Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Modelagem de Aprendizagem por Reforço e Controle em Nível Meta para melhorar a Performance da Comunicação em Gerência de Tráfego Aéreo

Daniela Pereira Alves

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientador
Prof. Dr. Li Weigang

Brasília
2006

Universidade de Brasília – UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Profª. Drª. Alba Cristina Magalhães Alves de Melo

Banca examinadora composta por:

Prof. Dr. Li Weigang (Orientador) – UNB
Prof. Dr. Anisio Brasileiro de Freitas Dourado – UFPE
Profª. Drª. Célia Ghedini Ralha – UNB

CIP – Catalogação Internacional na Publicação

Daniela Pereira Alves .

Modelagem de Aprendizagem por Reforço e Controle em Nível Meta para melhorar a Performance da Comunicação em Gerência de Tráfego Aéreo/ Daniela Pereira Alves . Brasília : UnB, 2006.
122 p. : il. ; 29,5 cm.

Dissertação (Mestre) – Universidade de Brasília, Brasília, 2006.

1. Agentes inteligentes. Controle em nível meta. Aprendizagem por reforço. Processo decisório de Markov. Gerência de fluxo de tráfego aéreo.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro – Asa Norte
CEP 70910–900
Brasília – DF – Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Modelagem de Aprendizagem por Reforço e Controle em Nível Meta para melhorar a Performance da Comunicação em Gerência de Tráfego Aéreo

Daniela Pereira Alves

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr. Li Weigang (Orientador)
UNB

Prof. Dr. Anisio Brasileiro de Freitas Dourado
UFPE

Profª Drª Célia Ghedini Ralha
UNB

Profª Drª Alba Cristina Magalhães Alves de Melo
Coordenadora do Mestrado em Informática

Brasília, 09 de Novembro de 2006

Dedicatória

Dedico este trabalho ao meu noivo, Janderson,
à minha família, Rivaldo, Cléria, Patrícia e Juliana,
ao amigo Bueno e a todos que de alguma forma
contribuíram para o seu acontecimento.

Agradecimentos

Meus sinceros agradecimentos ao meu orientador professor doutor Li Weigang, pelo aprendizado, acompanhamento e orientação que possibilitaram esta conquista. Obrigada por permitir a realização deste sonho!

A Janderson Vieira de Souza, meu noivo, pela sua enorme paciência, compreensão, apoio e dedicação. E pela valiosa convivência ao longo destes nove anos.

A toda minha família, em especial aos meus pais, Rivaldo e Cléria e minhas irmãs, Patrícia e Juliana. Pela compreensão da minha ausência, pelo apoio, incentivo, motivação e exemplos de luta.

A Bueno Souza, companheiro de projeto no mestrado, pela intensa convivência, dedicação e amizade nos vários meses de duração do projeto. A Adriane, esposa do Bueno e minha amiga, pela imensa compreensão e amizade.

Aos demais amigos e colegas de Mestrado, especialmente para Marcos Francisco, Deborah Alves, Lorena, Nelson, Edward, Roberto, José Geraldo, Alexandre, Richardson, Rosemberg, Débora Nery, Caetano, Hemerson e Yan. Obrigada pelo apoio, incentivo e convivência.

Aos professores doutores da UnB, em especial àqueles que tive o privilégio de cursar suas disciplinas: Celia, Alba, Maria Emília e Mauricio Ayala. Às funcionárias da Secretaria da Pós-graduação, Rosa e Gláucia, pela atenção, empenho e dedição.

Às amigas que sempre me acompanharam durante esta trajetória: Fabiana, Fabrícia e Elaine.

A todos os colegas do Instituto Brasileiro de Informação em Ciência e Tecnologia. Citarei alguns nomes para representar os demais: Eny, Carla, Gabriel, Marcos Sigismundo, Sueli Maffia, Francisca e Simone. Pela compreensão, convivência, apoio e força nos momentos difíceis.

A Deus, Nosso Senhor, que sempre esteve ao meu lado nesta difícil empreitada, encorajando-me, dando força, paciência, esperança e motivação para vencer todos os obstáculos enfrentados.

Sumário

Lista de Figuras

Lista de Símbolos

α, Taxa de aprendizagem

ϵ, Fator que determina a porcentagem a ser considerada no sorteio entre usufruir e explorar

η, Porcentagem que define a influência da heurística no usofruto do agente

γ, Fator de desconto para reforços futuros

s_t, Estado atual do agente

s_{t+1}, Próximo estado do agente

S, Conjunto finito de estados

A, Conjunto finito de ações

a_t, Ação tomada pelo agente

τ, Função de transição de estado

R, Função de recompensa

H, Função heurística

π^*, Política ótima

π, Política

Q, Função valor-ação

Q^*, Função valor-ação ótima

V, Função valor

V^*, A função valor ótima

Lista de Abreviaturas

AR : Aprendizado por Reforço

ATFM : *Air Traffic Flow Management*

ATC : *Air Traffic Control*

BLF : *Belief Desire Intention*, Crença Desejo Intenção

DT : Diferença Temporal

IA : Inteligência Artificial

MAR : Módulo de Aprendizagem por Reforço

MC : *Monte Carlo*

MDP : *Markov Decision Process*, Processo Decisório de Markov

MGM : Meta Gerente de Mensagens

MLC : *Meta-Level Control*, Controle em Nível Meta

MODEC : Módulo de Decisão e Controle

PBA : Padrão de Balanceamento de Aeroporto

PD : Programação Dinâmica

SARSA : *State Action Reward State Action*, Estado Ação Recompensa Estado Ação

SMA : Sistemas Multiagentes

UML : *Unified Modeling Language*, Linguagem de Modelagem Unificada

Resumo

Uma solução computacional que utiliza troca de mensagens lida com a dificuldade em decidir qual a melhor ação a ser executada à medida que uma nova mensagem chega. No caso específico da área de tráfego aéreo, o uso de troca de mensagens é empregado para manter consistentes as informações distribuídas entre os aeroportos, sujeitas às características estocásticas deste contexto.

O uso de gerência em nível meta e a aprendizagem por reforço foram empregados, neste trabalho, com intuito de apresentar uma estratégia para tratar o problema de gerência da imensa quantidade de mensagens geradas no contexto de tráfego aéreo. A estratégia proposta fundamenta-se na busca pela adaptação por meio da aprendizagem durante o processo de tomada de decisão. A idéia é empregar uma camada adicional de controle em nível meta sobre a camada de controle já existente no sistema hospedeiro para auxiliar o processo de tomada de decisão. A decisão passa a ser tomada com uso da experiência adquirida pelo agente com a aprendizagem por reforço melhorada por heurísticas propostas.

O trabalho, então, propõe um modelo de computação inteligente para auxílio do processo de tomada de decisão de um sistema distribuído aplicado a *Air Traffic Flow Management* (ATFM). Ele é indicado para atuar na comunicação via troca de mensagens entre aeroportos, trabalhando como uma camada adicional em um aeroporto que usa os metadados das mensagens em suas decisões, com vistas à otimização na definição de uma hierarquia para atendimento às mensagens.

O modelo é considerado inovador porque usa aprendizagem por reforço adequada às características deste ambiente estocástico, preocupando-se com a velocidade e qualidade do processo de tomada de decisão.

Na modelagem, três estratégias foram propostas para a aprendizagem: heurística inicial, epsilon adaptativo e heurística baseada em performance. Elas são combinadas aos algoritmos de aprendizado por reforço: *Q-learning* e SARSA. Os estudos de caso avaliam o desempenho, a qualidade do aprendizado quanto às três melhorias propostas e também o comportamento do *Q-learning* quando são alterados parâmetros do algoritmo.

Palavras-chave: Agentes inteligentes. Controle em nível meta. Aprendizagem por reforço. Processo decisório de Markov. Gerência de fluxo de tráfego aéreo.

Abstract

A computational solution which uses message exchange deals with difficulty to decide what is the best action to execute when a new message arrives. In the specific case of Air Traffic field, the use of message exchange is employed to keep consistency among distributed airport information which are subject to random characteristics of the context.

In this work meta-level management and reinforcement learning is employed, with the intention to present one strategy to deal with the problem of managing huge quantity of messages that are created in the aero air traffic context. The proposed strategy is based in the search for adapt action through the learning during the decision make process. The idea is to employ one additional meta-level control layer over the existing control layer in the host system to assist the decision process. The decision is then made using the experience acquired by the agent with the improved heuristical proposals.

This work proposes one intelligent computational model to assist the decision make process in a distributed systems applied to the Air Traffic Flow Management - ATFM. It is indicated to deal with the communication through message exchanges between airports, working like an additional layer in an airport that uses message's metadata in its decision of pursuing the optimization in the hierarchy to attendance messages.

The model is considered innovative because it uses reinforcement learning adjusted to the characteristics of the random environment, concerned with the speed and quality in decision make process.

In the modeling, three strategy was proposed for learning: initial heuristics, adaptative heuristics and performance heuristics. They are combined with algorithms: Q-learning and SARSA. The case studies evaluate by the three enhancements proposed - performance, learning quality and Q-learning behavior when parameters is modified.

Keywords: Intelligent agents. Meta-level control. Reinforcement learning. Markov decision process. Air traffic flow management.

Capítulo 1

Introdução

O tráfego aéreo exerce influência em diversos setores da economia do Brasil, como transporte de alguns produtos, Produto Interno Bruto (PIB) e transporte de pessoas, entre outros. Assim, os cidadãos brasileiros, sofre a influência do tráfego aéreo de forma direta, quando estão viajando, ou de forma indireta, por meio dos possíveis prejuízos na economia do nosso país.

A ocorrência de problemas em tráfego aéreo afetam os usuários por intermédio dos atrasos dos vôos e, em casos mais extremos, até no cancelamento de um vôo. Conforme a gravidade do problema, a suspensão temporária dos vôos em determinada região pode ser exigida. Um exemplo de conseqüência dos problemas em tráfego aéreo foi o acidente envolvendo dois aviões ocorrido em 29 de setembro de 2006 que vitimou 154 pessoas. Esse acidente trouxe a população brasileira à discussão de como está a situação atual do tráfego aéreo brasileiro.

Dentre os motivos que ocasionam os problemas em tráfego aéreo, destacam-se a intensa sobrecarga de trabalho imposta aos controladores, o número insuficiente de profissionais para atender a quantidade de vôos, os feriados que contribuem com o aumento do número de vendas de passagens aéreas, o horário do dia em que existe maior interesse, a intenção das companhias aéreas em utilizar determinada rota e também a situação do clima no instante em que se deseja voar.

Um aspecto importante no tráfego aéreo brasileiro é a intensa participação de trabalho humano. Um exemplo desta participação intensa está no profissional *controlador*. As condições de trabalho desta categoria não é tão favorável para a escolha dos melhores profissionais por causa dos baixos salários e da carga de trabalho intensa. Uma ferramenta, portanto, que os auxiliem é necessária e tem urgência.

Os motivos citados no parágrafo anterior evidencia a necessidade de se adotarem estratégias mais indicadas do que as atualmente empregadas, ou seja, o investimento em meios que visem ao uso mais eficiente dos recursos já existentes

nos aeroportos. Em outras palavras, há necessidade de pesquisar soluções em Air Traffic Flow Management (ATFM) para que os vôos ocorram de forma segura, rápida, ordenada e econômica.

A intenção deste trabalho é melhorar o tráfego aéreo, auxiliando a definição mais eficiente do escalonamento dos vôos, contribuindo, assim, com o melhor uso dos recursos em um menor prazo, quando for possível. Com isso, objetiva-se melhorar o aproveitamento do trabalho humano e a diminuição da carga de trabalho. Contudo, uma solução eficiente para o problema da situação do tráfego aéreo não é simples e envolve intensa atividade de pesquisa.

Este estudo, então, propõe um modelo de computação inteligente para auxílio do processo de tomada de decisão de um sistema distribuído aplicado a *Air Traffic Flow Management* (ATFM). Ele é indicado para atuar na comunicação via troca de mensagens entre aeroportos, trabalhando como uma camada adicional em um aeroporto que usa os metadados das mensagens e o aprendizado em suas decisões buscando a otimização na definição de uma hierarquia para atendimento às mensagens.

1.1 Motivação

O excesso de mensagens no ambiente ATFM ocorre por causa da necessidade de negociação entre os aeroportos para o estabelecimento da escala dos vôos. A negociação é necessária, porque a presença de incerteza influencia os eventos que interferem no escalonamento em tráfego aéreo.

Uma proposta de solução para o problema de escalonamento de vôos em tráfego aéreo é um modelo que atua em gerência de tráfego aéreo, isto é, em ATFM. Esse modelo usa informações distribuídas que são compartilhadas via troca de mensagens. Essa proposta recebeu o nome de ATFM Grid Service - ATFMGS, (59).

O ATFMGS teve como origem as pesquisas na área de tráfego aéreo, que são encontradas em Weigang nos anos de 1994 (57) e 1997 (58), Bonzano et al. no ano de 1996 (8), Prevot no ano de 2002 (36), Nguyen-Duc et. al. no ano de 2003 (34).

O ATFMGS auxilia o processo de tomada de decisões em sistemas distribuídos que atuam com o planejamento tático, podendo estender algumas subfunções do planejamento estratégico, como o replanejamento de escala de vôos e de monitoração e controle como uma função que simula o controle de tráfego, de modo que o sistema possa ter uma abrangência mais completa.

No sistema ATFMGS, emprega-se o conceito de sistema distribuído, que se caracteriza pela intensa troca de mensagens entre os aeroportos envolvidos. Estas mensagens ocorrem com a finalidade de coordenar, negociar e trocar informações entre os agentes que compõem o sistema.

O sistema de informações distribuídas ATFMGS permitiu melhor eficiência no tratamento do problema de tráfego aéreo, atuando sobre o escalonamento dos vôos até uma hora e trinta minutos antes das decolagens e pousos. Tal sistema distribuiu as informações entre os aeroportos e se mostrou compensador por vários pontos:

- deixa de existir um ponto único que, na verdade, torna-se o gargalo;
- as consultas sobre as informações podem ocorrer de forma mais rápida;
- não são todas as informações que precisam ser mantidas em todos os pontos, mas somente aquelas que são importantes e que exercem influência no escalonamento local.

Apesar das vantagens citadas, observa-se que a distribuição da informação, também apresenta algumas desvantagens como:

- capacidade de gerar grande quantidade de mensagens que, se não forem bem tratadas, levam a situações indesejáveis, como a paralisação da comunicação;
- a escalabilidade do modelo se mostra comprometida de acordo com o número de aeroportos considerados, por exemplo, se dobrar a quantidade de aeroportos;
- não são analisadas as características das mensagens para se tomar a decisão de qual a ação mais indicada a ser executada em determinado instante, considerando a situação atual do ambiente;
- não existe um processo de aprendizado que permita a adaptação do agente ao longo do tempo.

No trabalho do ATFMGS, vários aspectos podem ser identificados, por interferirem na quantidade de mensagens trocadas na negociação:

- inicialmente, o mau tempo, que interfere no escalonamento dos vôos previamente planejados;

- depois, o interesse das companhias aéreas em disponibilizar vôos nos mesmos horários por motivo de demanda garantida. Exemplo disso são os vôos às 18 horas;
- em seguida, efeitos sazonais, como férias e feriados;
- por último, eventos menos freqüentes - um alerta terrorista, por exemplo - que, apesar de não acontecerem no Brasil, influenciam os vôos internacionais que partem de aeroportos brasileiros.

A presença de incerteza influencia diretamente o escalonamento em tráfego aéreo, logo esse ambiente pode ser definido como estocástico. Algumas referências reforçam essa afirmação: Goodchild et al. em 2001 (24; 23) e Servé et al. em 2003 (46). Os fatores já mencionados - mau tempo, congestionamento, demanda de recursos maior que a disponibilidade - ocorrem de forma, nem sempre, previsível. Portanto, soluções que trabalham com processos estocásticos definem melhor esse ambiente.

