passam nos critérios acima e (2) selecionam as ofertas com menor preço. No caso de empate entre duas ofertas de agentes, a oferta "vencedora" será escolhida aleatoriamente entre as ofertas empatadas. O agente "vencedor" com uma oferta enviada para uma RFQ no dia d, recebe do cliente um pedido no dia $d+1$. Portanto, no dia seguinte à realização da oferta, cada agente recebe um retorno sobre o sucesso na venda pretendida.

Pedidos de clientes são finalizados quando agentes enviam produtos para os clientes. Devido ao fato de produtos enviados no dia d chegarem no dia $d+1$, $0 \leq d \leq 219$, o envio deve ser feito no dia anterior à data de vencimento para evitar penalidade. Agentes são informados diariamente de entregas atrasadas, penalidades sofridas e cancelamentos de pedidos devido à falha em entregar pedidos na data prometida.

2.6 O Cenário do TAC-SCM com Perturbações no Processo de Produção

Nos jogos e competições do TAC-SCM, a indicação de desempenho de uma fábrica é baseada somente em resultados financeiros, de custo de armazenamento por pedido e penalidades pagas. O custo total é balanceado com o lucro obtido em vendas. Em princípio, a fábrica com maior saldo bancário no final de um jogo é considerada vencedora. Essa medida de desempenho dá uma boa indicação de qual fábrica é mais eficiente do que outra em termos de custo e benefício. Contudo, não dá uma boa indicação sobre a robustez da fábrica quando perturbações locais no processo de produção são consideradas.

Para mostrar a robustez de uma fábrica, é necessária uma medida de desempenho que indique apenas a capacidade de uma fábrica de tratar perturbações de uma forma flexível. Os resultados financeiros de uma fábrica dão uma indicação do desempenho geral, mas a robustez em relação a perturbações locais no processo de produção é mais uma indicação do desempenho de uma fábrica. Portanto, o cenário do TAC-SCM foi modificado para levar em conta estas perturbações. Para isso, uma nova medida de desempenho foi considerada: a percentagem de pedidos enviados em tempo para consumidores finais, isto é, se a entrega de pedidos é realizada antes ou no prazo de entrega. Esta medida de desempenho foi considerada uma boa medida para a robustez de uma fábrica, pois ela indica a capacidade de uma fábrica entregar pedidos em tempo mesmo quando perturbações no processo de produção estão acontecendo.

O simulador do TAC-SCM não gera, por padrão, perturbações no processo de produção. Portanto, para testar a robustez de diferentes abordagens para as fábricas, um tipo de perturbação que acontece freqüentemente na prática tem sido adicionado no simulador. Na versão alterada do simulador TAC-SCM, cada componente que é entregue por um fornecedor para uma fábrica tem $n\%$ de chances de se tornar inutilizável na produção. A razão porque componentes se tornam inutilizáveis reflete na realidade um componente que pode está danificado, quebrado ou faltando na entrega. O simulador foi alterado de modo que o valor de n possa ser definido antes de cada competição. Neste trabalho, foram considerados três diferentes configurações para o valor de n:

- $n = 0$: este caso corresponde ao cenário tradicional do TAC-SCM, ou seja, sem perturbações no processo de produção.
- $n = 5$: neste caso, cada componente tem 5% de chance de ser avariado.
- $n = 10$: neste caso, cada componente tem 10% de chance de ser avariado.

2.7 Considerações Finais

Como mostrado, uma agência ou um agente no cenário TAC-SCM tem que realizar atividades interdependentes, como a obtenção de componentes que serão utilizados na montagem de PCs vendidos a consumidores finais. Portanto, um problema consiste na coordenação

de decisões tomadas para se resolver diferentes problemas. No caso de sistemas multi-agente, surge o problema de coordenar as decisões de diferentes agentes, que são responsáveis por diferentes atividades.

Outro problema que surge é o tratamento de perturbações no processo de produção, de modo manter altos níveis de serviço aos clientes, onde nível de serviço ao cliente significa a percentagem de pedidos enviados em tempo. O resultado do nível de serviço de uma fábrica pode ser afetado por perturbações no processo de produção. O tratamento de perturbações consiste em detectar alguma falha na etapa de produção e resolver esta falha de modo a não prejudicar o desempenho da fábrica. No TAC-SCM com perturbações, o problema de lidar com perturbações no processo de produção é dado em termos de avaria de componentes, significando que componentes previamente alocados para uso no processo de produção podem se tornar inutilizáveis.

Informalmente, dado um pedido P para q PCs do tipo j com a configuração (c_1, c_2, c_3, c_4), onde c_i é o código de um dos tipos de componentes utilizados na montagem do PC e j é a SKU do produto, são necessário q unidades de componentes de cada tipo presente na configuração do PC para suprir a produção de q PCs com a configuração dada. Assim, estes componentes precisam ser alocados para a produção de um PC do tipo j. Contudo, se uma quantidade $q' \leq q$ de componentes previamente alocados não puderem ser utilizados no momento da produção (porque foram avariados), pelo menos quatro ações podem ser tomadas para minimizar o impacto do problema de produção no nível de serviço da fábrica: (1) utilizar componentes previamente alocados para outros pedidos para suprir a produção de PCs do pedido P, (2) obter mais componentes de fornecedores, de modo a suprir a produção de produtos para o pedido P, (3) utilizar o estoque de segurança e (4) cancelar o pedido. Ações combinadas também podem ser realizadas, como utilizar o estoque de segurança e obter mais componentes de fornecedores caso o estoque de segurança não seja suficiente. O problema de decisão de um agente ou de uma agência consiste em realizar uma ou mais ações que maximize o nível de serviço, minimizando, dessa forma, o pagamento de penalidades aos clientes.

No próximo capítulo são apresentadas abordagens em duas categorias: abordagens projetadas para o ambiente TAC-SCM sem perturbações no processo de produção e abordagens projetadas para o ambiente TAC-SCM com perturbações no processo de produção.

3 Trabalhos relacionados

Vários trabalhos têm utilizado o conceito de SMA para lidar com o caráter distribuído de cadeias de suprimentos. Basicamente, os trabalhos abordados se baseiam em pelo menos duas abordagens gerais: (1) soluções altamente distribuídas, utilizando a decomposição do domínio do problema para efetivamente lidar com problemas mais fáceis de serem gerenciados e (2) abordagens focadas em problemas específicos, como previsão de preços e de demanda e otimização no uso de recursos. No último caso, soluções obtidas para cada um dos problemas separadamente são combinadas para a concepção de um planejamento para ações futuras. Ambas as abordagens podem ser concebidas com base em SMA, contudo a primeira abordagem faz uso direto das características de um SMA para lidar mais profundamente com a natureza distribuída de uma cadeia de suprimentos.

O objetivo deste capítulo é apresentar diferentes abordagens de sucesso para TAC-SCM e que serviram de embasamento para a concepção da abordagem concebida neste trabalho. As abordagens são agrupadas em duas seções: Na Seção 3.1 são apresentadas abordagens projetadas para o TAC-SCM sem perturbações no processo de fabricação. Na Seção 3.2 é apresentada a abordagem do sistema GRUNN, projetada para o cenário do TAC-SCM com perturbações no processo de fabricação. Na Seção 3.3 é apresentado um paralelo entre os dois grupos de abordagens, destacando-se a diferença entre os grupos.

3.1 Abordagens clássicas para o *Trading Agent Competion for Supply Chain*

As abordagens clássicas são as abordagens que não tiveram como ênfase a resolução de problemas de perturbações no processo de fabricação (ou simplesmente problemas locais de produção). Os principais agentes das abordagens clássicas são: o agente TacTex-06, o agente Mertacor, o agente Phant, o agente Southampton-SCM e o agente RedAgent.

3.1.1 O Agente TacTex-06

TacTex-06 (PARDOE; STONE, 2009) foi o vencedor da competição TAC-SCM em 2006. A estratégia desse agente consiste em fazer predições sobre o futuro da economia, como preços que serão cobrados por fornecedores de componentes e o nível da demanda dos consumidores. Com base nessas informações, o agente TacTex-06 planeja suas ações futuras com a

finalidade de maximizar seu lucro. O componente fundamental desse agente é sua habilidade para adaptar estas predições baseado no comportamento observado de outros agentes.

O TacTex-06 é baseado na ideia de que uma abordagem eficaz de agente requer o desenvolvimento de módulos fortemente acoplados para interação com fornecedores, consumidores e a fábrica. Foram elencadas cinco tarefas que um agente TAC-SCM deve realizar: (1) enviar pedidos de orçamentos de componentes para fornecedores, (2) decidir quais ofertas de fornecedores aceitarem, (3) realizar lances (ofertas) para pedidos de orçamento de PCs enviados por consumidores, (4) enviar a programação diária da produção para a fábrica e (5) entregar PCs finalizados.

Com base nas tarefas elencadas para agentes do TAC-SCM, o agente TacTex-06 foi projetado como mostra a Figura 3.1.

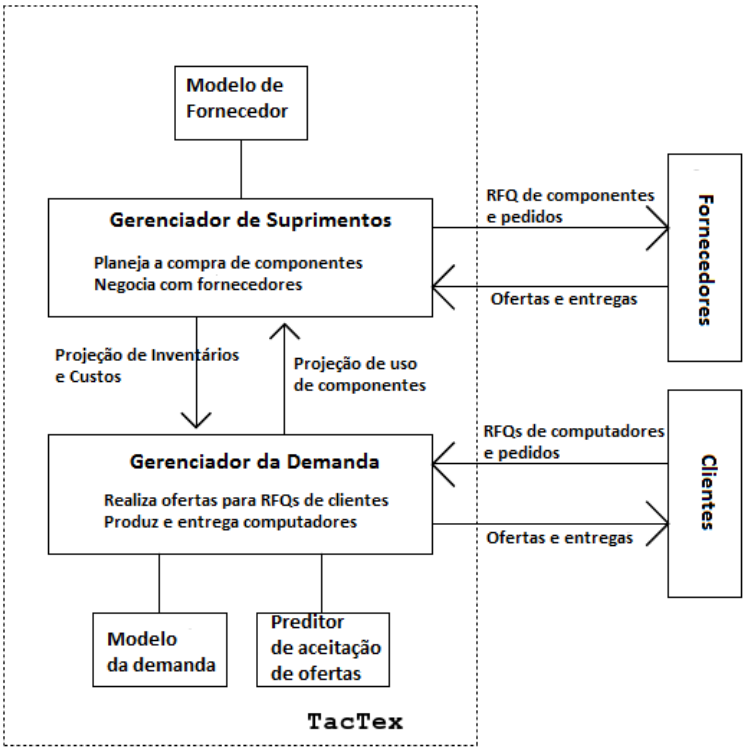


Figura 3.1: Uma visão geral dos componentes do TacTex-06 (PARDOE; STONE, 2009).

As duas primeiras tarefas foram atribuídas para o módulo gerenciador de suprimento e as últimas três tarefas ao módulo gerenciador de demanda. O gerenciador de suprimento controla todo planejamento relacionado ao inventário de componentes e compras e não requer nenhuma informação sobre a produção de computadores, exceto para uma projeção de uso futuro de componentes, o que é provido pelo gerenciador de demanda. As únicas informações requeridas pelo gerenciador de demanda é a projeção do inventário atual e a entrega futura de componentes, junto com o custo de substituição estimado para cada componente utilizado. Estas informações são fornecidas pelo gerente de suprimento. Estes módulos são auxiliados por três modelos de predição: um modelo preditivo do comportamento de fornecedores (modelo de fornecedor), um modelo de predição da demanda (modelo da demanda) e um previsor de aceitação de ofertas. Os módulos gerenciadores utilizam estes modelos para alocação eficiente de recursos e para

minimizar as incertezas do mercado. Portanto, trata-se de um planejamento centralizado das tarefas da cadeia de suprimento.

Observa-se que a solução incorporada no agente TacTex-06 foca em problemas específicos e as soluções para estes problemas são combinadas para se obter os resultados esperados.

3.1.2 O Agente Mertacor

O agente *Mertacor* (CHATZIDIMITRIOU et al., 2008) utiliza desde heurísticas simples a técnicas de mineração de dados para gerenciar uma fábrica em um ambiente dinâmico e competitivo que é o TAC-SCM. O agente Mertacor esteve entre os primeiros colocados nas competições de 2005 e 2006 e tem sido utilizado em muitas pesquisas para realização de experimentos controlados (CHATZIDIMITRIOU et al., 2008).

O agente Mertacor é composto de módulos que colaboram entre si para a resolução de quatro subproblemas do gerenciamento de cadeias de suprimento: aquisição de componentes, gerenciamento de inventário, produção e entrega de produtos e oferta para clientes (venda). A Figura 3.2 mostra a arquitetura do agente Mertacor.

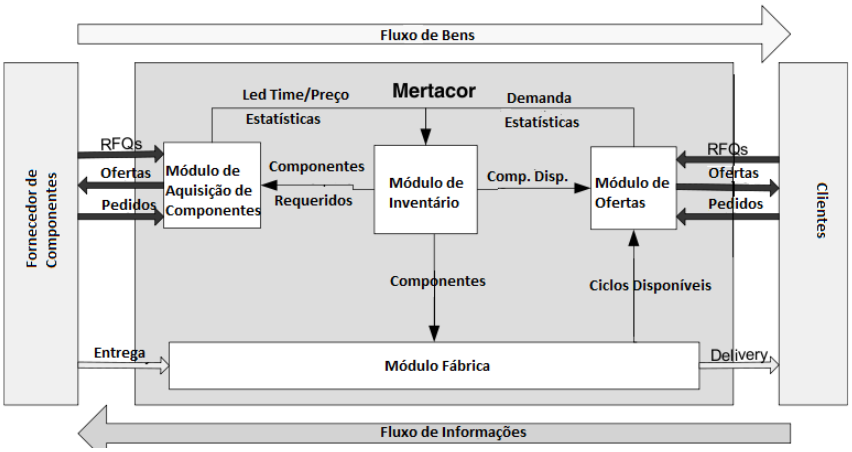


Figura 3.2: Arquitetura do agente Mertacor (CHATZIDIMITRIOU et al., 2008)

3.1.2.1 Módulos do Agente

A arquitetura do agente Mertacor é ilustrada na Figura 3.2, onde quatro módulos podem ser identificados:

- o módulo de inventário,
- o módulo de aquisição de componentes,
- o módulo fábrica e
- o módulo de ofertas.

3.1.2.2 Gerenciamento de Inventário

O agente Mertacor utiliza uma estratégia de gerenciamento de inventário de alto-nível *assembler-to-order* (ATO). Esta estratégia é uma solução hibrida de dois outros paradigmas: *make-to-stock* e *make-to-order*. Na estratégia *make-to-order*, a aquisição de componentes é tentada apenas após pedidos terem sido realizados, então é buscado a obtenção de componentes para suprir a produção de pedidos realizados. Já na estratégia *make-to-stock*, componentes são adquiridos e utilizados na produção de produtos que ficarão em estoque até que pedidos sejam realizados. E na estratégia *assembler-to-order*, componentes são adquiridos e mantidos em estoque, mas somente serão utilizados na produção após a realização de pedidos.

A estratégia ATO é adequada em ambientes onde o tempo de montagem ou fabricação é menor do que o tempo de reposição (CHENG et al., 2000), como o caso do jogo TAC-SCM. Adicionalmente, na estratégia ATO nenhum produto montado é armazenado no inventário e, desde que componentes são compartilhados por muitos produtos, o nível de componentes é baixo devido à agregação da demanda, a troca de componentes internamente é aplicável (a fim de evitar atraso na entrega de produtos e penalidades) e é observado baixo custo de armazenamento. A demanda de componentes é calculada considerando a demanda por segmentos de mercado, como mostrado por Chatzidimitriou et al. (2008).

3.1.2.3 Aquisição de Componentes

O desempenho do agente Mertacor é altamente dependente de dois fatores: (a) ter um inventário composto de componentes baratos e (b) satisfazer o nível de inventário, desde que o atraso na entrega de componentes gera penalidades, onerando o lucro final do agente. Portanto, é adotada uma estratégia que combina a aquisição de componentes baratos e a aquisição de componentes para manter o nível de estoque. Assim, nos primeiros dois dias do jogo, componentes são obtidos para satisfazer um inventário inicial de modo que o agente possa iniciar a produção imediatamente e iniciar a oferta para pedidos nos primeiros dias do jogo. Os componentes obtidos são previsivelmente caros, contudo, sua utilidade é alta, desde que no início do jogo a competição não é forte, permitindo a obtenção de pedidos a preços mais elevados. No dia 1 outro pacote de requisição de componentes é enviado aos fornecedores, permitindo obter uma grande quantidade de produtos baratos para os dias que se seguem.

No dia 2, o agente Mertacor troca para uma estratégia de obtenção de componentes projetada para satisfazer os objetivos anteriores. Ele explora todas as possíveis requisições por componentes (RFQs) e por fornecedor a cada dia, de modo a maximizar o conhecimento dos preços de componentes vendidos pelos fornecedores. Baseado nessa estratégia de consulta, o agente seleciona as ofertas mais promissoras em termos de baixo custo e quantidade oferecida. Em geral, as RFQs enviadas pelo agente Mertacor podem está em uma de três categorias:

- Estratégia normal de obtenção: ajuda a satisfazer os níveis de inventário, reordenando os níveis de inventário e permitindo oferta para pedidos de clientes no futuro próximo. Quantidades para estas RFQs são calculadas com base no nível de reordenamento do módulo de inventário.

- Estratégia de aquisição crítica: um estado especial em que os agentes procuram componentes a fim de satisfazerem pedidos críticos de clientes. Pedidos críticos são pedidos não entregues e cuja prazo de entrega está próximo (dois dias ou menos).
- Estratégia de aquisição antecipada: um esforço para obter componentes a baixos custo. Estas RFQs indicam quantidades que, se satisfeitas, poderiam fazer o inventário exceder os níveis de reordenamento, assim não haveria nenhuma necessidade de obter componentes por meio da estratégia normal em algum momento do jogo.

O módulo de aquisição ainda mantém o histórico de pedidos atrasados pelos fornecedores e mantém uma média e um desvio padrão calculados em tempo real destes atrasos para cada fornecedor, a fim de informar o módulo de inventário possíveis atrasos na entrega de pedidos pelos fornecedores.

3.1.2.4 Módulo Fábrica

Este módulo é responsável por: (a) gerar programação da produção e entrega, (b) ajustar os níveis de inventário, e (c) ajustar os ciclos de fábrica disponíveis. Este módulo é responsável por integrar os módulos de venda, obtenção de componentes e inventário. Ele funciona por meio de uma estratégia de simulação dos dias futuros de produção, tentando prever a disponibilidade futura de ciclos de fábrica, informando ao módulo de vendas o quanto ele pode vender de modo a manter baixos níveis de atraso na entrega.

3.1.2.5 O Módulo de Oferta

O módulo de oferta do agente Mertacor é baseado em modelos de regressão para a obtenção do preço de oferta ótimo, baseado no lucro previsto. A hipótese inicial é que toda oferta obterá sucesso na obtenção de pedido. O mecanismo de oferta é baseado em técnicas de aprendizagem de máquina, onde os dados de treinamento são obtidos de logs de jogos anteriores. Certas propriedades das RFQs de clientes e o estado do ambiente TAC-SCM são usados para prever o orçamento para cada RFQ de cliente e a probabilidade do cliente aceitar o orçamento. Além disso, é utilizado um mecanismo de segurança em caso de mudanças inesperadas no ambiente, onde os modelos de predição extraídos se tornam inválidos. O mecanismo desenvolvido mostrou ser acurado, obedecendo às regras de variação de mercado e, dessa forma, realístico e aplicável em ambientes de gerenciamento de cadeias de suprimento reais.

3.1.3 O Agente Phant

O agente Phant (STAN; STAN; FLOREA, 2006) foi o segundo colocado da competição TAC-SCM de 2006. As principais tarefas executadas diariamente pelo agente Phant são: i) negociar venda de computadores, ii) negociar a compra de componentes e iii) controlar a programação da fábrica. Estas tarefas são atribuidas a diferentes módulos: módulo de computador, módulo de componentes e módulo fábrica, como mostrado na Figura 3.3.

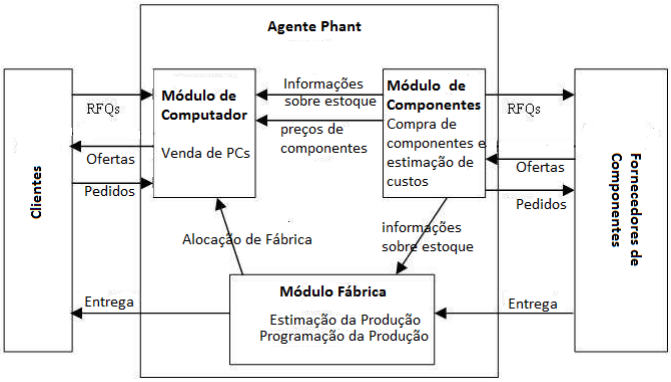


Figura 3.3: Arquitetura do agente Phant (STAN; STAN; FLOREA, 2006)

3.1.3.1 O Módulo de Componentes

O módulo de Componentes é responsável por garantir que haja sempre componentes em estoque, a fim de manter a fábrica em máxima produção, enquanto ajuda a manter preços de componentes comprados o mais baixo possível. Há também a necessidade de manter uma quantidade mínima de componentes em estoque, a fim de reduzir os custos de armazenamento, lidando com o *trade-off* entre adaptabilidade do tamanho do estoque e o preço de compra. Devido às restrições de tempo do TAC-SCM, há uma quantidade restrita de estratégias que podem ser utilizadas. No caso do agente Phant, são enviadas múltiplas RFQs para consultar os fornecedores sobre o preço de venda de componentes. Isso pode ser feito apenas configurando a quantidade de componentes requeridos em cada RFQ para zero e dividindo um pedido grande em vários pedidos menores, desde que pedidos menores tem maiores chances de serem atendidos pelo fornecedor. O processo completo é descrito por Stan, Stan e Florea (2006).

Outra função do módulo de componentes é prover estimativas dos preços de componentes, que serão utilizadas para calcular o lucro estimado da venda de computadores ou determinar o preço de reserva para comprar componentes.

3.1.3.2 O Módulo de Computador

O módulo de computador é responsável por selecionar a cada dia o que deve ser vendido e a que preço. A estratégia de definição de preço de venda é baseada no relatório diário de preços disponível no mercado. Este relatório contém o menor e o maior preço de pedido para cada tipo de computador até o momento. O menor preço é desprezado na suposição de que o maior preço é o melhor preço que um agente conseguiu vender um computador nos dias anteriores. Seja p_h o maior preço de pedido de clientes dos últimos três dias e f uma variável real inicialmente configurada com o valor 1.0. O preço de oferta é configurado como $p_h * f$, onde antes de ser multiplicado, f é ajustado de acordo com os níveis de produção e uso da fábrica.

3.1.3.3 O Módulo Fábrica

Este módulo decide quais pedidos podem ser enviados para entrega até o prazo de entrega e estima a utilização da fábrica necessária para produzir todos os pedidos ainda não enviados para entrega. É utiliza uma regra geral que diz que todos pedidos devem ser enviados para entrega dentro de 3 a 4 dias. Além disso, se no dia atual não há mais pedidos a serem produzidos e a fábrica ainda tem uma capacidade restante, computadores de todos os tipos são produzidos uniformemente, enquanto o estoque para cada tipo de computador estiver abaixo de um limiar. Segundo Stan, Stan e Florea (2006), a escolha do valor de limiar é crítica para um bom desempenho do agente. Valores muito baixos somente permitirão à fábrica ser utilizada 100% se sempre há uma grande quantidade de pedidos, o que não permite uma política de oferta agressiva o bastante, enquanto valores muito altos para o limiar levarão a fábrica a produzir muitos computadores quando os preços estiverem altos e o lucro baixo. Foi escolhido um valor de limiar fixo para o agente *Phant*, embora os autores ressaltem que é possível adaptar este limiar de acordo com determinadas condições de mercado, como incrementar o valor quando o preço de componentes não estiver alto, mas o lucro é baixo devido a uma demanda muito baixa.

3.1.4 Southampton-SCM

Southampton-SCM (HE et al., 2006) foi o vencedor da competição TAC-SCM 2005. Este agente realiza a compra de componentes utilizando estratégia de aquisição mista (combinando planejamento de longo e curto prazo) e define seus preços de acordo com as condições de mercado dominantes e seu próprio nível de estoque. *Southampton-SCM* é composto de três módulos (como mostra a Figura 3.4). O agente de clientes recebe RFQs enviadas por clientes e decide para quais RFQs deve realizar ofertas. Além disso, comunica-se com o agente fábrica para obter o nível de inventário atualizado e para enviar os pedidos de PCs relevantes feitos por clientes.

O agente de componentes decide quais RFQs de componentes enviar para quais fornecedores. O agente fábrica recebe os suprimentos enviados por fornecedores e decide baseado nos recursos disponíveis (quantidade de componentes e ciclos de fábrica) quais pedidos feitos por clientes deveriam ser produzidos. Além disso, determina uma programação para entrega dos PCs finalizados para clientes.

O componente fundamental da arquitetura do *Southampton-SCM* é o agente de clientes. Este componente decide para quais pedidos de orçamento de clientes deve realizar ofertas e qual o valor de cada oferta. A definição do preço é importante para o sucesso do agente, dado que vender um produto a um preço muito baixo poderá acarretar prejuízo; por outro lado, oferecer um preço muito alto poderá fazer com que o agente não consiga vender o suficiente para obter um bom resultado. Para definir um preço adequado, o agente utiliza um conjunto de regras nebulosas de acordo com nível do inventário, a demanda do mercado e o tempo decorrido no jogo.

O agente de clientes recebe de potenciais clientes RFQs para PCs especificando algumas preferências: quantidade de um tipo particular de PC para ser entregue em um dia

especificado e um preço de reserva (preço máximo que o cliente está disposto a pagar pelo produto). Quando seleciona para quais RFQs responder com ofertas, o agente ordena estas ofertas de acordo o potencial lucro que elas podem trazer e de acordo com inventário mantido pelo agente. Dessa forma, suponha que uma RFQ seja representada por uma tupla $(i, q, p_{res}, c_{penalty}, d_{due})$(HE et al., 2006), onde $i \in \{1, ..., 16\}$ é o tipo de PC que o cliente deseja, $q > 0$ é a quantidade, p_{res} é o preço de reserva, $c_{penalty} > 0$ é a multa paga se os computadores não são entregues em tempo, e d_{due} é a data de entrega desejada. A cada dia, o agente recebe um pacote de RFQs e as classifica em ordem decrescente da margem de lucro do tipo de PC requisitado (como descrito por He et al. (2006)):

$$p_{res} - p_{base} - \frac{c_{penalty}}{q},$$

onde p_{base} é o custo de comprar componentes (soma do preço de compra de cada componente). Assim, é esperado que o agente primeiro servirá clientes com alta margem de lucro e baixa penalidade. Isso é devido ao fato de que um p_{res} mais alto possibilita um lucro maior do que p_{res} mais baixos.

Para cada RFQ na lista, o agente primeiro verifica se ela pode ser suprida a partir do seu estoque de PCs finalizados. Se sim, o inventário de PCs correspondente é decrementado. Caso contrário, o agente verifica se ele contém componentes suficientes em seu inventário atual e em sua fila de entrega esperada (componentes pedidos, mas ainda não recebidos), e também se tem suficiente capacidade de produção restando para fabricar os PCs requisitados. Se sim, o agente decrementa a quantidade de componentes disponível em inventário.

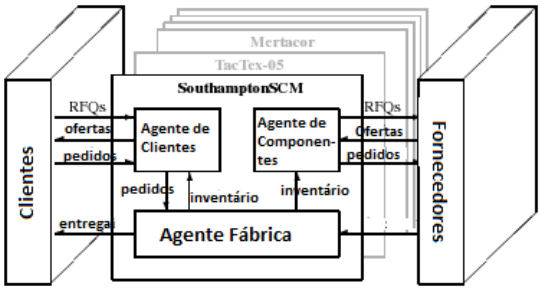


Figura 3.4: Arquitetura do agente Southampton-SCM (HE et al., 2006).

O agente de componentes é responsável por garantir que sempre haverá componentes suficientes para lidar com mudanças de demanda de consumidores. Ao fazer isso, ele lida com um desafio que é comum para todas as cadeias de suprimentos diante de mudanças na demanda de consumidores. Isto é, deve adquirir componente a um baixo custo, enquanto simultaneamente mantém um inventário de componente mínimo a fim de reduzir o custo diário de armazenamento e a possibilidade de se livrar de estoque redundante caso demanda de consumidores se modifiquem. Para isso, são utilizados modelos de previsão da demanda a longo e curto prazo.

O principal desafio do agente fábrica é planejar a produção. A estratégia utilizada

envolve fabricar PCs de acordo com pedidos de clientes (*build-to-order*) e satisfazer pedidos com menor prazo de entrega.

3.1.5 RedAgent

Um primeiro projeto de sucesso de SMA para TAC-SCM foi o *RedAgent*, vencedor da primeira competição do TAC-SCM (KELLER; DUGUAY; PRECUP, 2004). *RedAgent* é baseado em um projeto multi-agente em que agentes de heurísticas simples gerenciam tarefas como satisfazer pedidos de clientes ou obter recursos específicos. A idéia chave é utilizar mercados internos como o principal mecanismo de decisão, a fim de determinar em quais produtos investir e como alocar recursos existentes. Os mercados internos garantem a coordenação dos agentes de forma individual, mas ao mesmo tempo proveem uma estimativa de preço para bens, uma característica fundamental deste domínio.

Dessa forma, *RedAgent* tem uma arquitetura altamente distribuída, em que agentes simples baseados em heurísticas são incumbidos de lidar com aspectos individuais do jogo, como obtenção de componentes e produção de pedidos. Estes agentes se comunicam através de um mecanismo de mercado com a finalidade de determinar, coletivamente, quais componentes comprar, quais tipos de PCs produzir, como alocar os componentes disponíveis e ciclos de produção e o que vender para os clientes.

Os criadores do *RedAgent* destacaram algumas vantagens da abordagem baseada em mercados internos em relação às abordagens de otimização tradicionais: (1) oferece a oportunidade de um projeto modular, em que diferentes agentes, responsáveis por diferentes recursos, podem se comunicar no mercado; (2) mercados oferecem um mecanismo de procura mais eficiente que abordagens tradicionais, pois nas abordagens tradicionais cursos de ação alternativos não são considerados explicitamente; e (3) em adição à alocação de recursos, o mercado realiza a estimativa de preços anexados aos recursos.