Além da imensa quantidade de mensagens, outro aspecto a ser considerado em ATFM é que o sistema multiagente desenvolvido para auxiliar este contexto apresente boa performance e, se possível, consiga operar bem em tempo real. Todavia, lidar com imensa quantidade de informações pode influenciar no sentido de que a qualidade na seleção da decisão a ser tomada pode ser prejudicada, porque exigiria muito tempo nas análises das possibilidades. Assim, os dois fatores - desempenho e imensa quantidade de informações - devem ser balanceados para que se consiga obter boas soluções ao problema.

Sobre as condições anteriormente mencionadas, verifica-se que somente o alto desempenho da tecnologia de redes e dos processadores modernos não são suficientes para tratar a quantidade de mensagens originadas no contexto ATFM porque a operação ocorre em condições extremas. Outro fato que reforça tal afirmação é que, em várias situações, as redes não são projetadas para operar em sua demanda máxima por causa do alto custo do projeto.

Embora os recursos das redes estejam bastante evoluídos em relação aos de anos atrás, hoje se mostram limitados em razão da complexidade dos problemas atuais. E boas soluções para o problema envolvem tanto *hardware*, quanto *software*.

Em termos de *software*, há crescente necessidade de que as soluções computacionais sejam cada vez mais bem elaboradas e que contemplem o dinamismo e a incerteza presentes nos problemas atuais. Tal necessidade existe porque os problemas atuais lidam com grandes quantidades de informações que se alteram

de maneira freqüente ao longo tempo. Portanto, o uso de conceitos de agentes inteligentes, controle em nível meta e aprendizagem por reforço se mostram interessantes.

1.2 Objetivos

No aspecto geral, o modelo aqui proposto busca conciliar as vantagens da aprendizagem por reforço e do controle em nível meta, com intuito de auxiliar o processo de tomada de decisão em um sistema de computação que mantém as informações distribuídas consistentes via troca de mensagens. Tende assim, a apresentar melhor desempenho com o seu uso, possibilitando a melhoria da performance em sistemas de informações distribuídas.

No aspecto específico, foi estudada a aplicação do modelo à área de tráfego aéreo tomando como referência o sistema de informações distribuídas ATFMGS, em Dib (14).

Para atender aos aspectos geral e específico, o propósito do trabalho é definir um modelo multiagentes que atue em um ambiente controlado em nível meta sustentando-se nos conceitos apresentados por Raja e Lesser (38) e que use aprendizagem por reforço, Sutton e Barto (50). O intuito, ao se empregar as duas metodologias é possibilitar melhor eficiência no processo de tomada de decisão de um sistema distribuído ATFM refletindo, por conseguinte, no tratamento das mensagens com uso de aprendizagem por reforço.

O problema tratado nesta dissertação é aprender, no ambiente controlado em nível meta proposto, mediante uma aprendizagem por reforço melhorada por heurísticas, e, com isso, poder auxiliar no processo de decisão do controlador em nível meta.

Na aprendizagem por reforço definida nos multiagentes, usam-se estratégias com intuito de tomar decisões rápidas, aprender de forma eficiente e aumentar a velocidade da aprendizagem. A aprendizagem por reforço visa a contribuir com o controle em nível meta para que o mesmo consiga tomar decisões melhores, uma vez que há análise crítica em cima das possibilidades estimadas.

No modelo meta gerente de mensagens proposto, a aprendizagem por reforço se mostra favorável porque ocorre por meio da experiência adquirida pelo agente com sua atuação no ambiente. Uma experiência boa adquirida por ele serve de exemplo para decisões futuras a serem tomadas. As mudanças que ocorrem nesse ambiente estocástico são aprendidas pelo agente mediante sua atuação, que é refletida nas recompensas recebidas por ele ao longo do tempo.

A análise crítica realizada pela aprendizagem por reforço modelada se baseia no modelo matemático definido pelo processo decisório de Markov, Clarke e Disney (12) e ainda pela adoção de uma política de escolha de ações que busca maximizar os resultados obtidos. Assim, espera-se contribuir com um modelo fundamentado matematicamente que busque a melhoria dos resultados ao longo do tempo.

O controlador em nível meta proposto se caracteriza por apresentar um gerente em nível meta que monitora o processo de troca de mensagens entre os agentes envolvidos no ambiente. Em soluções gerais, sem a proposta do uso de controle em nível meta, não existe um mecanismo preocupado diretamente com o controle do sistema do aeroporto. Com a abordagem proposta por este trabalho, haverá, portanto, uma estrutura que trabalhará as informações vindas do controle, com o intuito de se melhorar a performance do sistema mediante o tratamento das possibilidades de execução existentes.

1.3 Contribuições

Em uma visão geral, com o modelo definido nesta pesquisa, espera-se contribuir com sistemas que empreguem agentes inteligentes que se comuniquem via troca de mensagens e que apresentem a necessidade de aprender ao longo do tempo, em ambientes estocásticos. Para tanto, há necessidade de os agentes envolvidos habitarem um ambiente controlado em nível meta.

Em uma visão específica, o modelo aqui definido pretende auxiliar na tomada de decisão da camada de controle e, por conseguinte, na melhoria de performance do processo de escalonamento dos vôos. O modelo meta gerente de mensagens se mostra interessante de ser combinado com a proposta de distribuição de informações entre os aeroportos por vários aspectos:

- pelo fato de inserir um processo de avaliação com o intuito de escolher melhor ação a ser tomada acerca de certos critérios (prioridade das ações, congestionamento de mensagens, grande tempo para avaliação, entre outros);

- na aprendizagem, é observada a característica estocástica do ambiente. Não são consideradas somente regras estáticas, que *a priori* têm de prever as situações buscando a priorização da maioria dos casos, e sim uma abordagem que se adapta às mudanças do ambiente;

- a aprendizagem usa estratégias heurísticas no processo, permitindo a tendência

à adaptação ao longo do tempo.

A aprendizagem por reforço aqui proposta foi modelada para trabalhar de maneira que se consiga boa performance, por meio da consideração dos estados como parâmetros binários, o que possibilita operações mais rápidas. A aprendizagem por reforço proposta é combinada a uma heurística baseada em domínio. Assim, espera-se contribuir com uma política de aprendizado que possibilite ao agente combinar eficiência na atuação com a qualidade de aprendizado.

Na aprendizagem também foi proposta a estratégia de adotar um *epsilon* adaptativo que influencia na porcentagem da distribuição de probabilidades da decisão do agente quando este deve decidir entre usufruir ou explorar.

As estratégias empregadas, heurística inicial e heurística baseada em performance, são combinadas aos algoritmos tradicionais de aprendizagem por reforço, no intuito de acelerar o processo de aprendizagem do agente, conforme apresenta Bianchi e Costa (6).

A heurística inicial possibilita acelerar o processo de aprendizagem, uma vez que algumas sugestões de ações para os agentes poderão auxiliá-los nos momentos iniciais que ainda não se tem algum conhecimento do ambiente. E, no caso da heurística baseada em performance, por meio de fatores que exercem influência na satisfação das ações sugeridas pelos agentes.

1.4 Estrutura do Documento

O restante do documento está dividido em sete capítulos e três apêndices.

O capítulo 2 aborda o tema Introdução a Sistemas Multiagentes. Nele, são mostrados conceitos, características e definições para uma arquitetura de sistemas multiagentes. Também traz uma seção que esclarece as diferenças entre os conceitos agentes inteligentes e objetos.

O capítulo 3 descreve os aspectos relacionados ao tema aprendizagem por reforço. Entre eles, estado, ação, recompensa, política e valor-estado. Os algoritmos de aprendizagem por reforço, *Q-learning* e SARSA, são descritos em detalhes.

No capítulo 4, são apresentados aspectos referentes ao tema de controle em nível meta. O controle em nível meta pode ser compreendido, de forma concisa, como uma camada adicional de controle sob a camada já existente de controle do sistema de informação.

O capítulo 5 aborda os conceitos de gerência de tráfego aéreo que são relevantes neste trabalho. O sistema de informações distribuídas ATFMGS para área de tráfego aéreo é apresentado e discutido.

O capítulo 6 apresenta o modelo proposto. Nele, apresentam-se as estratégias de aprendizagem por reforço propostas para auxiliar o processo de tomada de decisão em sistemas distribuídos que se comunicam via troca de mensagens, que são heurística inicial, *epsilon* adaptativo e heurística em performance.

No capítulo 7, é mostrado o estudo de caso realizado. Nele, o modelo é avaliado considerando quatro interesses: performance do algoritmo, qualidade da decisão tomada, situação do aeroporto em análise e o comportamento dos agentes *Q-learning* quanto às alterações dos parâmetros *alpha* e *gamma*.

O capítulo 8 apresenta a conclusão e também a sugestão de trabalhos futuros. Nele, a conformidade do modelo ao contexto que ele foi aplicado, tráfego aéreo, é discutida, assim como as principais diferenças entre os resultados obtidos com o algoritmo *Q-learning* e o algoritmo SARSA. Em trabalhos futuros, há indicações de outros estudos relacionados que podem ser interessantes.

O apêndice A apresenta o processo decisório de Markov. Os conceitos nesse apêndice são importantes para entender como funcionam os módulos de decisão e controle e o módulo de aprendizagem por reforço, que compõem o modelo meta gerente de mensagens, porque eles o seguem como formalismo matemático.

O apêndice B mostra a representação abstrata de estados do agente MGM. Cada parâmetro apresentado neste apêndice interfere no processo de tomada de decisão no modelo proposto.

O apêndice C apresenta a descrição do *Framework* de aprendizagem por reforço empregado. Nele, são apresentadas as quatro classes e as duas interfaces que compõem o *Framework*. Elas são implementadas de forma abstrata, podendo ser utilizadas em vários problemas que se enquadrem nas características de aprendizagem por reforço.

Capítulo 2

Introdução sobre Sistemas Multiagentes

O estudo de sistemas multiagentes começou há aproximadamente duas décadas. Hoje esses sistemas são tanto objetos de pesquisa, quanto ítens importantes para aplicações industriais, comerciais e acadêmicas, Weiss (60).

Em geral, os computadores não são bons identificadores de ações que devem ser realizadas. Entretanto, são bons executores para as ações, quando elas são projetadas, antecipadas, planejadas e codificadas previamente. Para muitas aplicações, um computador obediente é totalmente aceitável, mas, em outros casos, são requeridos sistemas que possam decidir o que fazer por si mesmos, para alcançar seus objetivos.

A inteligência dos sistemas computacionais podem ser bem modeladas usando os conceitos de agentes inteligentes.

2.1 Conceituação de Agentes Inteligentes

Entre os pesquisadores de inteligência artificial, há vários pontos de vista sobre a definição de agentes inteligentes. Muitos estudiosos do assunto, como Russel e Norvig (44) e também Luger (30), acreditam que agentes inteligentes teriam comportamentos parecidos com o dos seres humanos no que diz respeito ao raciocínio e sua reação às situações do mundo à sua volta. Por seu turno, a definição mais geral de um agente como sendo um sistema computacional é baseada em autonomia, capacidade de interatividade e reatividade.

Para Weiss (60), *Agentes* são aplicações que podem operar com robustez, em ambientes que se modificam rapidamente e que necessitam de respostas precisas e rápidas aos eventos que podem ser inesperados. Eles conseguem reagir rapidamente e possuem características que os permitem atuar em situações não

programadas. Eles têm a capacidade de interagir com outros, sejam humanos, ou agentes.

Na opinião de Bradshaw (9), quando os agentes se comunicam com o objetivo de alcançar uma meta comum, ou negociar algo, estão, na verdade, trabalhando em um sistema multiagentes com toda infra-estrutura necessária à comunicação.

Agentes inteligentes são entidades computacionais autônomas. Eles agem sem intervenção de outros sistemas e podem ser vistos como perceptores do ambiente que se encontram por intermédio dos sensores e atuam nesse ambiente produzindo efeitos, Weiss (60). De acordo com Weiss, esses agentes são inteligentes quando operam de forma flexível e racional em uma variedade de circunstâncias ambientais a partir das informações que possuem e da capacidade efetiva e perceptiva. O comportamento flexível e racional é alcançado pelo agente na base de processos-chave, como resolução de problemas, planejamento, tomada de decisão e aprendizado.

Na visão de Russel e Norvig (44), um agente é uma entidade computacional como um programa ou um robô que pode ser enxergado ou percebido como agindo sobre um ambiente de maneira flexível e autônoma, e seu comportamento depende parcialmente de sua experiência. A figura 2.1 ilustra a idéia. Nela, o agente observa o ambiente por meio de seus sensores e interage com ele por intermédio das ações. No agente, existe uma camada de controle e outra que avalia os efeitos.

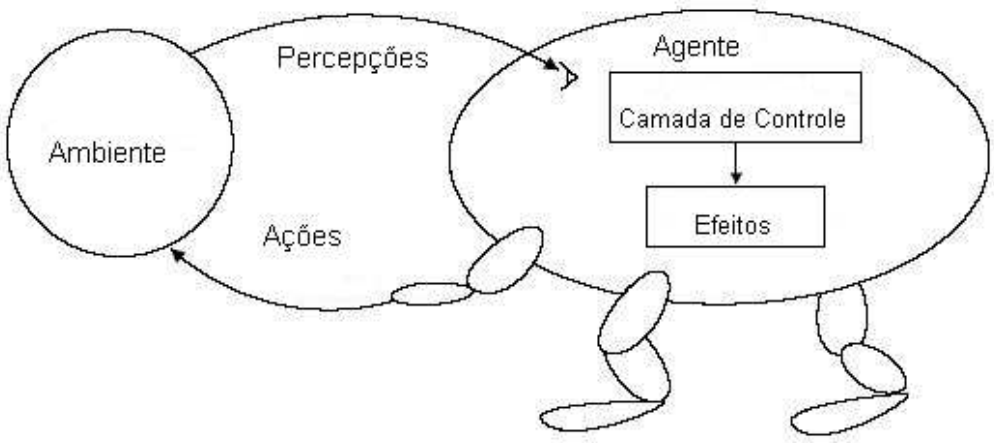

Figura 2.1: Arquitetura clássica de agentes inteligentes, Russel e Norvig (44).

Agentes de software são entidades computacionais persistentes e ativas que percebem, raciocinam, agem e se comunicam em um ambiente. Uma noção mais elaborada de agente inclui, na sua definição, atitudes de informação acerca do

mundo (conhecimento e crença) e proatitudes (intenções e obrigações) que guiam as ações do agente. Este tem a chamada arquitetura Crença, Desejo e Intenção, ou seja, *Belief, Desire, Intention* (BDI), em que a estrutura tripla - crença, desejo, intenção - executa um papel ativo no processo cognitivo do agente.

Na opinião de Rezende (40), quando se diz que o agente é *interativo*, significa que seu comportamento pode ser afetado por outro agente, que pode ser humano, ou por uma reação do ambiente.

Para Tanenbaum e Steen (52), os agentes podem decidir por si mesmos o que necessitam fazer de forma a satisfazer seus objetivos. Alguns atributos que caracterizam agentes inteligentes são:

- reatividade - capacidade de perceber o ambiente e respondê-lo em tempo hábil às mudanças ocorridas;
- proatividade - tomam iniciativas para poder atingir suas metas;
- habilidade social - interagem com outros agentes e, possivelmente, com seres humanos.

Segundo Weiss (60), um agente inteligente apresenta cinco características básicas:

- **Modelagem do Agente** - Um agente precisa de um modelo de seu mundo. Se esse mundo contém agentes, então pode ser benéfico modelar os outros agentes também.
- **Planejamento Multiagentes** - Agentes, em muitos casos, compartilham seus planos de modo a coordenar seus comportamentos ou atingir uma meta, usando os outros agentes para isso.
- **Relacionamento Social** - Agentes podem ter comportamento social com outros agentes. Por exemplo, se um agente executou um serviço para outro agente, o segundo pode estar com a obrigação de reciprocidade de algum modo. Se dois agentes estão trabalhando juntos para atingir uma tarefa, eles são tipicamente de cooperação.
- **Interação** - Agentes podem interagir. Em um mundo, multiagentes cuja interação não é predefinida podem necessitar de modelos dos outros agentes, para decidir como interagir para obter o sucesso ou falha. Isso pode impactar, de diferentes maneiras, o relacionamento social entre os agentes.

- **Comunicação** - Agentes podem comunicar-se para explorar a interação e assegurar a coordenação. Um agente pode persuadir outros a adotar seus planos.

Um sistema multiagentes precisa conter pelo menos um agente autônomo para viabilizar a geração das metas.

2.2 Tipos de Arquiteturas Abstratas em Sistemas Multiagentes

Existem várias formas de desenvolver agentes, mas quatro delas, por serem mais utilizadas, serão citadas a seguir, conforme apresenta Weiss (60):

1. **Arquiteturas Baseadas em Lógica** (*Logic Based Arquitectures*) - Nela, a tomada de decisão é vista como deduções lógicas. O processo envolve decidir qual ação executar e é reduzido a um teorema de prova. Essa abordagem possui a vantagem de ser clara semanticamente e permite usar conceitos de lógica e prova de teoremas (utilizados em inteligência artificial e ciência da computação há anos). No entanto, possui a desvantagem de que a arquitetura puramente lógica não satisfaz aos domínios sujeitos às restrições de tempo.

2. **Arquiteturas Reativas** (*Reactive Arquitectures*) - Sua principal característica se dá na abstenção de representações simbólicas e resolução baseada na relação entre percepção e ação do agente. Trata-se de uma arquitetura mais econômica em termos computacionais, pois é bem adaptável a ambientes que requerem performance em tempo real.

3. **Arquiteturas de Crenças-Vontades-Intenções** (*Belief-Desire-Intention*) - Nela, a tomada de decisão é vista como raciocínio prático de crenças e verdades sobre o mundo à sua volta e ocorre de acordo com as opções disponíveis para os agentes. Finalmente, utiliza as intenções e ações. Esse processo é similar ao raciocínio humano usado diariamente para decidir o que fazer.

4. **Arquitetura de Agentes em Camadas** (*Layered Agent Architectures*) - O processo de decisão é particionado em várias e diferentes camadas de tomada de decisão. Cada camada lida com o ambiente do agente em diferentes níveis de abstração. Essa abordagem prevê uma maneira natural de decompor as funcionalidades dos agentes.

2.3 Aspectos Importantes em um Projeto de Sistemas Multiagentes

Para Tanenbaum e Steen (52), as plataformas de computação modernas estão cada vez mais descentralizadas e heterogêneas, ou seja, seguem a filosofia de sistemas distribuídos. A complexidade nas aplicações está cada vez mais incrementada porque os sistemas não são mais isolados, e sim compartilhados, distribuídos, e requerem muito processamento em locais geograficamente distintos.

Na visão de Weiss (60), existem quatro técnicas mais importantes relacionadas com tamanho e complexidade dos sistemas de informação: modularidade, distribuição, abstração e inteligência. O uso de módulos inteligentes e distribuídos combina essas quatro características de acordo com a inteligência artificial distribuída.

Russel e Norvig (44) afirmam que os agentes operam e existem em algum ambiente que é tipicamente computacional e físico. O ambiente pode ser aberto, no qual novos agentes podem ser inseridos a qualquer momento, ou fechado. Embora alguns agentes possam trabalhar sós, na maioria dos casos precisam estar em conjunto com outros para alcançar um objetivo.

Quando existem vários agentes trabalhando juntos, faz-se necessária uma infra-estrutura computacional para que possam interagir coerentemente. Essa infra-estrutura inclui protocolos de interação que habilitam a conversação entre agentes e permitem a coordenação das atividades, Weiss (60).

A *cooperação* é a coordenação entre agentes que não competem por algo, enquanto a *negociação* é a coordenação entre agentes competitivos ou que não possuem interesses em comum.

Um *sistema multiagentes* é um sistema computacional em que dois ou mais agentes interagem ou trabalham em conjunto de forma a desempenhar determinadas tarefas ou satisfazer um conjunto de objetivos (60). Em sistemas nos quais os agentes são cooperadores, eles tentam conquistar o que um só não conseguiria. Em sistemas competitivos, os agentes disputam o que apenas um poderá obter.

A idéia em sistemas multiagentes consiste em coordenar o comportamento inteligente de um conjunto de agentes autônomos, cuja existência pode ser anterior ao surgimento de um problema particular. Esses agentes devem raciocinar a respeito das ações que devem tomar e sobre o processo de coordenação entre si. A organização do sistema está sempre sujeita a mudanças, visando à adaptação do sistema às variações no ambiente e no problema a ser resolvido.

Os agentes têm conhecimento do problema que devem tratar e encontrar uma

solução, sendo que eles podem obter esse conhecimento solicitando informações ao usuário e percebendo variações no ambiente por meio da comunicação com outros agentes. Eles são capazes de analisar um conjunto possível de soluções para atingir seus objetivos, dependendo de sua capacidade de tomar decisões para selecionar a melhor solução para o problema.

Existem alguns aspectos fundamentais na elaboração de projetos de sistemas multiagentes que devem ser considerados (60):

- **Organização** - Uma organização pode ser entendida como um conjunto de agentes que possuem compromissos mútuos e globais, compartilham crenças comuns e passam a ter intenções conjuntas quando atuam juntos para atingir o mesmo objetivo. O grupo de agentes desenvolve suas atividades em conjunto formando um time capaz de alcançar os resultados desejados.

- **Coordenação** - Os agentes interagem de forma que suas atividades sejam desenvolvidas e integradas no sentido de obter uma solução global; os integrantes do grupo de trabalho atuam de forma determinada e consistente. Em grupos de agentes, faz-se necessário o estabelecimento de convenções sociais a serem compartilhadas por todos os agentes, especificando como um determinado agente deve se comportar com relação aos membros da comunidade, quando os seus compromissos são alterados.

- **Negociação** - Permite que os agentes autônomos cheguem a um consenso em relação a um determinado problema. Nesse processo de negociação, pode acontecer modificação nos planos locais dos agentes envolvidos com a finalidade de chegar a uma decisão comum. Esse mecanismo de negociação também é utilizado na resolução de conflitos.

- **Cooperação** - É necessária em um sistema multiagentes, pois estabelece a maneira como eles demonstram suas necessidades a outros agentes com a finalidade de realizar uma determinada tarefa. Um agente, ao realizar uma tarefa, geralmente necessita da ajuda dos outros agentes que compõem o sistema, ou atua no auxílio dos outros para satisfazer o objetivo geral. Os agentes precisam, pois, saber como encontrar alguém com quem cooperar, visando à busca do objetivo.

- **Planejamento** - Planos devem ser regularmente monitorados e revisados de forma a responder à existência de eventos que fogem ao controle dos agentes envolvidos na aplicação. No planejamento multiagentes centralizado, o

planejamento é elaborado antes da execução das ações, e o agente elaborador recebe os planos dos agentes e os modifica de forma que os possíveis conflitos sejam removidos. No planejamento distribuído, cada agente elabora seus planos individuais e troca informações com outros agentes para identificar e resolver conflitos eventuais.

- **Comunicação e interação** - A interação entre agentes pode ocorrer por meio da comunicação ou em função da modificação do ambiente no qual eles estejam atuando. A intenção propicia a um conjunto de agentes combinar esforços na busca de solução de problemas distribuídos.

Diferenças entre Agentes Inteligentes e Objetos

A programação baseada em orientação a objetos tem sido muito utilizada nos últimos anos, conforme pode ser observado na quantidade de livros que descreve o assunto e são vendidos a cada ano: Deitel (13), Larman (28), entre outros. Da mesma forma, a adoção dos conceitos objetos e agentes inteligentes é freqüente.

Esta seção procura deixar claro tanto as similaridades, quanto as divergências que permeiam os conceitos: objetos e agentes inteligentes. A importância desta seção se faz pelo fato de que o paradigma de programação adotado na implementação deste trabalho é a orientação a objetos.

Objetos são definidos como entidades que encapsulam um estado, estão hábeis a executar ações ou métodos e se comunicam por meio de mensagens.

Existem semelhanças e diferenças significativas entre os agentes e os objetos. Ambos são autônomos, contudo os objetos têm controle de seu estado, e não têm controle de seu comportamento. Por exemplo, se um agente i requer ao agente j a execução de uma ação, o agente i pode ou não executá-la. A execução depende somente da decisão a ser tomada pelo agente. Em contrapartida, aos objetos não há outra escolha, a não ser a de executar a tarefa solicitada por meio de uma mensagem.

No paradigma de programação orientada a objetos, não existe integração de elementos importantes aos agentes, como interatividade, reatividade, proatividade e sociabilidade.

2.4 Considerações Finais

Os agentes inteligentes se mostram uma estratégia interessante a ser utilizada para problemas complexos, contudo, nem sempre, atendem aos anseios atuais. Esse fato ocorre em razão da dinâmica e incerteza presentes nos problemas a serem tratados, e, ainda, se não forem bem modelados, os agentes podem apresentar desempenho abaixo do esperado. Nesse sentido, mecanismo que os permita a melhoraria de performance, como a consideração do controle em nível meta, Raja (37) nos sistemas e também que os possibilitem se adaptarem ao dinamismo e incerteza, como a aprendizagem por reforço, Sutton e Barto (50) podem ser indicados em alguns casos.

Esta pesquisa utilizará os conceitos presentes neste capítulo para modelar um ambiente multiagentes.

Em linhas gerais, o ambiente multiagentes definido neste trabalho se caracteriza pela incerteza presente no ambiente de atuação dos agentes e pelo dinamismo presente. O conjunto de agentes trabalhará com base na filosofia de controle em nível meta, que será assunto do capítulo 4, e o modelo será mais bem esclarecido no capítulo 6, no qual é definido.

Capítulo 3

Aprendizagem por Reforço

Em abordagens tradicionais do aprendizado automático, geralmente os agentes aprendem por meio de exemplos de pares de entrada e saída, que indicam o comportamento do sistema, tendo como tarefa aprender uma função que poderia ter gerado tais pares, Russel e Norvig (44).

Os métodos tradicionais são apropriados quando existe uma espécie de "professor" fornecendo valores corretos, ou se a saída da função representa uma predição do futuro, que pode ser verificada pelas percepções do agente, no próximo passo da iteração (44).

Mas, quando se deseja que o agente tenha autonomia total, este terá de ser capaz de aprender, com base em outras informações, como por exemplo, recompensas, ou reforços fornecidos por uma "entidade crítica", ou ainda pelo ambiente em análise. Em alguns casos, é possível que o próprio agente determine suas recompensas através da observação da transição de estados que realize no ambiente, passando este a "experimentar", de maneira autônoma, o ambiente no qual está inserido.

Segundo Sutton e Barto (50), o aprendizado por reforço é uma abordagem da inteligência artificial que permite a uma entidade aprender, a partir de sua interação com o ambiente, por meio do conhecimento sobre o estado do indivíduo no ambiente, das ações efetuadas dentro deste ambiente e das mudanças de estado que aconteceram depois de efetuadas as ações, que é um conceito básico na área de aprendizado de máquina.

O aprendizado por reforço é indicado quando se deseja obter a política ótima, isto é, a representação do comportamento que o agente segue para atingir o objetivo, nos casos em que não se conhece *a priori* a função que modela tal política. O agente deve interagir com o seu ambiente diretamente para obter informações, que serão processadas pelo algoritmo apropriado, a fim de executar a ação que maximize a satisfação dos seus objetivos, nos estados do ambiente.

Em linhas gerais, o aprendizado por reforço consiste no aprendizado do mapeamento de estados em ações, de modo que um valor numérico de retorno seja maximizado. A princípio, o sistema não precisa conhecer as ações que devem ser tomadas, mas deve descobrir quais ações o levam a obter maiores valores de retorno, que podem ser vistos como imediatos locais, ou como retornos a longo prazo e globais, neste caso permitindo ao indivíduo alcançar um dado objetivo, Sutton e Barto (50).

3.1 Característica da Aprendizagem por Reforço

Existem quatro características comuns aos sistemas que aplicam conceitos de aprendizado por reforço, que são aprendizado pela interação, retorno atrasado, orientação ao objetivo e usufruto *versus* exploração. A seguir, cada uma dessas características será apresentada em detalhe, (50):

1. **Aprendizado pela Interação** - Esta é a característica que define um problema de aprendizado por reforço. Um agente de aprendizado por reforço atua no ambiente e aguarda pelo valor de reforço que o ambiente deve informá-lo como resposta perante a ação tomada. Ele deve assimilar, através do aprendizado, o valor de reforço obtido para tomar decisões posteriores.

2. **Retorno Atrasado** - Um máximo valor de reforço que o ambiente envia para o agente, que não quer dizer, necessariamente, que a ação tomada pelo agente foi a melhor. Uma ação é produto de uma decisão local no ambiente, sendo seu efeito imediato de natureza local, enquanto, em um sistema de aprendizagem por reforço, busca-se alcançar objetivos globais no ambiente. Assim, as ações tomadas devem levar a maximizar o retorno total, isto é, a qualidade das ações tomadas é vista pelas soluções encontradas em longo prazo.

3. **Orientação ao Objetivo** - Em aprendizagem por reforço, o problema tratado é considerado como um ambiente que dá respostas perante ações efetuadas, não sendo necessário conhecer detalhes da modelagem desse ambiente. Simplesmente existe um agente que atua dentro do ambiente desconhecido, tentando alcançar um objetivo. O objetivo é geralmente otimizar algum comportamento dentro do ambiente.

4. **Usufruto *versus* Exploração** - Em aprendizagem por reforço, os agentes vivem um dilema conhecido na literatura como usufruto *versus* exploração

(ou do termo em inglês, *The exploration X exploitation dilemma*), que consiste em decidir quando se deve aprender e quando não se deve aprender sobre o ambiente, mas usar a informação já obtida até o momento. Para que um sistema seja realmente autônomo, essa decisão deve ser tomada pelo próprio sistema. A decisão é fundamentalmente uma escolha entre agir baseando-se na melhor informação de que se dispõe no momento, ou agir para obter novas informações sobre o ambiente que possam permitir níveis de desempenho melhores no futuro. Como ambas formas trazem, em momentos específicos, benefícios à solução dos problemas, uma boa estratégia é mesclar as duas formas.

3.2 O Problema de Aprendizado por Reforço

Esta seção faz uma revisão bibliográfica de aprendizagem por reforço segundo Sutton e Barto (50).

Um sistema típico de aprendizado por reforço constitui-se basicamente de um agente inteligente interagindo em um ambiente via percepção e ação, conforme mostra a figura 3.1.

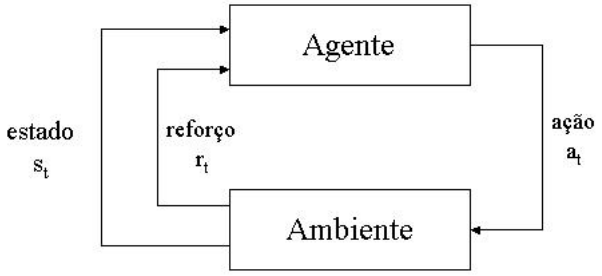

Figura 3.1: Figura clássica de aprendizagem por reforço, Sutton e Barto (50).

O agente percebe as situações atuais no ambiente, pelo menos parcialmente, e, com base nessas medições, seleciona uma ação a tomar no ambiente.