Na abordagem do *RedAgent* é observado o indício de um padrão de resolução de problemas que consiste no conceito de agentes como um conjunto de responsabilidades. A responsabilidade pela resolução de um problema complexo foi tirada de um ou mais planejadores centrais e atribuída entidades mais simples, contudo, capazes de resolverem subproblemas mais fáceis de serem gerenciados. Essa percepção de agentes como conjuntos de responsabilidades foi tomada emprestada das perspectivas de Fowler (do paradigma de programação orientada a objetos), onde no nível conceitual objetos são definidos como conjuntos de responsabilidades (FOWLER, 2000). Dado que isso venha a levantar muitos questionamentos, trata-se apenas de uma analogia que revela a essência da abordagem utilizada (não explicita) no projeto de *RedAgent*: delegação de responsabilidades.

3.2 Abordagens para o *Trading Agent Competition for Supply Chain* com Perturbações no Processo de Fabricação

A primeira abordagem a tratar diretamente com perturbações no processo de produção foi o sistema GRUNN (MEYER; WORTMANN, 2009). A abordagem utilizada foi baseada no conceito de produtos inteligentes para a concepção de uma solução distribuída em que entidades auto-gerenciáveis são responsáveis por pedidos de clientes. A seguir, o sistema GRUNN é apresentado.

3.2.1 GRUNN

Trabalhos baseados em diferentes inspirações têm chegado a soluções parecidas, mas não iguais. Meyer e Wortmann (2009) apresentaram um SMA baseado em uma abordagem descentralizada de controle e planejamento inspirada no conceito de produtos inteligentes (MEYER; FRäMLING; HOLMSTRöM, 2009). O resultado foi uma agência denominada *GRUNN.* que é caracterizada por ser uma solução altamente distribuída, assim como a abordagem dos criadores do *RedAgent*. Contudo, a abordagem dos autores do *GRUNN* é focada no conceito de produtos inteligentes. Segundo Mcfarlane et al. (2003), um produto inteligente é definido como uma representação física e baseada em informação de um produto. A Figura 3.5 mostra um exemplo de tal produto. Nesta figura, a jarra de molho de espaguete é o produto físico, a representação baseada em informação do produto está armazenada em uma base de dados e a inteligência é provida pelo agente de tomada de decisão.

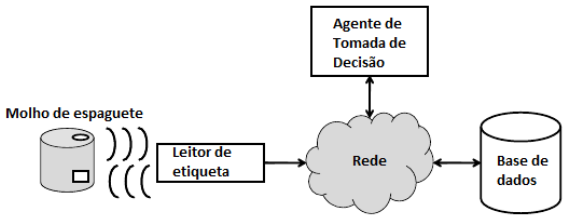


Figura 3.5: Um produto inteligente (WONG et al., 2002).

Meyer e Wortmann (2009) destacam que a idéia fundamental por trás dos produtos inteligentes é que produtos individuais na cadeia de suprimento controlam a si próprios. Isso é equivalente a dizer que produtos inteligentes são responsáveis por si próprios. Assim, mais uma analogia com a definição de objeto no nível conceitual (de acordo com as perspectivas de Fowler): objetos são responsáveis por si próprios. Aqui há uma sutil diferença entre as abordagens do SMA do *GRUNN* e do SMA do *RedAgent*: o propósito da abordagem distribuída. O propósito dos mercados internos do *RedAgent* por meio de leilões era prover um protocolo relativamente simples para agentes trocarem informações sobre suas necessidades e variações. Enquanto o propósito dos produtos inteligentes é focado diretamente sobre a idéia de entidades responsáveis por si próprias.

No caso do projeto do *GRUNN*, cada pedido de cliente é tomado como um produto inteligente, responsável por si próprio, ou seja, responsável por obter os recursos necessários para se concretizar (chegar ao consumidor final, o requerente do pedido). O resultado dessa concepção de SMA baseado em produtos inteligentes é mostrado na Figura 3.6, um SMA composto de quatro agentes planejadores. Dependendo de suas funções, cada um destes agentes pode delegar suas responsabilidades para outros agentes. Neste caso, o agente de compras (*PurchasePlannerAgent*) e o agente de vendas (*SalesPlannerAgent*) delegam suas responsabilidades para agentes do tipo *ComponentTypeAgent* e *ProductTypeAgent*, respectivamente. Os agentes de cada tipo de produto (representados pela classe *ProdutTypeAgent*) ao fecharem pedidos com clientes geram objetos do tipo *ProductAgent*. Cada agente do tipo *ProductAgent* fica encarregado do pedido para qual foi incumbido, ou seja, objetos do tipo *ProductAgent* são representações no nível de implementação do elemento de tomada de decisão de um produto inteligente.

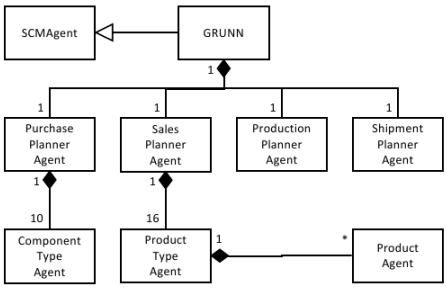


Figura 3.6: Diagrama de classes da estrutura interna do agente *GRUNN (MEYER; WORTMANN, 2009).*

O SMA *GRUNN* foi inspirado em produtos inteligentes para tratar os problemas nos quais a função centralizada de planejamento apresenta dificuldades em controlar, que são pequenas perturbações de transporte e produção, eventos comuns na prática. Portanto, os autores alteraram o ambiente do TAC-SCM para suportar a análise de mais uma métrica: a robustez encarnada na quantidade de pedidos de cliente entregues antes ou até a data de entrega devida. Para realmente analisar os efeitos das perturbações, o TAC-SCM foi modificado de forma que todo componente entregue por um fornecedor tenha *n porcento* de chance de não ser utilizável. Os resultados obtidos mostraram que, para a medida de desempenho adotada, a abordagem inspirada nos produtos inteligentes realmente é robusta. No entanto, ainda inferior à abordagem de planejamento centralizado no caso geral.

3.3 Considerações Finais

Os dois conjuntos de abordagens apresentados diferem pelo foco da pesquisa de seus projetistas. Enquanto, o projeto do agente TacTex06 esteve focado na questão de como lidar com aprendizagem no domínio do TAC-SCM, o projeto do sistema GRUNN esteve focado na questão dos problemas locais de produção. Outra diferença está na natureza da função de planejamento da produção. Enquanto o sistema GRUNN é distribuído no que diz respeito ao processo

de produção e gerenciamento de pedidos, outros agentes focam na resolução distribuída de problemas, exceto pelo agente RedAgent que provê uma solução altamente distribuída, mas não lida diretamente com problemas ou perturbações no processo de produção (ou, como algumas vezes é denominado, problemas locais de produção).

De fato, o trabalho de (MEYER; WORTMANN, 2009) foi o primeiro a levantar a questão, no contexto do TAC-SCM, sobre se uma solução distribuída não lida de forma mais eficaz com perturbações no processo de produção. Outras abordagens lidam com estes problemas de forma clássica, como manter um estoque de segurança. Apesar do trabalho de (MEYER; WORTMANN, 2009) ter apresentado uma arquitetura altamente distribuída no que diz respeito ao gerenciamento de pedidos, os elementos desta arquitetura não foram explorados do ponto de vista de outras métricas que não o nível de serviço aos clientes.

No próximo capítulo, são apresentados os fundamentos que nortearam a concepção da abordagem multi-agente proposta.

4 Fundamentação Teórica

Para se atingir o objetivo deste trabalho, a Inteligência Artificial desempenha um papel crucial. Para lidar com problemas complexos como o gerenciamento de cadeias de suprimento, a tecnologia multi-agente tem se mostrado bastante eficaz, pois prover um paradigma que facilita a decomposição de problemas complexos em problemas mais fáceis de serem gerenciados por entidades autônomas que colaboram entre si para atingirem um objetivo comum. Portanto, um agente é um sistema computacional que está situado em um ambiente e é capaz de ações autônomas neste ambiente com o propósito de atingir seus objetivos (WEISS, 2000).

O objetivo deste capítulo é apresentar os fundamentos que nortearam a concepção da abordagem multi-agente proposta. Para isso, Na Seção 4.1 é apresentado o conceito de agentes racionais como uma abordagem de inteligência artificial. Na Seção 4.2, são apresentados programas do agente reativo e do agente reativo com estado interno. Na Seção 4.3, são apresentados os fundamentos de sistemas multi-agente. Na Seção 4.4, é apresentado o conceito de aprendizagem de máquina. Na Seção 4.5, são apresentados os fundamentos de redes neurais de múltiplas camadas e o algoritmo *backpropagation*. Na Seção 4.6, são apresentados os fundamentos do algoritmo KNN. Na Seção 4.7, são apresentados os fundamentos dos métodos de combinação de classificadores e o algoritmo *AdaBoost.M1*. Finalmente, na Seção 4.8, são apresentadas as considerações finais sobre os algoritmos e técnicas utilizadas.

4.1 Agentes Racionais

Russell e Norvig (2002) defendem o conceito de agente como central para a Inteligência Artificial, tida como o desenvolvimento de sistemas que agem de forma racional. Nessa abordagem, um agente racional é aquele que age para alcançar o melhor resultado em um ambiente ou, quando há incerteza, o melhor resultado esperado. O padrão de racionalidade é encarnado por uma medida de desempenho, ou seja, um o critério objetivo para se medir o sucesso do agente no ambiente.

Segundo Russell e Norvig (2002), o estudo da Inteligência Artificial como o projeto de agentes racionais tem pelo menos duas vantagens em relação às abordagens classificas, como a abordagem de leis de pensamento: em primeiro lugar, o projeto de agentes racionais é mais geral que a abordagem de “leis de pensamento”, porque a inferência correta é apenas uma dentre vários mecanismos possíveis para se alcançar a racionalidade; em segundo lugar, é mais acessível ao desenvolvimento científico do que as estratégias baseadas no comportamento ou

no pensamento humano, porque o padrão de racionalidade é definido com clareza e é completamente geral. No caso, o padrão de racionalidade é encarnado por uma medida de desempenho. A medida de desempenho é o critério para se medir o sucesso de um agente em um ambiente.

Uma recomendação para a concepção de agentes racionais é a especificação de um ambiente de tarefa. Ambientes de tarefa são os problemas para os quais os agentes racionais são as soluções. Especificar um ambiente de tarefa consiste em definir: a medida de desempenho, o ambiente, os atuadores e os sensores do agente. Portanto, ao projetar um agente, a primeira tarefa é especificar o ambiente de tarefa de forma tão completa quanto possível (RUSSELL; NORVIG, 2002).

4.2 Programas de Agentes

Apesar de não haver um consenso sobre a definição de agente, pode-se dizer que eles interagem com o ambiente por meio de sensores e atuadores. Além disso, agentes implementam uma **função de agente**, que mapeia qualquer seqüência de percepções em ações. A função é uma descrição abstrata desse mapeamento, com seu domínio (conjunto de percepções) e contradomínio (conjunto de ações) bem definidos. Contudo, a implementação dessa função pode assumir diversas formas, não havendo uma implementação mais adequada do que todas as outras em todos os contextos. Uma implementação da função de agente é o que se denomina **programa de agente**. A função é uma descrição matemática abstrata, enquanto um programa de agente é uma implementação concreta, executando sobre uma arquitetura de agente (RUSSELL; NORVIG, 2002). Esta arquitetura consiste dos dispositivos computacionais com seus sensores e atuadores sobre os quais o programa de agente executa. Segundo Russell e Norvig (2002), o papel da Inteligência Artificial é projetar programas de agente que implementam a função de agente para mapeamento de percepções em ações.

Um agente de software executando em leilões virtuais, por exemplo, teria como percepções cotas estabelecidas pelos compradores que especificam restrições nas compras. Nesse ambiente, os agentes deveriam realizar ofertas de acordo com restrições estabelecidas pelos compradores, como preço de reserva ou preço máximo que estão dispostos a pagar por determinado produto. Nesse caso, as percepções seriam as cotas e as informações retornadas pelo ambiente sobre a situação dos leilões, como se o agente venceu ou não. As ações do agente seriam ofertas aos clientes. O papel da inteligência artificial é conceber programas de agente, mapeando o histórico de percepções em ações.

Vários autores têm discutido a natureza de agentes e suas respectivas propriedades. Russell e Norvig (2002) apresentam diversos modelos de agentes que são adequados a diversos tipos de ambientes de tarefa: reativos simples, reativos baseados em modelo, baseados em objetivos, baseados em utilidade e agente com aprendizagem. Estes agentes podem ser distribuídos em dois grupos: agentes reativos e agentes deliberativos. Os agentes reativos não modelam o mundo para determinar suas ações, enquanto que os agentes deliberativos contêm um modelo do mundo, possivelmente incluindo ele próprio. O modelo é "pré-concebido", mas seu estado é alterado pelo agente em resposta às novas informações sobre o ambiente, percebidas pelos

sensores do agente. O agente estima quais ações serão necessárias para alcançar um determinado objetivo através da interpretação deste modelo, e então executa ações que levarão à sua realização.

4.2.1 Agentes Reativos

Os agentes puramente reativos devem responder continuamente às mudanças que ocorrem em seu ambiente. Dependendo do ambiente de tarefa, o projeto deste tipo de agente pode ser muito simples. Por exemplo, para o caso em que o ambiente é observável e determinístico a arquitetura abstrata do agente padrão pode ser refinada dando origem aos agentes reativos (WOOLDRIDGE, 2000). A Figura 4.1 ilustra algumas das informações e os subsistemas (módulos) processadores de informação propostos em um primeiro refinamento. O módulo processador *Ver* identifica os sensores componentes da arquitetura física do agente. O módulo processador *Ação* identifica a função tomada de decisão intrínseca ao programa agente que realizada o mapeamento percepção-ação.

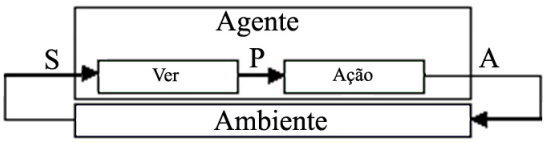


Figura 4.1: Agente Padrão (WEISS, 2000)

Mais especificamente, o funcionamento da arquitetura do agente padrão envolve quatro passos, ou seja: (1) por meio dos sensores o agente recebe informações do ambiente que são seqüências de estados definidos em $S = s_1, ..., s_n$; (2) um subsistema de percepção, $Ver : S \rightarrow P$, processa cada estado de uma seqüência S^* e mapeia em uma seqüências de m percepções, $P = p_1, ..., p_m$, que são representações de aspectos dos estados de S que estão acessíveis ao agente para a tomada de decisão; (3) um subsistema de tomada de decisão Ação: $P^* \rightarrow A$ processa as seqüências perceptivas P^*, resultantes de S^*, e seleciona uma ação em $A = a_1, ..., a_l$; (4) por meio de atuadores o agente envia a ação selecionada para o ambiente.

Agentes puramente reativos não armazenam seqüências perceptivas P^*, mas tomam decisões estritamente baseados nas percepções correntes, P, sem levar em consideração qualquer tipo de informação histórica. Isto implica em uma especialização do subsistema de tomada de decisão do agente, ou seja, *Ação*: $P \rightarrow A$. É mais simples conceber agentes reativos para ambientes estáticos e observáveis. De maneira equivalente, Russell e Norvig (2002) incorporaram um conjunto de regras condição-ação no subsistema de tomada de decisão do agente reativo.

As regras condição-ação são formalizações de associações comuns observadas entre certas percepções e ações possíveis para o agente. Assim, O agente tem pré-computado um conjunto de ações para várias situações previstas, o que simplifica a implementação do mapeamento percepção-ação.

4.2.2 Agentes Reativos com Estado Interno

Um refinamento do agente reativo simples é o agente reativo baseado em modelo (conhecimento a respeito do estado do ambiente). A arquitetura de um agente com estado interno é adequada às situações de projeto em que o ambiente é parcialmente observável e sua geografia é desconhecida. Para lidar com a falta de acessibilidade o agente mantém em uma estrutura de dados interna, uma descrição do estado do ambiente, que é atualizada à medida que os episódios vão acontecendo. Esta nova arquitetura consiste em um refinamento do subsistema de tomada de decisão do agente puramente reativo (WEISS, 2000) e é apresentada na Figura 4.2.

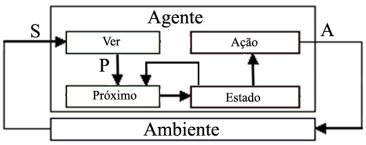


Figura 4.2: Arquitetura do Agente Reativo com Estado Interno.

Comparando com o agente reativo simples apresentado na Figura 4.1, o agente baseado em modelo possui um subsistema de processamento de informação a mais: *Próximo*. Isto implica em realizar uma nova decomposição no subsistema de tomada de decisão do agente padrão. Em termos de funcionamento esta decomposição equivale à decomposição do passo (3) do ciclo de operação do agente puramente reativo descrito na seção anterior. Mais especificamente, depois do subsistema de percepção *Ver* mapear um estado do ambiente em S em uma percepção em P, (passo 3.1) um subsistema de atualização de estado interno, *Próximo*: $IxP \rightarrow I$, mapeia a percepção em P e o estado interno corrente em $I = i_1, ..., i_m$, em um novo estado interno; que, por sua vez, (passo 3.2) é processado pelo subsistema de tomada de decisão, *Ação*: $I \rightarrow A$, para selecionar uma ação possível em A.

Vale ressaltar, conforme proposto para o agente puramente reativo, as regras condição-ação podem ser incorporadas no módulo de tomada de decisão, desta vez levando em consideração o estado interno mantido a respeito de seu ambiente, em vez da informação que chega diretamente dos sensores do agente.

4.3 Sistemas Multi-Agente

Em muitos casos, vários agentes são necessários para a resolução de problemas complexos. Como consequência disso, surgiu o conceito de sistemas multi-agente. Sistemas Multi-Agente (SMA) podem ser vistos como um novo paradigma para compreender e construir sistemas distribuídos, onde é assumido que os componentes computacionais são autônomos: capazes de controlar seu próprio comportamento para alcançar seus objetivos (WOOLDRIDGE, 2009).

Dessa forma, Sistemas Multi-Agente são sistemas computacionais em que dois ou mais agentes interagem ou trabalham em conjunto de forma a desempenhar determinadas tarefas e ou satisfazer um conjunto de objetivos (LESSER, 1999). Um dos pontos principais para permitir a

construção de sociedades de agentes consiste em conseguir gerir as interações e as dependências das atividades dos diferentes agentes no contexto do sistema multi-agente. Ainda segundo Lesser (1999), a investigação em Sistemas Multi-Agente está focada no desenvolvimento de princípios e modelos computacionais para construir, descrever, implementar e analisar as formas de interação e coordenação de agentes em sociedades de reduzida ou elevada dimensão.

Outro problema que surge quando se considera dois ou mais agentes interagindo é como conceber a comunicação entre estes agentes. A comunicação entre agentes é um dos principais problemas a serem tratados na área de sistemas multi-agente. Foi analisado anteriormente que um agente, para ser considerado como tal, possui capacidades de percepção, processamento e atuação em um dado ambiente. Além disso, um agente deliberativo (ou mesmo híbrido) possui uma representação interna de seu ambiente, conhecimento e capacidade de raciocinar baseado em seu conhecimento, de forma a decidir em cada instante qual a melhor ação a executar. Da mesma forma, é assumido aqui que um agente possui a capacidade de se comunicar com outros agentes em seu ambiente.

De forma a incluir a capacidade de comunicação em um agente, é usual incluir um módulo de comunicações na sua arquitetura que se subdivide nos componentes de percepção (recepção de mensagens) e de ação (envio de mensagens). Este módulo de comunicações está diretamente ligado ao módulo central do agente (módulo deliberativo) permitindo desta forma ao módulo deliberativo do agente ter acesso às mensagens recebidas e definir quais as mensagens a enviar.

A comunicação tem dois objetivos: compartilhamento do conhecimento, informação, crenças ou planos com outros agentes e coordenação de atividades entre agentes. No entanto, a realização de comunicação que permita atingir estas duas metas, requer a definição de uma linguagem comum ou partilhada, caracterizada por: sintaxe, semântica, vocabulário, conjunto de regras de ação e fórmulas de interpretação dos símbolos utilizados na comunicação; e modelo do domínio do discurso, ou o significado que um conjunto de símbolos assume quando interpretado em um determinado contexto de conversação.

Dado que os agentes em um sistema multi-agente se comunicam, uma forma comum de ordenar a comunicação entre agentes é por meio da definição de um protocolo. Os protocolos podem ser definidos a vários níveis. Os níveis inferiores definem o método de interligação dos agentes. Os níveis intermediários definem a sintaxe da informação transmitida. Nos níveis superiores encontram-se a semântica da informação.

4.3.1 Protocolos de Comunicação e Atos de Fala

Genericamente podemos definir que um protocolo contém a seguinte estrutura de dados (HUHNS; STEPHENS, 1999):

- Emissor;
- Receptor(es);

- Linguagem utilizada;
- Ações que o receptor deve executar.

A Teoria dos Atos da Fala (AUSTIN, 1962; SEARLE, 1969) é utilizada como modelo de comunicação aplicada à comunicação dos agentes. A idéia principal da Teoria dos Atos de Fala é entender a linguagem como uma forma de ação ("todo dizer é um fazer"). Assim, diversos tipos de ações humanas poderiam se realizar através da linguagem: os "atos de fala", (ou *Speech acts*, em inglês). Austin (1962) distinguiu três aspectos essenciais dos atos de fala:

- A Locução: ato físico da fala;
- A Elocução: sentido atribuído à locução;
- A *Perlocução*: efeito da ação resultante da locução.

John Searle estendeu o trabalho de Austin (SEARLE, 1969) e identificou diversas propriedades necessárias para o sucesso de atos de discurso. E entre estas condições destacam-se as condições normais de transmissão (input/output), condições preparatórias e condições de sinceridade. Searle identificou ainda um conjunto de classes para os atos de discurso, incluindo (SEARLE, 1969): diretivas (pedidos), promessas, expressivas, representativas (informações) e declarativas. A Teoria do Ato de Fala utiliza o termo *performativa* para identificar a elocução.

As *performativas* podem ser verbos como: prometer, convencer, insistir, dizer, pedir, oferecer, requerer, etc. A utilização das performativas destina-se exclusivamente a definir inequivocamente a elocução desejada. Assim, definem o sentido com que deve ser interpretado o conteúdo da mensagem. O resultado reflete a ação esperada do outro interveniente face ao primeiro ato de fala.

Vários formatos e padrões foram definidos para comunicação entre agentes. Estas linguagens definem essencialmente a estrutura exterior da mensagem e definem um conjunto de performativas que facilitam, entre outras coisas, a negociação entre agentes.

4.3.2 Ontologias

Para que agentes que compõem um sistema multi-agente possam interagir é necessário uma plataforma de comunicação, uma linguagem de comunicação e que os agentes possuam um vocabulário comum bem definido. No entanto, é comum que diferentes agentes possuam diferentes terminologias para o mesmo significado ou idênticas terminologias para diferentes significados. Este problema pode ser resolvido se eles compartilharem uma ontologia comum. Weiss (2000) define uma ontologia como uma especificação de objetos, conceitos e relacionamentos de uma determinada área de interesse.

No contexto de um Sistema Multi-Agente, uma ontologia é uma representação formal de conceitos, características e relacionamentos em um dado domínio específico, permitindo o

entendimento comum da área de conhecimento entre pessoas e agentes de software. Existem muitas linguagens de representação de conhecimento utilizadas para modelagem de ontologias, como XML (*eXtensible Markup Language* e UML (Unified Modelling Language) (FOWLER, 2000).

4.3.3 Regulamentação Social

Regulamentação social é um tipo de regulamentação de mercado. A palavra regulamentação usualmente conota um tipo de guia administrativo de mercado (SHERMAN, 2001). Esta regulamentação externa é diferenciada da regulamentação por competição: aumento de preço quando suprimentos estão escassos e redução de preços quando suprimentos estão abundantes. Regulamentação externa suplementa competição e aparece em três formas principais: leis para controle de monopólio, regulamentação da indústria e regulamentação social como proteção ambiental. A definição de regulamentação social como proteção ambiental provê uma inspiração para um mecanismo de controle e planejamento baseado em concessões a partir de entidades regulamentadoras. A idéia surgiu a partir dos exemplos apresentados em Sherman (2001), como segue.

Problemas ambientais, como poluição e congestionamento, são difíceis de resolver. direitos de poluir e direitos de utilizar avenidas podem ser criados para ajudar a resolver estes problemas e eles são exemplos de idéias econômicas na prática. Eles não resolvem todos os problemas, mas mostram como a criação de novos direitos pode regular efeitos externos por motivar sua redução de forma eficiente. Por exemplo, criar direitos de poluir o ar pode, paradoxalmente, ajudar a controlar a poluição. A solução de "direitos de poluir" para controlar a poluição permite que estes direitos sejam comprados ou vendidos. Definir direito de poluir é difícil porque requer definição de métricas e coerção entre diferentes tipos de valores, mas uma vez que estas dificuldades são superadas, a quantidade total de poluição, o direito total de poluir, pode ser especificado. Isso significa que o nível de poluição permitido pode ser especificado, como pode agora ser feito para limitar emissões de dióxido sulfúrico nos Estados Unidos para controlar as chuvas ácidas.

Uma vez que direitos de poluir são definidos e uma dada forma de suprimento destes direitos é estabelecida, um preço de mercado pode ser determinado. Desta forma, quem conseguir reduzir a poluição eficientemente, isto é, para menos que o valor de um direito de poluir, reduzirá poluição e venderá seus direitos de poluir para outros. Estes que estão diante de um aumento de poluição podem comprar os direitos de poluir e usá-los para emitir poluição. Desta forma, em um equilíbrio de mercado, o preço dos direitos de poluir reflete o custo marginal de controlar poluição a um nível que a disponibilidade de direitos de poluir permitir.

Pode-se pensar, assim como foi pensado para o controle da poluição, em direitos de utilizar avenidas. No caso geral, o que se deseja por meio do mecanismo de concessão de direitos é controlar os efeitos externos gerados pelas atividades econômicas.

Em sistemas multi-agente, a regulamentação social pode ser utilizada como mecanismo de coordenação entre diferentes agentes que competem por recursos internos do sistema.

Assim, diferentes agentes seriam controlados por um módulo ou outro agente cujo papel seria a regulamentação no uso de recursos ou no acesso às informações internas do sistema.

4.4 Aprendizagem de Máquina

Segundo Mitchell (1997), o campo de aprendizagem de máquina se ocupa em responder a questão de como construir programas de computador que melhoram com a experiência. Assim, Mitchell (1997) propõe uma definição ampla para o que é aprendizado: é dito que um programa de computador aprende da experiência E em relação a uma classe de tarefas T e medida de desempenho P, se sua medida de desempenho na classe de tarefas T, medida por P, melhora com a experiência E. Assim, para se ter um problema de aprendizagem bem definido, deve-se identificar três características: a classe de tarefas, a medida de desempenho a ser melhorada e a fonte de experiência.

Aprendizagem toma muitas formas, dependendo da natureza do sistema de desempenho, do componente de generalização e do tipo de retro-alimentação disponível. O tipo de retro-alimentação é freqüentemente o fator mais importante em se determinar a natureza do problema de aprendizagem que um agente deve tratar. O campo de aprendizagem de máquina geralmente distingue dois casos: aprendizagem supervisionada e aprendizagem não-supervisionada.

O problema de aprendizagem supervisionada consiste em aprender uma função a partir de exemplos de suas entradas e saídas. O problema de aprendizagem não-supervisionada envolve aprender padrões na entrada quando nenhuma saída específica é fornecida. O problema de aprendizagem por reforço é o mais geral das três categorias. Um agente de aprendizagem por reforço deve aprender a partir do reforço, ou seja, uma informação indicando a qualidade das ações executadas pelo agente. Essas informações são obtidas indiretamente do ambiente. Aprendizagem por reforço tipicamente inclui o subproblema de aprender como o ambiente trabalha. A escolha de qual tipo de aprendizado utilizar depende da forma como a experiência de treinamento (exemplos) está disponível (RUSSELL; NORVIG, 2002).

A representação da informação aprendida também desempenha um papel muito importante em determinar como o algoritmo de aprendizagem deve trabalhar. Alguns componentes de um agente podem ser representados por algum esquema de representação. Exemplos de representação são: polinômios lineares ponderados para funções utilidade para avaliar a ação de programas em jogos; sentenças em lógica proposicional e em lógica de primeira-ordem para os componentes em um agente lógico; e descrições probabilísticas tais quais redes bayesianas para os componentes de inferência de um agente baseado em teoria-da-decisão. Neste trabalho foram utilizados métodos baseados em redes neurais, algoritmo KNN (*K-Nearest Neighboard*) e combinação de classificadores utilizando o algorítimo *AdaBoost*. Na próxima seção serão apresentados os principais conceitos sobre redes neurais e sobre outros algoritmos de aprendizagem utilizados neste trabalho.

4.5 Redes Neurais

Redes neurais artificiais (RNA) fornecem um método geral e prático para a aprendizagem de funções reais e discretas a partir de exemplos. As redes neurais artificiais, ou simplesmente redes neurais, foram inspiradas em teorias biológicas sobre o funcionamento de redes neurais em seres vivos. Segundo Haykin (1999), uma rede neural é um processador maciçamente paralelamente distribuído, composto de unidades de processamento simples denominadas neurônios. A força da conexão entre estas unidades é representada por meio de pesos sinápticos e é nestes pesos que o conhecimento adquirido é armazenado. A plasticidade de uma rede neural é conseguida por meio de alterações no peso de suas conexões, em um processo de aprendizagem.

Vários modelos e técnicas de redes neurais foram desenvolvidos de forma a possibilitar modelos com aprendizagem. Algoritmos como o de retro-propagação do erro (*backpropagation*) utilizam gradiente descendente para ajustar os parâmetros da rede a um conjunto de treinamento disposto na forma de pares de entrada-saída. A partir de exemplos conhecidos do que se deseja aprender, pode-se aproximar uma função real para, por exemplo, realizar previsões sobre valores futuros.

Arquiteturas neurais são tipicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior. A seguir, serão abordadas as redes de múltiplas camadas como uma evolução de um primeiro modelo denominado *Perceptron*. A próxima seção aborda o modelo de redes neurais utilizado nesse trabalho, as redes de múltiplas camadas e o algoritmo de treinamento de retro-propagação do erro ou *backpropagation*. Mais informações sobre outros modelos de redes neurais podem ser encontradas em Braga, Carvalho e Ludemir (2000), Haykin (1999).