Em um sistema de Aprendizado por Reforço (AR), ou ainda, Reinforcement Learning (RL), o estado do ambiente é representado por:

um conjunto de variáveis de estado percebidas pelo agente, no qual o conjunto de combinações de valores dessas variáveis formam o conjunto de estados discretos do agente ou conjunto de agentes. E denota-se por s;

um conjunto de ações discretas, que, escolhidas por um agente, mudam o estado do ambiente. E denota-se por $A(s)$;

valor de transição de estado, que é passado ao agente por um sinal de reforço denominado ganho e apresenta valores tipicamente entre [0,1].

O objetivo do método é levar o agente a escolher a seqüência de ações que tende a aumentar a soma de valores de reforço, ou seja, auxilie no encontro da política π, definida como o mapeamento de estados em ações, que maximize as medidas do reforço acumuladas, ao longo do tempo.

O problema de aprendizado por reforço apresenta seis fatores fundamentais: ambiente, política, reforço e retorno, função de retorno, função valor-estado e função valor-ação. A seguir, serão apresentados os seis fatores fundamentais relacionados ao problema de aprendizado por reforço.

Ambiente

Todo sistema de aprendizado por reforço aprende um mapeamento de situações e ações, por experimentação em um ambiente dinâmico.

O ambiente no qual está inserido o sistema deve ser pelo menos parcialmente observável por meio de sensores, descrições simbólicas, ou situações mentais. Também é possível, entretanto, que toda informação relevante do ambiente esteja perfeitamente disponível. Nesse caso, o agente poderá escolher ações baseadas em estados reais do ambiente.

Política

Uma política expressa pelo termo π representa o comportamento que o sistema de aprendizado por reforço segue para alcançar o objetivo. Em outras palavras, uma política π é um mapeamento de estados s e ações a em um valor π(s,a). Assim, se um agente de aprendizado por reforço muda sua política, as probabilidades de seleção de ações sofrem mudanças e, conseqüentemente, o comportamento do sistema apresenta variações, à medida que o agente vai acumulando experiência, por causa das interações com o ambiente. Portanto, o processo de aprendizado, no sistema de aprendizado por reforço, pode ser expresso em termos da convergência até uma política $\pi^*(s, a)$ que conduz à solução do problema de forma ótima.

Reforço e retorno

O reforço é um sinal do tipo escalar (r_{t+1}), que é devolvido pelo ambiente

ao agente, assim que uma ação tenha sido efetuada e uma transição de estado ($s_t \rightarrow s_{t+1}$) tenha ocorrido. Existem diferentes formas de defini-lo. O reforço no ambiente pode ser gerado com funções de reforço, que intrinsecamente expressam o objetivo que o sistema de aprendizado por reforço deve alcançar. O agente deve maximizar a quantidade total de reforços recebidos chamada *retorno*, o que nem sempre significa maximizar o reforço imediato a receber, mas o reforço acumulado durante a execução total.

De modo geral, o sistema de aprendizado por reforço busca maximizar o valor esperado de retorno e, com isso, pode ser definido como uma função da seqüencia de valores até um tempo T final.

No caso mais simples, é um somatório como aparece na equação seguinte:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_{t+n} \tag{3.1}$$

Em muitos casos, a interação entre o agente e o ambiente não termina naturalmente em um episódio (seqüência de estados que chegam até o estado final), mas continua sem limite, como, por exemplo, em tarefas de controle contínuo. Para essas tarefas, a formulação do retorno é um problema, pois $T = \infty$ e o retorno que se deseja também tenderá ao infinito ($R_T = \infty$). Para esses problemas, foi criada a taxa de amortização (γ), a qual determina o grau de influência que têm os valores futuros sobre o reforço total. Assim, a expressão do retorno aplicando a taxa de amortização é expressa pela seguinte equação:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{K=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.2}$$

Onde, $0 \leq \gamma \leq 1$, se $\gamma \rightarrow 0$, o agente tem uma visão míope dos reforços, maximizando apenas os reforços imediatos, mas, se $\gamma \rightarrow 1$, a visão do reforço abrange todos os estados futuros, dando maior importância ao estado final, desde que a seqüencia R_t seja limitada.

Um sistema AR faz um mapeamento de estados em ações, baseado nos reforços recebidos. Assim, o objetivo do AR é definido usando-se o conceito de função de reforço, a qual é uma função dos reforços futuros que o agente procura maximizar. Ao maximizar essa função, o objetivo será alcançado de forma ótima. A função de reforço define quais são bons e maus eventos para os agentes.

Função de retorno

As funções de retorno podem ser bastantes complicadas, porém existem pelo menos três classes de funções freqüentemente usadas para criar as funções

adequadas de acordo com o tipo de problema:

- **Reforço só no estado final** - Nesta classe de funções, as recompensas são todas zero, exceto no estado final, em que o agente recebe uma recompensa real (por exemplo: +1) ou uma penalidade (por exemplo: -1). Como o objetivo é maximizar o reforço, o agente irá aprender que os estados correspondentes a uma recompensa são bons e os que levam a uma penalidade devem ser evitados.

- **Tempo mínimo ao objetivo** - Funções de reforço, nessa classe, fazem com que o agente realize ações que produzam o caminho ou trajetória mais curta, para um estado objetivo. Toda ação tem penalidade (-1), sendo que o estado final é 0. Como o agente tenta maximizar valores de reforço, ele aprende a escolher ações que minimizam o tempo que leva a alcançar o estado final.

- **Minimizar reforços** - Nem sempre, o agente precisa ou deve tentar maximizar a função de reforço, podendo, às vezes, aprender a minimizá-las. Isso é útil quando o reforço é uma função para recursos limitados, e o agente deve aprender a conservá-los, ao mesmo tempo que alcança o objetivo.

Função valor-estado

Define-se uma função valor-estado como o mapeamento do estado, ou par estado-ação em um valor que é obtido a partir do reforço atual e dos reforços futuros.

Se a função valor-estado considera somente o estado s, ela é denotada por *V(s)*. De outra forma, se é considerado o par estado-ação (s,a), a função valor-estado é denotada por função valor-ação *Q(s,a)*.

Uma vez que os reforços futuros mantém dependências das ações futuras, as funções-valor dependem também da política π que o algoritmo de aprendizado por reforço segue. Em um processo de decisão markoviano, define-se uma função valor-estado $V^{\pi}(s)$ dependente da política π. Como a equação:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} \tag{3.3}$$

$$V^{\pi}(s) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \tag{3.4}$$

onde a função $V^{\pi}(s)$ é o valor esperado do retorno para o estado $s_t = s$. Isto é, o somatório dos reforços aplicando a taxa de amortização γ.

Função valor-ação

Se considerarmos o par estado-ação, a equação para a função valor-estado $Q^{\pi}(s, a)$ será a seguinte:

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\} \tag{3.5}$$

$$Q^{\pi}(s, a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\} \tag{3.6}$$

Semelhante à anterior, só que considerando o reforço esperado para um estado s_t = s *e uma ação* $a_t = a$.

3.3 Fundamentos Matemáticos

Existem dois conceitos que devem ser conhecidos para facilitar a modelagem de um problema como um sistema de aprendizagem por reforço: a propriedade de Markov e os processos de decisão markoviano.

Propriedade de Markov

Quando a probabilidade de transição de um estado s para um estado s' depende apenas do estado s e da ação a adotada em s, isso significa que o estado corrente fornece informação suficiente para o sistema de aprendizado decidir que ação deve ser tomada. Quando o sistema possui essa característica, diz-se que ele satisfaz a propriedade de Markov, Clarke e Disney (12).

No caso mais geral, se a resposta em *t+1* para uma ação efetuada em t depende de todo o histórico de ações até o momento atual, a dinâmica do ambiente é definida pela especificação completa da distribuição de probabilidades, como mostra a equação a seguir:

$$P_r = \{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_t - 1, a_t - 1, ..., s_1, s_0, s_0\} \tag{3.7}$$

onde a probabilidade (P_r) do estado s_{t+1} é igual ao estado s' e o reforço r_{t+1} igual a r é uma função que depende de todos os estados, ações e reforços passados.

Se a resposta do ambiente em *t+1* depende apenas dos estados e reforços em *t*, a probabilidade da transição para o estado *s'* é dada pela equação abaixo:

$$P_{s,s'}{}^{a} = P_r = \{s_{t+1} = s' | s_t = s, a_t = a\} \tag{3.8}$$

A probabilidade de transição satisfaz às seguintes condições:

$$P_{s,s'} \geq 0, \quad \forall_{s,s'} \in S, \forall a \in A(s); \tag{3.9}$$

$$\sum s' \in S, \quad P_{s,s'} = 1, \forall a \in A(s) \tag{3.10}$$

A *propriedade de Markov* é de fundamental importância na aprendizagem por reforço, uma vez que tanto as decisões quanto os valores são funções apenas do estado atual, abrindo a possibilidade de métodos de soluções incrementais, onde se pode obter soluções a partir do estado atual e para cada um dos estados futuros, como é feito no método de programação dinâmica.

Processos de decisão markoviano

Segundo Raja e Lesser (39), um processo de decisão markoviano é definido como um conjunto de estados *s*, ∀

$$s$$

$\in S$, um conjunto de ações *A(s)*, um conjunto de transições entre os estados. Assim, dado um par de estado e ação, a probabilidade de o estado *s* passar a um estado *s'* é:

$$P_{s,s'}{}^{a} = P_r\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{3.11}$$

onde P_r é o operador de probabilidade, neste caso representa a probabilidade do estado s_{t+1} ser *s'*, sempre que o estado s_t for igual a *s* e a ação a_t for igual a *s*. Assim, a dependência de o estado seguinte s_{t+1} ser estado*s'* está relacionada a tomar a ação *a* no instante *t*.

De forma análoga, dados estado e ação atuais e estado seguinte *s'*, o valor esperado do retorno é:

$$R_{s,s'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{3.12}$$

onde *E* é o valor esperado do retorno r_{t+1}, sempre que o estado s_t no instante *t* passa a ser o estado *s'* no instante *t+1*.

Os valores de probabilidades $P_{s,s'}$ e o retorno esperado $R_{s,s'}$ determinam os aspectos mais importantes da dinâmica de um problema de decisão markoviano finito. Ele pode ser caracterizado como:

1. um ambiente que evolui probabilisticamente baseado em um conjunto finito e discreto de estados;
2. para cada estado do ambiente, existe um conjunto finito de ações possíveis;
3. para cada passo em que o sistema de aprendizado executar uma ação, é verificado um custo positivo ou negativo para o ambiente em relação à ação;
4. estados são observados, ações são executadas e reforços, relacionados.

Assim, para quase todos os problemas de aprendizagem por reforço é suposto que tenha a forma de um processo de decisão markoviano, desde que seja satisfeita a propriedade de Markov no ambiente. Nem todos os algoritmos de aprendizagem por reforço necessitam de uma modelagem como processo decisório de Markov inteiro do ambiente, mas é necessário ter pelo menos a visão do ambiente como um conjunto de estados e ações.

3.4 Métodos de Solução

Para solucionar o problema de aprendizagem por reforço (avaliar a política e encontrar a política ótima), existem três classes de métodos fundamentais: programação dinâmica, Monte Carlo e diferença temporal, Sutton e Barto (50).

Programação dinâmica

De acordo com (50), o termo "programação dinâmica" faz referência a uma coleção de algoritmos que podem ser usados para computar políticas ótimas, dado que se tem um modelo perfeito do ambiente como um processo decisório de Markov. Esses algoritmos, portanto, são considerados limitados, uma vez que, necessitam do modelo perfeito do ambiente.

Segundo Bellman (4), a programação dinâmica tem a vantagem de ser matematicamente bem fundamentada, mas exige uma modelagem bem precisa do ambiente como um processo de decisão markoviano.

A programação dinâmica é uma coleção de algoritmos que podem obter políticas ótimas sempre que exista uma modelagem perfeita do ambiente como um pro-

cesso decisório markoviano, isto é, como um conjunto de estados, ações, retornos e probabilidades de transição em todos os estados.

Os algoritmos clássicos de programação dinâmica são usados de forma limitada, uma vez que a modelagem perfeita do ambiente como um problema decisório markoviano exige grande custo computacional, porém fornece um bom fundamento para o conhecimento dos outros métodos usados na solução do problema de aprendizado por reforço e um padrão de comparação.

A idéia por trás da programação dinâmica é usar valores de funções para organizar e estruturar as buscas pelas melhores políticas.

A dinâmica do sistema é dada por um conjunto de probabilidades de transição de estado $P_{ss'} = P_r\{s_{t+1} = s', s_t = s, a_{t+1} = a\}$ e por um conjunto de reforços imediatos esperados,

$$R_{s,s} = E\{r_{t+1}|a_t = a, s_t, s_{t+1} = s'\}$$

para todo *s*, *s'* ∈ *S*, *a* ∈ *A(s)*.

Monte Carlo

O método Monte Carlo não precisa da modelagem do ambiente e se apresenta de forma simples em termos conceituais. Ele se baseia no cálculo da média dos retornos obtidos em seqüências. Para assegurar-se que exista um valor de retorno bem definido, o método Monte Carlo é utilizado apenas para tarefas episódicas, isto é, a experiência é dividida em episódios que de algum modo alcançam o estado final sem importar as ações que foram selecionadas (exemplo: jogo de xadrez).

Assim, somente depois da conclusão de um episódio, o valor de retorno é obtido e as políticas são atualizadas. Entretanto, não são visíveis quando a solução do problema é possível apenas de forma incremental, porque, para se atualizar, o método Monte Carlo exige que seja alcançado o estado final no processo que, portanto, pode ser lento.

Uma vantagem do método Monte Carlo é que, diferente do método de programação dinâmica, não necessita da informação completa do ambiente e ainda pode levar a um comportamento ótimo. Em Monte Carlo, deve-se gerar transições de estados, sem precisar todo o conjunto de distribuição de probabilidades para todas as possíveis transições, como é exigida pela programação dinâmica.

Diferença temporal

O método diferença temporal não exige um modelo exato do sistema e permite ser incremental, da mesma forma que o método de Monte Carlo.

O método diferença temporal é uma combinação de características dos métodos Monte Carlo com as idéias da programação dinâmica, que buscam estimar valores de utilidade para cada estado no ambiente. Em outras palavras, quanto mais próximo da convergência do método, mais o agente tem certeza de qual ação tomar em cada estado.

O aprendizado é feito diretamente a partir da experiência, sem a necessidade da modelagem completa do ambiente, como característica do método Monte Carlo, mas leva vantagem em cima deste, por atualizar as estimativas da função valor a partir de outras estimativas já aprendidas em estados sucessivos, sem a necessidade de alcançar o estado final de um episódio antes da atualização. Nesse caso, a avaliação de uma política é abordada como um problema de predição, isto é, estimar a função-valor V^{π} sob a política π.

3.5 Algoritmos de Aprendizagem por Reforço

Para que o agente decida qual a ação mais indicada a ser tomada, ele utiliza um algoritmo de aprendizagem por reforço. A seguir, nesta seção, dois desses algoritmos são discutidos: *Q-learning* e SARSA.

Q-learning

Um dos maiores avanços na área de aprendizado por reforço foi o desenvolvimento de um algoritmo baseado em diferenças temporais que dispensa política (*off-policy methods*) conhecido como *Q-learning* . A versão mais simples, *one-step Q-learning*, Watkins (56):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max{}_a Q(s_{t+1}, a_t) - Q(s_t, a_t)] \tag{3.13}$$

onde a função valor-ação $Q(s_t, a_t)$ é atualizada a partir do seu valor atual, o reforço imediato r_{t+1}, e a diferença entre a máxima função valor no estado seguinte (encontrando e selecionando a ação do estado seguinte que a maximize), menos o valor da função valor-ação no tempo atual. O fato de selecionar a ação que maximize a função valor no estado seguinte permite achar de uma forma simples a função valor estimada.

Uma característica do *Q-learning* é que a função valor-ação Q aprendida, aproxima-se diretamente da função valor-ação ótima Q^* sem depender da política

que está sendo utilizada. Tal fato simplifica bastante a análise do algoritmo e permite fazer testes iniciais de convergência. A política ainda mantém um efeito a determinar quais pares estado-ação devem ser visitados e atualizados, porém para que a convergência seja garantida, é necessário que todos os pares estado-ação sejam visitados e atualizados continuamente, por isso *Q-learning* é um método *off-policy*.

O algoritmo *Q-learning* propõe que o agente, ao invés de maximizar V^*, aprenda uma função de recompensa esperada com desconto Q, conhecida como função valor-ação. Esta função de estimativa Q é definida como sendo a soma de reforço recebido pelo agente por ter realizado a ação a_t no estado s_t em um momento t, mais o valor descontado de γ de seguir a política ótima daí por diante:

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma V^*(s_{t+1}) \tag{3.14}$$

Reescrevendo a equação na forma não-determinística, tem-se:

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \Sigma s_{t+1} \in S \;\; T(s_t, a_t, s_{t+1} V^*(s_{t+1}) \gamma \tag{3.15}$$

Pode-se concluir que:

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \Sigma s_{t+1} \in S \;\; T(s_t, a_t, s_{t+1} \max{}_{at+1} \;\; Q^*(s_{t+1}, a_{t+1}) \tag{3.16}$$

Outra conseqüência importante dessa formulação é que a política ótima pode ser obtida diretamente.

Seja Q_t *a estimativa de* Q$^*(s_t, a_t)$*noinstantet.Oalgoritmo*Q-learning*aproximaiterativamen*

Onde:

$\boldsymbol{s_t}$: estado atual

$\boldsymbol{a_t}$: ação realizada em s_t

$\boldsymbol{r(s_t, a_t)}$: é o reforço recebido após realizar a_t em s_t

$\boldsymbol{s_{t+1}}$: é o novo estado

γ : é o fator de desconto $(0 \leq \gamma \leq 1)$

α : é a taxa de aprendizagem. Está compreendida em $(0 \leq \alpha \leq 1)$. α pode ser definida como:

$$\frac{1}{visitas(s, a)}$$

sendo visitas(s,a) , o número de visitas já realizadas ao estado s, com ação a escolhida e realizada.

Uma propriedade importante deste algoritmo é que as ações usadas durante o processo iterativo de aproximação da função Q podem ser escolhidas usando qualquer estratégia de exploração (ou usufruto). Desde que cada par (estado, ação) seja visitado muitas vezes, a convergência dos valores de Q para Q^* é garantida, (56), porém essa convergência é extremamente lenta.

Uma estratégia para escolha das ações bastante utilizada em implementações do *Q-learning* é a exploração aleatória $\epsilon - Greedy$, na qual o agente executa a ação aleatória com probabilidade ϵ. Neste caso, a transição de estados é dada por uma regra de transição de estados, para a decisão, um valor q é sorteado e comparado com o valor epsilon; se é menor, o agente deverá explorar; caso contrário, deverá usufruir o conhecimento adquirido.
q - é um valor escolhido de maneira aleatória com distribuição de probalidade uniforme sobre [0,1] e $(0 \leq \epsilon \leq 1)$ e o parâmetro que define a taxa de exploração e usufruto: quanto menor é o valor de ϵ, menor a probabilidade de se fazer uma escolha aleatória.

a - é uma ação aleatória selecionada entre as ações possíveis de serem executadas no estado s_t.

A seguir, será apresentado em pseudocódigo o algoritmo Q-learning, retirado de (56):

Algoritmo: Q-learning

$\forall t \;\; Q_t(s,a) = random$
repeat
Selecione ação a de acordo com regra de transição de estados
Execute ação a
Receba o reforço r(s,a) e o observe o próximo estado s_{t+1}
Atualize o valores de Q(s,a) de acordo com a regra de atualização:
$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha[r(s_t, a_t)\gamma \max a_{t+1} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$
for all estado u **do**
Atualize os valores de V(u) de acordo com a regra de atualizaçao:
$V_{t+1}(u) \leftarrow V_t + \alpha[r(s,a) + \gamma V_t(s') - V_t(s)]e(u)$

end for
Atualize o estado s = s_{t+1}
until Critério de parada ser atingido

SARSA

O algoritmo SARSA pode ser compreendido como um algoritmo *Q-learning* que sofreu modificações, Sutton e Barto (50). Nele, propõe-se que a política seja aprendida em tempo de execução, estima-se o valor de uma política ao mesmo tempo que a usa para interagir com o ambiente. Para tanto, a equação de aprendizado usada no SARSA elimina a maximização das ações existentes no *Q-learning*, sendo que a regra fica:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha[r(s_t, a_t)\gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)] \tag{3.18}$$

Se a_{t+1} for escolhido segundo uma política gulosa, o algoritmo se torna igual ao *Q-learning* e $a_{t+1} = \max a_{t+1} Q_t(s_{t+1}, a_{t+1})$. Porém, neste método a ação a_{t+1} é escolhida de maneira aleatória (a partir de uma determinada distribuição de probabilidades), alcançando um rendimento melhor que o *Q-learning*, em casos onde o conjunto de ações é muito grande ou onde o ambiente apresenta apenas penalidades (50).

Para realizar o aprendizado *on-line*, a cada iteração estima-se Q_π a partir de π, ao mesmo tempo que se modifica a distribuição de probabilidades que define a escolha da próxima ação (ou seja, π).

Finalmente, o nome do algoritmo advém do fato de que a regra de aprendizado utiliza todos os elementos da quíntupla $(s_t, a_t, r, s_{t+1}, a_{t+1})$ que define a transição de um par estado-ação para o próximo, para atualizar a estimativa de Q. A seguir, será apresentado em pseudocódigo o algoritmo SARSA:

Algoritmo: SARSA

$\forall t \;\; V_t = random$
$e(S) = 0$
repeat
Visite o estado s de acordo com a poltica π
Execute ação a
Receba o reforço r(s,a) e o observe o próximo estado s_{t+1}
Atualize elegíveis:
e(s) = e(s) + 1
for all estado u **do**
Atualize os valores de V(u) de acordo com a regra de atualização:
$V_{t+1}(u) \leftarrow V_t + \alpha[r(s,a) + \gamma V_t(s') - V_t(s)]e(u)$

Atualize e(u) = $\gamma\lambda e(u)$
until Critério de parada ser atingido

3.6 Uso de Heurística para Acelerar a Aprendizagem por Reforço

A tendência no trabalho tomado como referência, Bianchi e Costa (6), é combinar as vantagens da aprendizagem por reforço com outras estratégias de maneira a torná-la mais rápida. Uma dessas estratégias pode ser o uso de heurísticas baseada no domínio do problema em análise.

A idéia é construir uma base de conhecimento prévia, que sugere ações a serem tomadas pelo agente, de acordo com o estado atual que ele se encontra.

Resultados experimentais, conforme apresentado por Bianchi e Costa (6), mostram que estratégias heurísticas resultam em um aumento de performance, no desempenho significativo do algoritmo AR utilizado.

Função heurística

A idéia é que uma função heurística H influencie as escolhas das ações a serem tomadas durante o processo de aprendizagem. A heurística será usada

somente quando uma escolha de ação deverá ser realizada. A intenção é não alterar a essência dos algoritmos clássicos de aprendizagem por reforço, preservando, assim, muitas de suas propriedades. Ao invés disso, os algoritmos são adaptados para operar de maneira mais rápida aos domínios.

Uma proposta interessante é a apresentada por (6). A função heurística é usada somente na regra de transição de estados, que escolhe a ação a_t a ser tomada no estado s_t.

A função H deverá fazer uma estimativa de pares estado-ação futuros, tomando como base a distância entre o desempenho atual e o desempenho ideal e tendo como ponto de partida o estado atual.

Uma estratégia para a escolha de ações é a exploração aleatória $\epsilon - Greedy$, na qual, além da estimativa das funções de probabilidade de transição T e da recompensa R, a função H é considerada. A regra de transição de estados proposta é dada pela figura 3.2, Regra de transição de estados de Bianchi e Costa (6).

$$\pi(s_t) = \begin{cases} a_{random} & \text{se } q \leq \epsilon, \\ \arg\max_{a_t} \left[\mathsf{F}_t(s_t, a_t) \bowtie \xi H_t(s_t, a_t)^\beta\right] & \text{caso contrário,} \end{cases}$$

Figura 3.2: Regra de transição de estados, Bianchi e Costa (6).

F: S X A $\rightarrow \Re$ - é uma estimativa de função valor que descreve a recompensa esperada acumulada. Por exemplo: se $F_t(s_t, a_t)$ for substituído por $Q_t(s_t, a_t)$, tem-se o algoritmo *Q-learning*.

H: S X A $\rightarrow \Re$ - é uma função heurística que influencia a escolha da ação. $H_t(s_t, a_t)$ define a importância de se executar a ação a_t estando no estado s_t.

$\bowtie$ - é uma função matemática que deve operar sobre números reais e produzir um valor pertecente a um conjunto ordenado (que suporte a operação de maximização).

ξ e β são variáveis reais usadas para ponderar a influência da função heurística.

q - é um valor escolhido de maneira aleatória com distribuição de probabilidade uniforme sobre [0,1], e $\in (0 \leq \epsilon \leq 1)$ é o parâmetro que define a taxa de exploração e usufruto quanto menor a probabilidade de se fazer uma escolha aleatória.

a - é uma ação aleatória selecionada entre as ações possíveis de serem executadas no estado s_t.

3.7 Considerações Finais

No modelo definido neste trabalho, a aceleração se baseará na combinação da função-valor do agente com uma heurística que será definida baseando-se no domínio. Espera-se, com essa abordagem, auxiliar o ambiente controlado em nível meta no processo de tomada de decisão e, assim, que ele ocorra de maneira mais rápida do que sem o uso dos meios que o acelerem. No modelo, a função valor-estado será alterada e combinada a função H e permitirá ao Módulo de Decisão e Controle decidir qual ação tomar em termos do valor *epsilon* sorteado. Assim, ela indicará uma ação de usufruto que deverá ser realizada.

A combinação adequada entre a função H e o usufruto tomou como base o estudo feito por Bianchi (6) apresentado neste capítulo.

O capítulo 4, a seguir, discute o tema controle em nível meta.

Capítulo 4

Controle em Nível Meta

Neste trabalho, o ambiente de atuação dos agentes é um ambiente controlado em nível meta, e a idéia central é aplicar o aprendizado por reforço com intuito de obter o melhoramento da performance, no processo de tomada de decisão dos agentes.

O controle em nível meta pode ser entendido como uma camada adicional inserida nos sistemas de informações, a qual atua sobre a camada de controle já existente neles. A intenção é acrescentar o raciocínio em nível meta, com limitações de custo computacional, sendo que ele será projetado para conseguir melhoras significativas, na atuação dos agentes individuais e nos sistemas multiagentes cooperativos. Entretanto, o enfoque principal deste trabalho é a aprendizagem.

Este capítulo buscará descrever os aspectos relevantes e essenciais para o entendimento geral da filosofia de controle em nível meta. Nele, não se pretende fazer uma cobertura ampla e completa do assunto. Assim, o capítulo abordará os aspectos essenciais e de relevância a esta pesquisa.

4.1 Descrição do Controle em Nível Meta

O controle em nível meta lida em ambientes "abertos", os quais os agentes disputam entre si por recursos limitados, Raja e Lesser (39). Tais ambientes se caracterizam por serem incertos e dinâmicos, e os agentes inteligentes que os operam são considerados complexos. Assim, os agentes "complexos" devem raciocinar sobre as ações que resolvem seus problemas locais, coordenando-se, quando necessário, com outros agentes, para que o esforço contínuo deles permita a realização das tarefas. Todo processo ocorre depois do planejamento e seleção da ação a ser executada.

Na visão de Raja e Lesser (39), as deliberações (ou do termo em inglês, *deliberation*) devem envolver computações e atrasos, que são originados a partir da

espera pela chegada da informação apropriada. A *deliberação* é a consideração explícita de várias alternativas possíveis de ação através de:

- geração de alternativas;
- escolha de uma dentre várias alternativas;
- raciocínio de um agente deliberativo sobre como alcançar o objetivo e encerrar suas atividades, quando seus objetivos forem atingidos.

O processo realizado, no controle em nível meta, é feito considerando a situação em análise: recursos limitados, incerteza sobre qual ação é a melhor a ser considerada e a limitação de se ter de trabalhar na realidade de tempo-real. Deve-se levar em consideração, pelos agentes inteligentes existentes no ambiente, que a qualquer momento, novas tarefas podem chegar. As tarefas apresentam tempo de término (ou do termo em inglês, *deadlines*) restrito e podem apresentar baixa utilidade ao sistema.

O controle em nível meta deve decidir quais ações deliberativas devem ser executadas e se deve deliberar ou executar as ações de domínio, que são resultado das ações deliberativas anteriores, Raja e Lesser (38).

As ações deliberativas são referenciadas, neste capítulo, como ações de controle. Para que seja possível a otimização, um agente deve ter conhecimento sobre o efeito de todas as combinações de ações, ao longo do tempo, que são intratáveis para algum problema de complexidade que é intratável.

Para Raja e Lesser (38), o principal foco do uso do controle em nível meta é como aproximar a seleção ideal, as seqüências e as ações de controle, sem despender esforços computacionais impraticáveis.

Segundo Raja (37), existem três classes de ações deliberativas, podendo ser de planejamento, seqüenciamento e coordenação.

As três classes de ações apresentam a característica de não serem triviais e requererem tempo de processamento, na ordem exponencial no número do domínio de ações.

Esquemas sofisticados que controlam as complexidades das ações podem ser utilizados nas implementações. Eles são exemplos da abstração de características importantes. A seguir, os três tipos de ações deliberataivas são descritos.

Primeiro tipo de ação deliberativa - É a coleta de informação que podem ser classificadas em duas possibilidades. A *primeira possibilidade* para coleta de informação é coletar dados sobre o ambiente, que inclui o estado dos outros agentes. Esta informação é usada pelo controlador em nível meta, para

determinar as ações de controle que são relevantes. Estas ações não utilizam o tempo de processamento local, mas atrasam o processo de deliberação em nível meta. A *segunda possibilidade* para a ação de coleta de informação é a determinação de fatores dos estados complexos dos agentes, que envolvem uma quantidade significativa de computação. Estes fatores podem, por exemplo, computar o tempo de ajuste, ou ainda a sintonia detalhada, a substituição e as informações de prioridade sobre as ações primitivas a serem executadas para completar as tarefas dos agentes. Os agentes devem tornar explícitas as decisões de controle em nível meta e determinar quais são os fatores complexos apropriados.

Segundo tipo de ação deliberativa - Envolve planejamento e escalonamento. *Planejamento* é o processo no qual o agente utiliza suas crenças sobre ações e suas conseqüências, para procurar por soluções de uma ou mais tarefas de alto nível, isto é, objetivos sobre o espaço possível de planos. Ele determina quais ações de domínio devem ser tomadas, para que sejam realizadas as tarefas. *Escalonamento* é o processo de decidir quando e onde cada uma das ações deve ser realizada.

Terceiro tipo de ação deliberativa - É a coordenação. *Coordenação* é o processo no qual um grupo de agentes realizam tarefas, em um ambiente, de maneira compartilhada. Para Lesser e Raja (38), a *coordenação* é um processo de negociação interagente, que estabelece compromissos em complementos de tempos, considerando as tarefas e os métodos.