4.5.1 Perceptron de Múltiplas Camadas (MLP)

As redes de múltiplas camadas com realimentação adiante evoluíram a partir do modelo *Perceptron*. Em redes com múltiplas camadas, os neurônios da camada de entrada recebem estímulos do ambiente. Os neurônios da camada escondida processam os dados obtidos pela camada de entrada e os enviam para a camada de saída. Todas as camadas podem realizar processamento. Com a adição da camada escondida, problemas não-linearmente separáveis, como o XOR, podem ser resolvidos, pois nesse caso a rede pode representar funções mais complexas do que as lineares. Contudo, surge o problema de atribuição de crédito: determinar qual a influência que a camada escondida tem sobre o resultado da rede. O período entre 1958 e 1982 foi dedicado a resolver esse problema. A solução foi o algoritmo conhecido como *Backpropagation* (algoritmo de retro-propagação do erro). A seguir, as redes MLP são apresentadas quanto aos seus aspectos estruturais, funcionais e de aprendizagem.

4.5.1.1 Aspectos Estruturais das Redes de Múltiplas Camadas

A Figura 4.3 ilustra uma rede neural MLP com três camadas. Estas redes são do tipo *feedforward* ou de alimentação adiante. Isso significa que a informação flui em uma única direção ao longo da rede: capturada pela camada de entrada, processada pelas camadas escondidas até que o resultado seja enviado para o ambiente por meio da camada de saída. Cada neurônio consiste em uma função que processa sua entrada e produz uma saída correspondente. A saída de um neurônio ou servirá de entrada para um próximo neurônio ou será enviado para o ambiente (no caso, quando for um neurônio de saída). No caso das redes de múltiplas camadas com alimentação adiante, os neurônios são funções contínuas e, portanto, suas entradas são valores contínuos vindos de outros neurônios em camadas anteriores ou do ambiente. Na Figura 4.3, a rede é composta de quatro neurônios na camada de entrada, assim as percepções são representadas por vetores com quatro componentes reais.

A camada de saída é composta por dois neurônios que produzem a reposta da rede ao estímulo do ambiente (o vetor de entrada). Todas as camadas são completamente conectadas; cada neurônio da camada de entrada está conectado a todos os neurônios da camada intermediária e neurônios desta camada estão conectados a todos os neurônios da camada de saída.

Há várias funções que podem ser utilizadas para representação dos neurônios ou função de ativação e a escolha desta função é um dos principais aspectos do projeto de uma rede neural. Neste trabalho foi escolhida a função sigmóide para os neurônios. A função sigmóide, $y = 1/(1+e^{-x})$ é usada em um rede neural porque é contínua e é necessário continuidade para se utilizar o gradiente descendente. Por exemplo, o valor da função sigmóide, y, aproxima-se de 1 quando x assume valores positivos cada vez maiores e 0 quando x assume valores negativos cada vez menores. Além disso, y é igual a 0.5 quando $x = 0$. A derivada em relação à saída y relativo à entrada x da função sigmóide é simples de ser calculada e é igual a: $y(1-y)$, ou seja, é expressa como uma simples função da saída.

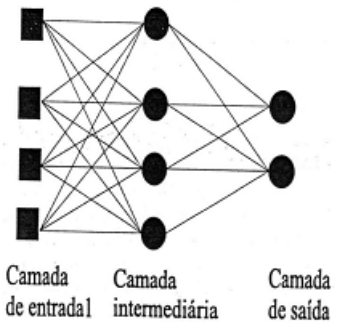


Figura 4.3: Uma rede neural com múltiplas camadas.

4.5.1.2 Treinamento das Redes de Múltiplas Camadas

O processo de aprendizagem de redes neurais por meio do algoritmo *backpropagation* ocorre em duas etapas: *forward* e *backward*.

Inicialmente, exemplos são apresentados à rede. Cada exemplo é representado por um par de vetores de entrada e saída deseja ou $\langle \mathbf{x}, \mathbf{y} \rangle$. O erro é calculado como a diferença entre o vetor de saída desejado e o vetor de saída produzido pela rede.

O desempenho da rede precisa ser mensurado para se determinar o quão bem ela está aproximando a função objetivo nos exemplos de treinamento. No caso em que a saída da rede é um escalar, é utilizada a função desempenho $P = \frac{-1}{2}(y - y')^2$, onde y' é a saída esperada para algum exemplo especificado e y é a saída produzida pela rede para aquele exemplo. De acordo com o erro, os pesos na rede são alterados de acordo com a regra delta generalizada, que é baseado na regra delta proposta por Widrow e Hoff (1960). O algoritmo é apresentado em duas fases. A fase *forward* envolve os passos mostrados na Figura 4.4.

1. Apresentação da entrada para a primeira camada da rede, a camada C^0.
2. Para cada camada C^i a partir da camada de entrada
   (a) Após os nodos da camada C^i ($i > 0$) calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada C^{i+1}.

Figura 4.4: Fase *forward* do algoritmo *backpropagation*

A fase *backward* envolve os passos mostrados na Figura 4.5.

1. A partir da última camada, até chegar na camada de entrada
   (a) Os nodos da camada atual ajustam seus pesos de modo a reduzir seus erros.
   (b) O erro de um neurônio das camadas intermediárias é calculado utilizando os erros dos neurônios da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre eles.

Figura 4.5: Fase *backard* do algoritmo *backpropagation*

O algoritmo *backpropagation* faz uso dessas duas etapas, como mostra a Figura 4.6. Os ajustes dos pesos em cada camada são realizados por meio do gradiente. Nesse caso, uma função de custo é dada em função do erro e o objetivo do treinamento é minimizar essa função de custo. O custo é definido pela soma dos erros quadráticos médios e representada pela função:

$$E = \frac{1}{2} \sum_{p} \sum_{i=1}^{k} (d_i^p - y_i^p)^2 \tag{4.1}$$

onde E é a medida de erro total, p é o número de padrões, k é o número de unidades de saída, d_i é a i-ésima saída desejada e y_i é a i-ésima saída gerada pela rede. A Equação 4.1 dá o erro total cometido pela rede, ou a quantidade em que, para todos os exemplos p de um dado conjunto, as saídas geradas pela rede diferem das saídas desejadas.

1. Inicializar pesos e parâmetros
2. Repetir até o erro ser mínimo ou até a realização de um dado número de ciclos:
   (a) Para cada padrão de treinamento **x**
      i. Definir a saída da rede através da fase *forward*.
      ii. Comparar as saídas produzidas com as saídas desejadas.
      iii. Atualizar pesos dos nodos através da fase backward.

Figura 4.6: Algoritmo *Backpropagation*

4.6 Aprendizado Baseado em Instâncias

Métodos de aprendizagem baseados em instância simplesmente armazenam os exemplos de treinamento. A generalização ocorre quando uma nova instância deve ser classificada, comparando-a com as instâncias conhecidas nos exemplos previamente armazenados. Métodos baseados em instâncias são classificados como preguiçosos (*lazy*) porque eles adiam o processamento até que uma nova instância precise ser classificada. Há vários métodos de aprendizado baseado em instâncias. Aqui será apresentado o método dos k vizinhos mais próximos (KNN - K Nearest Neighbor).

4.6.1 K Vizinhos mais Próximos

Este é o método de aprendizado baseado em instâncias mais básico. É assumido que todas as instâncias correspondem a pontos em um espaço n-dimensional $\mathbb{R}^n$. Os vizinhos mais próximos de uma instância são definidos em termos da distância euclidiana padrão. Para esclarecer como essa distância é calculada, considere que cada instância seja representada por um vetor $\langle x_k(1), x_k(2), ..., x_k(n)\rangle$, onde x_k se refere à instância considerada e $x_k(i)$, o valor da instância para o atributo i. Seja duas instâncias x_p e x_q, a distância entre estas duas instâncias, representada por $d(x_p, x_q)$, é calculada como: $d(x_p, x_q) = \sum_{j=1}^{n} (x_p(j) - x_q(j))^2$

O método KNN aproxima a função objetivo f ao comparar uma nova instância à todas as instâncias no conjunto de treinamento atual. A função objetivo f mapeia uma nova instância para uma classe em um conjunto $C = \{c_1, c_2, ..., c_m\}$ e é definida como: $f : R^n \rightarrow C$. Para o método KNN, considere o conjunto de treinamento $D = x_1, x_2, ..., x_t$. Os valores $f(x_i)$, para $1 \leq i \leq t$, são conhecidos. Assim, cada exemplo é representado pelo par $(x_i, f(x_i))$. O algoritmo KNN é apresentado na Figura 4.7.

Treinamento

- Para cada exemplo $(x_i, f(x_i))$ adicione o exemplo para a lista de exemplos (exemplos)

Classificação:

- Dada uma instancia de consulta x_q a ser classificada:
  - Seja $x_1, x_2, ..., x_k$ as k instancias na lista de exemplos que são mais próximas a x_q
    - retorne $f^{'} := \underset{c \in C}{argmax} \sum_i^k \delta(c, f(x_i))$, onde $\delta(a,b) = 1$ se $a = b$ e $\delta(a,b) = 0$, caso contrário.

Figura 4.7: Algoritmo KNN.

Um refinamento do algoritmo KNN para resolver essas situações consiste em ponderar a influência dos vizinhos mais próximos pela distância dos mesmos à instância a ser classificada. Para o refinamento do algoritmo KNN, a última linha do algoritmo mostrado na Figura 4.7 é alterada pela Equação 4.2.

$$f^{'} := \underset{c \in C}{argmax} \sum_{i}^{k} w_i \delta(c, f(x_i)) \tag{4.2}$$

onde $w_i = \frac{1}{d(x_p, x_q)^2}$ e $d(x_p, x_q)$ é a distância entre as instâncias x_p e x_q.

4.7 Métodos de Combinação (Ensembles Methods)

Métodos de combinação são algoritmos de aprendizagem que constroem um conjunto de classificadores e classificam novas instâncias por meio do voto ponderado de suas predições (DIETTERICH, 2000). O método de combinação original é o de Média Bayesiana, contudo, métodos mais recentes são incluídos, como *Bagging* e *Boosting*. Neste trabalho foi utilizado o algoritmo *AdaBoost*, portanto esta seção enfatiza principalmente este algoritmo de *boosting*.

Em um problema de aprendizagem supervisionada, exemplos de treinamento são dados para um programa de aprendizagem, como mostrado anteriormente. Estes exemplos estão na forma $((x_1, f(x_1)), (x_2, f(x_2))..., (x_n, f(x_n)))$, onde f é a função objetivo e x_i, $1 \leq i \leq n$, é tipicamente um vetor na forma $\langle x_i(1), x_i(2), ..., x_i(m) \rangle$ cujos componentes são valores discretos ou contínuos. O valor de $f(x_i)$ é geralmente um valor do conjunto $\{1, 2, 3, ..., K\}$. Se a função f não é conhecida, algoritmos de aprendizagem de máquina buscam por uma função $f^{'}$ compatível com os exemplos de treinamento, supondo que essa função seja uma boa aproximação para f. A compatibilidade significa que $f^{'}$ classifica corretamente todas as instâncias no conjunto de treinamento. Considerando que pode haver mais de uma função $f^{'}$ compatível com os exemplos de treinamento, cada função possível é uma hipótese para f.

Dado um conjunto C de exemplos de treinamento, um algoritmo de aprendizagem produz um classificador. O classificador é uma hipótese sobre a função f. Dado uma nova instância x_i, ela prediz o valor $f(x_i)$ correspondente. Os classificadores podem ser denotados por $h_1, h_2, ..., h_L$.

Um método de combinação de classificadores é um conjunto de classificadores cujas

decisões individuais são combinadas de alguma forma para classificar novos exemplos. Segundo Dietterich (2000), métodos de classificação são freqüentemente mais acurados do que classificadores individuais. Uma condição necessária e suficiente para uma combinação de classificadores ser mais acurado do que qualquer um de seus membros tomados individualmente é se os classificadores são acurados e diversos. Um classificador acurado tem sua taxa de erro menor do que um classificador aleatório sobre uma nova instância x_i. Dois classificadores são diversos se eles produzem diferentes erros sobre novas instâncias. Para perceber porque acurácia e diversidade são necessárias, imagine que há uma combinação de classificadores $\{h_1, h_2, h_3\}$ e considere uma nova instância x. Se os três classificadores são idênticos (isto é, não diversos), então quando $h_1(x)$ está errado, $h_2(x)$ e h_3 também estão errados. Contudo, se os erros produzidos pelos classificadores não são relacionados, então quando $h_1(x)$ está errado, $h_2(x)$ e $h_3(x)$ podem está corretos, assim o voto da maioria corretamente classifica x. De fato, se a taxa de erro de L hipóteses h_l são todas iguais a $p < 0.5$ e se os erros são independentes, então a probabilidade que o voto da maioria esteja errado será a área sob a distribuição binomial onde mais do que $L/2$ hipóteses estão erradas. A Figura 4.8 mostra isso para uma combinação simulada de 21 hipóteses, cada uma tendo uma taxa de erro 0.3.

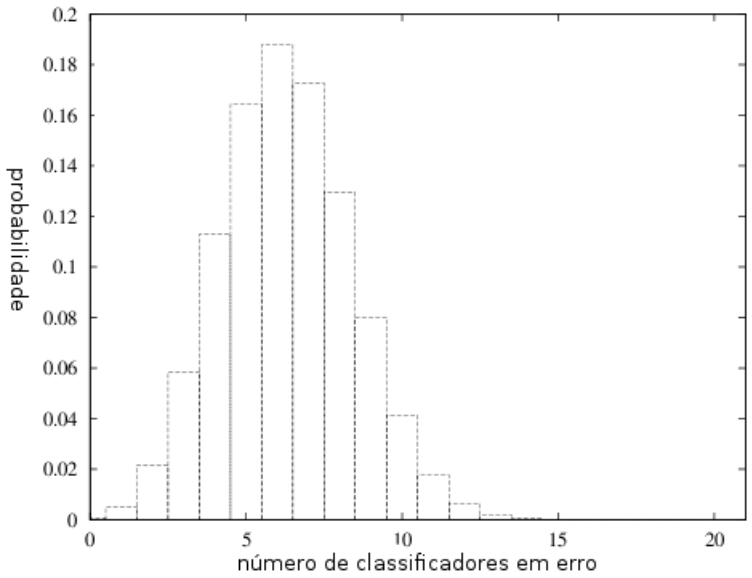


Figura 4.8: A probabilidade de exatamente l (de 21) hipóteses produzirem erro, assumindo que cada hipótese tem uma taxa de erro de 0.3 e produz seus erros independentemente de outras hipóteses (DIETTERICH, 2000).

A área sob a curva para 11 ou mais hipóteses erradas é igual a 0.026, que é muito menor do que a taxa de erro de uma hipótese tomada individualmente.

Existem vários métodos para técnicas combinação de classificadores. Neste trabalho foi utilizado o algoritmo AdaBoost (FREUND; SCHAPIRE, 1996). A seguir será apresentado os fundamentos dos métodos de *boosting* e, mais especificamente, o algoritmo *AdaBoost*.

4.7.1 Métodos de Boosting e o Algoritmo AdaBoost

Boosting é um método geral para melhorar o desempenho de qualquer algoritmo de aprendizagem. Em teoria, *boosting* pode ser usado para significativamente reduzir o erro de qualquer algoritmo de aprendizagem "fraco"que consistentemente geram classificadores que precisam apenas ser um pouco melhores que um classificador aleatório. Métodos de *boosting* trabalham por repetidamente executar um algoritmo de aprendizagem fraco em várias distribuições sobre os dados de treinamento, combinando os classificadores produzidos por este algoritmos de aprendizagem fracos em um único classificador composto. Nesta seção, será descrito o algoritmo AdaBoostM1, utilizado pela sua praticidade e facilidade de implementação (FREUND; SCHAPIRE, 1996). Um algoritmo de *boosting* recebe como entrada um conjunto de treinamento com m exemplos de treinamento, $S = \langle (x_1, f(x_1)), (x_2, f(x_2)), ..., (x_m, f(x_m)) \rangle$, onde x_i é uma instância obtida de algum espaço X e representada por um vetor de valores de atributos. Ainda, $f(x_i) \in Y$ é um rótulo de classe associado com x_i. Para simplificar a notação, será utilizado $y_i = f(x_i)$. Aqui será considerado que o conjunto de possíveis rótulos Y é de cardinalidade finita k.

Adicionalmente, um algoritmo de *boosting* tem acesso a outro algoritmo de aprendizagem de máquina não especificado, chamado de classificador fraco, denotado por **ClassificadorFraco** (embora, algumas variações possam conter algoritmos de aprendizagem forte, como redes neurais de múltiplas camadas). O algoritmo de *boosting* chama repetidamente **ClassificadorFraco** em uma série de etapas. Em uma etapa t, o algoritmo de *boosting* fornece para **ClassificadorFraco** uma distribuição D_t dos dados de treinamento em S. Em resposta, **ClassificadorFraco** encontra uma hipótese $h_t : X \to Y$ que deveria classificar corretamente uma parte do conjunto de treinamento que tem uma grande probabilidade relativo a D_t. Isto é, o objetivo do classificador fraco é encontrar uma hipótese h_t que minimiza o erro de treinamento $\varepsilon_t = Pr_{i \sim D_t}(h_t(x_i)) \neq y_i$. Observe que este erro é medido em relação à distribuição D_t que foi fornecida pelo classificador fraco. Este processo continua ao ongo de t etapas e, por último, o método de *boosting* combina as hipóteses fracas $h_1, ..., h_t$ em uma única hipótese final h_{fin}.

Duas questões permanecem em aberto na descrição geral de um algoritmo de *boosting* dada no parágrafo anterior: (1) de que forma D_t é calculada em cada rodada e (2) como h_{fin} é calculada. Diferentes esquemas de *boosting* respondem estas duas questões de diferentes formas. Aqui, o foco será sobre o algoritmo *AdaBoost.M1*, utilizado neste trabalho na avaliação de preços de oferta em leilões fechados. O algoritmo é mostrado na Figura 4.9. No algoritmo *AdaBoost.M1*, a distribuição inicial D_1 é uniforme sobre S, dessa forma, $D_1(i) = 1/m$ para todo i. Para calcular a distribuição D_{t+1} de D_t e da última hipótese fraca h_t, o peso do exemplo i é multiplicado por algum número $\beta_t \in [0,1)$ se h_t classifica x_i corretamente, caso contrário, o peso não é alterado. Os pesos são normalizados ao serem divididos pela constante de normalização Z_t. Efetivamente, exemplos "fáceis"que são corretamente classificados por muitas das hipóteses fracas, recebem um peso menor, enquanto exemplos difíceis que tendem

Algoritmo AdaBoost.M1

Entrada:

sequência de m exemplos $\langle (x_1, y_1), ..., (x_m, y_m) \rangle$ com rótulos $y_i \in Y = \{1, ..., k\}$
algoritmo de aprendizagem fraco **WeakLearn**
um inteiro T especificando o número de iterações

Início:

$D_l(i) = 1/M$ para todo i.
Para $t = 1, 2, ..., T$:

1. Chamar **WeakLearn**, fornecendo a ele a distribuição D_t.
2. Obter uma hipótese $h_t : X \to Y$.
3. Calcular o erro de $h_t : \varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} (D_t(i))$.
   Se $\varepsilon_t > 1/2$, então faça $T = t - 1$ e termina o laço.
4. Faça $\beta_t = \varepsilon/(1 - \varepsilon_t)$.
5. Atualize a distribuição D_t:
   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t \text{ se } h_t(x_i) = y_i \\ 1 \text{ caso contrário} \end{cases}$$
   onde Z_t é uma constante de normalização (escolhida assim que D_{t+1} se tornar uma distribuição).

Saída: a hipótese final
$h_{fin}(x) = \arg max_{y \in Y} \sum_{t:h_t(x)=y} log(\frac{1}{\beta_t})$

Figura 4.9: Algorítmo *AdaBoost.M1*

O número β_t é calculado como mostrado na Figura 4.9 como uma função de ε_t. A hipótese final h_{fin} é calculada como o voto ponderado (isto é, o limiar linear ponderado) das hipóteses fracas. Isto é, para uma dada instância x, h_{fin} produz um rótulo y que maximiza a soma dos pesos das hipóteses fracas predizendo este rótulo. O peso da hipótese h_t é definido como $log(1/\beta_t)$, assim, os pesos maiores são dados a hipóteses com menores erros.

Uma importante propriedade sobre o algoritmo *AdaBoost.M1* é que se hipóteses fracas tem consistentemente erro apenas um pouco melhor do que $1/2$, então o erro de treinamento da hipótese final cai rapidamente para próximo de zero. Para problemas de classificação binária (o caso em que o algoritmo é utilizado neste trabalho), hipóteses fracas precisam ser apenas um pouco melhor do que um classificador aleatório (DIETTERICH, 2000).

4.8 Considerações Finais

Os conceitos abordados foram desde agentes simples até sistemas multi-agente e técnicas de aprendizagem. No que diz respeito aos sistemas multi-agente, o problema de coordenação entre os múltiplos agentes que compartilham recursos poder ser resolvido por meio da metáfora da regulamentação social. Técnicas de aprendizagem podem ser utilizadas nos módulos de decisão dos agentes para aprenderem, com base em logs de jogos passados e no *feedback* do ambiente, a realizar ações melhores.

No próximo capítulo é descrito uma abordagem que faz uso dos conceitos apresentados nesta seção na concepção de uma abordagem multi-agente para o problema de gerenciamento de cadeias de suprimento com perturbações no processo de produção de PCs.

5 UECEGRUNN: Uma Abordagem Multi-Agente para Competição de Agentes Negociadores

Uma solução multi-agente para o controle e planejamento da cadeia de suprimentos contempla ambas as inspirações: produtos inteligentes como guias da cadeia de suprimento e regulação social para controlar o uso de recursos [1]. O controle do uso de recursos está presente nas soluções centralizadas, mas neste trabalho este controle é baseado em mecanismos de mercado, como leilões. O gerenciamento de pedidos é realizado de forma descentralizada, com cada pedido sendo gerenciado por um agente denominado *Intelligent Product* (IP). Aprendizagem de máquina foi utilizada na definição do preço de venda de computadores.

O objetivo deste capítulo é apresentar a abordagem multi-agente proposta, doravante denominada UECEGRUNN. Para isso, na Seção 5.1 é apresentada uma visão geral da interações entre os agentes do sistema. Na seções seguintes os diferentes agentes que compõem o sistema são apresentados.

5.1 Organização e Funcionamento do Sistema Multi-Agente

A estrutura geral do sistema UECEGRUNN e os principais relacionamentos entre os agentes do sistema são mostrados na Figura 5.1. O SMA UECEGRUNN gerencia uma fábrica de computadores e cada agente do sistema desempenha papéis bem definidos. O agente central (AC) realiza a coordenação de outros quatro agentes, que são responsáveis por diferentes atividades da cadeia de suprimentos. A coordenação significa que (1) o AC determina quais agentes devem tratar quais eventos de mercado e (2) o AC gerencia o uso de recursos pelos outros agentes. O relacionamento de gerenciamento é entre agente e leilão e determina qual agente é responsável pelo início, pelo fechamento do leilão e pela execução das regras do leilão. A instanciação significa a criação de uma instância de um tipo específico de agente, que passa a existir e a agir de forma autônoma. A negociação é um relacionamento que implica na compra ou venda de bens em leilões. A interação com o mercado é realizada por meio de eventos, dentre os principais eventos estão os seguintes:

- Clientes enviam RFQs aos agentes competidores [2].

[1] Recursos podem ser componentes utilizados na produção de PCs, ciclos de produção (capacidade de produção diária da fábrica) ou PCs finalizados armazenados em estoque.

[2] Agentes ou sistemas multi-agente que representam fábricas que montam PCs e vendem estes PCs no varejo.

- Agentes competidores enviam ofertas aos clientes em resposta às RFQs.
- Clientes enviam pedidos aos agentes em reposta às ofertas enviadas pelos agentes competidores.
- Fornecedores enviam ofertas de componentes aos agentes em resposta às requisições de orçamento (RFQs) enviadas por agentes.
- Fornecedores realizam entrega de componentes em resposta aos pedidos enviados por agentes.

O agente central percebe os eventos do mercado e determina quais agentes devem tratar quais eventos. Como resultado, internamente os agentes recebem mensagens do AC, estas mensagens notificam quais eventos devem ser tratados. Os agentes interagem entre si para resolverem diferentes problemas da cadeia de suprimentos.

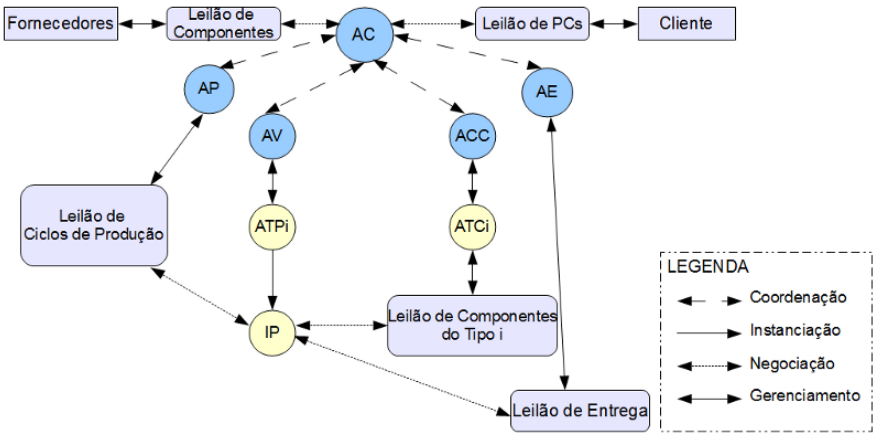


Figura 5.1: Organização do sistema UECEGRUNN.

No que diz respeito às interações entre o AC e os agentes envolvidos com a venda de PCs (doravante denominadas interações central-vendas), considerando D um dia simulado e $0 \leq D \leq 219$, os seguintes eventos podem ocorrer:

CV1. O AC recebe RFQs de clientes e, em seguida, envia ao AV uma lista das RFQs recebidas.

CV2. O AV separa a lista de RFQs em conjuntos de RFQs, um conjunto P_i para cada tipo de produto i, onde $i \in \{1, 2, ..., 16\}$.

CV3. O AV envia para cada ATP_i uma mensagem com o conjunto de RFQs P_i.

CV4. Para cada RFQ recebida, um agente ATP_i envia uma proposta de oferta ao AV.

CV5. No final do dia, o AV seleciona as ofertas mais promissoras (em relação há algum critério específico) e solicita ao AC o envio destas ofertas aos clientes, notificando aos ATPs quais de suas propostas de oferta foram aceitas.

CV6. O AC recebe pedidos de clientes, como resposta a $RFQs$ enviadas no dia $D-1$.

CV7. O AC notifica ao AV os pedidos realizados por clientes.

CV8. O AV separa os pedidos de clientes em conjuntos O_i para cada tipo de produto i.

CV9. O AV envia o conjunto O_i para o agente ATP_i.

CV10. Para cada pedido recebido, um agente ATP_i instancia um agente de produto (IP) que irá gerenciar o pedido recebido até a entrega do mesmo.

No que diz respeito às interações entre o AC e os agentes envolvidos com as compras de componentes (doravante denominadas interações central-compras), os seguintes eventos podem ocorrer:

CC1. No início de um dia D, o agente de compras (ACC - Agente de Compra de Componentes) determina quais RFQs enviar para fornecedores e solicita ao agente central o envio destas RFQs. A decisão sobre quais tipos e a quantidade de cada tipo de componente obter é baseada na demanda média, no desvio padrão da demanda dos últimos dez dias e em estratégias estáticas definidas empiricamente.

CC2. O AC confirma o envio das RFQs para fornecedores no dia D.

CC3. No dia $D+1$, fornecedores que receberam RFQs no dia D respondem com ofertas. O AC envia uma mensagem ao ACC notificando as ofertas recebidas.

CC4. Ainda no dia $D+1$, o ACC decide quais ofertas de fornecedores deve aceitar e, para isso, solicita ao AC o envio de pedidos de componentes aos fornecedores que tiveram ofertas aceitas.

CC5. O AC atualiza suas informações sobre componentes esperados no futuro com base nos pedidos enviados aos fornecedores.

CC6. Se um fornecedor prometeu entregar um componente do tipo i no dia $D+k$, onde $k > 0$, o AC espera até o dia $D+k$ a notificação da fábrica sobre o recebimento do pedido com data de entrega no dia $D+k$. Se o AC não receber o pedido no dia prometido, atualiza suas informações sobre a disponibilidade de componentes, subtraindo o que não foi recebido. Isso acontece porque fornecedores ou entregam o pedido dentro do prazo prometido ou cancelam o pedido. Se o fornecedor entrega o pedido na data acertada, o AC recebe uma notificação de recebimento dos componentes comprados e envia uma mensagem ao ACC sobre a disponibilidade de novos componentes. A mensagem enviada pelo AC diz a quantidade Q de componentes do tipo C recebidos no dia D. O ACC armazena essa informação internamente.

CC7. Paralelamente ao envio de RFQs e realização de pedidos, no início de um dia D, um agente de tipo de componente i (ATC_i) envia uma mensagem ao ACC solicitando a quantidade disponível de componentes do tipo i.

CC8. O ACC, com base nas notificações de pedidos, responde ao ATC_i com uma mensagem contendo a quantidade de componentes do tipo i disponível.

CC9. Com base na quantidade disponível de componentes, um agente ATC_i abre um leilão de oferta de componente do tipo *i* para agentes de tipo de produto ou *Intelligent Products* (IPs).

CC10. Durante o dia *D*, cada um dos ATCs espera mensagens com lances de agentes do tipo IP. No final do dia, os ATCs fecham seus respectivos leilões e selecionam os melhores lances. Cada ATC notifica aos agentes do tipo IP que enviaram lances o resultado de seu respectivo leilão.

CC11. Cada um dos ATCs envia uma mensagem ao ACC solicitando a reserva de componentes leiloados; o ACC, por sua vez, atualiza suas informações internas subtraindo a quantidade de componentes reservados da quantidade de componentes disponível.

No que diz respeito às interações entre o AC e o agente de produção (interações central-produção), os seguintes eventos podem ocorrer:

CP1 No primeiro dia simulado, o AP recebe uma mensagem do AC notificando a capacidade diária da fábrica em ciclos de produção. O AP utiliza essa informação para determinar no início de cada dia a quantidade de ciclos de produção a ser ofertada no leilão do dia.

CP2 No início de um dia *D*, o AP abre o leilão de oferta de ciclos de produção.

CP3 Durante o dia *D*, o AP espera lances por ciclo de produção de agentes do tipo IP.

CP4 No final do dia o leilão é fechado e os agentes do tipo IP que enviaram lances são notificados do resultado do leilão de ciclos de produção.

CP5 O AP solicita ao AC a produção dos produtos correspondentes aos lances vitoriosos enviados por agentes do tipo IP.