O problema em sistema multiagentes é que eles não raciocinam explicitamente sobre o custo da computação nas deliberações, enquanto não as tenham executado, embora muitos sistemas não tenham maneira de compartilhar os recursos disponíveis para as ações de deliberações e domínios de ações. Um agente não age racionalmente, se ele falhar em dimensionar o custo de computar uma solução. Isso porque ele lida com ações sem significado operacional.

Um agente apresenta racionalidade, no contexto de recursos limitados, se ele maximiza a utilidade esperada, tendo como informação a computação necessária e os limites dos outros recursos.

A intenção do controle em nível meta é mostrar que o acréscimo de raciocínio, em nível meta, com limitações de custo computacional adicional, *overhead*, pode ser projetado com melhoras significativas de performance dos agentes individuais.

Nos sistemas multiagentes cooperativos, se recursos significativos são gastos no processo de tomada de decisão em nível meta, estas decisões devem ser toma-

das, somente se o uso dos recursos for compensatório. Em contrapartida, se o processo de raciocínio em nível meta tem custo computacional baixo, não existirá a necessidade de raciocínio em nível meta, de maneira explícita.

A idéia adotada é permitir que o controle em nível meta, com custo computacional limitado, permita aos agentes complexos resolver seus problemas, de maneira mais eficiente em ambientes abertos e dinâmicos.

O controle em nível meta mostra-se possível para computação, mediante o uso de uma representação abstrata dos estados do agente. A representação abstrata captura a informação crítica necessária para o processo de tomada de decisão. Nas políticas de controle em nível meta, existem a limitação de custo do controle em nível meta e o uso de aprendizado automático.

Agentes sofisticados que operam em ambientes abertos devem tomar decisões complexas, de controle em tempo real, para programar e coordenar as atividades de domínio. Tais decisões são tomadas no contexto de recursos limitados e considerando incertezas sobre os resultados das atividades. A execução de maneira otimizada das atividades computacionais, sem consumir muitos recursos no processo, é o alvo do controle em nível meta, para um agente que dispõe de recursos limitados.

O enfoque do controle em nível meta é prover distribuição efetiva de recursos computacionais e melhorar o desempenho de agentes individuais, em um sistema multiagente cooperativo, conforme afirmam Raja e Lesser (39). Isso é feito pela aproximação da solução ideal para decisões, em controle nível meta, pelos agentes de aprendizado por reforço.

A arquitetura do controle em nível meta apóia decisões em várias situações. Quando aceitar, retardar ou rejeitar uma nova tarefa. Quando é apropriado negociar com outro agente. Quando deve ser realizada nova negociação, no momento que uma tarefa de negociação falhar. Quanto esforço colocar em execução. Quando se deve raciocinar sobre uma tarefa nova. E se deve planejar novamente, no momento em que o desempenho de execução atual diverge do desempenho esperado.

O controle em nível meta é um modelo que usa raciocínio detalhado acerca dos custos de programação e coordenação dos agentes. Para tanto, uma representação abstrata do estado de cada agente é usado, por meio de estratégias heurísticas, com o intuito de se tomarem decisões de controle em nível meta. Uma aproximação, baseada em aprendizagem por reforço, aprende políticas de maneira automática.

4.2 Motivação do Uso do Controle em Nível Meta

Hierarquia de espaço de estados das decisões dos agentes

Um agente é uma entidade que recebe sensações, originadas do ambiente, e responde com ações que afetam o ambiente, causando efeitos, Russel e Norvig (44), conforme foi apresentado na seção 2.1. Na opinião de Sutton e Barto (50), um agente ideal será aquele que sempre age com o intuito de maximizar seus critérios de performance, não considerando somente benefícios locais, mas também os globais.

Na arquitetura clássica de agentes, conforme apresenta a figura 2.1 do Capítulo 2 - Arquitetura Clássica de Agentes, quando uma percepção é observada por intermédio dos sensores, a camada de controle é imediatamente disparada pelo estado corrente. A camada de controle determina como as percepções devem ser processadas e mapeadas em seqüências de ações de domínio.

As ações dos agentes, na arquitetura clássica, são de dois tipos: ações de domínio e ações de controle. As ações de execução, ou de domínio, são ações primitivas. As seqüências dessas ações alcançam vários objetivos, que afetam o ambiente.

A pesquisa usada como referência, Raja e Lesser (38), toma como referência uma visão simplificada das ações dos agentes, que somente considera o raciocínio sobre o processamento dos recursos necessários para que as ações possam ser completadas. O comportamento da execução de uma ação primitiva é caracterizado usando distribuições estatísticas, a quantidade de recursos utilizados, a possibilidade de completar com sucesso a ação e contribuir para a utilidade da tarefa.

Ações tomadas pela camada de controle são chamadas *ações de controle*. Os agentes são caracterizados pelo tipo de ações de controle que realizam: *agentes reflexivos* agem de maneira reativa, respondendo de maneira imediata às percepções; *agentes baseados em meta* agem de maneira a atingir seus objetivos; *agentes baseados em utilidades* agem de maneira a maximizar suas recompensas.

A camada de controle faz parte da arquitetura clássica de agentes e tem somente uma opção para o processamento de controle, com o custo fixado, em termos de uso dos recursos. O processamento do controle irá raciocinar sobre as ações de domínio, que precisam ser realizadas pelos efetuadores.

Para facilitar o raciocínio explícito sobre as ações de controle e seus custos, uma nova categoria de ações, chamadas *ações de controle em nível meta*, é intro-

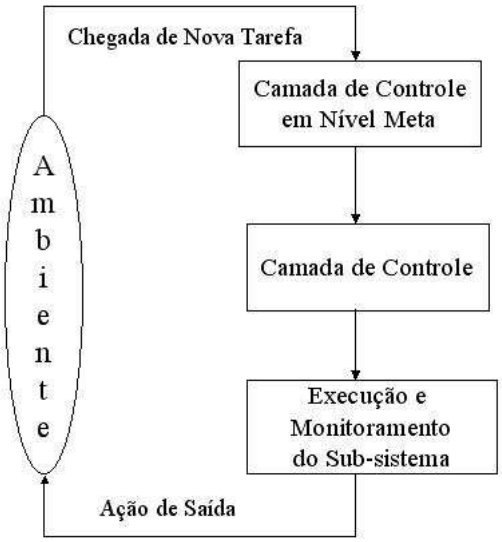

Figura 4.1: Fluxo de controle em um agente racional, Raja (37).

duzida, conforme pode ser observado na figura 4.1. Nela, pode ser observada a presença de três níveis: camada de controle em nível meta, camada de controle e, por último, execução e monitoramento do subsistema. Na figura, a chegada de nova mensagem aciona o controle em nível meta.

A arquitetura de agentes clássica é extendida com controles em nível meta que raciocina, com o auxílio da aprendizagem por reforço, sobre as ações de controle e verifica às maneiras alternativas para realizá-las.

A chegada de disparos perceptivos, na camada de controle em nível meta, determina quais tarefas os agentes devem realizar. A camada de controle dos agentes determina como as tarefas são escolhidas, serão processadas e mapeadas, em seqüências de ações.

As ações de controle podem ser descritas em um ou mais caminhos. Um caminho é baseado na visão das ações de controle, como um algoritmo que pode ser interrompido, em qualquer ponto, para obter a solução. O uso dos recursos pode ser controlado pela determinação de quando parar o algoritmo. O outro caminho é o uso de diferentes algoritmos que realizam ações de controle, representando o compartilhamento em termos de uso dos recursos e utilidade com precisão.

As ações de controle em nível meta otimizam a performance mediante a escolha do domínio de seqüenciamento e das ações de controle. Para tanto, incluem-se as ações de acordo com a forma como as tarefas são tratadas, levando em consideração seus prazos de execução, *deadlines* e também a quantidade de processamento a ser alocada. A quantidade de processamento a ser alocada deve

ser requerida previamente, assim como outros recursos de domínio e ações de controle, no tempo apropriado.

O controle em nível meta pode ser também visto como um problema de decisão seqüencial. A essência de problemas de decisão seqüencial é que as decisões são tomadas tendo como referência os efeitos em termos imediatos e distantes e a escolha da melhor ação corrente. Essa ação corrente depende criticamente do tipo das ações futuras e também das ações que serão tomadas nos pontos de decisão no futuro, justificativa para o uso de aprendizagem. A limitação dos recursos dos agentes causa a escolha de ações em nível meta correntes, que influenciam na disponibilidade dos recursos para escolhas de ações no futuro.

O controle em nível meta efetivo necessita das informações de performance alcançadas, no passado para prever sobre o futuro, e também, para não tomar decisões cegas, em cada ponto do processo de tomada de decisão.

Razões que justificam a dificuldade do problema

As dificuldades em lidar com ambientes incertos e dinâmicos são conseqüência da:

1. complexidade da informação, que caracteriza o estado do agente ou dos outros agentes que interagem entre si;
2. variedade de resposta com custos diferenciados, em termos da disponibilidade dos parâmetros;
3. prazos de execução, associados com as tarefas;
4. média alta de incerteza, causada pela chegada de tarefas, de maneira não determinística, e ações de domínio primitivas;
5. decisões que são freqüentemente não-observáveis, de maneira imediata, e provavelmente têm efeitos de decréscimo significantes.

No controle em nível meta proposto por Raja e Lesser (38), os agentes são capazes de possuir múltiplas tarefas e objetivos concorrentes. Cada tarefa é representada usando a estrutura de tarefa com prazo de execução e a utilidade potencial, que pode ser atingida como resultado da completude.

A estrutura das tarefas descreve uma ou mais das possibilidades para que as tarefas sejam executadas. Tais alternativas são expressas, como uma hierarquia de abstração, que são como instanciações de ações básicas. As instanciações de ações primitivas são chamadas de *ações de domínio*.

As ações de domínio são caracterizadas por apresentarem qualidades esperadas e distribuição de duração, podendo ser escalonadas e executadas. As informações sobre os relacionamentos das tarefas indicam quão básica são as ações, ou quão abstrata é a tarefa.

As informações sobre os relacionamentos que afetam as características, qualidade e tempo, mediante a estrutura das tarefas. As características dos recursos, os quais as tarefas estão em consumo, são também inseridas nas estruturas das tarefas.

Classificação das decisões em nível meta

Existem cinco tipos de disparos que acionam o controle em nível meta:

1. chegada da nova tarefa, vinda do ambiente;
2. presença de uma tarefa, no conjunto de tarefas atuais, ou seja, que estão no escalonamento corrente e requeiram negociação com um agente que não é local;
3. falha de negociação para atingir um compromisso;
4. decisão de escalonar um novo conjunto de tarefas, ou permanência das atuais;
5. desvio significante da performance esperada.

O controlador em nível meta é invocado quando um dos cinco disparos ocorre. A escolha de qual evento de disparo foi deliberado em conseqüência do domínio de alocação de tarefas e de outros domínios similares. Esses eventos ocorrem freqüentemente em horizontes finitos com determinadas considerações.

O controlador em nível meta é definido, de maneira específica, como um mecanismo baseado em disparos, e não como um componente que é invocado em intervalos de tempo bem definidos, pelas seguintes razões: é uma ativação periódica que pode ocorrer em ambientes restritos e lida com um custo computacional desnecessário.

O controle em nível meta pode acontecer, de maneira que não seja freqüente e efetiva, nos vários ambientes dinâmicos. O seu mecanismo baseado em disparo é uma solução geral que manipula o espectro inteiro do ambiente, sem varrê-lo por completo. Por conseguinte, um mecanismo baseado em disparo é apropriado para domínios nos quais as atividades requeiram controle em nível meta e

não chegaram, necessariamente, em forma de distribuição uniforme. Se as atividades chegaram recentemente, métodos de ativação periódicas, que não devem ser freqüente, durante períodos recentes, devem ser aproveitados em períodos de inatividade.

Um exemplo de evento de disparo, que não foi incluído, é a situação em que uma, dentre várias ações primitivas, sai fora do controle e continua a execução. Nesse caso, o tempo de término é esperado como baixo uso de alocação dos recursos. Embora o evento esteja lidando com a baixa performance, não adiciona disparos por duas razões: não ocorre freqüentemente; e, desde que outros disparos ocorram com muita freqüência, o controlador em nível meta será invocado, brevemente, ao invés de posteriormente, e ainda será reconhecido o baixo uso dos recursos e será abortada a ação que aconteceu com erros.

Algumas características de problemas de domínios, como o tipo controle em nível meta, descrito neste capítulo, devem fornecer melhorias na performance - é quando o agente internamente gera um novo objetivo como resultado das percepções observadas. Esses objetivos podem ser revisados, como resultado das ações de percepções internas e seus efeitos no ambiente. As análises dos objetivos encadeiam, por meio da camada de controle, o processo no qual o agente recebe um objetivo, ou conjunto de objetivos, sendo os mesmos entradas e saídas de uma seqüencia de métodos executáveis, com restrições no tempo de início e término e na utilidade esperada. O controle em nível meta é um processo de decidir entre as seguintes opções:

- descartar o objetivo e não analisá-lo;
- atrasar a análise do objetivo;
- raciocinar sobre a quantidade de objetivos e prosseguir na análise deles;
- determinar o contexto da análise do objetivo - se analisar um objetivo único, ou vários com uma perspectiva de agente única, ou ainda analisar os objetivos únicos, ou múltiplos, no contexto para facilitar as metas dos agentes.

O controle em nível meta é útil em situações que a análise dos objetivos são caras, em outras palavras, nas quais o custo acumulado afeta a performance dos agentes. O controle em nível meta é útil também quando o custo da análise de um objetivo é significantemente mais caro que o custo das ações de controle em nível meta. Ele também é útil quando uma escolha tem de ser feita sobre o tipo de análise do objetivo e as opções para análise do objetivo têm custos diferentes, de maneira significativa, e produz resultados com utilidades diferentes.

Em algumas situações, o controle em nível meta deve ser visto como efetivo e, ainda como uma alternativa não tão cara, para tomar decisões em análise de objetivos mais caros. Por exemplo, quando um objetivo chega, e o seu prazo para término é apertado, o controle em nível meta irá rapidamente, de maneira não tão cara computacionalmente, determinar qual objetivo deverá ser descartado e, portanto, desconsiderado. A análise do objetivo deve tomar a mesma decisão depois de completar as computações que têm custos próprios associados. Na falta do controle em nível meta, estes custos podem contribuir para a degradação da performance.

4.3 Fluxo de Controle da Arquitetura do Agente

Os componentes da arquitetura do agente são o ambiente, a camada de controle em nível meta, camada de controle e a camada executora. A camada de controle consiste de diferentes componentes de escalonamento e negociação. A camada executora é o subsistema de execução.