No que diz respeito às interações entre o AC e o agente de entrega (interações central-entrega), os seguintes eventos podem ocorrer:

CE1 em um dia *D*, o AE abre um leilão de entrega.

CE2 Durante o dia *D*, o AE espera lances por entrega de produtos de agentes do tipo IP.

CE3 No final do dia o leilão é fechado e os agentes do tipo IP que enviaram lances são notificados do resultado do leilão de entrega.

CE4 Com base nos lances vencedores, o AE solicita ao AC o envio de pedidos correspondentes aos agentes que venceram no leilão.

Os eventos acima podem ocorrer paralelamente, portanto os agentes do SMA UECEGRUNN foram concebidos no nível de implementação como *threads* [1] que se comunicam via rede por meio do protocolo HTTP (BERNERS-LEE; FIELDING; FRYSTYK, 1996).

[1] Um fluxo de controle seqüencial isolado dentro de um programa. Como um programa seqüencial qualquer, um thread tem um começo, um fim, e uma seqüência de comandos. Entretanto, um thread em Java não é um programa, não roda sozinho, roda dentro de um programa

Com relação aos eventos das interações central-entrega, o leilão gerenciado pelo AE foi configurado de modo a aceitar todos os lances, dado que o simulador do TAC-SCM não impõe qualquer limite à capacidade de entrega das fábricas.

Nas próximas seções, os agentes presentes no sistema UECEGRUNN são apresentados e as interações central-vendas (CV), central-compras (CC), central-produção (CP) e central-entrega (CE) são referenciadas no contexto da apresentação de cada agente.

5.2 O Agente Central (AC)

O AC é responsável pelo controle de acesso de informações sobre inventário e pela comunicação com a fábrica, que informa a capacidade de produção disponível. Além disso, o AC coordena a interação dos agentes internos do SMA UECEGRUNN com o mercado. A Figura 5.2 mostra as interações do AC com os agentes que gerenciam a venda de computadores, a obtenção de componentes, o processo de produção e o envio de pedidos.

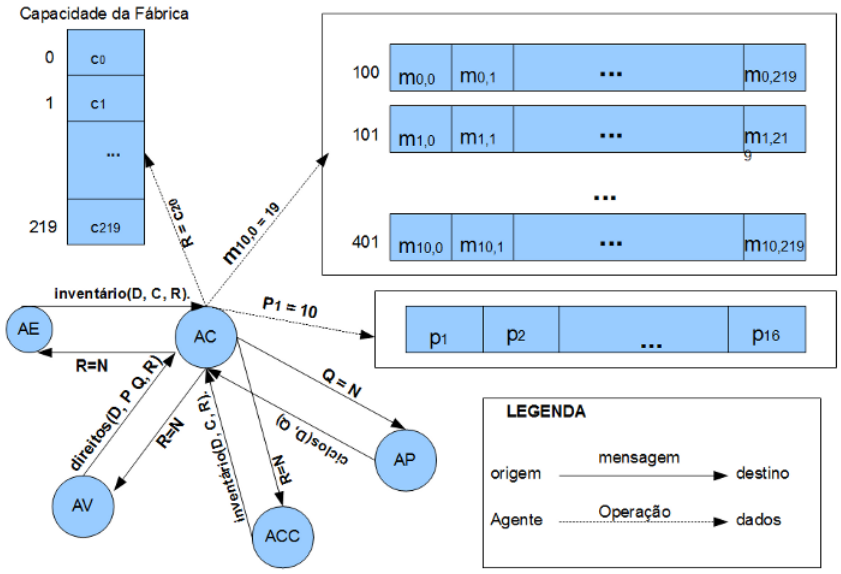


Figura 5.2: Ilustração das interações do agente central (AC).

Uma seta continua indica uma mensagem do emissor (origem) ao destinatário (destino). Um texto sobre a seta consiste no conteúdo da mensagem. Os tipos de mensagem enviadas ao AC são: (1) *inventário(D, C, R)*, verificar os níveis de inventário no dia D do produto ou componente com o código C e armazena o resultado em R; (2) *direitos(D, P, Q, R)*, solicita Q unidades de direitos de venda de produtos do tipo P válidos até o dia D e coloca o resultado em R; (3) *ciclos(D,R)*, verifica a quantidade de ciclos de produção disponível no dia D e armazena o resultado em R. Quando uma mensagem como *inventário(D, C, R)* é recebida pelo AC, os valores de D e C são especificados e R é uma variável, logo o AC responde com um valor para o argumento de saída R que case com valores específicos dos argumentos de entrada D e C.

Uma operação sobre as estruturas de dados é representada por uma seta tracejada com a descrição da operação sobre a seta. Por exemplo $m_{i,j} = 10$, faz com que o elemento $m_{i,j}$ da

matrix $[m_{i,j}]_{10\times219}$ tenha seu conteúdo alterado para conter o valor 10.

Os dados sobre o estado atual do SMA UECEGRUNN são armazenados em estruturas como vetores ou matrizes. Assim, há um vetor de inteiros com capacidade para 220 elementos: $capacidadeFabrica = (c_0, c_1, ..., c_{219})$, onde c_i é a capacidade da fábrica em ciclos de produção disponível no dia i. Informações sobre o inventário atual de produtos (quantidade atual de produtos em estoque) são armazenadas em um vetor de 16 elementos, um para cada tipo de produto. Cada elemento p_i do vetor de produtos representa a quantidade de produtos do tipo i disponível em estoque. Informações sobre a disponibilidade de componentes são armazenadas em uma matriz, onde cada linha da matriz contém 220 elementos com informações sobre a disponibilidade de um tipo de componente ao longo dos 220 dias simulados. Seja $[m_{ij}]_{10\times220}$ uma matriz com informações sobre os dez tipos de componentes ao longo do jogo. Um elemento m_{ij} representa a quantidade de componentes do tipo correspondente ao índice i recebido de fornecedores no dia j e ainda não utilizada. Os índices são mapeados para códigos de componentes de acordo com a Tabela 5.1.

Tabela 5.1: Mapeamento de índice para código de componente.

Índice	Código de Componente
0	100
1	101
2	110
3	111
4	200
5	210
6	300
7	301
8	400
9	401

A quantidade de componentes disponível do tipo associado a um índice i no dia j, é dada pela Equação 5.1.

$$q_{i,j} = \sum_{k=0}^{j} m_{i,k} \tag{5.1}$$

O AC sabe quantos ciclos de produção são necessários para a produção de uma unidade de PC de determinado tipo. Nas interações central-vendas (interações que vão de CV1 a CV10) , quando um AV solicita n unidades de direito de venda de um produto do tipo i para o dia D, o AC precisa determinar a quantidade de direitos de venda disponível para produtos do tipo i, o que é realizado por meio da Equação 5.2.

$$u_{fc}(i,D) = min(\frac{fc}{f_i},\ min(q_{comp_1,D},\ q_{comp_2,D},\ q_{comp_3,D},\ q_{comp_4,D})) \tag{5.2}$$

onde f_i é a quantidade de ciclos de produção necessária para se produzir uma unidade de produto do tipo i, fc é a capacidade livre da fábrica em ciclos de produção, i é um código de produto,

D é data de verificação, $comp_j$ é o índice associado ao código do componente e *min* é uma função que retorna o menor escalar de uma lista de escalares dada. O valor calculado u_{f_c} é o que a fábrica garante produzir caso nenhum problema de produção venha ocorrer e se apenas produtos do tipo i forem produzidos. Portanto, quando o AC recebe do AV uma mensagem do tipo $direitos(D,P,Q,R)$ executa o procedimento da Figura 5.3.

mensagem recebida do AV: $direitos(D,P,Q,R)$, onde D é o dia para o qual os direitos foram solicitados; P, código do tipo de produto para os quais os direitos foram solicitados; Q, a quantidade de direitos solicitados; e R, o argumento de saída para a resposta ao AV (ou agente de vendas).

$requisitado$ = quantidade de direitos solicitada pelo AV

$disponivel = u_{fc}(i,D)$, de acordo com a Equação 5.2 e fazendo $i = P$.

declarar variável rc, fazendo $rc = Q$.

Início:

Se $requisitado \leq disponivel$ então

Enviar mensagem de resposta para o AV com conteúdo $R = Q$

Atualiza estado interno:

Para $k = 1$ a 4 faça

Para $j = D$ a 0 faça

Se $m_{comp_k,j} > rc$, Então Faça

$m_{comp_k,j} = m_{comp_k,j} - rc$

$rc = 0$

Caso contrário

$rc = rc - m_{comp_k,j}$

$m_{comp_k,j} = 0$

Fim Se

Se $rc \leq 0$, Então sai do laço atual; Fim Se

Fim Para

Fim Para

$c_D = c_D - Q \times f_i$, fazendo $i = P$

Caso Contrário

Enviar mensagem de resposta para o AV com o conteúdo $R = 0$

Fim Se

Fim.

Figura 5.3: Procedimento do agente central para liberação de direitos.

Como pode ser observado no procedimento apresentado, os recursos correspondentes à quantidade de direitos solicitada pelo AV (Q) são deduzidos da quantidade de recursos disponível. Contudo Se o AV não utilizar os direitos solicitados, envia uma mensagem *devolverDireitos(D, P, Q)*, que significa a devolução de direitos de venda de Q produtos do tipo P para o dia D. O recebimento de uma mensagem do tipo *devolverDireitos(D, P, Q)* vinda do AV faz com que o AC realize as operações:

1. $m_{comp_j,D} = m_{comp_j,D} + Q$, para $1 \leq j \leq 4$, sendo que $comp_j$ é o código de um tipo de componente necessário para se produzir uma unidade de produto do tipo P.

2. $c_D = c_D + Q \cdot f_P$, onde f_P é a quantidade de ciclos necessária para a produção de uma unidade de produto do tipo P.

É importante notar que direitos de venda não significam que os recursos serão alocados definitivamente. O AC possuiu a informação sobre a disponibilidade de ciclos de produção e componentes. Quando o AC recebe notificações sobre pedidos enviados aos fornecedores (interação central-compras CC5), ele atualiza a quantidade de componentes disponível, adicionando a quantidade de componentes pedidos. Os ciclos de produção são constantes, fixado no simulador do TAC-SCM como 2000. Portanto, os ciclos diários são definidos inicialmente com o valor 2000. À medida que direitos são liberados ao AV, a quantidade de componentes disponível é reduzida de acordo com a quantidade de direitos liberada. O mesmo ocorrendo com a capacidade de produção da fábrica. Essa informação é interna ao AV. Isso significa que o ATC, quando realiza seus leilões de componentes, não conhece essa informação interna ao AC (interações central-compras de CC8 a CC11). O mesmo ocorre com o AP com a quantidade de componentes disponível.

Outra observação importante sobre a distribuição da informação é sobre a possibilidade de falhas do AC. O AC armazena informações de quantidade de componentes com base em pedidos realizados (interações central-compras CC5) e não com base em componentes efetivamente recebidos. Assim, os pedidos podem falhar e não resultarem na entrega de componentes por parte do fornecedor. Já a informação de quantidade de componentes que o ACC guarda é sobre componentes efetivamente recebidos (interações central-compras CC6). Como um ATC recebe a quantidade de componentes disponíveis do ACC, os componentes leiloados é que definitivamente serão alocados para o processo de produção. A distribuição de direitos, dessa perspectiva, constitui única e exclusivamente em uma permissão de venda e não na alocação eficaz de recursos. Esta alocação ocorre nos leilões, quando componentes efetivamente são liberados para produção. O mesmo processo ocorre na venda de ciclos de produção. Neste caso, o AP tem uma informação inicial da quantidade de ciclos de fábrica disponível ao longo do jogo (interações central-produção CC1) e utiliza essa informação ao longo de todo o jogo sem consultar o AC. Portanto, quem tem a palavra definitiva sobre o uso de ciclos de produção é o AP (interações que vão de CP2 a CP5).

Nas interações central-produção (interações de CP2 a CP5), quando o AP decide, com base no resultado de leilões de ciclo de produção, quais produtos devem ser produzidos, solicita ao AC que reserve ciclos necessários para suprir os produtos de pedidos de IPs que venceram o leilão. Assim, se a solicitação ocorreu no dia D, somente no dia $D+1$ é que a produção de produtos cuja solicitação ocorreu no dia D será realizada.

Nas interações central-entrega (interações de CE1 a CE4), como não há limites para a capacidade de envio de pedidos, a interação entre AE e AC consiste na solicitação de envio de todos os pedidos produzidos.

5.3 O Agente de Compras (ACC)

As atividades do agente de compras podem ser divididas em duas etapas: manutenção de inventário e obtenção de componentes. O programa do ACC é baseado em um agente reativo com estado interno, exceto pelo fato de que foram utilizadas heurísticas para a seleção de ações,

além de um conjunto de regras *condição-ação*. Uma visão geral das atividades realizadas pelo ACC é ilustrada na Figura 5.4, apresentando as seguintes mensagens e operações que especificam algumas etapas das interações central-compras:

1. Mensagem $RFQ_i(S,C,PR,Q,PE)$ = Requisição de orçamento i para componentes do tipo C enviada ao fornecedor identificado com S, especificando: um preço de reserva (PR), quantidade (Q) e prazo de entrega (PE).

2. Mensagem $oferta(i,PO,Q,PE)$ = oferta em resposta à RFQ i, especificando: um preço de oferta (PO), uma quantidade que o fornecedor pode atender (Q) e um prazo de entrega do fornecedor (PE).

3. Operação atualiza = operação de atualização da sequencia história, as informações do dia atual sobre a demanda por componentes

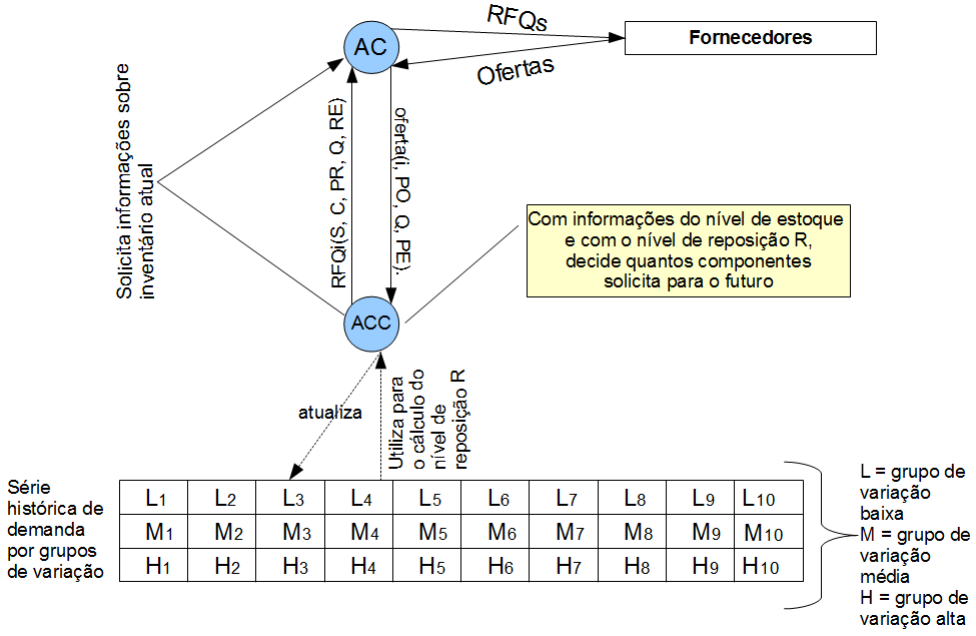


Figura 5.4: Atividades do agente de compras.

5.3.1 Manutenção de Inventário

No contexto das interações central-compras (nas interações que vão de CC1 a CC2), a quantidade de componentes especificada em uma RFQ é definida por meio de uma estratégia ATO *Assembler to Order* (ATO) com base em um nível de reposição. Em uma estratégia ATO, componentes são adquiridos, estocados e um conjunto de produtos pré-configurados é montado de acordo com pedidos. Dessa forma, produtos pré-configurados são enviados imediatamente para suprir os pedidos para os quais foram montados, sem a necessidade de serem estocados. A estratégia adotada é baseada na estratégia geral do agente *Mertacor*, descrita por Chatzidimitriou et al. (2008).

A estratégia ATO trabalha como segue: o objetivo principal do sistema é calcular certo nível de inventário (limiar) que precisa ser satisfeito e abaixo do qual substituição é necessária. O limiar para cada componente é calculado em tempo real utilizando a Equação 5.3, como descrita por Simchi-Levi (2005):

$$R = D_{AVG}L_{AVG} + z\sqrt{L_{AVG}D_{STD}^2 + D_{AVG}^2 L_{STD}^2} \tag{5.3}$$

onde, D_{AVG} é a demanda média para um componente específico, L_{AVG} é o tempo médio de entrega do fornecedor, D_{STD} é o desvio padrão da demanda, L_{STD} é o desvio padrão do tempo de entrega e z é um fator de segurança denotando o nível de serviço fornecido ao usuário. O segundo termo da Equação 5.3 determina que o nível de reposição fique acima da demanda esperada (representada pelo primeiro termo) em um dado intervalo de tempo, reduzindo as chances de falta de componentes em estoque. Quanto maior o valor de z, maior será o nível de reposição. Quanto mais componentes em estoque, menor a chance de pedidos atrasados por atraso na produção por falta de suprimentos, garantindo níveis de serviço mais altos. Simchi-Levi (2005) especifica alguns valores de z e o os níveis de serviços conseguidos a partir da definição destes valores, como mostra a Tabela 5.2. Um nível de serviço de 99%, por exemplo, significa que 99% de todos os pedidos de clientes seriam produzidos a tempo (até a data de entrega contratada). A demanda é calculada com base nos pedidos de produtos realizados por clientes. Valores médios e os desvios padrões são calculados sobre um período de 10 dias do histórico de transações.

Estatísticas da demanda de produtos podem variar substancialmente, tornando os limiares (níveis de reposição) instáveis. Isso acontece por causa de dois fatores: (1) variabilidade diária da demanda de clientes e (2) a existência de outros competidores. Contudo, pequenas variações podem ser obtidas se a demanda for aproximada em termos de variações de custo de produtos, considerando grupos de variação de custo: produtos de alto custo (ou variação de alto custo), produtos de médio custo (ou variação de médio custo) e produtos de baixo custo (ou variação de baixo custo). Portanto, a demanda é calculada como demanda nos grupos de variação de custo de produtos sobre os pedidos de produtos realizados por clientes. Por exemplo, pedidos para produtos do tipo 1, 2, 9, 10 e 11 serão computados como demanda no grupo de variação de baixo custo, pois como mostra a Tabela 2.2, os produtos do tipo 1, 2, 9, 10 e 11 são variações de baixo custo.

Tabela 5.2: Níveis de serviço e os valores de z correspondentes. Adaptado de (SIMCHI-LEVI, 2005)

Nível de Serviço	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%	100%
z	1,29	1,34	1,41	1,48	1,56	1,65	1,75	1,88	2,05	2,33	3,08

O mapeamento da demanda por variação de custo de produtos para demanda por componentes é ilustrada na Figura 5.5. Componentes são compartilhados por muitos produtos, pois um componente pode ser necessário na produção de um ou mais tipo de produto. Assim, é possível fazer o mapeamento da demanda por grupos de variação de custo de produto (baixo custo, alto custo e médio custo) para a demanda por componente. O grau de partição de um compo-

nente em um grupo de variação de custo de produto é definido como um valor que determina a frequência de participação do componente em diferentes grupos. Dessa forma, o grau de participação de cada componente para cada grupo de variação é calculo como mostra a Figura 5.5. Por exemplo, o componente associado ao índice 1 tem grau de participação 0.4, 0.33 e 0.0 nos grupos de variação de baixo custo, de médio custo e de alto custo, respectivamente. Pegando o grupo de variação de baixo custo, percebe-se que há cinco produtos participando desse grupo (produto 1, 2, 9, 10 e 11). Destes produtos, uma unidade de componente do tipo 100 (que é o componente associado ao índice 1), como pode ser visto na Tabela 2.2, é utilizada na produção de uma unidade dos produtos 1 e 2. Portanto, o grau de participação do componente associado ao índice 1 (componente 100) é calculado como $2.0/5.0 = 0.4$.

A demanda média por componente i, D_{AVG_i}, é calculada de acordo com a Equação 5.4.

$$D_{AVG_i} = l_i \cdot L_{AVG} + m_i \cdot M_{AVG} + h_i \cdot H_{AVG} \tag{5.4}$$

onde, L_{AVG}, M_{AVG} e H_{AVG} representam, respectivamente, as demandas médias nos grupos de variação custo alto, médio e baixo e l_i, m_i e h_i, representam os graus de participação de um componente associado ao índice i nos grupos de variação de custo alto, médio e baixo, respectivamente.

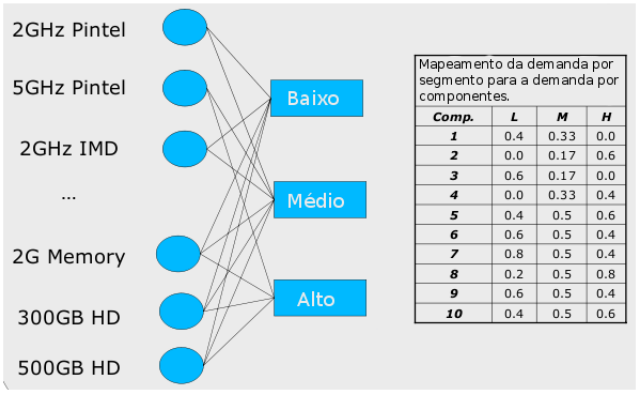


Figura 5.5: Mapeamento da demanda por variação para a demanda por componentes. A demanda em cada grupo de variação é multiplicada pelo número corresponde para o cálculo da demanda final de componente.

5.3.2 Estratégias de Aquisição de Componentes

O desempenho global do sistema UECEGRUNN é altamente dependente de dois fatores: (1) ter um inventário composto por componentes baratos e (b) satisfazer os níveis de inventário, desde que pedidos atrasados implicam em penalidades e após cinco dias de atraso, os pedidos são cancelado. Para satisfazer estes requisitos, o agente ACC foi dotado de uma heurística simples, dividida em duas fases: de estado inicial e de estado normal.

A estratégia de estado inicial é executada nos primeiros dois dias do jogo, consistindo em RFQs de dois tipos: primárias e secundárias. As RFQs primárias são geradas pelo ACC

no dia simulado 0 para consolidar um inventário inicial, a fim de possibilitar rapidamente o inicio da produção de computadores (que é controlada pelo agente de produção), portanto, possibilitando ao agente responsável pelas vendas a realização de ofertas logo nos primeiros dias do jogo. Os componentes adquiridos dessa forma são previsivelmente caros, contudo de alta utilidade, pois no início do jogo não há uma forte competição, possibilitando a venda de computadores a preços mais elevados. Na estratégia de estado inicial, para cada tipo de componente, são enviadas 5 RFQs para cada fornecedor de um tipo de componente considerado. Os parâmetros das RFQs foram definidos como segue:

Tipo de componente: é definido o tipo (o código) do componente especificado;

Quantidade: cada RFQ específica 200 unidades de componentes de um tipo específico;

Prazo de entrega: prazos de entrega foram definidos de forma empírica, para os dias 3, 6, 9, 12 e 15;

Preço de reserva: o preços de reserva foi fixado como 90% do preço base do componente.

As RFQs secundárias são enviadas no dia simulado 1, sendo geradas pelo ACC para tentar obter componentes baratos no futuro. Para isso, são enviadas 5 RFQs para cada fornecedor de componente, sendo que cada umas das cinco RFQs é para um prazo de entrega diferente. Os prazos de entrega das RFQs para cada tipo de componente foram determinados de forma empírica, para os dias simulados 20, 30, 40, 50 e 60. Os outros parâmetros das RFQs por tipo de componente foram determinados como segue:

Tipo de componente: é definido o tipo (o código) do componente especificado;

Quantidade: a quantidade de componentes para RFQs com prazo de entrega nos dias $20, 30, 40, 50$ e 60 foi definida como $50, 50, 100, 100, 150$, respectivamente.

Preço de reserva: o preço de reserva para RFQs com prazo de entrega nos dias $20, 30, 40, 50$ e 60 foi definido como 80%, 75%, 70%, 70% e 65% do preço base do componente, respectivamente. Estes preços também foram definidos empiricamente.

A partir do terceiro dia (no dia 2), o ACC substitui a estratégia de obtenção para uma estratégia normal para satisfazer níveis de reposição de estoque, dividindo as RFQs enviadas para fornecedores em duas categorias:

1. Normal: para satisfazer os níveis de reposição e possibilitar ofertas para clientes.
2. RFQs de Longo Prazo: para obter componentes baratos no futuro; desta forma, os níveis de reposição podem ser satisfeitos antecipadamente, mas com componentes baratos, incrementando os lucros nas vendas.

Na estratégia normal, o prazo de entrega é definido como $D+2$, onde D é o dia atual. Para cada componente é calculado nível de reposição (RL - *Reposition Level*), como mostra a Equação 5.3, e as seguintes informações são obtidas (solicitadas ao AC):

1. Quantidade de componentes disponível para componente do tipo i em estoque (Q_i);
2. Quantidade de componentes do tipo i em pedidos enviados aos fornecedores e a serem recebidos nos próximos dois dias (OO_i - *On Order*);
3. Quantidade de componentes do tipo i em pedidos pendentes (e que devem ser resolvidos nos próximos dois dias) de clientes (BO_i - *Back Order*).

Dessa forma, o nível de estoque atual para componentes do tipo i é calculado como $IL_i = Q_i + OO_i - BO_i$. Na estratégia normal, a quantidade de componentes de determinado tipo e o preço de reserva das RFQs é determinado de acordo com o procedimento da Figura 5.6.

Fornecedores possuem mais propensão para cumprirem prazos de entrega maiores, dado que contam com mais tempo para planejar a produção e o envio de pedidos de componentes. A estratégia de longo prazo é baseada neste fato de que solicitações por componentes no futuro podem ser atendidas mais facilmente pelos fornecedores. Como conseqüência, pode ocorrer a obtenção de componentes baratos com um prazo de entrega mais longo, isso foi percebido também por meio de experimentos realizados. Preços acessíveis significam preços mais baixos que o preço médio de mercado em determinado momento, possibilitando maiores lucros. A estratégia de longo prazo é uma estratégia fixa, executada nos dias simulados 2, 3, 4, 5 e 6. Para cada dia $D \in \{2,3,4,5,6\}$, são enviadas cinco RFQs para fornecedores de cada tipo de componente com as seguintes propriedades:

Quantidade: uma quantidade constante igual a 100.

Tipo de Componente: é especificado o tipo de componente para o qual o orçamento foi solicitado.

Dia de Entrega: são definas 5 RFQs com dias de entrega definidos como: $prazo_1 = D + 2$, $prazo_5 = 215 - D \cdot 8$ e $prazo_k = prazo_{k+1} - 40$, para $k = 4,3,2$. Neste caso, $prazo_4 = prazo_5 - 40$, $prazo_3 = prazo_4 - 40$ e $prazo_2 = prazo_3 - 40$, onde $prazo_j$ é o dia de entrega máximo definido para a RFQ j. Observe que serão enviadas cinco RFQs para cada fornecedor de componente.

Preço de reserva: para as RFQs 1, 2, 3, 4 e 5, os preços de reserva serão definidos, respectivamente, como 100%, 90%, 85%, 75% e 65% do preço base de cada componente.

Quando recebe os componentes dos fornecedores, o AC notifica ao ACC o recebimento destes componentes. O ACC, por sua vez, notifica aos agentes de tipo de componente de modo que estes possam atualizar suas informações sobre o inventário atual do tipo de componente que representam. Os agentes de tipo de componente utilizam esse inventário para determinar a quantidade de componentes que disponibilizam em leilões de oferta de componentes.

Variáveis e Funções:

$quantidade_i$: quantidade de componentes do tipo i definida em cada RFQ a ser enviada aos fornecedores de componentes do tipo i.

$precoReserva_i$: preço de reserva para RFQ de componentes do tipo i.

$precoBaseComponente_i$: preço base de componentes do tipo i.

$random(a,b)$: gera um número real no intervalo real $[a,b)$.

D_{AVG_i}: demanda média do componente do tipo i em termos de grupo de variação de custo de produtos nos últimos dez dias.

D: dia atual

$RFQ_i(S,C,PR,Q,PE)$ = Requisição de orçamento i para componentes do tipo C enviada ao fornecedor identificado com S, especificando: um preço de reserva (PR), quantidade (Q) e prazo de entrega (PE).

$taxa_{j,i}$: a proporção de componentes que deve ser obtido do fornecedor j para componentes do tipo i[a].

IL_i: nível de estoque do componente i.

RL_i: nível de reposição do componente i.

Para cada componente i faça

Se $IL_i < RL_i$ Então

$quantidade_i = RL_i - IL_i$

$precoReserva_i = precoBaseComponente_i$

Caso contrário, Se $Q_i + OO_i < 2 \cdot RL$ então

$quantidade_i = D_{AVG_i}$

$precoReserva_i = (0.7 - 0.1 \cdot random(0.0, 0.10)) * precoBaseComponente_i$

Caso contrário

$quantidade_i = 0$

$precoReserva_i = 0$

Fim Se

Se $quantidade_i > 0$ então

Para j = 1 a n faça (comentário: n pode ser 1 ou 2; para $i \leq 111$, n = 1, caso contrário, n = 2).

obtém taxa ($taxa_{j,i}$) do fornecedor j para o componente i ($fornecedor_{(j,i)}$).

Se $taxa_{j,i} > 0$ então

Envia mensagem ao AC solicitando envio de uma mensagem contendo uma RFQ como $RFQ(fornecedor_{(j,i)}, i, precoReserva, quantidade \cdot taxa_{(j,i)}, D+2)$ para $fornecedor_{(j,i)}$.

Fim Se

Fim Para

Fim Se

Fim Para

[a]Para $i <= 111$, $taxa_{j,i} = 1.0$, pois há um único fornecedor para componentes do tipo $100, 101, 110$ e 111. Para outros tipos de componentes, há dois fornecedores. A $taxa_{j,i}$ é definida como 90% para o fornecedor que teve o último preço de oferta menor e 10% para o outro fornecedor.

Figura 5.6: Procedimento para execução pelo agente de produção da estratégia normal de obtenção de componentes.

5.4 Agentes de Tipo de Componente

O objetivo dos agentes de tipo de componente é a distribuição de componentes internamente, de modo a suprir a fabricação de PCs para pedidos de cliente. Para isso, o agente possui uma arquitetura reativa com estado interno, ou seja, o estado vai sendo atualizado com as ordenações que vão sendo feitas para a decisão tomada no procedimento da Figura 5.7. No contexto das interações central-compras (interações que vão de CC1 a CC11), os agentes de tipo

de componente interagem o ACC (interações CC7, CC8 e CC11) e com agentes do tipo IP (interações CC9 e CC10), resolvendo o problema de gerenciamento de cada tipo de componente. Dessa forma, há dez agentes de tipo de componente, cada um responsável por componentes de um determinado tipo. Estes agentes controlam todas as informações sobre os componentes que gerenciam. A distribuição de componentes é realizada por meio de leilões, sendo que cada leilão é gerenciado como mostra a Figura 5.7. Portanto, as interações entre o ACC e os IPs ocorrem por meio de leilões (nas interações CC9 e CC10).

Variáveis e funções:

$ofertas$ = lista de ofertas

$oferta_i$ = oferta na posição i da lista $ofertas$.

$preco(oferta_i)$ = preço da oferta i

inv total de componentes disponíveis em estoque no dia atual

$tamanho$ = função que recebe uma lista e retorna a quantidade de elementos na lista

$Quantidade(oferta_i)$ = quantidade de componentes solicitados na oferta i

$agente(oferta_i)$ = agente que enviou a oferta i, no caso um agente do tipo IP

Início:

1. classifica a lista $ofertas$ em ordem decrescente de preço
2. Para $i = 1$ a $tamanho(ofertas)$ faça
   (a) Se $Quantidade(oferta_i) <= inv$
      i. $inv = inv - Quantidade(oferta_i)$
      ii. Envia mensagem para $agente(oferta_i)$ notificando a aceitação da oferta
   (b) Caso contrário, envia mensagem para $agente(oferta_i)$ notificando que a oferta não foi aceita
   (c) Fim Se
3. Fim Para

Fim

Figura 5.7: Procedimento executado pelos agentes de tipo de componente para seleção de ofertas no leilão de componentes.

Para cada tipo de componente é realizado um leilão para componentes daquele tipo. No início de cada dia, cada agente de tipo de componente inicia um leilão de oferta de componentes. O leilão fica aberto até o final do dia esperando que agentes do tipo IP enviem ofertas por componente. Cada oferta enviada por um agente do tipo IP tem os seguintes campos: identificador da oferta, preço de oferta, quantidade de componentes requerida e o identificador do agente que enviou a oferta. No final do dia, as ofertas são verificadas em ordem decrescente de seus respectivos preços e são atendidas até que não haja mais componentes em estoque. No final do leilão, todos os agentes recebem uma notificação se foram atendidos ou não.

5.5 Os Agentes de Produção e Entrega - AP e AE

O agente de produção é responsável por definir para quais pedidos deve alocar ciclos de produção. As atividades de produção e entrega são ilustradas na Figura 5.8. Nas interações central-produção (interações que vão de CP1 a CP5), o AP gerencia um leilão de ciclos de produção. O leilão de ciclos de produção é controlado por meio do procedimento da Figura 5.9.

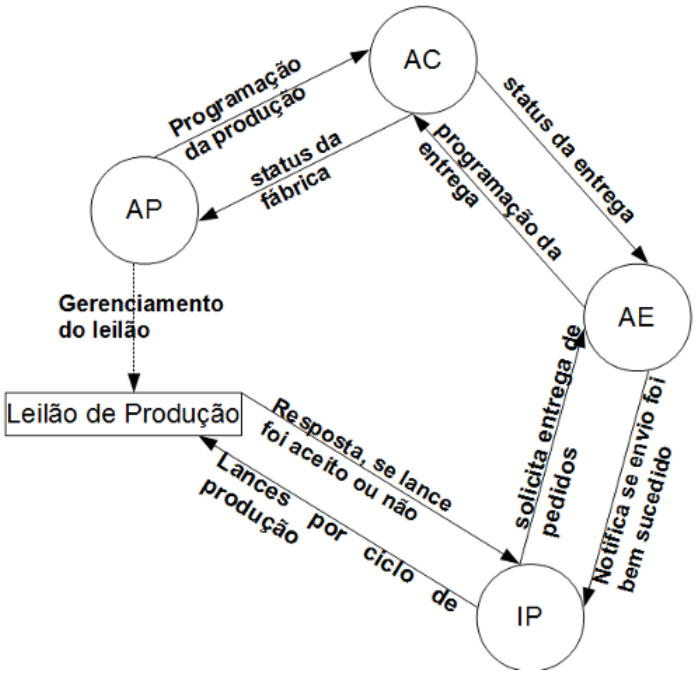


Figura 5.8: Atividades de planejamento da produção e envio de pedidos.

Os agentes do tipo IP participam deste leilão (o que pode ser observado nas interações CP2 a CP4), concorrendo entre si para conseguirem alocar uma parte da capacidade da fábrica para seus pedidos. O leilão gerenciado pelo AP é aberto no início do dia e, no final do dia, o leilão é fechado para verificação de quais IPs tiveram suas ofertas de compra de ciclos de produção aceitas. Depois de decidir quais IPs devem ser atendidos, o AP gera uma programação de produção de produtos no dia d para produção no dia $d+1$. Esta programação é enviada ao agente central, que notifica à fábrica quais produtos devem ser produzidos.

O agente de entrega simplesmente recebe solicitações de agentes do tipo IP para envio de pedidos que estes agentes representam (interação CE2). Em seguida, solicitam ao agente central, no final do dia d o envio da programação de entrega que deve ser realizada no dia $d+1$ (interação CE4). Esta programação é enviada pelo agente central para a fábrica, que se encarrega de enviar os pedidos finalizados aos seus respectivos clientes.

O agente de entrega foi colocado na estrutura do SMA UECEGRUNN apenas para manter a coerência, dado que no TAC-SCM não há limite de entrega, portanto, não há o que gerenciar em termos de entrega. Todos os pedidos que possam ser atendidos com PCs já produzidos, podem ser enviados, sem restrição de capacidade de entrega. Na implementação atual, AE simplesmente repassa a informação de entrega para o AC, o que poderia ser realizado diretamente pelos IPs.

Variáveis e funções:

$ofertas$ = lista de ofertas

$oferta_i$ = oferta na posição i da lista $ofertas$.

$preco(oferta_i)$ = preço da oferta i

$capacidade$ = capacidade da fábrica em ciclos de produção, inicialmente com valor igual a 2000

$tamanho$ = função que recebe uma lista e retorna a quantidade de elementos na lista

$Ciclos(oferta_i)$ = quantidade de ciclos solicitados no oferta i

$agente(oferta_i)$ = agente que enviou a oferta i

Início:

1. classifica a lista $ofertas$ em ordem decrescente do preço de oferta
2. Para $i = 1$ a $tamanho(ofertas)$ faça
   (a) Se $Ciclos(oferta_i) <= capacidade$
      i. $capacidade = capacidade - Ciclos(oferta_i)$
      ii. Envia mensagem para $agente(oferta_i)$ notificando a aceitação da oferta
   (b) Caso contrário, envia mensagem para $agente(oferta_i)$ notificando que a oferta não foi aceita
   (c) Fim Se
3. Fim Para

Fim

Figura 5.9: Procedimento executado pelo agente de produção para o leilão de ciclos de produção.

5.6 Agente de Vendas (AV)

O agente de vendas (AV) é responsável pela decisão final de quais ofertas devem ser enviadas em reposta às RFQs de clientes. No contexto das interações central-vendas (CV1 a CV10), o AC notifica ao AV os eventos diários sobre RFQs (interação CV1) e pedidos de clientes (interações CV6 e CV7). Em resposta, o AC recebe solicitações de direitos para venda de computadores (interação CV5). Estes direitos podem ser utilizados até uma data especificada, que é definida pelo próprio AV. A Figura 5.10 ilustra as principais atividades do AV.

O AV espera que todos os ATPs configurem ofertas em resposta às RFQs de clientes (o que ocorre ao longo das interações CV3, CV4 e CV5). Um ATP do tipo i é responsável por RFQs para produtos do tipo i. O AV define quais ATPs recebem quais RFQs (interações CV8 e CV9). No final de um dia simulado, o AV ordena as ofertas geradas pelos ATPs em ordem decrescente do lucro esperado (o que ocorre na interação CV5), dado pela Equação 5.5:

$$L(k) = (pv_k - pc_k) \cdot p_k \tag{5.5}$$

onde $L(k)$ é o lucro esperado de uma oferta para uma RFQ de um produto do tipo k, pv_k é o preço de oferta de uma unidade de produto do tipo k, pc_k é o custo aproximado de uma unidade

de produto do tipo k e p_k é probabilidade da oferta ser aceita. Ao percorrer a lista de ofertas, o AV consulta o AC se há uma quantidade de direitos suficientes para suprir a oferta. Isso é feito por meio da mensagem $direitos(D,P,q_k,R)$ (interações CV5 e CV6). Se a resposta do AC for uma mensagem com $R = q_k$, onde q_k é a quantidade de direitos necessários para a oferta a ser enviada, a oferta é enviada ao cliente. No dia seguinte, o ATP que enviou a oferta é notificado se ela foi aceita pelo cliente ou não. Se uma oferta não foi aceita pelo cliente, o ATP solicita ao AV a liberação dos direitos reservados para a oferta. O AV então envia uma mensagem $devolverDireitos(D,P,q_k)$ ao agente AC. Como resposta, o AC libera q_k direitos para serem utilizados no futuro. É importante notar que o AC somente libera uma quantidade de direitos se houver direitos suficientes para a quantidade solicitada, nunca envia menos ou mais.

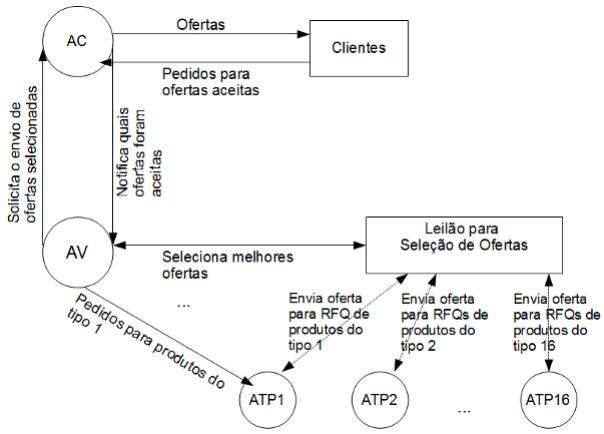


Figura 5.10: Atividades de vendas e interação entre agentes no processo de vendas.

5.7 Agentes de Tipo de Produto

O objetivo dos agentes de tipo de produto é a geração de ofertas em resposta a RFQs enviadas por clientes (nas interações CV1, CV2 e CV3). Nas interações central-vendas (CV3, CV4 e CV5), para cada RFQ recebida de clientes para um tipo de produto i, um agente de tipo de produto (ATP) i é acionado com o objetivo de gerar uma oferta para a RFQ correspondente. Para cada RFQ de cliente, um ATP deve:

- Gerar uma oferta com preço definido (interação CV4).
- Para cada oferta aceita pelos clientes, os ATPs devem gerar um agente do tipo IP que irá se encarregar pelo processamento e finalização do pedido correspondente (interação CV10).

Na geração de uma oferta, é necessário definir um preço. A seguir é descrito como um ATP define este preço de oferta, enquanto que a Seção 5.8 especifica os agentes do tipo IP.

5.7.1 O Programa de um Agente de Tipo de Produto

Um agente de tipo de produto (ATP) tem um programa baseado em um agente reativo simples, como mostra a Figura 5.11. Para RFQ recebida, um ATP deve decidir um preço de oferta apropriado. Diferentes estratégias podem ser projetadas para um ATP definir o preço de oferta que seja mais promissor em termos de lucro e probabilidade de aceitação.

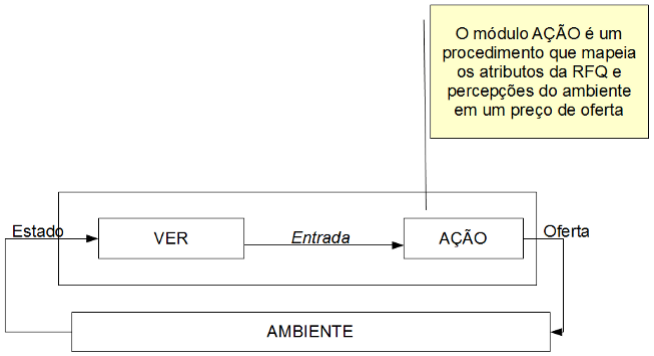


Figura 5.11: Estrutura do agente de tipo de produto.

O módulo VER identifica os atributos de cada RFQ juntamente com informações disponíveis no ambiente e gera uma representação dos mesmos em termos de um vetor de números reais. Por exemplo, seja uma RFQ no dia D com preço de reserva PR para um produto do tipo i, especificando Q unidades do produto a serem enviadas até uma data de entrega DE. Além disso, é informado: uma penalidade P por dia de atraso na entrega, a demanda total (DT) de produtos no dia D, o preço mais baixo (LP) e o preço mais alto (HP) pago por tipo de produto no dia $D-1$. Essas informações são resumidas em um vetor, como mostrado na Equação 5.6.

$$Entrada = (D, DE, Q, PR, P, LP, HP, DT) \tag{5.6}$$

O módulo $AÇÃO$ mapeia o vetor resultante do módulo VER em um preço de oferta para a RFQ recebida. Três versões do módulo ação dos ATPs foram concebidas e avaliadas: (1) procedimento baseado em *AdaBoost.M1*, (2) uma heurística simples e (3) redes neurais com KNN e heurística. A seguir, as três versões do módulo $AÇÃO$ são apresentadas. A estratégia com *AdaBoost.M1* foi denominada de E1; estratégia utilizando somente heurística simples foi denominada de E2 e a estratégia com redes neurais de múltiplas camadas, KNN e heurística, foi denominada E3. O módulo de definição de preços nos ATPs foi denominado *bidder*.

5.7.2 Estratégia com *AdaBoost.M1* (E1)

A primeira abordagem experimentada foi baseada no trabalho de Pardoe e Stone (2005). Nessa abordagem com aprendizagem, o problema de definir o preço de oferta consiste em determinar a probabilidade de uma oferta com preço correspondente a uma fração do **preço base**[1] ser

[1] O preço base é um preço de referência por meio do qual o preço de reserva especificado por clientes em RFQs é definido. O preço base é constante e definido no inicio do jogo para cada tipo de produto.

aceita pelo cliente de uma dada RFQ. Contudo, a estratégia definida por Pardoe e Stone (2005) utiliza aprendizagem para gerar um único modelo de previsão para todos os tipos produtos. Esta estratégia não se mostrou adequada para o caso distribuído, que dispõe de um agente para cada tipo de produto, executando possivelmente em diferentes máquinas. Por isso, a estratégia utilizada neste trabalho foi modificada de modo a gerar um modelo para cada tipo de produto, sem definir o tipo de produto como entrada no modelo. No caso da estratégia originalmente definida por Pardoe e Stone (2005), o tipo de produto era definido como um atributo de entrada. A seguir, a estratégia modificada é apresentada.

Dado um preço x, determinar a probabilidade de x ser aceito pelo cliente é equivalente a determinar a probabilidade do preço ofertado pelos outros agentes serem superiores a x, dado que os clientes sempre selecionam o menor preço de oferta. Se há logs de jogos passados com cada entrada contendo os atributos do vetor da equação 5.6 e o preço de oferta dado por outros agentes em resposta à RFQ representada pelos dados da entrada, é possível extrair dos dados no log uma hipótese sobre o preço de oferta dos agentes.

Na estratégia com *AdaBoost.M1*, diferentes frações do preço base são consideradas e uma hipótese é aprendida para cada fração, especificando a probabilidade do preço correspondente a esta fração ser aceito pelos clientes. Estas frações são definidas por um intervalo $I = [f_{inicio}, f_{fim}]$ e um incremento δ, por meio do qual n partições do intervalo I são geradas. O tamanho de uma seqüência F é dada pela função $\eta(F)$. Por exemplo, para $I = [0.9, 1.05]$ e $\delta = 0.05$, a seqüência de partições considerada é $F = (0.9, 0.95, 1.0, 1.05)$, com $\eta(F) = 4$. Sendo $p_{b,c}$ o preço base associado a um produto do tipo c, o preço de oferta correspondente a uma fração f_i é dado por: $f(i) \cdot p_{b,c}$.

Neste trabalho, foi considerada a seqüência de frações definida pelo intervalo $[0.55, 1.25]$ com $\delta = 0.05$, resultando na seqüência:

$$F = (0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.90, 0.95, 1.0, 1.05, 1.10, 1.15, 1.20, 1.25)$$

Essa sequência foi escolhida com base no trabalho de Pardoe e Stone (2005) e no fato de que preços abaixo de $0.55 \times p_{b,c}$ não possibilitarem lucros e preços acima de $1.25 \times p_{b,c}$ terem probabilidade próxima a zero de serem aceitos pelos clientes. A Figura 5.12 mostra como uma sequência de preços de oferta foi gerada para um produto com preço base igual 1650.

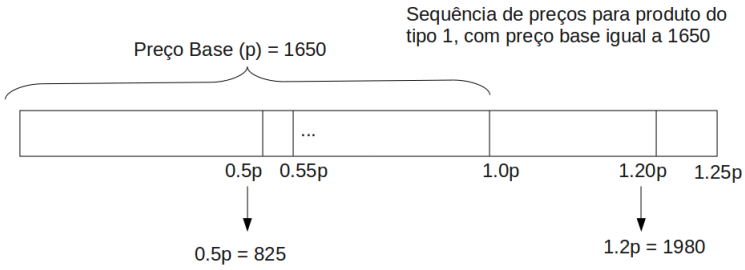


Figura 5.12: Sequência de preços de oferta gerada para um produto com preço base igual a 1650.

Considerando um produto c, para cada fração f_i na seqüência F é calculada a probabilidade do preço de oferta vencedor (aceito pelo cliente) ser maior do que $f_i \cdot p_{b,c}$, represen-

tando essa probabilidade por $P(f_i, p_{b,c})$. Isso é equivalente a aprender a função de densidade de probabilidade considerando intervalos com limites superiores sendo preços determinados pelas partições f_i. Para aprender a função objetivo $P(f_i, p_{b,c})$, foi utilizado um método de combinação de classificadores. Os atributos tomados como entrada para o método de aprendizagem foram variáveis do ambiente TAC-SCM disponíveis durante o jogo e informações percebidas e manipuladas pelos ATPs durante os jogos. Estas informações são apresentadas na Tabela 5.3.

data	Data de laçamento do RFQ
Prazo	Prazo de entrega exigido pelo cliente que enviou a RFQ
Penalidade	Penalidade por dia de atraso na entrega
Preço de Reserva	Preço máximo que o cliente está disposto a pagar pelo produto
Quantidade	Quantidade de computadores requisitados na RFQ atual
Demanda total	Quantidade total de computadores solicitados no dia atual
Preços máximos mínimos	Os preços mais altos e os preços mais baixos de venda nos últimos cinco dias, dando um total de 10 atributos

Tabela 5.3: Atributos selecionados para treinamento no processo de aprendizagem utilizando o algoritmo *AdaBoost.M1*.

Deve ser observado que é gerado para cada partição um modelo que calcula a probabilidade do preço resultante da partição ser menor que o preço de oferta de outros agentes. Este modelo é uma hipótese P' para a função objetivo P. Assim, haverá $\eta(F)$ modelos diferentes utilizados pelos agentes de tipo de produto (ATP) para configurar as ofertas com preços que são promissores em termos de probabilidade de aceitação e lucro. Durante a execução do sistema, no total, há $k \cdot \eta(F)$ modelos sendo utilizados por todos os k ATPs. De posse dos modelos necessários, cada ATP executará o algoritmo mostrado na Figura 5.13, sendo que a hipótese P' foi aprendida por meio do algoritmo *AdaBoost.M1*.

O algoritmo na Figura 5.13 seleciona a partição f_i com uma probabilidade igual a $P'(f_i, p_{b,c})$. Uma vez selecionada a partição f_i, o preço correspondente $f_i \cdot p_{b,c}$ é dado como o preço de oferta (quando o algoritmo termina o laço de repetição que começa na linha 1). Na linha 2, é escolhido o mínimo entre o preço de reserva e o preço de oferta selecionado no passo anterior, pois o preço de reserva é o valor máximo que o cliente está disposto a pagar pelo produto.

Variáveis:

r = uma variável que armazena um número real gerado uniformemente no intervalo $[0,1)$

P' = uma hipótese de P gerada por meio do algoritmo AdaBoost

c = identificação do tipo de produto representado pelo agente de tipo de produto em execução

pr = preço de reserva

Início:

1. Para $i = \eta(F)$ a 1, faça
   (a) $oferta = p_{b,c}$
   (b) gerar um número aleatório entre $[0,1)$ e atribui a r
   (c) Se $r < P'(f_i, p_{b,c})$ então
      i. $oferta = f_i \times p_{b,c}$
      ii. Sair do laço
   (d) Fim Se
2. Fim Para
3. $oferta = min(pr, oferta)$

Fim

Figura 5.13: Procedimento baseado no algoritmo AdaBoost.M1 para definição de preços de oferta do agente de tipo de produto.

O treinamento ocorreu com base em logs de jogos passados. A geração dos exemplos ocorreu de acordo com o procedimento da Figura 5.14. Para cada fração possível de preço base, foi gerado um conjunto de exemplos correspondente.

Inicio:

Dado uma fração do preço base $f_i \cdot p_{b,c}$

Para cada entrada (vetor de entrada) e saída (menor preço de oferta dos agentes da competição) faça

Se o melhor preço de oferta dos competidores for maior do que $f_i \cdot p_{b,c}$ Então

rotule a entrada com 1.0

Caso contrário

rotule a entrada com 0.0

Fim Se

Fim Para

Fim

Figura 5.14: Procedimento para geração de exemplos de treinamento para a estratégia baseada em AdaBoost.M1.

Em uma dada entrada do conjunto de treinamento, se o preço de oferta dos competidores for maior do que a fração do preço base correspondente, considera-se que a probabilidade de a fração do preço base considerada ser menor do que o preço de oferta de outros competidores é igual a 100% e a entrada é rotulada com 1; caso contrário, a entrada é rotulada com

0. Depois, os exemplos com entradas rotuladas são utilizados para geração da hipótese sobre o preço de oferta.

5.7.3 Estratégia com Heurística Simples (E2)

Por meio de heurísticas simples é possível obter bons resultados, inclusive complementando técnicas de aprendizagem de máquina. Heurísticas tem sido desenvolvidas baseadas no relatório de preços gerado diariamente durante o jogo, como no trabalho de Stan, Stan e Florea (2006), Chatzidimitriou et al. (2008). Neste trabalho, foi desenvolvida uma heurística baseada nesse relatório de preços. As únicas informações contidas nesse relatório são o maior e o menor preço pago por PC no dia anterior, para cada tipo de PC. Apesar dessa informação ser limitada, ela fornece informações recentes sobre o preço de oferta, definido em um intervalo com limites inferior e superior definidos por pl_c e ph_c, o menor e o maior preço, respectivamente, pago por PC do tipo c no dia $d-1$. No dia d, um agente considera uma distribuição de preços no intervalo $I = [pl_c, ph_c]$ para um produto do tipo c. Essa distribuição dependente não somente das condições de mercado atual, mas dos outros agentes no ambiente. Neste trabalho foi adotada a hipótese de que os preços de oferta mais promissores no dia d estão distribuídos uniformemente no intervalo I. Assim, considerando um preço de oferta po_c, para um produto do tipo c, supondo uma distribuição de probabilidade uniforme no intervalo I, é obtida uma hipótese sobre a probabilidade de po ser aceito como:

$$P'(po) = \frac{(ph_c - po)}{(ph_c - pl_c)} \tag{5.7}$$

A Equação 5.7 constitui uma hipótese bastante fraca quando é levado em conta apenas o relatório de preços do dia anterior. Portanto, a hipótese final no dia d, P'', foi estimada por meio da média ponderada de P' nos cinco dias anteriores. Dado os vetores de preço $vl_c = \langle pl_1^c, pl_2^c, ..., pl_5^c \rangle$ e $vh = \langle ph_1^c, ph_2^c, ..., ph_5^c \rangle$, onde pl_i^c e ph_i^c especificam o menor e o maior preço, respectivamente, pago por produto do tipo c no dia $d-i$, onde d é o dia atual no jogo. Assim, foi obtido:

$$P''(po) = \frac{\sum_{i=n}^{1} (P'_{(i)}(po) \times (n-i+1))}{\sum_{i=n}^{1} (n-i+1)} \tag{5.8}$$

onde: $P'_i(po) = \frac{(ph_i^c - po)}{(ph_i^c - pl_i^c)}$. Nesse caso, se $po > ph_i^c$, $P''_i(po) = 0$ e se $po < pl_i^c$, $P''_i(po) = 1$. A hipótese de que P'' provê uma boa estimativa da probabilidade de aceitação do preço po se mostrou satisfatória quando comparada com resultados obtidos com aprendizagem de máquina.

5.7.4 Estratégia com Redes Neurais, KNN e Heurística (E3)

Uma terceira estratégia consistiu no uso de redes neurais para prever o preço de oferta. A rede foi treinada por meio do algoritmo *backpropagation*. Foram utilizadas 16 redes neurais, uma para cada tipo de produto prevendo o melhor preço de oferta para RFQs de clientes. Os atributos selecionados como entrada da rede são aqueles mostrados na Tabela 5.4. A saída

consiste na previsão do menor preço de oferta que outros competidores poderiam enviar.

data (x_1)	Data de laçamento da RFQ
Prazo (x_2)	Prazo de entrega exigido pelo cliente que enviou a RFQ
Penalidade (x_3)	Penalidade por dia de atraso na entrega
Preço de Reserva (x_4)	Preço máximo que o cliente está disposto a pagar pelo produto
Quantidade (x_5)	Quantidade de computadores requisitados na RFQ atual
Demanda total (x_6)	Quantidade total de computadores solicitados no dia atual
Preço máximo (x_7)	O maior preço de pedido no dia anterior
Preço mínimo (x_8)	O menor preço de pedido no dia anterior

Tabela 5.4: Atributos selecionados para entrada da rede na previsão do preço de oferta.

A topologia das redes utilizadas é ilustrada na Figura 5.15, escolhida de forma empírica. Além da hipótese aprendida com redes neurais, foi utilizado o algoritmo KNN para verificar de modo *online* se a hipótese representada pela rede neural era confiável nas condições do momento da verificação.

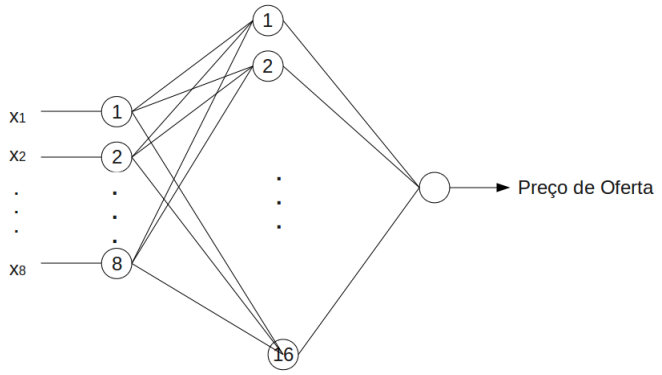


Figura 5.15: Topologia da rede neural utilizada para predição do melhor preço de oferta.

O algoritmo KNN consistia em uma base de exemplos das 100 últimas ofertas enviadas por um ATP. Cada oferta era representada por um vetor de números reais contendo os valores da RFQ enviada e o preço de oferta definido em resposta à RFQ. Como, para cada oferta enviada, o agente tem a retro-alimentação se uma oferta foi aceita ou não pelo cliente, o vetor é rotulado com o valor 1 caso a oferta correspondente tenha sido aceita pelo cliente com o valor -1 em caso contrário. O vetor é representado por $\mathbf{x} = \langle x_1, x_2, ..., x_n, oferta \rangle$ contendo os dados de uma RFQ e o preço de oferta escolhido pelo agente. Os valores dos atributos sobres as RFQs x_1, x_2, x_n são conseguidos no dia d, extraídos da própria RFQ; o preço de oferta ($oferta$) é determinado pelo agente. O rótulo de cada instância (contendo 1 para sucesso e -1 para fracasso) é conseguido no dia $d+1$ a partir da confirmação de aceitação da oferta por parte dos clientes (ou seja, da realização de pedidos de clientes). As ofertas que não receberem pedidos são marcadas e as instâncias que as representam são rotuladas com -1.

Dada uma nova instância x', com dados de uma RFQ recebida e com o preço de oferta gerado pela rede neural, um ATP verifica por meio do algoritmo KNN se esta instância será

aceita ou não. Seja $oferta$ o preço de oferta previsto pela rede neural. O agente utiliza o algoritmo KNN para verificar para uma dada RFQ se o preço $oferta$ é aceito ou não. Caso o algoritmo KNN retorne 1, o preço de oferta é configurado com o valor previsto pela rede neural. Caso o algoritmo KNN retorne -1, significa que o preço de oferta não é promissor. Nesse caso, é utilizado o menor preço entre o preço gerado pela heurística simples e o preço gerado pela rede neural. O algoritmo na Figura 5.16 resume a estratégia E3.

Variáveis e notações:

Algoritmo KNN: foi configurado com $k = 1$ e utilizando como exemplos as 100 últimas ofertas dos últimos dias no jogo em andamento.

P' = hipótese representada por uma rede neural treinada com o algoritmo de retropropagação do erro.

P'' = heurística simples representada pela equação 5.8.

s = vetor de entrada, contento valores para os atributos mostrados na Tabela 5.4.

Início

1. Seja P' a hipótese (rede neural) aprendida por meio do algoritmo *backpropagation*,
2. P'' a heurística simples e
3. KNN um classificador KNN que recebe a entrada **x** e retorna 1 se o preço for aceito e -1 se o preço não for aceito
4. O preço de oferta é obtido: $oferta = P'((s))$
5. Dado a entrada $\mathbf{x} = \langle x_1, x_2, ..., x_n, oferta \rangle$
6. Se $KNN(\mathbf{x}) \neq 1$ então
   (a) $ofertaSimples = P''(\mathbf{x})$
   (b) Se $ofertaSimples < oferta$ então $oferta = ofertaSimples$
7. retorna oferta

Fim

Figura 5.16: Procedimento com redes neurais, KNN e heurística simples para definição de preços de oferta do agente de tipo de produto.

5.8 Agente de Produto ou *Intelligent Product* (IP)

O objetivo de agentes do tipo IP é o gerenciamento de pedidos. Deve-se observar que os agentes do tipo IP integram os diferentes tipos de interações: central-vendas, central-compras, central-produção e central-entrega. O produto das interações central-compras é um pedido. E para cada pedido enviado por cliente, os agentes de tipo de produto (ATP) instanciam um agente do tipo IP que concorre com outros agentes do tipo IP por recursos para produção. Os recursos pelo quais agentes do tipo IP concorrem são componentes obtidos de fornecedores por meio das interações central-compras e ciclos de produção negociados durante as interações central-produção. Os agentes poderiam concorrer por recursos para entrega, mas como no TAC-SCM não tem limitação na capacidade de entrega, todos os agentes conseguem enviar

seus pedidos assim que conseguem os produtos necessários para isso e solicitam o envio destes produtos aos AE.