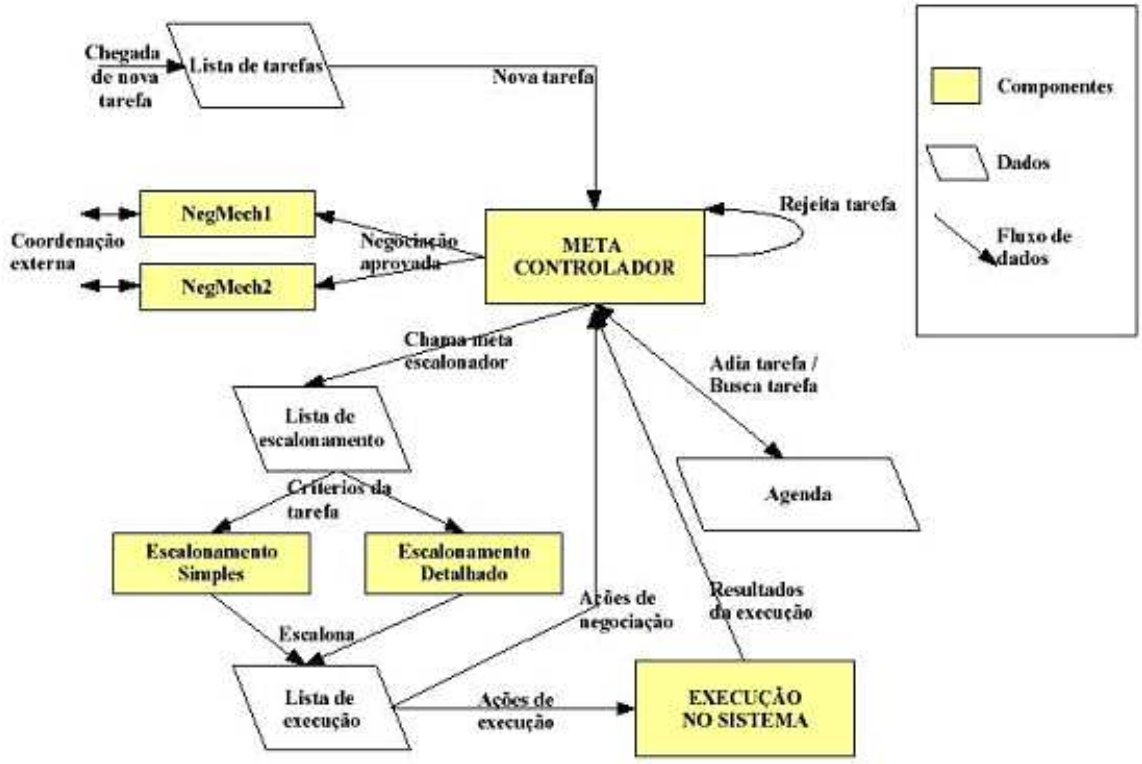

Figura 4.2: Controle em nível meta, Raja (37).

A figura 4.2 mostra o controle de fluxo no interior da arquitetura. Nessa com-

plexa arquitetura, os componentes de controle, tais como escalonadores, componentes de negociação e subsistema de execução, interagem com o componente de controle em nível meta. Tanto a camada de controle em nível meta, quanto os componentes da camada de controle podem influenciar no processo de tomada de decisão do agente. Há várias estruturas de dados que ajudam a manter a trajetória para o estado do agente: lista de novas tarefas, lista de agenda, lista de escalonamento e lista de execução.

A lista de novas tarefas contém as tarefas que acabaram de chegar do ambiente para o agente.

A lista de agenda é o conjunto de tarefas que chegaram ao agente, mas o raciocínio sobre como realizar as tarefas está atrasado. Nesse caso, as tarefas ainda não podem ser escalonadas.

A lista de escalonamento é o conjunto de tarefas de alto nível escolhidas para serem escalonadas e executadas.

A lista de execução é um conjunto de ações primitivas que foram escalonadas para realizar tarefas de alto nível em execução, ou que ainda serão executadas.

O ambiente consiste de um gerenciador de tarefas que gera tarefas para agentes individuais, baseado em um modelo de chegada. O controle em nível meta é invocado quando uma nova tarefa chega ao agente, mesmo se o agente está no meio da execução de outra tarefa. O subsistema de execução é invocado sempre que o agente tem de agir no ambiente. Estas ações podem, ou não, ter recompensas imediatas. Quando uma ação completa sua execução, o subsistema de execução envia as características da execução para o controle em nível meta, que é também o subsistema de monitoração.

A camada de controle consiste de escalonadores e protocolos de negociação. Os dois escalonadores, simples e complexo, diferem em seus perfis de desempenho. Adicionalmente, o escalonador complexo é parametrizado para que o nível de esforço e a quantidade de folga inserida no escalonamento sejam variável.

O escalonador simples

O escalonador simples é invocado pelo controle em nível meta e recebe a estrutura da tarefa e os critérios a serem atendidos como entradas. Ele usa informações abstratas, pré-computadas sobre a tarefa, para selecionar o escalonamento apropriado que atenda aos critérios definidos. Uma abstração de tarefa é uma estrutura de dados que captura a informação de diversas maneiras, para alcançar uma tarefa associada.

Abstrações de tarefas apóiam o controle reativo para situações altamente restritas no tempo. Quando um agente deve escalonar uma tarefa, mas não tem os recursos, ou o tempo, para chamar o escalonador complexo, as informações pré-computadas sobre os possíveis escalonamentos, a estrutura da tarefa, pode ser usadas para fornecer um escalonamento razoável, mas freqüentemente não ótimo.

O agente acumula conhecimento sobre todas as tarefas que é capaz de executar mediante análise *off-line*, em cada tarefa. Esse processo *off-line* constrói escalonamentos potenciais, em forma de uma seqüência linear de ações primitivas. Cada seqüência têm características de desempenho associadas, tais como distribuição de expectativa de qualidade, distribuição de expectativa de duração e incerteza sobre a expectativa de duração para alcançar tarefas de alto nível. Essas características de desempenho são descobertas pela busca sistemática sobre o espaço dos critérios a serem atingidos. A abstração da tarefa esconde os detalhes desses escalonamentos e fornece somente informações de alto nível necessárias para fazer escolhas em nível meta.

O escalonador complexo

O escalonador complexo é um processo de tempo real suave, que não envolve intenso uso dos recursos computacionais, cujo objetivo é encontrar um caminho de execução por uma cadeia de tarefa hierárquica, tal que o escalonamento resultante satisfaça certos critérios de projetos, como prazos finais de tempo real, limites de custo e preferências de utilidade.

No escalonamento complexo, uma linguagem foi projetada para o tratamento de um problema do tipo seqüenciamento seleção-de-ações, sendo que o processo seleciona um subconjunto de ações primitivas dentre um conjunto de ações candidatas. Depois disso, as ações devem ser ordenadas de forma que o resultado seja um escalonamento fim-a-fim das atividades do agente. O agente conhece as situações no contexto específico considerando os critérios a serem atendidos e um conjunto de parâmetros de escalonamento (entradas). E, assim, um escalonamento é retornado como uma sucessão de ações primitivas.

4.4 Heurísticas usadas nas Decisões do Escalonador em Nível Meta

Para tornar viável o uso do modelo que emprega controle em nível meta, duas heurísticas foram propostas: a estratégia simples e a estratégia sofisticada.

Tanto a estratégia simples, quanto à estratégia sofisticada usam características de estado e servem como um teste efetivo das características de estado do controle em nível meta.

As duas estratégias diferem entre si porque a simples (do termo em inglês, *naive*) não utiliza informação sobre tarefas que possam chegar no futuro durante o processo de raciocínio. Ela toma decisões míopes, ou seja, que só verificam os benefícios locais, e não os globais. Já a estratégia sofisticada toma decisões baseando-se nos eventos futuros que estão disponíveis, isto é, são analisados os benefícios globais.

As abordagens estratégia simples e sofisticada levam em consideração os agentes locais e o conjunto de agentes presentes no sistema multiagente.

4.5 Protocolos de Negociação

Os protocolos de negociação possibilitam ao controlador em nível meta estabelecer acordo entre os agentes que o compõem. Existem dois tipos de protocolos de negociação: NegMech1 e NegMech2, Raja (37). Eles serão descritos nos parágrafos seguintes desta seção.

A complexidade do protocolo de negociação é proporcional ao valor da tarefa negociada e da folga estabelecida no escalonamento. Esta folga disponível em outros agentes é um bom indicador de que a tarefa pode ser escalonada para execução. A seguir, os dois protocolos são apresentados:

1. NegMech1 é um protocolo de negociação de custo computacional baixo, cuja análise pode ser feita de maneira curta. De forma simples, o protocolo NegMech1 pode ser entendido como "um mecanismo que funciona no aspecto tudo ou nada". Uma proposta única é enviada, e uma resposta única é recebida. Apesar de ter as vantagens apresentadas acima, ele apresenta poucas chances de sucesso em relação ao protocolo NegMech2.

2. NegMech2 é um protocolo de negociação com múltiplos passos, que tenta concluir uma negociação no decorrer de uma seqüência de propostas para que se obtenha consenso, ou que o prazo de acordo se esgote. É mais caro computacionalmente que o protocolo NegMech1. O elevado custo ocorre em função dos atrasos no início da negociação, pelas sobrecargas na computação e pela comunicação envolvida na negociação; contudo tem grande chance de sucesso.

A seleção de qual será o protocolo mais indicado a ser utilizado em determinada situação dependerá do ganho relativo associado à tarefa e da probabilidade

de um outro agente realizar a tarefa.

4.6 Estado do Agente

O Controle em nível meta usa o estado atual do agente para tomar a decisão apropriada. O modelo no qual se baseia este capítulo, Raja e Lesser (39), faz distinção entre o estado real do agente e a representação abstrata, que captura somente algumas informações sobre o estado atual.

A análise do estado real do agente é muito complexa e resultaria em uma explosão combinatória, até mesmo para cenários simples. A representação abstrata reduz a doze características básicas, que o controle em nível meta deve considerar. Elas serão apresentadas a seguir:

1. o *status* das listas;
2. a boa utilidade de uma nova tarefa;
3. tensão do prazo de execução de uma nova tarefa;
4. a boa utilidade do escalonamento atual;
5. tensão do prazo de execução do escalonamento corrente;
6. chegada de uma nova tarefa com alto valor de utilidade;
7. quantidade de folga, em um escalonamento local;
8. desvio da performance esperada;
9. custo de descartar uma tarefa;
10. quantidade de folga em escalonamentos de outros agentes;
11. relação entre fragmentos de folga em escalonamentos locais com novas tarefas;
12. relação entre fragmentos de folga em agentes não locais com novas tarefas.

Cada característica pode ter um, de quatro valores: ALTO, MÉDIO, BAIXO e NÃO-INFORMADO. Assim, o tamanho do espaço de pesquisa se reduz a $4^{12} = 2^{24}$, que é cerca de 17 milhões de estados.

4.7 Modelo Formal

A estratégia heurística tem como enfoque principal a abordagem teórica do raciocínio de controle em nível meta, em sistemas multiagentes (39). A seguir, será feita a definição de uma formulação teórica de um problema de controle em nível meta.

1. Dado S como sendo um conjunto de estados de um agente, e $s_i \in$ S denota o estado do agente em um estágio i, sendo que i pode ser 0,1,2,3, ...,n.

2. A é o conjunto de possíveis ações de controle e $a_i \in$ A, a ação tomada pelo agente no estado s_i. Ações de controle não afetam, de maneira direta, a utilidade atingida pelos agentes, contudo afetam o estado interno deles. Trata-se de ações que consomem tempo e têm somente efeitos indiretos no mundo externo. As ações de controle são seguidas por intermédio da utilidade de execução, alcançadas nas ações de domínio. Estas ações de domínio são diretamente o resultado de ações de controle, no instante corrente e nas ações precedentes. As ações que não são explicitamente representadas no modelo são conseqüências das ações de controle.

3. A política π é uma descrição de comportamento do sistema. Uma política estacionária no controle em nível meta, $\pi : S \rightarrow A$, especifica, para cada estado, uma ação de controle a ser tomada. A política é definida para um ambiente específico. Um ambiente é definido por três distribuições descritas, pelo tipo da tarefa, taxa de chegada da tarefa e estreitamento do prazo de execução da tarefa. O controle em nível meta é um processo de decisão para escolher e seqüenciar ações de controle. Neste estudo, existem cinco eventos que disparam o processo de controle em nível meta. A ocorrência de algum desses disparos interrompe alguma outra atividade, o agente passa a ser automaticamente inserido e o controle é deslocado para o controlador em nível meta.

4. s_j é o novo estado em que o agente se encontra. Este estado atual, ao ser interrompido por um evento que requeira raciocínio, em termos de controle em nível meta, as ações serão consideradas de controle, quando estão sendo executadas segundo $\pi(s_i)$ e também, as ações de domínio que seguem a política $\pi(s_i)$.

5. R($s_i, \pi(s_i), s_j$) é a recompensa obtida no estado s_j, como conseqüência de tomar a ação de controle $\pi(s_i)$, no estado s_i e a execução no domínio de

ação que segue $\pi(s_i)$. A recompensa é o valor acumulado das tarefas e das ações de domínio, que são completadas entre transições de estado. Quando os valores são atingidos pelas tarefas associadas às incertezas, as funções de recompensa são representadas como uma distribuição de probabilidades.

6. $\mathrm{U}_\pi(s_i)$ é a utilidade do estado s_i sob a política π.

7. $\mathrm{P}(\mathrm{s}_j|\pi(s_i))$ é a probabilidade de o agente estar em um estado s_i, como resultado de tomar a ação $\pi(s_i)$, que é a ação sugerida pela política π ao estado s_i.

O modelo acima define um processo de decisão markoviano finito. De acordo com a teoria da decisão, uma ação ótima é aquela que maximiza a utilidade esperada pelo agente e é dada por:

$$E[U_\pi(s_i)] = E[\sum_{j=1}^{j} \gamma^j R(s_i), \pi(s_i), s_j)] \tag{4.1}$$

onde $\gamma \in [0, 1]$ é um parâmetro conhecido como *taxa de desconto* que determina o valor presente de utilidade de ganho futuro. Isso pode ser computado pela seguinte equação:

$$E[U_\pi(s_i)] = \sum {}^{n}{}_{j=1} P(s_j|\pi(s_i))[R(s_i), \pi(s_i), s_j) + \gamma^j E[U_\pi(s_i)]] \tag{4.2}$$

O problema de controle em nível meta é encontrar a melhor política π^* que maximiza o retorno esperado para todos os estados, Raja e Lesser (39). Tal política ótima pode ser encontrada usando programação dinâmica e aprendizado por reforço. Esses métodos irão implicitamente determinar a probabilidade de transição e a função de reforço, definida anteriormente.

Neste estudo, a complexidade do espaço de estados tem dificuldade de encontrar a política ótima. Então, aproxima-se a política de controle em nível meta, usando uma abstração da representação do espaço de estados, que irá capturar informação suficiente, no processo de tomada de decisão.

A camada de controle em nível meta será foco deste trabalho. As camadas de controle e execução serão tratadas no restante do texto como camadas abstratas que foram implementadas em situações em que alguns detalhes foram desconsiderados. Isso ocorreu no intuito de ser possível abordar o capítulo de maneira satisfatória, sem a inserção de grande quantidade de detalhes. Elas foram

apresentadas neste estudo no sentido de se abordar, de maneira geral, o que vem a ser controle em nível meta.

O capítulo 5, a seguir, apresenta o tema Gerência de Tráfego Aéreo.

Capítulo 5

Gerência de Tráfego Aéreo

Um importante fator que dificulta o escalonamento em tráfego aéreo é a presença de incerteza no processo. Tráfego aéreo é um ambiente estocástico. Fatores como mau tempo, congestionamento e demanda de recursos maior que disponibilidade ocorrem de forma nem sempre previsível. Assim, soluções que trabalham com processos estocásticos definem melhor esse ambiente.

Segundo Bian et al. (5), em razão do grande número de incertezas e imprevisibilidade de eventos que influenciam o ambiente de tráfego aéreo, o replanejamento das tabelas de vôos é uma atividade de vital importância para as companhias aéreas e precisa ser considerado em bases diárias. O replanejamento ocorre porque raramente se pode executar precisamente o planejamento original, portanto revisões precisam ser realizadas tempestivamente, a fim de garantir o provimento dos serviços de forma eficiente e segura.

Outro aspecto que também gera dificuldades ao escalonamento em tráfego aéreo está ligado ao fato de a demanda ser maior que a disponibilidade, em razão do aumento do número de vôos a cada ano. A discrepância entre demanda e disponibilidade de recursos contribui para a necessidade de se investir em estratégias que visem ao uso mais eficiente dos recursos disponíveis. Em outras palavras, a gerência de fluxo de tráfego aéreo, ou ainda *Air Traffic Flow Management* (ATFM), é útil no sentido de buscar melhora na situação indesejada.