A estratégia de gerenciamento de pedidos foi baseada no trabalho de Meyer e Wortmann (2009). Contudo, algumas modificações foram realizadas. No trabalho de Meyer e Wortmann (2009) há três leilões: de componentes, ciclos de produção e de entrega. Como o leilão de entrega é trivial, já que sempre solicitações de entrega são aceitas, este leilão foi removido do modelo. A Figura 5.17 ilustra o processo que vai desde a instanciação de agentes de produto até o gerenciamento de leilões pelos agentes AP, ATC e AE. Nessa ilustração, as numerações indicam a ordem em que as mensagens são enviadas.

Deve ser observado que a ilustração mostra apenas um agente de cada tipo, de modo a simplificar a apresentação. Contudo, de fato, há 16 agentes de tipo de produto (ATP) em execução recebendo pedidos e muitos agentes de produto são instanciados por cada ATP, tantos quantos são os pedidos recebidos. Outra observação é que os ATPs apenas consideram pedidos para os tipos de produtos que representam. Por exemplo, o ATP para o tipo de produto 2 ignora pedidos recebidos para produtos de outros tipos.

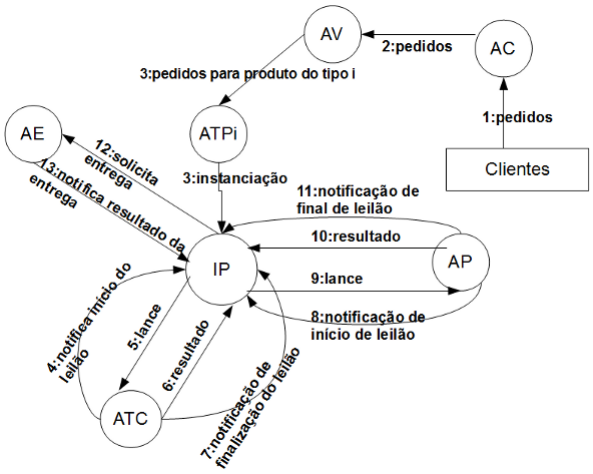


Figura 5.17: Atividades do gerenciamento de pedidos.

Após serem instanciados, os agentes de produto enviam lances por recursos para os leilões de componentes e produção. O preço de oferta é especificado de acordo com a Equação 5.9.

$$oferta(i) = \left(\frac{penalidade_i}{maxPenalidade}\right) - \left(\frac{e_i - d}{max_p}\right) \tag{5.9}$$

onde $oferta(i)$ é preço de oferta por unidade de componente ou ciclo de produção para um pedido i, e_i é o dia de envio do pedido i, d é o dia atual, $penalidade_i$ é penalidade paga por dia de atraso na entrega do pedido i, max_p é o valor máximo de $e_i - d$ (que no jogo TAC-SCM é igual a 12) e $maxPenalidade$ é o valor máximo da penalidade, especificado no início do jogo. Todos os valores são normalizados no intervalo $[0,1]$. A Tabela 5.5 mostra vários exemplos de preços de oferta para componentes a partir dos atributos de pedidos.

e_i	d	$penalidade_i$	max_p	$maxPenalidade$	$oferta(i)$
12	10	4000	12	8000	0,333
12	10	5000	12	8000	0,458
11	10	4000	12	8000	0,416
11	10	5000	12	8000	0,541

Tabela 5.5: Exemplo para mostrar como os preços de oferta são influenciados pelos valores das variáveis da Equação 5.9.

5.8.1 O Programa do Agente IP

O programa de um agente do tipo IP é baseado em um agente reativo simples, exceto que o módulo *AÇÃO* é representado pelo algoritmo ilustrado na Figura 5.18. Nota-se que a participação nos leilões por agentes do tipo IP é seqüencial, ou seja, primeiro agentes do tipo IP participam de leilões de componentes. Quando conseguem componentes necessários para a produção de PCs para suprir o pedido que representam, iniciam a participação em um leilão de ciclos de produção, enquanto não conseguem a quantidade necessária de ciclos de produção para suprir o pedido que representam, continuam enviando lances no leilão de ciclos de produção. Por último, participam de leilões de entrega. Neste caso, na primeira tentativa sempre conseguem o direito de envio de PCs para suprir o pedido que representam.

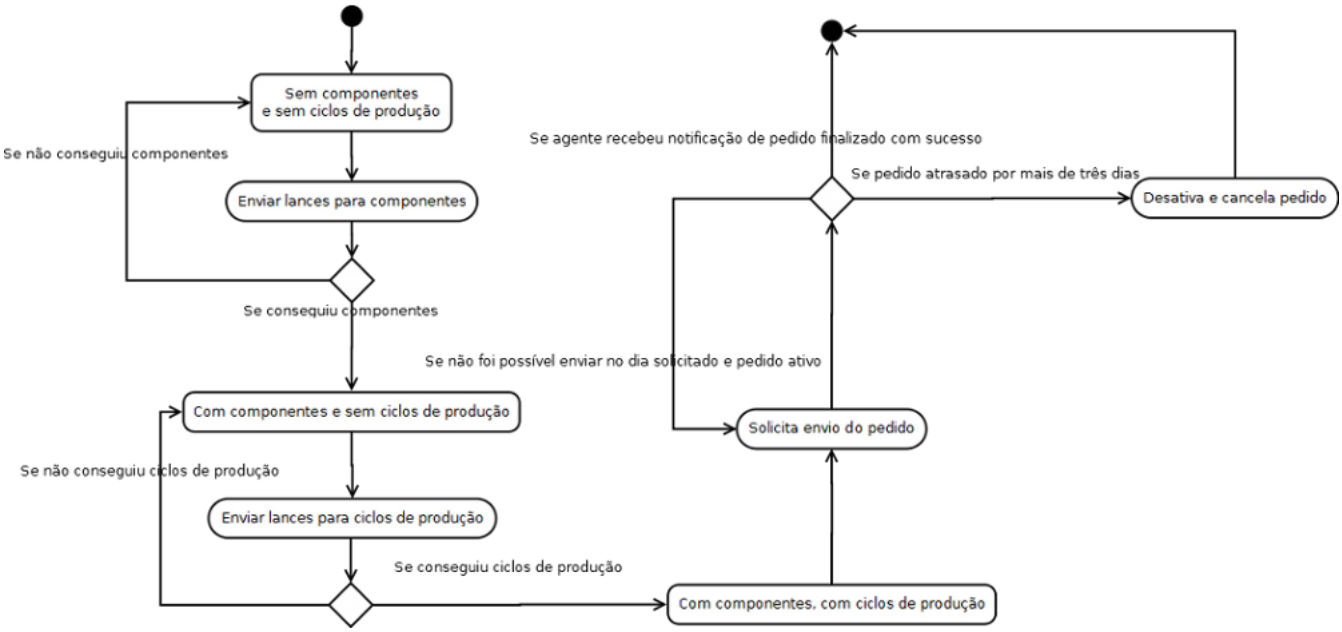


Figura 5.18: Programa de um agente do tipo IP.

Os leilões contemplam os vencedores reservando os recursos leiloados. Isso significa que estes recursos não serão mais disponibilizados em leilões futuros. Os vencedores (agentes do tipo IP que representam pedidos específicos) são notificados que os recursos adquiridos foram reservados e eles, portanto, podem determinar como tais recursos devem ser utilizados. Como se pode observar, todos os leilões são leilões de demanda, vencendo o participante que

fizer o maior lance de demanda (MASILI, 2004). Também é um tipo de leilão fechado que fica aberto durante um intervalo de tempo e os lances são enviados no intervalo de tempo determinado sem um participante conhecer o lance de outros participantes.

5.9 Considerações Finais

A abordagem apresentada nesta seção não se limita às estratégias e heurísticas apresentadas. De fato, os módulos de decisão e ação dos agentes poderiam evoluir em diferentes refinamentos. No que diz respeito à implementação atual da abordagem, no próximo capítulo é realizada uma avaliação das técnicas de aprendizagem utilizadas e do desempenho global de três versões implementadas, consistindo em diferentes implementações do módulo ação dos agentes de tipo de produto, descrito na Figura 5.18.

6 Avaliação, Experimentos e Resultados

Neste capítulo foram avaliadas as diferentes versões da estratégia multi-agente concebida. A validação do sistema ocorreu em duas etapas. A primeira etapa foi a validação dos métodos de aprendizagem de máquina (das hipóteses geradas) e da heurística simples de predição de preços. A segunda etapa consistiu na avaliação do SMA desenvolvido em diferentes versões em diferentes cenários de perturbações no processo de produção. Métricas como o erro quadrático médio (Root Mean Squared Error - RMSE) e matrizes de confusão foram utilizadas para comparar os diferentes métodos de aprendizagem de máquina. Para avaliar o desempenho das diferentes versões do SMA desenvolvido, foram consideradas várias simulações contra um conjunto de agentes considerados competitivos. Foram utilizados valores médios e técnicas estatísticas, como o teste de hipótese, para comparar diferentes resultados obtidos.

Os dados obtidos durante a fase de treinamento das redes neurais e de uso do algoritmo *AdaBoost.M1* foram extraídos por meio de componentes de software de terceiro e derivações projetadas especificamente para este trabalho, mas que podem ser reaproveitadas em trabalhos futuros. Para extrair informações de log, foi utilizada o software *The TAC SCM Game Data Toolkit*[1]. Além disso foi utilizada a ferramenta de análise *CMieux Analysis and Instrumentation Toolkit*[2] (BENISCH et al., 2005).

Este capítulo é organizado como segue: na Seção 6.1, é apresentado o ambiente de simulação utilizado; na Seção 6.2, são apresentadas as metodologias de avaliação dos métodos de aprendizagem utilizados; na Seção 6.3 é apresentado o resultado da avaliação dos métodos de aprendizagem; nas seções seguintes, os resultados obtidos pelo SMA UECEGRUNN em diferentes cenários são apresentados.

6.1 Ambiente de Simulação

Como mostra a Figura 6.1, o SMA UECEGRUNN foi implementado sobre uma arquitetura baseada na linguagem de programação Java, utilizando comunicação a nível de *sockets* para prover comunicação de baixo nível em rede entre os agentes do sistema. As implementações dos métodos de aprendizagem de máquina utilizados vieram de duas fontes: IASolver[3] e Weka[4].

[1] http://www.sics.se/tac, último acesso em 20 de março de 2010.
[2] http://www.cs.cmu.edu/ mbenisch/ait, último acesso em 20 de março de 2010.
[3] http://gilzamir.computacaosobral.net.br/softwares, último acesso em 20 de março de 2010.
[4] http://www.cs.waikato.ac.nz/ml/weka, último acesso em 20 de março 2010.

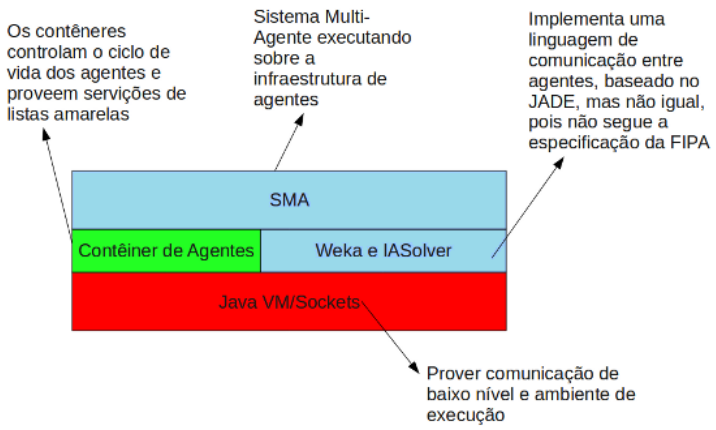


Figura 6.1: Infraestrutura de execução do SMA UECEGRUNN.

O ambiente de execução das simulações consistiu em uma versão modificada do servidor disponibilizado pela comunidade que organiza o TAC, o mesmo servidor utilizado nos experimentos descritos por Meyer e Wortmann (2009). O servidor do TAC, para a versão utilizada neste trabalho, foi modificado de modo a gerar a probabilidade de $P\%$ de uma unidade de componente recebido de fornecedores não poder ser utilizada na produção, onde $P \in \{0, 5, 10\}$.

Para garantir um ambiente de simulação com poder computacional suficiente para suportar vários agentes executando paralelamente, foram utilizados três computadores com as configurações mostradas na Tabela 6.1.

Tabela 6.1: Configuração do ambiente de simulação, descrevendo a configuração dos computadores e a localização dos agentes ou servidores entre os computadores disponibilizados.

Computador/Configuração	Memória	Processador	Agentes ou Servidor
Computador I	1.2 GB	Intel(R) Celeron(R) CPU 2.13 GHz	Servidor
Computador II	1.5 GB	AMD(R) Sempron(TM) 2600+	UECEGRUNN, Deep Maize e Dummy
Computador III	2 GB	Intel(R) Core(TM)2 Duo CPU T5550 1.83 GHz	TacTex, Phant, Mertacor

Os agentes foram instalados em dois computadores e o servidor em um terceiro computador, de modo a balancear a carga de execução: consumo de memória e de processamento. Para testar especificamente as estratégias de definição de preço de oferta, foi desenvolvido um simulador de vendas, descrito como segue.

6.1.1 O Simulador de Vendas

O simulador de vendas foi desenvolvido de modo a isolar do agente o problema de definição do preço de oferta dos outros problemas, como obter componentes, programar pro-

dução e entrega de pedidos. O simulador recebe como entrada *logs* de jogos passados e executa as seguintes etapas:

1. Obtém as informações de cada dia simulado (conforme apresenta a Tabela 6.2)
2. Para cada dia simulado, envia as informações obtidas para o agente *bidder* e obtém o preço de oferta do *bidder*. No caso, *bidder* é um módulo de definição de preços utilizado nos agentes de tipo de produto (ATPs).
3. Após passar todos os logs, gera um relatório do desempenho do agente com as informações: taxa de sucesso (número total de pedidos obtidos sobre o número de RFQs recebidas) e RMSE (erro quadrático médio).

Tabela 6.2: Dados enviados diariamente ao agente responsável pela definição do preço de oferta.

Informação	Descrição
Dia	Dia atual
Prazo de entrega	Dia de entrega exigido pelo cliente
Quantidade	Quantidade de produtos requisitados na RFQ atual
Preço de Reserva	Preço de reserva especificado na RFQ
Penalidade	Penalidade por dia de atraso na entrega
Menor preço	Menor preço pago pelo produto no dia anterior
Maior preço	Maior preço pago pelo produto no dia anterior
Demanda	Quantidade total de produtos requisitados no dia atual

6.2 Metodologia de Aprendizagem e Testes

Os métodos de aprendizagem de máquina foram concebidos e validados em três etapas: (1) treinamento, (2) validação e testes utilizando métricas como RMSE e matriz de confusão e (3) avaliação em um ambiente simulado com dados de jogos não presentes nos exemplos de treinamento.

Para validar as técnicas de aprendizagem de máquina, foram geradas hipóteses por meio de métodos de aprendizagem de máquina tomando como entrada um conjunto com 9553 exemplos. A maior dificuldade foi analisar a estratégia E1 (baseada *em AdaBoost.M1*), pois para esta estratégia há 15 hipóteses geradas, uma para cada limite de partição que representa uma fração do preço base do produto. A validação da estratégia combinando redes neurais, KNN e heurística foi realizada por meio apenas da métrica RMSE, pois se trata de um problema de regressão. Todos os métodos foram testados posteriormente por meio do simulador de vendas, que gera as mesmas condições de venda de jogos realizados anteriormente. Para gerar essa simulação, são utilizados os dados dos jogos armazenados em *logs*. Os dados para geração da simulação das condições de venda foram extraídos de *logs* não utilizados no treinamento dos modelos de aprendizagem de máquina.

Nas próximas seções, as etapas (1), (2) e (3) são apresentadas.

6.2.1 Treinamento

Os dados de treinamento foram selecionados a partir de logs disponibilizados pelo simulador do TAC-SCM. Os dados foram pré-processados e os valores de atributo selecionados foram extraídos. Além disso, algum trabalho extra foi realizado para reduzir a influência de ruídos e informações indesejadas. Por exemplo, utilizar dados dos primeiros 50 dias simulados e dos últimos 20 dias de cada simulação pode resultar em um modelo não adequado, pois os exemplos correspondentes a estes intervalos são influenciados por comportamentos diferenciados de agentes em momentos específicos do jogo. No final do jogo, por exemplo, um agente poderia vender produtos em estoque a um preço inferior do que faria normalmente, isso com o objetivo de terminar o jogo com poucos ou possivelmente nenhum item em estoque.

Os dados foram normalizados de acordo com a equação:

$$a'_i = \frac{a_i - max(a_i)}{max(a_i) - min(a_i)} \tag{6.1}$$

onde a'_i é o valor normalizado do atributo identificado por a_i, $max(a_i)$ é o maior valor do atributo a_i no conjunto de treinamento e $min(a_i)$ é o menor valor do atributo a_i no conjunto de treinamento.

Dessa forma, foram executados dez jogos contendo quatro agentes considerados competitivos: Mertacor, TacTex, DeepMaize e Phant. Os dados gerados foram pré-processados de acordo com as considerações anteriores. Utilizando dados destes dez jogos, foram gerados 9553 exemplos.

Witten e Frank (2005) discutem as problemáticas de treinamento e teste e predição de desempenho para métodos de aprendizagem de máquina. O objetivo das estratégias de validação é prover uma estimativa do desempenho de métodos de aprendizagem de máquina para casos além dos exemplos de treinamento.

6.2.2 Metodologia de Validação e Testes

Existem diferentes formas de validação de modelos de aprendizagem. Os dois métodos mais utilizados são os métodos *hold out* e o método de validação cruzada. O método *hold out* é normalmente utilizado quando se possui uma grande quantidade de dados. Neste procedimento, uma maior parte dos dados é usada como dados de treinamento e o restante é usado como dados de teste.

O método de validação cruzada é comumente usado quando a quantidade de dados disponível é pequena. Ele consiste na divisão de todos os dados em n subconjuntos de mesmo tamanho. Neste caso, o algoritmo é aplicado n vezes sendo que em cada vez um dos subconjuntos é reservado para teste e os outros subconjuntos são utilizados para treinamento. No fim dessas iterações todos os dados terão sido utilizados para treinamento e para teste. A taxa de acerto (validade) de uma regra será, então, a média das taxas de acerto encontradas nas n diferentes iterações.

Neste trabalho foi utilizada validação cruzada, separando o conjunto de treinamento em partes ou *folds*. Após a separação, 90% dos dados, ou $9/10$ das partes, foi utilizado para treinamento e o restante, 10%, para testes. Esse conjunto de técnicas utilizadas para o treinamento e teste dos métodos de aprendizagem considerados é similar à técnica de validação cruzada estratificada *n-fold*, nesse caso *10-fold*.

O treinamento dos métodos de aprendizagem baseados nas estratégias descritas foi realizado, de um modo geral, de acordo com as seguintes etapas (FEITOSA, 2009):

- O conjunto de treinamento construído (os dados conhecidos) foram separados em 10 partes, preocupando-se com a igual representação das classes de dados em cada uma das partes.
- As partes foram reagrupadas em 10 subconjuntos diferentes, cada um com 90% do total de exemplos. A parte restante, corresponde aos 10% não utilizados, foi utilizada para testes.
- Após a separação dos conjuntos, foram realizadas 10 baterias de treinamentos. Após cada treino, o conjunto remanescente foi testado.
- O treinamento ocorre por meio da apresentação repetitiva de um conjunto de exemplo ao supervisor, cada apresentação corresponde a uma época. O número de épocas foi escolhido de modo que os resultados pudessem convergir para um erro mínimo. No caso da aprendizagem com redes neurais, o número de épocas considerado foi de 200. Para o algoritmo *AdaBoost* foram consideradas 200 iterações. A ordem de apresentação dos exemplos foi aleatória, a cada época um nova ordem é escolhida aleatoriamente. Essa é uma estratégia comum para evitar uma eventual tendência no processo de aprendizagem.
- Finalmente, avaliou-se a eficiência das previsões obtidas por meio das redes neurais e do algoritmo *AdaBoost* utilizando as métricas de *Root Mean Squared Error* (RMSE), matriz-confusão e o erro médio.

6.3 Avaliação e Teste dos Métodos de Aprendizagem

6.3.1 Redes Neurais

A validação das redes neurais utilizadas foi realizada por meio da estratégia *10-folds*, discutida na Seção 6.2. É interessante notar que para cada tipo de produto (ver Tabela 2.2 para a lista completa de tipos) há uma rede neural que realiza a predição do preço para aquele produto. Portanto, é mostrado na Tabela 6.3 o resultado completo (um RMSE para cada *fold* ou partição) para o produto do tipo 1.

Tabela 6.3: Resultado do RMSE para cada *fold* no treinamento da rede neural para o produto do tipo 1.

Etapa do Treinamento/Teste	Valor do RMSE	Valor $*$ 100
1	0,064639	6,463
2	0,061539	6,153
3	0,055094	5,509
4	0,052108	5,210
5	0,055127	5,512
6	0,051839	5,183
7	0,055227	5,522
8	0,056438	5,643
9	0,058128	5,812
10	0,056689	5,668
Erro Global	0,056	5,668

A Tabela 6.4 mostra a média do RMSE ao longo dos testes para outros produtos. Assim, para o *fold* 1, o resultado é mostrado na linha que apresenta o valor 1 na primeira coluna. O valor global do RMSE é a média do RMSE para todos os *folds*.

Tabela 6.4: Resultado do RMSE médio global obtido pelas redes neurais para cada tipo de produto.

Produto	Valor do RMSE	Valor $*$ 100
1	0,566	5,660
2	0,681	6,810
3	0,544	5,440
4	0,0624	6,240
5	0,0563	5,630
6	0,0696	6,960
7	0,0469	4,690
8	0,0467	4,670
9	0,0396	3,960
10	0,0534	5,340
11	0,0390	3,900
12	0,0435	4,350
13	0,0599	5,990
14	0,0658	6,590
15	0,0475	4,750
16	0,0478	4,780

Além do RMSE do teste durante as etapas da validação cruzada, a rede neural foi testada em experimentos com o simulador de vendas. Nas simulações realizadas, foram utilizados 1294 exemplos de entrada. O erro quadrático médio (RMSE) obtido pelas redes neurais para cada tipo de produto durante a simulação de vendas é mostrado na Tabela 6.5.

Utilizando apenas redes neurais para prever o melhor preço de oferta, foram obtidos na simulação de vendas os resultados mostrados na Tabela 6.5, com uma taxa média de sucesso

de aproximadamente 75%. Isso significa que 75% das ofertas enviadas resultariam em pedidos de cliente.

Tabela 6.5: Desempenho por tipo de produto obtido quando apenas redes neurais são utilizadas durante a simulação de vendas.

Produto	RMSE	$RMSE \times 100$	Taxa de Sucesso (%)
1	0,04693	4,693	68,21
2	0,08045	8,045	79,93
3	0,04899	4,899	75,09
4	0,07286	7,286	72,86
5	0,04182	4,182	76,87
6	0,06334	6,334	77,31
7	0,03702	3,702	73,38
8	0,05958	5,958	86,43
9	0,08761	8,761	78,21
10	0,05128	5,128	65,75
11	0,05848	5,848	56,17
12	0,04210	4,210	85,79
13	0,04075	4,075	75,87
14	0,04537	4,537	66,74
15	0,03222	3,222	71,48
16	0,04040	4,040	79.11
Média	0,05308	5,308	74,92

Tabela 6.6: Desempenho obtido pela estratégia híbrida (que combina RNA, KNN e heurística) para cada tipo de produto durante a simulação de vendas.

Produto	RMSE	$RMSE \times 100$	Taxa de Sucesso (%)
1	0.0533	5,33	80,43
2	0.0848	8,48	84,56
3	0.0554	5,54	84,66
4	0.0782	7,82	87,87
5	0.0482	4,82	82,87
6	0.0675	6,75	81,50
7	0.0449	4,49	86,04
8	0.0578	5,78	90,55
9	0.0283	2,83	89,24
10	0.0572	5,72	76,35
11	0.0233	2,33	86,83
12	0.0449	4,49	90,56
13	0.0457	4,57	82,16
14	0.0520	5,20	77,65
15	0.0380	3,80	83,99
16	0.0438	4,38	86,99
Média	0.0515	5,15	84,52

6.3.2 Heurística Simples

A heurística simples apresentada na Seção 5.7 foi avaliada no ambiente simulado de vendas. Os resultados obtidos são mostrados na Tabela 6.7.

Tabela 6.7: Resultado do RMSE médio global para cada tipo de produto obtido pelo procedimento utilizando somente a heurística simples.

Produto	RMSE	RMSE $*$ 100	Taxa de Sucesso (%)
1	0,0527	5,27	65,04
2	0,0684	6,84	60,59
3	0,0545	5,45	72,45
4	0,0600	6,00	61,90
5	0,0463	4,63	68,65
6	0,0666	6,66	62,07
7	0,0434	4,34	76,92
8	0,0444	4,44	66,57
9	0,0240	2,40	72,53
10	0,0618	6,18	58,96
11	0,0226	2,26	79,17
12	0,0362	3,62	66,01
13	0,0429	4,29	70,27
14	0,0523	5,23	65,48
15	0,0360	3,60	75,29
16	0,0382	3,82	68,94
Média	0,0469	4,69	68,18

6.3.3 AdaBoost.M1

Análise da estratégia com *AdaBoost.M1* ou estratégia E1 apresenta algumas dificuldades. Para cada produto, de dezesseis produtos possíveis, há 15 modelos. Portanto, a avaliação é sobre 240 modelos. Essa é uma desvantagem importante da estratégia com AdaBoost.M1, pois há perda de desempenho e um maior consumo de memória na execução desses 240 modelos.

Uma primeira avaliação foi quanto ao erro quadrático médio (RMSE). A média global do RMSE obtida para cada tipo de produto na classificação de cada fração do preço base durante a etapa de treinamento e teste é mostrada na Tabela 6.8. Observe em cada coluna que o RMSE aumenta com o valor da fração do preço base até um limite, a partir do qual o valor do RMSE começa a diminuir novamente (até que fique igual a zero). Uma conclusão é que com uma fração do preço base muito baixa (por exemplo, $0,55$), aproximadamente 100% dos exemplos serão positivos, dado que a oferta a um preço muito baixo tem uma grande chance de ser aceita. Por outro lado, com um valor grande de fração do preço base (por exemplo, $1,25$), dificilmente um pedido seria aceito. Nesse caso, aproximadamente 100% dos exemplos obtidos seriam negativos. Portanto, devido ao desbalanceamento de classes, obtemos um RMSE aparentemente razoável nos limites do intervalo que determina as frações do preço base que serão consideradas como preço de oferta ao longo do jogo. Neste trabalho, a questão do balanceamento de classes

não foi abordado.

Tabela 6.8: RMSE médio obtido pelo algoritmo AdaBoost.M1 ao classificar cada fração do preço base como positivo (uma oferta a esta fração do preço base teria sucesso na obtenção de um pedido) ou negativo (uma oferta a esta fração do preço base falharia na obtenção de um pedido).

Fração/Produto	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0,55	0,08	0,06	0,05	0,05	0,14	0,15	0,16	0,13	0,07	0,09	0,08	0,10	0,08	0,11	0,11	0,11
0,60	0,12	0,10	0,12	0,10	0,17	0,17	0,14	0,15	0,18	0,15	0,16	0,15	0,17	0,14	0,16	0,15
0,65	0,15	0,15	0,15	0,15	0,18	0,18	0,15	0,18	0,20	0,17	0,19	0,16	0,18	0,18	0,17	0,15
0,70	0,20	0,18	0,16	0,15	0,20	0,17	0,17	0,16	0,19	0,18	0,18	0,17	0,22	0,18	0,16	0,15
0,75	0,26	0,20	0,20	0,17	0,21	0,19	0,20	0,19	0,20	0,19	0,18	0,19	0,25	0,21	0,24	0,21
0,80	0,29	0,24	0,21	0,21	0,29	0,25	0,26	0,25	0,23	0,22	0,23	0,22	0,26	0,25	0,24	0,26
0,85	0,30	0,27	0,24	0,21	0,27	0,27	0,24	0,26	0,23	0,24	0,20	0,23	0,27	0,29	0,22	0,25
0,90	0,25	0,28	0,25	0,24	0,26	0,24	0,24	0,22	0,22	0,25	0,23	0,24	0,24	0,27	0,22	0,23
0,95	0,17	0,29	0,28	0,27	0,24	0,22	0,23	0,24	0,22	0,24	0,22	0,24	0,21	0,25	0,22	0,23
1,00	0,11	0,22	0,21	0,27	0,20	0,20	0,19	0,21	0,19	0,22	0,20	0,22	0,16	0,20	0,17	0,20
1,05	0,08	0,14	0,13	0,18	0,12	0,15	0,13	0,17	0,13	0,16	0,16	0,18	0,12	0,15	0,14	0,15
1,10	0,06	0,09	0,08	0,12	0,08	0,09	0,08	0,11	0,08	0,10	0,12	0,12	0,10	0,13	0,11	0,10
1,15	0,03	0,05	0,05	0,08	0,06	0,06	0,04	0,07	0,06	0,06	0,07	0,07	0,08	0,08	0,07	0,07
1,20	0,00	0,03	0,02	0,04	0,03	0,03	0,03	0,03	0,03	0,03	0,04	0,04	0,05	0,05	0,03	0,03
1,25	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Média	0,15	0.15	0.14	0.149	0.16	0.157	0.15	0.157	0.148	0.15	0.15	0.155	0.159	0.165	0.15	0.15

Outra análise da estratégia de classificação foi quanto ao uso de matrizes de confusão. Uma matriz de confusão fornece uma medida eficaz do modelo de classificação de uma hipótese h, ao mostrar o número de classificações corretas *versus* as classificações preditas para cada classe, sobre um conjunto de exemplos D. Como na estratégia de classificação há 240 modelos ou hipótese, não é prático apresentar aqui as matrizes de confusão para cada modelo. Contudo, a fim de comparar o RMSE com medidas obtidas por meio da matriz de confusão, a Tabela 6.9 apresenta as matrizes de confusão para a fração do preço base $0,60$.

Tabela 6.9: Matriz de confusão para os *folds* de 6 a 10 obtidas por meio dos modelos de classificação de produtos do tipo 1 e com uma fração do preço base igual a 60%. Cada sub-tabela sob o rótulo de um *fold* correspondente à matriz de confusão quando os testes foram realizados para este *fold*

	fold 6		fold 7		fold 8		fold 9		fold 10	
Verdadeiras/Preditas	P	N	P	N	P	N	P	N	P	N
P	5493	78	6403	96	7311	116	8229	126	9149	134
N	33	129	38	151	43	173	49	194	50	220

Um aspecto a ser observado na matriz de confusão é quanto à disposição das informações. Os valores previstos pela hipótese estão dispostos verticalmente, enquanto a classificação real está disposta horizontalmente. Dessa forma, na diagonal principal de cada matriz há o total de acerto do modelo nas duas classes (positiva e negativa). Assim, a análise da hipótese gerada que classifica a fração de 60% do preço base revelou uma taxa de verdadeiros positivos igual a aproximadamente 98%, enquanto a taxa de verdadeiros negativos ficou em torno de 81%. Outra métrica utilizada foi a área sob a curva ROC (*Receiver Operating Characteristic curve*) (GöNEN, 2007). Para este caso, a área sob a curva ROC foi igual a 0.98, que sob um ponto de vista acadêmico clássico indica um excelente desempenho da hipótese.

Até aqui foram discutidos apenas alguns aspectos da avaliação dos modelos de aprendizagem em si, sem levar em conta o desempenho global da estratégia de classificação, que foi analisada na simulação de vendas, a partir de dados de um jogo que não estavam presentes nos exemplos de treinamento (assim como ocorreu com a avaliação das outras estratégias). A Tabela 6.10 mostra os resultados obtidos pela estratégia de classificação durante a simulação

de vendas. A taxa média de sucesso foi de $75,63\%$, aparentemente um pouco melhor do que o resultado obtido pelo uso da heurística simples, aproximadamente igual ao uso somente de redes neural e inferior à estratégia que combina redes neurais, KNN e heurística simples.

Tabela 6.10: Resultado do RMSE médio global para cada tipo de produto obtido pela estratégia E1, baseada no algoritmo *AdaBoost.M1*.

Produto	RMSE	RMSE $*$ 100	Taxa de Sucesso (%)
1	0,0888	8,88	72,54
2	0,1146	11,46	73,86
3	0,0951	9,51	75,63
4	0,1076	10,76	81,45
5	0,0796	79,60	76,39
6	0,1089	10,89	73,45
7	0,0723	72,30	77,26
8	0,0908	9,08	77,60
9	0,0495	4,95	75,84
10	0,0805	8,05	71,92
11	0,0523	5,23	74,35
12	0,0559	5,59	77,34
13	0,0787	7,87	76,26
14	0,0886	8,86	72,89
15	0,0690	6,90	76,74
16	0,0616	6,16	7662,
Média	0,0809	8,09	75,63

6.3.4 Considerações finais sobre as estratégias utilizadas

As estratégias de venda analisadas mostraram resultados promissores, embora mais investigação seja necessária para validação em outros contextos. O diferencial das abordagens apresentadas é que são aplicadas a cada produto separadamente, contudo levando em conta informações sobre todos os produtos. Por exemplo, como entrada para cada rede neural que representa cada um dos 16 possíveis produtos há a demanda total de produtos em um determinado dia. Além disso, foi possível obter resultados satisfatórios com exemplos de treinamento de poucos jogos (6 jogos). Estratégias que exigem muitos modelos, como a estratégia E1 podem se tornar inviáveis na prática. Na estratégia com *AdaBoost.M1*, para cada novo tipo de produto, seria necessário gerar 15 novos modelos ou hipóteses, algo que pode se tornar impeditivo a partir de uma grande quantidade de tipos de produtos. A Tabela 6.11 resume os resultados obtidos na simulação de vendas.

O resultado obtido pela estratégia E3 (que combina redes neurais, KNN e heurística), foi superior ao resultado obtido pelas outras estratégias. Um dos motivos é o fato desta estratégia utilizar o *feedback* sobre o resultado das ofertas mais recentes enviadas aos clientes.

O próximo passo é a análise do sistema concebido como um todo, para cada versão possível. Dado o custo de gerar simulações para todas as versões possíveis, as três seguintes

Tabela 6.11: Resumo dos resultados obtidos na simulação de vendas

Estratégia	Taxa de Sucesso (%)
Redes Neurais , KNN e Heurística	84,52
Estratégia de Classificação (com AdaBoost.M1)	75,63
Redes Neurais	74,92
Heurísticas Simples	68,18

estratégias foram selecionadas para análise em simulações envolvendo três cenários do gerenciamento de cadeias de suprimento: (1) estratégia utilizando somente a heurística simples (E2), (2) estratégia combinando redes neurais, KNN e heurística simples (E3) e (3) estratégia de classificação com *AdaBoost.M1* (E1). O motivo de escolher estas estratégias foi analisar o impacto que o uso do *feedback* de cliente em relação às ofertas enviadas teria em termos de desempenho. No caso, a estratégia com *feedback* de ofertas combina redes neurais, KNN e a heurística simples concebida. A heurística concebida apresentou desempenho satisfatório nas avaliações realizadas. O uso de *AdaBoost.M1* foi motivado pelos resultados apresentados por Pardoe e Stone (2005), que utilizaram métodos de combinação de preditores. O objetivo era analisar o desempenho de um algoritmo de combinação de preditores em um cenário distribuído. A implementação utilizada do algoritmo *AdaBoost.M1* foi a disponibilizada na biblioteca Weka.

Os três cenários nos quais cada estratégia foi avaliada são os cenários com 0%, 5% e 10% de probabilidade de componentes individuais não poderem ser utilizados na produção de PCs. Finalmente, os resultados obtidos pela versão do SMA UECEGRUNN com melhor desempenho são comparados com os resultados obtidos pelo sistema GRUNN.

6.4 Avaliação da Estratégia Multi-Agente

O sistema multi-agente desenvolvido foi avaliado com suas diferentes estratégias de venda. Os resultados obtidos foram comparados com resultados obtidos por outros agentes. O conjunto de agentes escolhido foi: TacTex, Phant, Mertacor e DeepMaize. Estes agentes foram escolhidos por terem ficado entre os primeiros colocados em competições recentes do TAC-SCM e, portanto, a presença destes agentes no mercado tornam o ambiente altamente competitivo. As três estratégias de venda concebidas para este trabalho foram comparadas em termos de lucro quando submetidas em jogos contra o conjunto de agentes competidores. Uma outra possibilidade seria em um mesmo jogo adicionar diferentes versões do mesmo agente. Contudo, segundo (SODOMKA; COLLINS; GINI, 2007), analisar o desempenho de agentes jogando contra si mesmo não é representativo de como o agente deveria executar em situações do mundo real. Além disso, foi observado que quando duas ou mais versões de um mesmo agente executam em um mesmo jogo, há um decremento no desempenho (HE et al., 2006).

Sodomka, Collins e Gini (2007) citam três diferentes métodos para se testar agentes:

- caso base: diferentes versões de um agente são testadas em conjuntos de simulações

independentes, ou seja, simulações geradas aleatoriamente. Contudo, alguns parâmetros das simulações são constantes, como o competidores contra os quais cada versão irá atuar.

- caso de lucro ajustado à demanda: similar ao caso base, exceto que níveis de lucro são ajustados de acordo com o nível da demanda no mercado. Isso significa que o lucro é ponderado em uma fator inversamente proporcional à demanda. Assim, agentes que lucram mais em momentos de baixa demanda são beneficiados se conseguem obter pelo menos o mesmo lucro de outros agentes quando a demanda esta alta.

- caso de mercado pareado: similar ao caso base, exceto que as condições de mercado são repetidas em conjuntos de simulações para as versões do agente que estão sendo avaliadas. Isso significa que os agentes são testados nas mesmas condições de mercado, havendo um controle sobre a definição de valores para parâmetros do jogo.

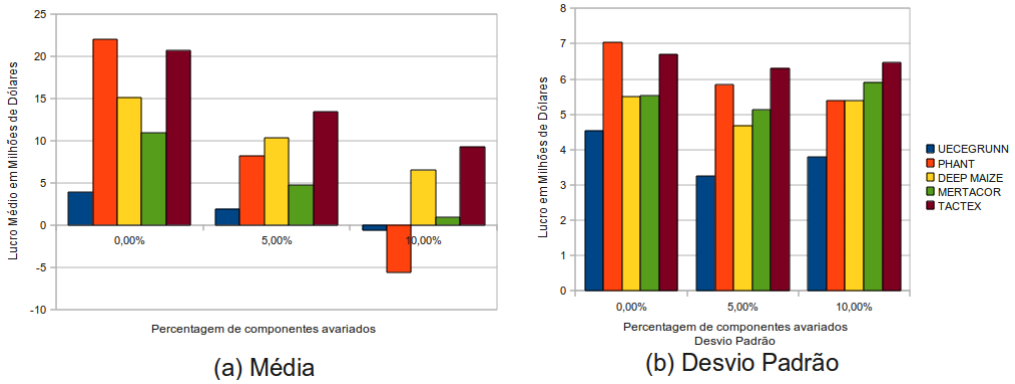
Neste trabalho foi adotado o caso base, pois nos outros dois casos são necessários simuladores especializados que não estavam disponíveis no momento em que as simulações foram realizadas. Cada versão do SMA UECEGRUNN foi testada em 84 simulações. As simulações foram divididas em três grupos de 28 simulações:

- grupo 1: simulações com 0% de componentes avariados
- grupo 2: simulações com 5% de componentes avariados
- grupo 3: simulações com 10% de componentes avariados

Nas próximas seções, os resultados são apresentados e discutidos para cada agente e o desempenho destes agentes são comparados em relação a três métricas: lucro, pedidos finalizados em tempo e custo por pedido.

6.5 Sistema Multi-Agente UECEGRUNN Utilizando a Estratégia com *AdaBoost.M1* (E1)

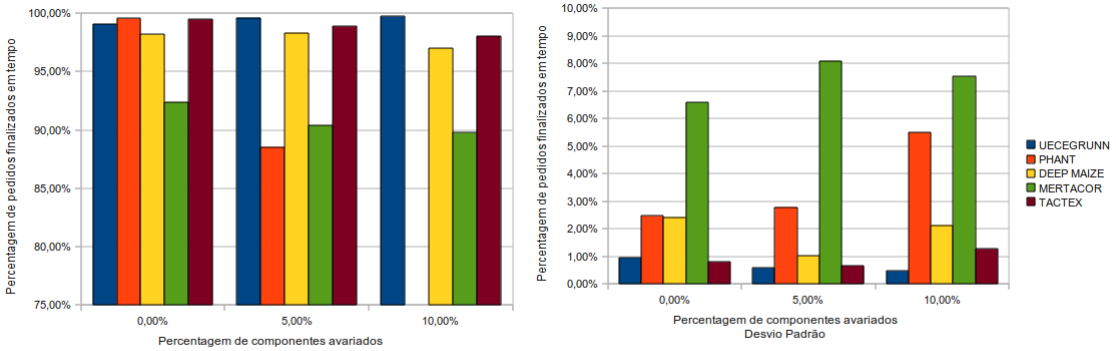
A primeira versão avaliada do sistema UECEGRUNN foi a estratégia de classificação utilizando o algoritmo *AdaBoost.M1*. A Figura 6.2 (a) mostra o lucro do UECEGRUNN com a estratégia E1 e o lucro dos outros agentes. Todos os agentes pioraram seus respectivos desempenhos à medida que a probabilidade de componentes avariados foi crescendo de 0% para 5% e de 5% para 10%.



(a) Média (b) Desvio Padrão

Figura 6.2: Desempenho das fábricas em termos de lucro e considerando o SMA UECEGRUNN com a estratégia baseada em *AdaBoost.M1*.

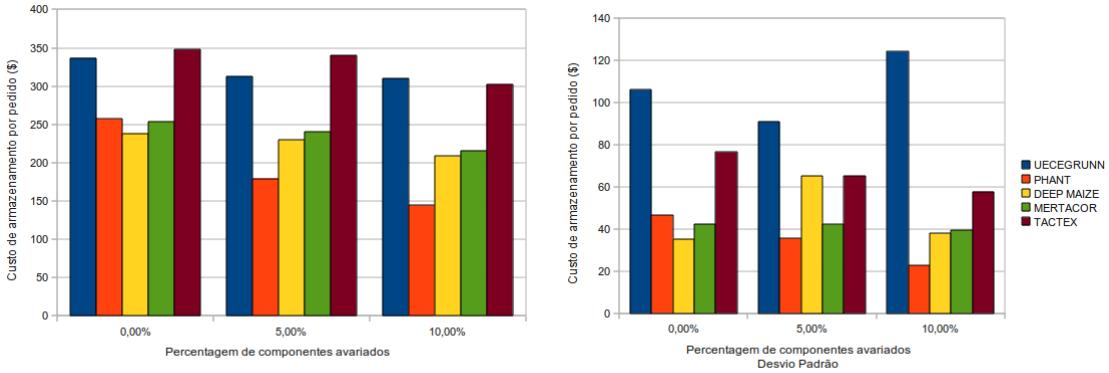
Na Figura 6.3 (a), a percentagem de pedidos finalizados em tempo do sistema UECEGRUNN foi mantida um pouco abaixo dos 100%, enquanto os outros agentes apresentaram resultados inferiores em relação à percentagem de pedidos finalizados a tempo. O agente Phant, por exemplo, apresentou 75% de pedidos finalizados em tempo (ou seja, 25% dos pedidos não foi finalizado em tempo).



(a) Média (b) Desvio Padrão

Figura 6.3: Desempenho das fábricas em termos de pedidos finalizados em tempo e considerando o SMA UECEGRUNN com a estratégia baseada em *AdaBoost.M1*.

A Figura 6.4 mostra a evolução do custo de armazenamento por pedido nos três cenários. Como pode ser observado, o custo de armazenamento do SMA UECEGRUNN só foi inferior ao custo do agente TacTex. Como será mostrado em seguida, o custo de armazenamento por pedido obtido pelo SMA UECEGRUNN quando a estratégia E1 foi utilizada é superior ao custo das outras duas estratégias (E2 e E3), que não apresentam diferenças significativas quanto a este custo. A partir da análise de logs dos jogos realizados, foi percebido que o sistema UECEGRUNN com a estratégia E1 nem sempre conseguia concluir todas decisões dentro do limite de tempo de 15 segundos que representam um dia simulado. Isso está de acordo com a análise de Stan, Stan e Florea (2006), segundo a qual no TAC-SCM há uma quantidade restrita de estratégias que podem ser utilizadas, dado que decisões devem ser tomadas em tempo real.

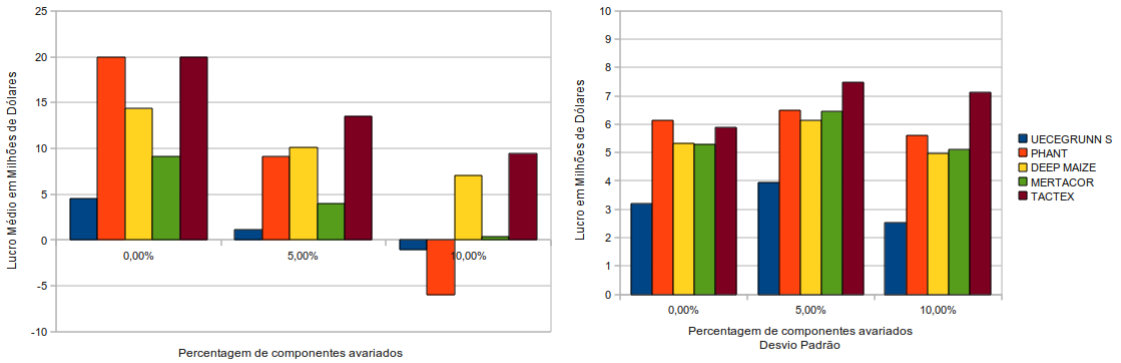


(a) Média (b) Desvio Padrão

Figura 6.4: Custo de armazenamento por pedido, considerado o SMA UECEGRUNN com a estratégia baseada em *AdaBoost.M1*.

6.6 Sistema Multi-Agente UECEGRUNN Utilizando como Estratégia de Oferta a Heurística Simples (E2)

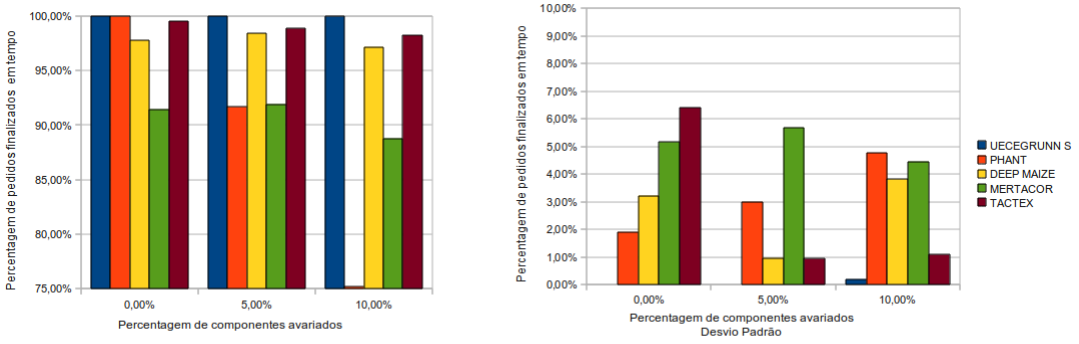
A Figura 6.5 (a) mostra lucro médio do SMA UECEGRUNN e dos outros agentes competidores. O desempenho do SMA UECEGRUNN com heurística simples não pareceu significativamente inferior ao desempenho do SMA UECEGRUNN com *AdaBoost.M1*.



(a) Média (b) Desvio Padrão

Figura 6.5: Desempenho das fábricas em termos de lucro.

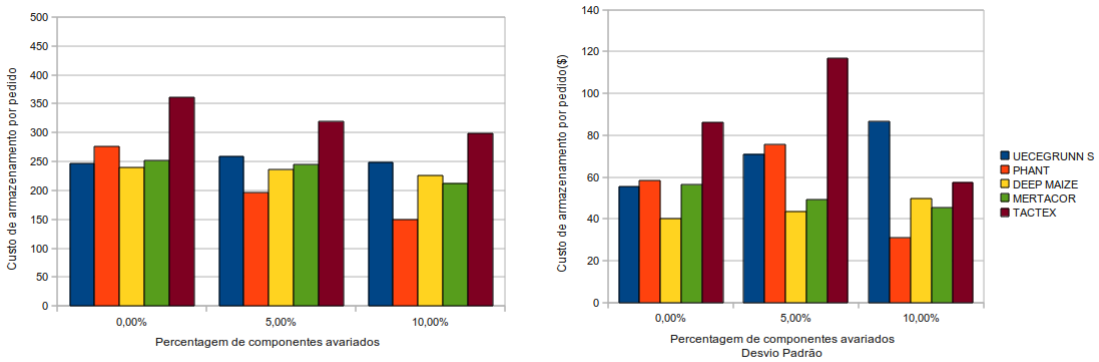
O desempenho dos agentes foi analisado em relação à taxa de pedidos finalizados em tempo e os resultados estão na Figura 6.6 (a). O SMA UECEGRUNN conseguiu manter uma taxa de pedidos finalizados em dia (pedidos finalizados em tempo) aproximadamente igual a 100% nos três cenários investigados.



(a) Média (b) Desvio Padrão

Figura 6.6: Desempenho das fábricas em termos de pedidos finalizados em tempo.

A Figura 6.7 mostra a evolução do custo de armazenamento por pedido nos três cenários. Como pode ser observado, o custo de armazenamento do SMA UECEGRUNN com a estratégia E2 não sofreu alterações significativas ao longo dos três cenários, mantendo-se próximo ao custo gerado por outros agentes. Contudo, quando comparado com SMA UECEGRUNN com a estratégia E1, o SMA com a estratégia E2 obteve um menor custo de armazenamento por pedido.

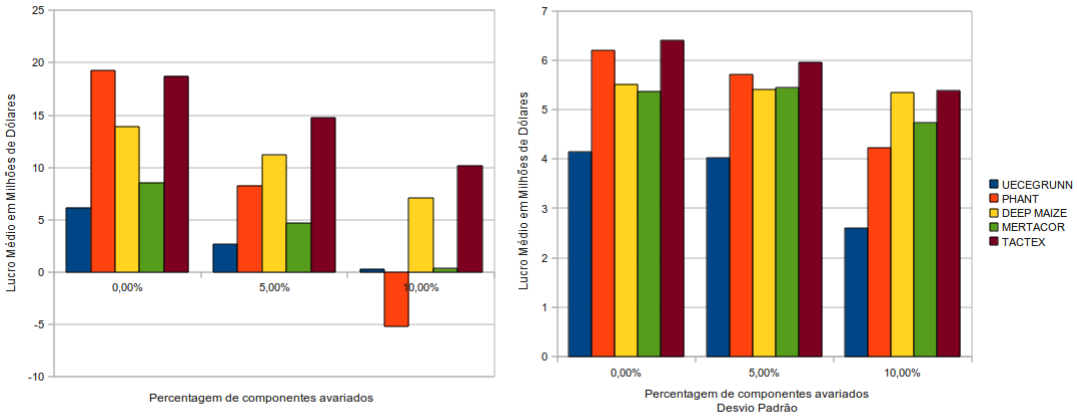


(a) Média (b) Desvio Padrão

Figura 6.7: Custo de armazenamento por pedido.

6.7 Sistema UECEGRUNN Utilizando a Estratégia de Oferta baseada em Redes Neurais, KNN e Heurística (E3)

Os resultados obtidos pelo SMA UECEGRUNN com a estratégia híbrida E3 confirmam a superioridade desta estratégia sobre as outras estratégias avaliadas. Os gráficos das Figuras 6.8 mostram o desempenho das fábricas quanto ao lucro.

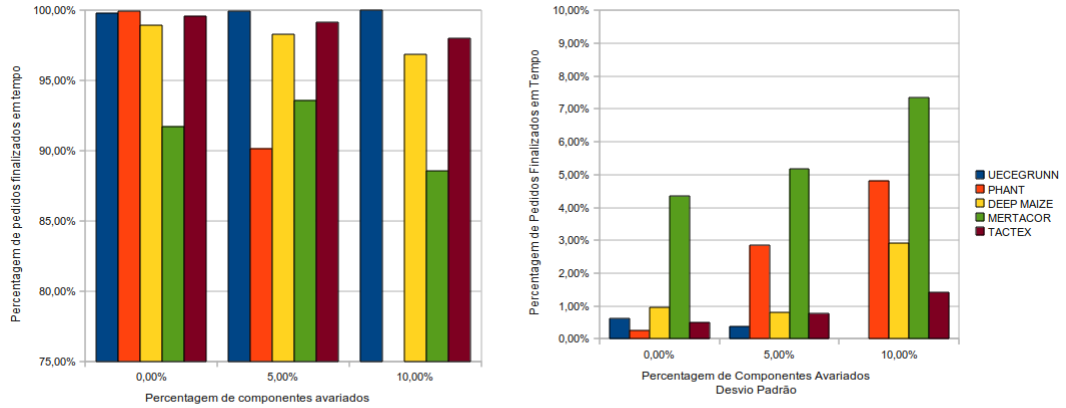


(a) Média (b) Desvio Padrão

Figura 6.8: Desempenho das fábricas em termos de lucro.

Com essa abordagem combinando redes neurais, KNN e heurística, o sistema UECEGRUNN obteve lucro médio positivo nos três cenários. No pior cenário avaliado, o sistema só consegue superar em termos de lucro o agente Phant e tem um lucro ligeiramente menor do que o lucro do agente Mertacor. Contudo, os agentes TacTex e DeepMaize ainda superam o SMA UECEGRUNN em termos de lucro.

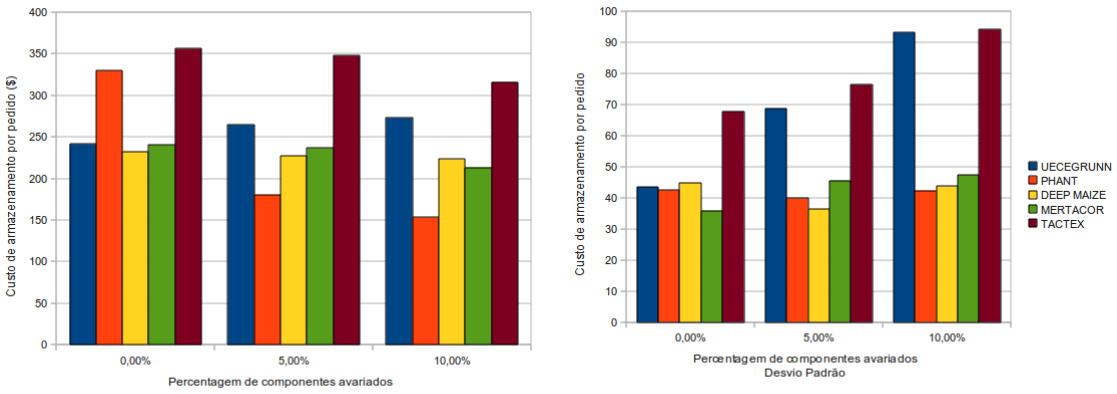
A Figura 6.9 mostra que o sistema consegue manter uma alta percentagem de pedidos finalizados em tempo, independente do cenário.



(a) Média (b) Desvio Padrão

Figura 6.9: Desempenho das fábricas em termos de pedidos finalizados em tempo.

O sistema UECEGRUNN manteve um alto nível de serviço sem aumentar significativamente o custo de armazenamento por pedido, como mostra a Figura 6.10. O custo de armazenamento por pedido do SMA UECEGRUNN utilizando a estratégia E3 foi inferior ao custo do mesmo sistema utilizando a estratégia E1, mas não foi significativamente diferente do sistema utilizando a estratégia E2.



(a) Média (b) Desvio Padrão

Figura 6.10: Custo de armazenamento por pedido.

Nas próximas seções, os resultados apresentados foram discutidos em três dimensões: comparação das estratégias entre si (E1, E2 e E3), comparação do SMA UECEGRUNN com as abordagens clássicas para o TAC-SCM e comparação do SMA UECEGRUNN com o sistema GRUNN.

6.8 Comparação entre as três versões do sistema UECEGRUNN

O resultados das simulações contra um conjunto de agentes competitivos confirmaram a superioridade da estratégia híbrida E3 (redes neurais, KNN e heurística) contra as estratégias com *AdaBoost.M1* e heurística simples. Para se chegar a essa conclusão, foram analisadas as diferenças entre o lucro médio do sistema UECEGRUNN utilizando cada uma das estratégias. Os resultados são resumidos na Tabela 6.12 e foram calculados como a média dos lucros obtidos ao longo de 84 simulações para cada estratégia, resultando em um total de 252 simulações.

Tabela 6.12: Lucro médio das versões do SMA UECEGRUNN com as estratégias: estratégia com *AdaBoost.M1* (E1), heurística simples (E2), redes neurais com KNN e heurística simples (E3).

Estratégia	Lucro Médio
E1	1698823
E2	1528314
E3	3032617

Não há diferença significativa entre os métodos E1 e E2. Já o método E3 apresenta um desempenho estatisticamente superior. Portanto, há três hipóteses: (1) a média de E3 é superior à media de E1, (2) a média de E3 é superior à media de E2 e (2) não há diferença significativa entre E1 e E2. Dados que as amostras são independentes, dois testes foram utilizados: teste Levene (LEVENE, 1960) para igualdade das variâncias e o teste t para comparação das médias. O teste de Lavene produziu seguintes resultados:

- Amostras de lucro de E1 e de E2 possuem variações iguais. No caso, foi obtido o intervalo de confiança $[0.75, 1.79]$ e a taxa de variância foi de 1,16.
- Amostras de lucro de E1 e de E3 possuem variações iguais. No caso, foi obtido o intervalo de confiança $[0.63, 1.50]$ e a taxa de variância foi de 0,97.
- Amostras de lucro de E2 e de E3 possuem variações iguais. No caso, foi obtido o intervalo de confiança $[0.54, 1.28]$ e a taxa de variância foi de 0,83.

Dado que não há evidências para se rejeitar a hipótese de que as amostras possuem a mesma variância, foi utilizado o teste t de *Student* para averiguar as hipóteses (1), (2) e (3). Os resultados foram resumidos na Tabela 6.13.

Tabela 6.13: Resultado do teste t sobre o lucro médio obtido. A hipótese alternativa é que as diferenças entre as médias são diferentes de zero.

Estratégias	E1 e E2	E1 e E3	E2 e E3
Intervalo de Confiança (95%)	-1089576 – 1430593	18028,80 – 2649560,58	235101,2 – 2773504,7
Significância	0,78	0,04	0,02
Diferença das médias	170509	1333794	1504303

Portanto, de acordo com os dados da Tabela 6.13, tomando um nível de significância de 5%, aceita-se a hipótese que a estratégia E3 apresentou nas simulações realizadas uma média significativamente superior às médias das estratégias E1 e E2. Não há evidência para se rejeitar a hipótese de que as médias das estratégias E1 e E2 sejam iguais.

A comparação das três estratégias quanto à percentagem de pedidos finalizados em tempo mostra que não há diferenças significativas entre os desempenhos das versões E2 e E3 do sistema UECEGRUNN. O comportamento do custo por armazenamento nos diferentes cenários não mostrou diferenças significativas para estas versões. O motivo destas duas versões do sistema UECEGRUNN terem apresentado desempenho aproximados em termos de pedidos enviados em tempo e custo de armazenamento por pedido é devido fundamentalmente ao fato de as três versões diferirem apenas em suas estratégias de definição de preço de oferta. Contudo, por que o custo de armazenamento da estratégia utilizando *AdaBoost.M1* foi maior do que o custo de armazenamento das outras versões? A resposta para esta questão está relacionada com o que já foi discutido sobre o desempenho do sistema quando utiliza a estratégia E1. Em cada dia simulado de 15 segundos o agente deve tomar as decisões referentes a este dia, como definir preços de oferta para cada RFQ que chega, esperar o resultado de leilões e enviar a programação de produção e entrega. Se o agente não consegue executar todas as ações dentro de 15 segundos, obviamente que atrasos poderão correr. Isto ocorreu quando o SMA UECEGRUNN utilizou a estratégia *AdaBoost.M1*, o que onerou o desempenho (tempo gasto para tomar decisões e quantidade de memória utilizada) do sistema como um todo, gerando pequenos atrasos. Com os atrasos na entrega, componentes e PCs permaneceram mais tempo em estoque, ocasionando um pequeno incremento no custo de armazenamento por pedido e piorando, em relação ao lucro, o desempenho do sistema UECEGRUNN com a estratégia E1.

6.9 Comparação com as Estratégias Centralizadas

Os resultados apresentados mostraram que todos os agentes pioram seus respectivos desempenhos à medida que a probabilidade de componentes avariados aumentou de 0% para 10%. No pior caso, que segundo Meyer e Wortmann (2009) é um cenário muito provável no mundo real, o SMA UECEGRUNN supera o agente Phant e fica muito próximo em termos de lucro do agente Mertacor. Em todos os casos, o agente TacTex apresenta lucro superior ao dos outros agentes, confirmando os resultados apresentados por Meyer e Wortmann (2009). A Tabela 6.14, baseada nos trabalhos de Ketter, Collins e Gini (2008), Janson, Finne e Eriksson (2006), apresenta os agentes avaliados com base no foco de suas arquiteturas e no compromisso de pesquisa seus projetistas.

Aqueles agentes que tiveram como foco o uso de técnicas de aprendizagem de máquina e otimização obtiveram os melhores resultados em relação ao lucro obtido e à percentagem de pedidos finalizados a tempo. O agente TacTex procura minimizar o custo de obtenção de componentes ao passo que procura maximizar o lucro de vendas. Para isso, dois componentes são dotados de modelos para previsão de preços e da demanda. Assim, o agente consegue prever quais computadores serão mais lucrativos no futuro.

O agente Phant, apesar de não ter como foco o uso de aprendizagem de máquina, apresentou um bom desempenho nos cenários onde não há o problema de avaria de componentes. Contudo, no geral, obteve resultados pobres em termos de lucro quando a probabilidade de avaria de componentes individuais serem avariados foi de 10%. O agente Mertacor utiliza aprendizagem de máquina no seu componente de oferta de computadores e tem mostrado um desempenho satisfatório no cenário sem problemas locais de produção, mantendo um lucro médio positivo, mas próximo de zero, nos outros dois cenários. O agente Mertacor não faz uso intensivo de aprendizagem de máquina para prever preço de componentes e a demanda de clientes, por exemplo. A negociação com fornecedores se resume à definição do preço de reserva, aceitando todas as ofertas de fornecedores. Enquanto agentes como TacTex e DeepMaize monitoram seus próprios níveis de reputação [1], podendo em determinando momento rejeitar ofertas de fornecedores, caso a rejeição dessas ofertas não venha a diminuir suas respectivas reputações.

Segundo Meyer e Wortmann (2009), uma forma simples de lidar com a avaria de componentes nos cenários com 5% e 10% de probabilidade de um componente se tornar inutilizável é o incremento do estoque de segurança do inventário por uma pequena margem. O fato de o agente Phant ter obtido o menor custo de armazenamento por pedido aliado ao fato dele ter sido o agente com pior desempenho em termos de pedidos finalizados em tempo, indica que este agente não utilizou a estratégia simples para lidar com a avaria de componentes e nem possuía uma estratégia robusta a estes problemas. O resultado foi que o principal motivo do agente Phant ter reduzido substancialmente seu desempenho nos cenários com chances de avaria de componentes foi justamente a alta percentagem de pedidos não finalizados em tempo. Em al-

[1] Os fornecedores mantém o histórico de aceitação de ofertas por parte dos agentes e possuem uma função que define o nível de reputação dos agentes com base nesse histórico. Agentes com mais alta reputação possuem mais chances de conseguirem ofertas de fornecedores.

guns cenários, a percentagem de pedidos atrasados do agente Phant chegou a 25%.

Tabela 6.14: Lista de agentes utilizados nas simulações e as principais técnicas utilizadas em seus respectivos programas.

Agente	Compromisso de Pesquisa	Ênfase da Arquitetura	Principais Técnicas Utilizadas
TacTex06	Aprendizagem de Máquina	***Framework*** externo de análise, pacotes de terceiros	Filtro de particulas para predição da probabilidade inicial de aceitação de ofertas e regressão aditiva com ***decision stump*** para prever mudanças futuras de preço de computadores. A predição da demanda utiliza uma abordagem ***bayesiana*** que envolve manter uma distribuição de probabilidade sobre os parâmetros do jogo. Previsão de preço de componentes. Utiliza dados online e dados de jogos passados para se adaptar a diferentes competidores
DeepMaize	Teoria dos Jogos Empirica, Otimização de Restrições, Aprendizagem de Máquina, Coordenação de decisões	Framework externo de análise, modularidade, pacotes de terceiros	A ideia central é manter valores marginais para cada produto finalizado e componente recebido da forma mais acurada possivel, dando predições sobre as condições de mercado e restrições de produção. Controle distribuido de feedback. Programação dinâmica e programação linear inteira para otimizar a programação da produção da fábrica
Mertacor	Coordenação de decisões, gerenciamento dinâmico de cadeias de suprimento	Flexibilidade, Modularidade	Coordenação de módulo por meio de gerenciamento de inventário. Algoritmo M5' para predição do preço de oferta de computadores, utilizado com KNN e heuristicas
Phant	Coordenação de decisões, lidar com incertezas & Heuristicas simples	Heuristicas simples	Coordenação de módulo por meio de gerenciamento de inventário. Heuristicas simples para definição do preço de oferta de computadores

O agente TacTex foi o que apresentou maior custo de armazenamento por pedido e é o agente que apresentou o segundo melhor desempenho em relação à percentagem de pedidos finalizados em tempo. De fato, a estratégia de inventário do TacTex mantém um nível mínimo de estoque alto (800 para componentes em geral e 400 para CPUs). O agente DeepMaize surpreendentemente mantém altos níveis de serviço (97% no pior caso e quando compete contra o SMA UECEGRUNN com a estratégia E3) mantendo baixos custos de armazenamento por pedido. Uma análise na estratégia do DeepMaize revelou que este agente coordena suas decisões por meio de um principio chamado "decomposição baseada em valor". Nesta abordagem, uma programação de produção de longo-prazo é construída por incrementalmente adicionar produtos com alto lucro esperado. Portanto, há um grande prazo para se planejar o processo de produção, o prazo necessário para que componentes sejam adquiridos e para que o processo de produção ocorra sem atrasos significativos.

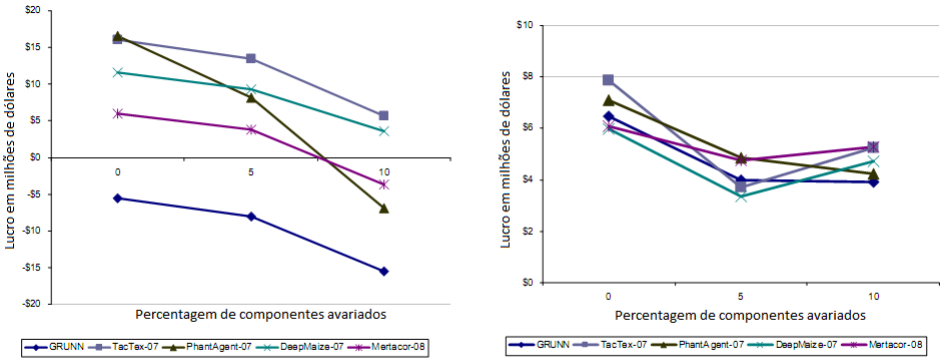
A Tabela 6.15 mostra o lucro médio dos agentes com abordagens centralizadas de planejamento da produção em conjuntos de simulações com diferentes versões do sistema UECEGRUNN. Os agentes apresentaram aproximadamente o mesmo lucro nos três grupos de simulação.

Tabela 6.15: Lucro dos agentes expresso em milhões de unidades monetárias nas simulações com diferentes versões do sistema UECEGRUNN.

Versão do UECEGRUNN/Competidores	TacTex06	Phant	DeepMaize	Mertacor
UECEGRUN E1	14,43	8,18	10,63	5,48
UECEGRUN E2	14,31	7,69	10,52	4,51
UECEGRUNN E3	14,56	7,45	10,73	4,53

6.10 Comparação com o SMA GRUNN

A Figura 6.11 mostra o lucro médio obtido pelo agente UECEGRUNN contra o mesmo conjunto de agentes utilizados nas simulações com o SMA UECEGRUNN.



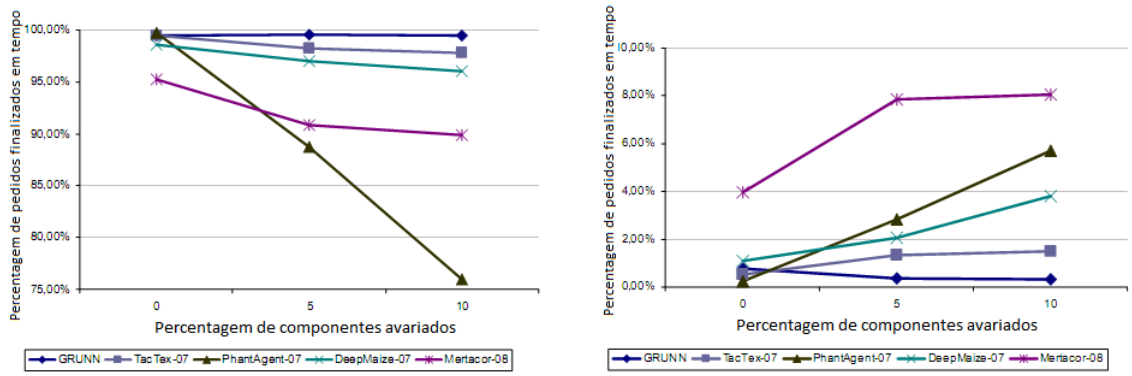
(a) Média (b) Desvio Padrão

Figura 6.11: Desempenho das fábricas quanto ao lucro. Adaptado de Meyer e Wortmann (2009).

Comparando os resultados com o agente GRUNN (MEYER; WORTMANN, 2009), enquanto o sistema GRUNN obteve um lucro negativo em todos os cenários da competição, o SMA UECEGRUNN com a estratégia E3 obteve um lucro positivo em todos os cenários, com uma diferença significativa em relação ao sistema GRUNN.

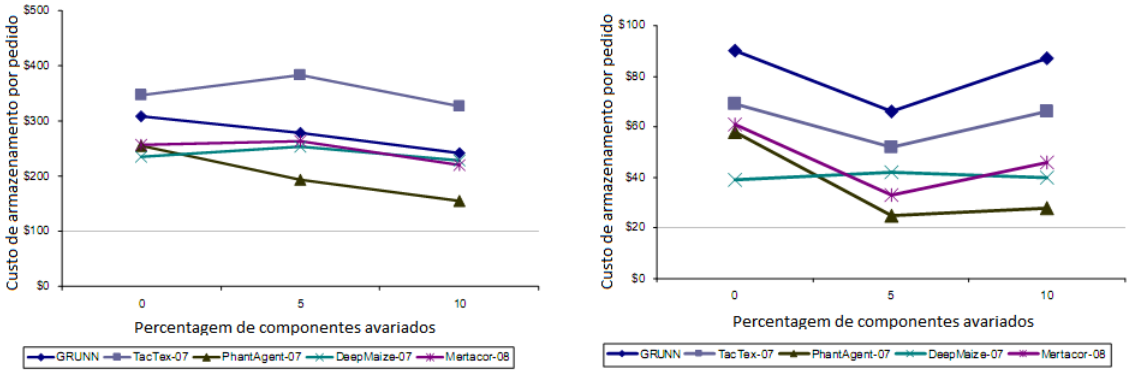
A percentagem de pedidos finalizados a tempo dos agentes UECEGRUNN e GRUNN não apresentam diferenças significativas. Os resultados de pedidos finalizados em dias para o agente GRUNN são apresentados na Tabela 6.12.

Comparando os custo de armazenamento por pedido do sistema GRUNN, como mostra a Figura 6.13, percebe-se este custo foi ligeiramente superior ao custo de armazenamento por pedido do sistema UECEGRUNN com a estratégia E3. Contudo os custos de ambos os agentes não são significativamente diferentes.



(a) Média (b) Desvio Padrão

Figura 6.12: Desempenho das fábricas quanto à percentagem de pedidos finalizados em tempo. Adaptado de Meyer e Wortmann (2009).



(a) Média (b) Desvio Padrão

Figura 6.13: Custo de armazenamento por pedido. Adaptado de Meyer e Wortmann (2009).

6.11 Considerações finais

O sistema UECEGRUNN utilizou o conceito de produtos inteligentes para gerenciamento de pedidos. Todos as informações sobre um pedido de cliente eram gerenciadas por agente de produto (denominado *Intelligent Product*). Com o uso dessa estratégia, o SMA UECEGRUNN conseguiu um alto nível de serviço aos clientes, superior ao nível de serviço das abordagens dos agentes TacTex, Phant, Mertacor e DeepMaize.

Quanto ao lucro, o agente obteve melhor resultado com a estratégia E3, que combina redes neurais, KNN e heurística na geração de preços de oferta de computadores. Os experimentos mostraram claramente que o uso da retro-alimentação do resultado de ofertas durante o jogo promoveu uma melhora significativa do desempenho da estratégia E3. Apesar de ter obtido resultados promissores, o lucro do SMA UECEGRUNN foi significativamente inferior ao lucro dos agentes TacTex e DeepMaize. Isso acontece porque os agentes TacTex e DeepMaize identificam melhores oportunidades de venda e compra. A Tabela 6.16 mostra o preço médio de vendas de cada agente. Como pode ser observado, o SMA UECEGRUNN foi o que apresentou o menor preço médio, como resultado, precisará de muitas vendas para obter um lucro maior.

Tabela 6.16: Média do preço de venda de PCs por agente. O cenário considerado foi o de 0% de componentes avariados.

Produto/Agente	TacTex	Phant	DeepMaize	Mertacor	UECEGRUNN
1	1366,02	1325,23	1319,93	1287,56	1249,96
2	1448,57	1423,22	1407,2	1386,5	1383,8
3	1454,45	1434,13	1409,78	1388,89	1365,42
4	1529,42	1518,29	1505,4	1487,58	1486,25
5	1752,66	1687,03	1713,36	1669,86	1652,65
6	1912,33	1838,8	1873,36	1845,86	1829,07
7	1905,96	1840,78	1870,71	1823,08	1782,36
8	1992,4	1931,3	1966,74	1920,06	1917,49
9	1374,24	1336,73	1344,58	1309,95	1271,36
10	1473,63	1444	1440,5	1413,34	1392,46
11	1459,84	1433,41	1444,28	1413,89	1389,69
12	1550,94	1528,27	1514,49	1496,41	1487,35
13	1754,11	1704,47	1738,58	1672,05	1659,21
14	1845,35	1799,17	1829,33	1775,53	1759,8
15	1847,92	1804,49	1828,73	1777,7	1762,47
16	1933,65	1911,67	1920,55	1879	1873,11
Média	2660,15	2596,1	2612,75	2554,73	2526,24

Como o preço médio de venda de PCs do SMA UECEGRUNN foi menor do que o preço médio de venda de PCs de outros agentes, a suposição de que SMA UECEGRUNN vendeu mais PCs poderia surgir. Esta suposição não se sustenta quando é observado a quantidade total de produtos vendidos por cada agente, como mostra a Tabela 6.17.

O que de fato acontece é que agentes como TacTex e DeepMaize identificam os momentos adequados de venda e de compra (KETTER et al., 2006). Assim, aproveitam oportunidades de venda e de compra, vendendo PCs com preços maiores e comprando componentes baratos.

O gerenciamento de inventário foi baseado na estratégia ATO do agente Mertacor. De acordo com os resultados mostrados na Figura 6.10, o custo de manutenção de estoque por pedidos não foi significativamente diferente do custo obtido por outros agentes. A estratégia de obtenção de componentes também foi baseada na estratégia do agente Mertacor. Nesse caso, não foram utilizadas estratégias de aprendizagem para previsão de preço de componentes e nem para prever o comportamento de fornecedores. O agente Mertacor e o agente UECEGRUNN foram os que obtiveram componentes há um preço médio mais elevado, como mostra a Tabela 6.18. Portanto, suas estratégias de obtenção de componentes precisam ser melhoradas.

Tabela 6.17: Quantidade de PCs vendidos em simulações quando há 0% de chance de componentes serem avariados.

Produto/Agente	TacTex	Phant	DeepMaize	Mertacor	UECEGRUNN
1	150699	129909	103974	123763	84167
2	130645	128831	112678	124176	87347
3	129299	124027	105312	123394	97145
4	120002	117898	110108	125252	102965
5	145504	128725	122752	123221	95155
6	127309	116297	113279	122330	101502
7	128996	111742	109135	121246	108084
8	112978	98360	106888	123903	119080
9	148482	128181	112880	121760	77967
10	129906	118006	113893	121827	88091
11	127113	122695	111147	124128	88888
12	112607	119178	110952	126064	99849
13	145832	128962	117599	122324	97166
14	127339	121592	119067	124776	108428
15	132843	121452	118078	129048	115946
16	117277	107241	117647	129223	130535
Média	208683,1	192309,6	180538,9	198643,5	160231,5

Quanto aos outros agentes, as seguintes considerações foram obtidas:

- O agente TacTex conseguiu obter bons resultados quanto ao lucro e manteve um alto nível de serviço aos clientes. Um ponto-fraco do agente TacTex é o custo de armazenamento por pedido, superior ao dos outros agentes.
- O agente Phant apresenta boas heurísticas de compra e venda, contudo, precisa melhorar o gerenciamento de pedidos, pois foi o agente que apresentou o pior resultado quanto à percentagem de pedidos finalizados em tempo.
- O agente DeepMaize obteve bons resultados quanto ao lucro, perdendo apenas para o agente TacTex. Quanto ao desempenho em termos de pedidos finalizados em tempo, o agente DeepMaize ficou em terceiro lugar. Dos resultados apresentados, foi percebido que este agente pode melhorar quando a estratégia de obtenção de componentes, dado que comprou componentes a um preço médio mais alto do que os agentes TacTex e Phant.
- O Mertacor apresentou um lucro médio inferior ao lucro obtido pelos agentes TacTex e DeepMaize e superou o agente Phant apenas no cenário com 10% de chance de componentes individuais serem avariados. Foi observado que o agente Mertacor precisa melhorar as estratégias de compra e de gerenciamento de pedidos, dado que foi o agente a comprar componentes a um preço médio mais alto e obteve o segundo pior resultado em relação à pedidos finalizados em tempo.

Tabela 6.18: Preço médio pago por componente ponderado pela quantidade comprada. O cenário considerado foi o de 0% de componentes avariados.

Agente/Componente	100	101	110	111	200	210	300	301	400	401	Média
TacTex	700	1059	711	1021	161	161	67	132	195	263	447
Phant	664	973	675	950	160	160	67	131	194	263	424
Deep Maize	703	1059	715	1033	166	165	69	135	198	266	451
Mertacor	707	1085	726	1054	170	169	70	138	206	277	460
UECEGRUNN(E3)	699	1076	716	1049	169	169	69	192	256	348	456

7 Conclusões e trabalhos futuros

Obter estratégias multi-agente para competir em ambientes altamente competitivos como o TAC-SCM não é uma tarefa trivial, dado que existem vários tipos de incertezas e a necessidade de coordenar a decisão de diversos agentes em atividades interdependentes. Foi percebido que o ambiente do TAC-SCM retrata com realismo o problema de coordenar atividades interdependentes e permite a análise de diferentes estratégias em relação a várias métricas de gerenciamento de cadeias de suprimento, como lucro, nível de serviço aos clientes e custo de operações internas.

Neste trabalho, uma solução multi-agente que atua no ambiente TAC-SCM foi descrita. Esta abordagem utiliza uma estratégia multi-agente baseada em produtos inteligentes para gerenciamento de pedidos. Para coordenar as decisões internas tomadas pelos diferentes agentes foi utilizado o conceito de regulamentação social. Diferentes estratégias de definição de preço de oferta foram utilizadas. Estas estratégias incluíram técnicas de aprendizado de máquina, como AdaBoost.M1 e redes neurais, e uma heurística simples.

Para avaliar a abordagem multi-agente agente concebida e outras abordagens encontradas na literatura em diferentes cenários de problemas locais de produção, o simulador do TAC-SCM foi modificado para suportar três casos de problemas locais de produção com as seguintes probabilidades de componentes individuais serem danificados: 0%, 5% e 10%.

Das três estratégias de vendas utilizadas, a que obteve o melhor resultado foi a estratégia E3, que combina o uso de redes neurais, KNN e heurística. O melhor desempenho dessa estratégia é justificado pelo uso do *feedback* das ofertas enviadas, informação esta ignorada pelas outras estratégias.

A abordagem multi-agente desenvolvida apresentou desempenho inferior ao desempenho de outros agentes quanto ao lucro obtido. No cenário com 10% de probabilidade de avaria de componentes individuais, o SMA UECEGRUNN superou o agente Phant em termos de lucro e ficou próximo do agente Mertacor. Quanto à percentagem de pedidos enviados em dia, o SMA UECEGRUNN obteve o melhor resultado. O custo de armazenamento por pedido mostrou que o agente teve o segundo maior custo, contudo, com uma pequena diferença em relação ao custo gerado por outros agentes. De acordo com os resultados analisados, percebeu-se que o sistema UECEGRUN precisa melhorar em dois aspectos: vendas e compra. A simples previsão de preços juntamente com o uso interno de leilões não foi suficiente para garantir maiores lucros e, portanto, superar agentes como TacTex e DeepMaize. O agente UECEGRUNN manteve altos níveis de pedido, evitando o pagamento de multas, o que poderia onerar o lucro. Quando comparado com o sistema GRUNN, O SMA UECEGRUNN obteve

um lucro significativamente superior nos três cenários e manteve a propriedade de alta taxa de pedidos finalizados em tempo.

Estratégias de outros agentes foram avaliadas em relação a diferentes medidas consideradas e suas vantagens e desvantagens foram apontadas. O trabalho confirmou a superioridade de estratégias centralizadas para controle e planejamento da produção em relação ao lucro. O agente TacTex apresentou o melhor resultado, inclusive mantendo boa margem de lucro em cenários com problemas locais de produção. Os resultados obtidos pelo agente Phant revelaram que é necessário mais do que boas estratégias de venda de computadores e de compra de componentes para se obter bons resultados em diferentes situações de problemas locais de produção. O agente Phant obteve bons resultados no cenário de 0% de componentes avariados. Contudo, o lucro do agente caiu substancialmente nos cenários com 5% e 10% de chance de componentes individuais serem avariados.

Além das avaliações realizadas, a implementação do SMA UECEGRUNN pode ser útil em disciplinas com Inteligência Artificial Distribuída, servindo como ilustração da aplicação de teorias computacionais em um programa de agente concreto para a resolução de problemas complexos. A abordagem em si pode ser modificada para suportar estratégias para problemas específicos, dado a modularidade do sistema, com agentes responsáveis por atividades bem definidas. O uso de uma abordagem do agente Mertacor para gerenciamento de inventário e obtenção de componentes ilustra esse caso. Ainda poderia haver o uso combinado de abordagens de diferentes agentes para problemas específicos, todas tendo suas decisões coordenadas por meio de mecanismos de leilão.

Como trabalho futuro, os principais pontos a serem explorados são: a estratégia de compra e de vendas do SMA UECEGRUNN, possivelmente pela identificação de regimes de mercado, também apontada em outros trabalhos, como apresentado por Ketter et al. (2006); o uso de métodos de aprendizagem por reforço poderia melhorar a estratégia de vendas por meio do uso do *feedback* de sucesso ou fracasso de ofertas; percebe-se possível a inserção de mecanismos de adaptação, conforme a freqüência de transações e a aversão ao risco que o agente venha a ter, em qualquer momento da competição; quanto às ações dos IPs, percebe a necessidade de mecanismos de controle (centralizado) sobre as ações dos IPs; quanto à regulamentação social, percebe-se a necessidade de refinamento do protocolo de determinação do vencedor.

Referências Bibliográficas

AUSTIN, J. L. *How to do things with words*. [S.l.]: Cambridge (Mass.), 1962.

BENISCH, M. et al. *CMieux Analysis and Instrumentation Toolkit for TAC SCM*. [S.l.], September 2005.

BERNERS-LEE, T.; FIELDING, R.; FRYSTYK, H. *Hypertext Transfer Protocol – HTTP/1.0*. United States: RFC Editor, 1996.

BRAGA, A. de P.; CARVALHO, A. P. de Leon F. de; LUDEMIR, T. B. *Redes Neurais Artificiais: Teoria e Aplicações*. [S.l.]: LTC, 2000.

CHATZIDIMITRIOU, K. C. et al. Agent mertacor: A robust design for dealing with uncertainty and variation in scm environments. *Expert Systems with Applications*, v. 35, n. 3, p. 591 – 603, 2008. ISSN 0957-4174.

CHENG, F. et al. Inventory-service optimization in configure-to-order systems. *IBM Research Report, RC*, v. 21781, p. 114–132, 2000.

CHOPRA, S.; MEINDL, P. *Supply Chain Management: Strategy, Planning, and Operations*. Upper Saddle River: Prentice-Hall, Inc., 2001.

COLLINS, J. et al. *The Supply Chain Management Game for the 2007 Trading Agent Competition*. Pittsburgh, PA, 2006.

DIETTERICH, T. G. Ensemble methods in machine learning. In: . [S.l.]: Springer-Verlag, 2000. p. 1–15.

FEITOSA, R. G. F. *LACONIBOT: Um Agente para Atuar em Leilões do tipo CDA do TAC*. Dissertação (Mestrado) — Universidade Estadual do Ceará, 2009.

FOWLER, M. *UML Destiled: A Brief Guide to the Standart Object Modeling Language*. [S.l.]: Addison Wesley Longman, Inc, 2000.

FREUND, Y.; SCHAPIRE, R. E. *Experiments with a New Boosting Algorithm*. 1996.

GöNEN, M. S.-K. C. C. M. Receiver operating characteristic (roc) curves. In: *Statistics and Data Analysis*. [S.l.: s.n.], 2007.

HAYKIN, S. *Neural Networks: a comprehensive foundation*. [S.l.]: Prentice-Hall, Inc., 1999.

HE, M. et al. Designing a successful trading agent for supply chain management. In: *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2006. p. 1159–1166. ISBN 1-59593-303-4.

HUGOS, M. *Essentials of Supply Chain Management*. Hoboken, New Jersey: Jonh Wiley & Sons, 2003.

HUHNS, M. N.; STEPHENS, L. M. Multiagent systems and societies of agents. MIT Press, Cambridge, MA, USA, p. 79–120, 1999.

JANSON, S.; FINNE, N.; ERIKSSON, J. Evolution of a supply chain management game for the trading agent competition. *AI Communications*, v. 19, n. 1, p. 1–12, 2006.

KELLER, P. W.; DUGUAY, F.-O.; PRECUP, D. Redagent-2003: An autonomous market-based supply-chain management agent. *Autonomous Agents and Multiagent Systems, International Joint Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 3, p. 1182–1189, 2004.

KETTER, W.; COLLINS, J.; GINI, M. A survey of agent designs for tac scm. In: *Workshop for Trading Agent Design and Analysis*. Chicago, USA: [s.n.], 2008. p. –.

KETTER, W. et al. Identifying and forecasting economic regimes in TAC SCM. In: POUTRé, H. L.; SADEH, N.; JANSON, S. (Ed.). *Agent-Mediated Electronic Commerce: Designing Trading Agents and Mechanisms*. [S.l.]: Springer-Verlag, 2006, (Lecture Notes in Artificial Intelligence, v. 3937). p. 113–125.

LESSER, V. R. Cooperative multiagent systems: A personal view of the state of the art. *IEEE Transactions on Knowledge and Data Engineering*, IEEE Computer Society, Los Alamitos, CA, USA, v. 11, p. 133–142, 1999. ISSN 1041-4347.

LEVENE. Robust tests for equality of variance. In: OLKIN, I. et al. (Ed.). *Contributions to Probability and Statistics*. California: Stanford University Press, 1960. p. 278–292.

MASILI, G. S. *Metodologia e Software para Simulação de Leilões de Energia Elétrica do Mercado Brasileiro*. Dissertação (Mestrado) — Universidade Estadual de Campinas, 2004.

MCFARLANE, D. et al. Auto id systems and intelligent manufacturing control. *Engineering Applications of Artificial Intelligence*, v. 16, n. 4, p. 365–376, June 2003.

MEYER, G.; WORTMANN, H. Robust planning and control using intelligent products. In: *IJCAI-09 Workshop on Trading Agent Design and Analysis*. Pasadena, California, USA: [s.n.], 2009.

MEYER, G. G.; FRäMLING, K.; HOLMSTRöM, J. Intelligent products: A survey. *Comput. Ind.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 60, n. 3, p. 137–148, 2009. ISSN 0166-3615.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997.

PARDOE, D.; STONE, P. Bidding for customer orders in TAC SCM. In: FARATIN, P.; RODRIGUEZ-AGUILAR, J. (Ed.). *Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems (AMEC 2004)*. Berlin: Springer Verlag, 2005, (Lecture Notes in Artificial Intelligence, v. 3435). p. 143–157.

PARDOE, D.; STONE, P. An autonomous agent for supply chain management. In: ADOMAVICIUS, G.; GUPTA, A. (Ed.). *Handbooks in Information Systems Series: Business Computing*. [S.l.]: Emerald Group, 2009. v. 3, p. 141–72.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach (International Edition)*. [S.l.]: Pearson US Imports & PHIPEs, 2002. ISBN 0130803022.

SEARLE, J. *Speech Acts*. [S.l.]: Cambridge University Press, 1969.

SHERMAN, R. The future of market regulation. *Southern Economic Journal*, v. 67, n. 4, p. 782–800, April 2001.

SIMCHI-LEVI, D. *Designing And Managing The Supply Chain*. [S.l.]: Mcgraw-Hill College, 2005. ISBN 007298239X.

SODOMKA, E.; COLLINS, J.; GINI, M. Efficient statistical methods for evaluating trading agent performance. In: *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*. [S.l.]: AAAI Press, 2007. p. 770–775. ISBN 978-1-57735-323-2.

STAN, M.; STAN, B.; FLOREA, A. M. A dynamic strategy agent for supply chain management. *Symbolic and Numeric Algorithms for Scientific Computing, International Symposium on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 227–232, 2006.

WEISS, G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. [S.l.]: The MIT Press, 2000. Paperback. ISBN 0262731312.

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. *Adaptive switching circuits*, v. 4, p. 96–104, 1960.

WITTEN, I. H.; FRANK, E. *Data Mining: Pratical Machine Learning Tools and Techniques (The Morgan Kaufmann Series in Data Management Systems)*. [S.l.]: Morgan Kaufmann, 2005.

WONG, C. Y. et al. The intelligent product driven supply chain. 2002.

WOOLDRIDGE, M. Intelligent agents. In: WEISS, G. (Ed.). *Multiagent Systems A Modern Approach to Distributed Artificial Intelligence*. [S.l.]: MIT Press, 2000. p. 27–77.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. Hoboken, NJ: John Wiley & Son, 2009.