Nesta pesquisa, um tipo de solução ATFM é usado como referência para o entendimento da Distribuição de Informação em Aeroportos. Ela é um sistema multiagentes para sincronização e gerenciamento de fluxo de tráfego aéreo em tempo real, encontrada em Dib (14).

5.1 Conceitos de Gerenciamento de Fluxo de Tráfego Aéreo

A sincronização de tráfego aéreo em tempo real foi definida para oferecer melhoria no fluxo do próprio serviço, ou ainda para áreas em que a demanda excedia a capacidade disponível dos sistemas *Air Traffic Control* (ATC) em um certo tempo.

Para Targa et al. (53), o termo AFTM se aplica a toda atividade relacionada ao gerenciamento do fluxo de tráfego aéreo, garantindo que todos os vôos sejam efetuados de forma rápida, segura, ordenada e econômica, permitindo que a totalidade da demanda de tráfego aéreo controlada, em uma determinada área, seja compatível com a capacidade do sistema ATC.

Na opinião de Targa et al. (53), os objetivos em ATFM são evitar sobrecargas no sistema demanda *versus* capacidade garantir uma distribuição regular do tráfego aéreo otimizando o seu fluxo (otimização mais eficaz do espaço aéreo e aeroportos), fornecer apoio aos órgãos de controle e garantir a segurança de suas operações, relacionar-se com os sistemas dos outros países e proporcionar economia e segurança.

Em relação ao tempo, ATFM envolve três fases principais: planejamento estratégico, planejamento pré-tático e planejamento tático.

Fase de planejamento estratégico - Esta fase se estende de seis meses até a antevéspera do dia da realização dos vôos e compreende as atividades que têm a função de pesquisar, projetar, planejar e coordenar as atividades referentes a possíveis situações de desbalanceamento entre demanda e a capacidade;

Fase de planejamento pré-tático - Esta fase corresponde à véspera do dia de ocorrência dos vôos, e as atividades incluem ações da projeção de demanda junto com a coordenação dos vôos, e as atividades incluem ações da projeção de demanda junto com a coordenação dos operadores e órgãos de controle.

Fase tática - esta fase corresponde ao dia de ocorrência do vôo, e as atividades compreendem as medidas de regulação do dia que devem ser aplicadas em conjunto com o ATC.

Em situações extremas, nas quais problemas de desbalanceamento (demanda superar a capacidade) não podem ser resolvidos por redistribuição antecipada da demanda, ou por uma adequação da capacidade do sistema, o ATFM passa a

interferir diretamente no processo, com medidas de regulação. Para Odoni (35) *apud* Rizzi e Muller (42), as medidas podem ser dos seguintes tipo: atraso em solo (*the ground-delay*); atraso em vôo (*the air-delay*); mudanças de rota (*re-routing*); regulagem da razão de fluxo em pontos específicos (*metering*); restrição à velocidade da aeronave ou uma combinação delas.

Em conseqüência do mau gerenciamento do fluxo de tráfego aéreo, tem-se como opção que as aeronaves em solo sejam atrasadas em função daquelas que estão em ar. Essa abordagem é conhecida como Espera em Solo, Odoni(35). Ela faz sentido, se o atraso local é apropriado e imposto, no local de decolagem, como decorrência de anular o congestionamento em rota e os atrasos em órbita.

No intuito de minimizar os prejuízos e danos causados pela discrepância entre demanda e capacidade, várias soluções são discutidas e até implementadas. O espaço aéreo é composto por vários elementos limitados em sua capacidade, como aeródromos, setores de controle, aerovias, rotas e pontos de notificação, Rizzi e Muller (42).

A capacidade de um aeroporto pode ser expressa em relação à quantidade máxima de movimentos de aeronaves (pousos e decolagens) por hora, isso é determinado pelas características dos aeroportos, procedimentos de segurança e condições meteorológicas. Em contrapartida, as capacidades dos setores são definidas como o número de aeronaves que podem ser controladas simultaneamente pelos controladores de tráfego aéreo do setor, em um dado intervalo de tempo. As capacidades dos aeroportos e do espaço aéreo são as maiores causas dos congestionamentos, Alonso et al. (1).

As estratégias para reduzir ou eliminar os congestionamentos no tráfego aéreo podem ser agrupadas em quatro classes, Navazio e Romanin-Jacur (33) *apud* Dib (14):

1. **Longo prazo**, o qual consiste na construção de novos aeroportos ou na ampliação dos existentes e na revisão do espaço aéreo (rotas e setores); esse seria o procedimento natural para combater o congestionamento, mas essa é uma estratégia lenta e demanda grandes investimentos.

2. **Médio prazo**, o qual consiste em melhor distribuição da demanda tanto no espaço aéreo (por exemplo, utilizando diferentes aeroportos, com isso se utilizam diferentes rotas, no Brasil isso poderia acontecer em São Paulo, Belo Horizonte e eventualmente no Rio de Janeiro), quanto no horário de vôos. Assim, poderia utilizar melhor os intervalos de tempo do dia; tal estratégia requer um médio período de tempo (meses) e baixos investimentos, mas existem algumas dificuldades organizacionais junto às companhias

aéreas, por causa do interesse econômico em determinados horários, como por exemplo, 18 horas.

3. **Curto prazo**, o qual consiste em um gerenciamento ótimo dos recursos presentes, respeitando os horários dos vôos programados pelas companhias aéreas tanto quanto possível. Tal estratégia requer um curto período de tempo (horas) e baixos investimentos.

4. **Monitoração e controle**, monitoramento contínuo das aeronaves, da decolagem até o pouso. Envolve ainda verificar se a aeronave cumpre seu plano de vôo, dentro das restrições de seu perfil, podendo-se realizar modificações em ambos. Essas ações, em um setor, podem afetar os planejamentos de curto prazo e tático de setores vizinhos.

As estratégias em curto prazo tentam limitar os impactos causados pelos atrasos produzidos pelos congestionamentos, em outras palavras, gerencia o fluxo de tráfego, a fim de evitar que a demanda exceda a capacidade disponível. Essa atividade é conhecida como gerenciamento de fluxo de tráfego aéreo, Andretta et al. (2).

5.2 Planejamento Tático

Conforme apresenta Dib (14), o planejamento tático compreende as seguintes atividades:

Monitoramento das unidades de transportes aéreos - Tal processo consiste em verificar a capacidade das unidades de transportes aéreos, sob diferentes condições meteorológicas e de equipamentos, se ela suporta a demanda prevista, acionando um alarme, caso seja detectado um provável problema. Segundo Dib (14), as unidades de transportes aéreos são elementos como um aeroporto, uma pista, uma via aérea, ou um setor aéreo. Cada tipo de unidade possui medidas próprias para ajuste da demanda à capacidade. Em geral, essas medidas incluirão atrasos de aeronaves em vôo, atrasos de aeronaves em solo e mudanças de rotas de vôos em torno dos aeroportos.

Diagnóstico do problema - Detectado um problema durante o monitoramento, inicia-se uma análise detalhada da unidade de transporte aéreo para determinar sua causa e severidade. Por ser o problema resultado de uma previsão baseada em uma demanda projetada em uma condição meteorológica prenunciada, existe a probabilidade de esse problema não ocorrer. Os diagnósticos,

portanto, devem ser realizados com muito critério, sendo comum a solicitação de novo monitoramento ou de mais informações, para conclusão de um diagnóstico.

Geração da solução - O tipo de ação que será proposta para resolução do problema detectado dependerá de fatores como a severidade do problema, o horário provável para sua ocorrência e o tipo de unidade envolvida. O gerenciamento de um fluxo local pode envolver outras unidades que não aquela na qual foi diagnosticado o problema. Isso pode resultar em um número de soluções muito grande, requerendo um critério para escolha da melhor solução.

Implementação da solução - Escolhida uma solução, será preciso comunicá-la aos controladores de tráfego aéreo de outras companhias aéreas, que participarão de sua implementação.

Durante o planejamento tático, as previsões de demandas das unidades de transportes aéreos são monitoradas continuamente, e seus fluxos de tráfego, chegando e saindo, estruturam-se para melhor utilização de suas capacidades.

Para Dib (14), a unidade transporte aéreo considerada é o "aeroporto" e, como conseqüência, abrangerá também as pistas e seus destinos, pousos e decolagens e também os horários de funcionamento das operações, de forma a identificar a capacidade individual de cada aeroporto.

5.3 Identificação de Problemas no Gerenciamento de Fluxo de Tráfego Aéreo

O atual sistema ATFM visa, primariamente, à proteção das posições de controle de tráfego aéreo, contra o congestionamento de tráfego, Stoltz (48) *apud* Dib (14). Ele baseia-se na centralização da demanda de tráfego no espaço aéreo e no processamento desta para suavizar os picos de tráfego, quando a demanda excede a capacidade.

Os sistemas utilizados, atualmente, confiam em horários de decolagem e pousos programados, os quais consideram a rota prevista da aeronave. Se esses horários fossem rigorosamente cumpridos, as áreas congestionadas seriam protegidas contra sobrecarregamento, e suas capacidades, utilizadas mais eficientemente. Contudo, fatores randômicos ocorrem, como mal tempo e atrasos nos vôos, entre outros, e nem sempre o que foi planejado pode ser cumprido à risca.

Os vôos são planejados e agendados até meses antes de sua ocorrência. Entretanto, essas escalas de vôos raramente são seguidas fielmente, tendo em vista várias razões, dentre elas atrasos nas operações dos aeroportos e mudanças nas condições meteorológicas. Assim, a ocorrência de agrupamentos de aeronaves

para pousos e decolagens simultâneos é comum, em determinados aeroportos.

Uma vez previsto um grande congestionamento, entra em ação o controle de fluxo. No Brasil, esse trabalho de verificação é realizado pelo Centro de Gerenciamento da Navegação Aérea (CGNA).

Quando um centro de controle de área ou, *Area Center Control* - ACC, requisita o controle central de fluxo para regular o tráfego, planeja-se o necessário atraso em solo, vôo por vôo, para suavizar o nível de demanda à capacidade declarada. No entanto, as comunicações de atrasos em solo se efetuam com até 2 horas de antecedência ao horário de partida da aeronave. Após esse tempo, quaisquer condições adversas podem causar agrupamento e congestionamentos nos aeroportos.

5.4 Características da Centralização da Informação entre os Aeroportos

A abordagem centralizada tem uma estrutura simples, sendo mais facilmente implementada. Apresenta algumas vantagens inerentes aos ambientes centralizados, como economia de recursos, tempo de acesso e pessoal.

Segundo Weigang (58), o controle de fluxo centralizado entra em ação quando se prevê um congestionamento. Para aplicar medidas efetivas de controle, faz-se necessária uma previsão apurada do local, do tempo e da magnitude do congestionamento em tempo hábil, bem como de um sistema centralizado de informações.

Tanto os EUA, quanto a Europa possuem sistemas de gerenciamento de fluxo aéreo com controle centralizado. Esses centros mantêm um fluxo aceitável para todas as unidades de transportes aéreos.

Apesar das vantagens da abordagem centralizada, conclui-se que ela apresenta também uma série de desvantagens. A centralização dificulta o processamento de sistemas de resolução de conflitos, pois o aumento significativo da concentração de aeronaves, voando a altas velocidades, reduz significativamente o tempo para a resolução do problema e aplicação de uma solução.

O planejamento das tabelas de uma frota aérea comercial é de extrema importância para a eficiência da própria companhia aérea, mas pode tornar-se um problema de graves proporções, quando mal realizado. Uma tabela de vôos bem planejada e gerenciada pode levar a uma economia significativa e contribuir para elevar a satisfação dos clientes, o mais importante. Clientes que sofrem freqüentes atrasos, em uma determinada companhia, tendem a procurar outra companhia, Bian et al. (5).

Tether e Metcalfe (54) afirmam que "como uma competição efetiva requer o fornecimento de serviços similares em termos de freqüência e horários, isto significa que, quando companhias aéreas competem em uma rota, elas, tipicamente, tentam atingir a mesma freqüência que seus competidores, mesmo que signifique utilizar aeronaves menores."

Pela tentativa de adequar os horários de vôos à preferência dos consumidores, geralmente se tem uma aglomeração de pousos e decolagens em determinados horários.

O sistema ATFM, em função da necessidade *a priori* de suavizar os picos de tráfego aéreo, em rotas ou nos aeroportos, nas horas de congestionamento, tende a mudar o horário dos vôos das companhias aéreas.

De fato, no Brasil, o planejamento dos vôos traz implícitas situações de potenciais congestionamentos, uma vez que se constroem tabelas de vôos considerando os aspectos comerciais e o número de operações por hora, ou seja, a capacidade dos aeroportos são medidas em números de movimentos (pousos e decolagens) por hora. Os tempos de separação entre pousos e decolagens variam de acordo com as pistas, as aeronaves, a configuração de relevo, enfim, as características locais.

5.5 Características da Distribuição da Informação entre os Aeroportos

Normalmente, os congestionamentos localizam-se em aeroportos ou determinados setores, e sua influência sobre o fluxo geral do tráfego afeta mais intensamente os setores vizinhos, o que caracteriza uma situação localizada, de modo que a centralização da análise e decisão desses eventos localizados é bem mais difícil, pois podem estar acontecendo simultaneamente.

A abordagem distribuída, segundo Tidhar et al. (55), tem uma série de vantagens:

Distribuição natural - O domínio do problema é naturalmente distribuído quanto aos aspectos de conhecimento, funcionalidades e controle.

Autonomia - Maior flexibilidade para introduzir modificações no agendamento de horários dos aeroportos locais ou nos setores envolvidos.

Comunicação - Mesmo com toda cooperação entre as unidades envolvidas, transferir os "conhecimentos" locais para uma unidade central requer grande esforço, não despendido em uma solução descentralizada.

Confiança - Cada unidade descentralizada é, parcialmente, imune a falhas

de outras unidades.

As vantagens, especialmente a autonomia dos aeroportos locais, fazem dessa abordagem a mais indicada, principalmente para regiões como Europa e sudeste da Ásia, que possuem um número muito grande de aeroportos autônomos pertencentes a diferentes países e precisam coordenar suas atividades.

Adicionalmente, a abordagem distribuída facilita a interligação de sistemas legados, permitindo que diferentes sistemas em diferentes plataformas tecnológicas possam interagir, mediante um mesmo protocolo de negociação.

5.6 Sistema Multiagentes para Sincronização e Gerenciamento de Tráfego Aéreo

No trabalho tomado como referência, ATFMGS, Dib (14), foi proposto um sistema que, embora seja de processamento descentralizado, possui uma estrutura de sincronização de dados e informações, dado uma visão única do fluxo de tráfego aéreo e, dessa forma, comporta-se como se fosse centralizado. Assim, unem-se as vantagens do sistema centralizado e do descentralizado em uma única solução.

Ele busca auxiliar a tomada de decisão em sistemas distribuídos que atuam para o planejamento tático, podendo se estender em algumas subfunções do planejamento estratégico como o replanejamento de escala de vôos e de monitoração e controle como uma função que simula o controle de tráfego, de modo que o sistema possa ter uma abrangência completa.

A solução proposta pelo ATFMGS opera de forma a atuar sobre o congestionamento causado por tráfego de origem próxima, conforme pretende Stoltz (48). Nele, o estudo considerou o tráfego de origem próxima qualquer vôo de duração inferior a uma hora e trinta minutos antes do pouso ou decolagem.

Considerou-se o sistema atual de ATFM como uma série de filtros que permitem a adaptação diária da demanda à capacidade disponível. A atividade de sincronização de tráfego aéreo em tempo real é um filtro adicional, o mais próximo às operações reais do controle de tráfego aéreo.

No sistema ATFMGS, para mensurar o conceito de fluxo, foi necessário definir uma medida de fluxo e de congestionamento na utilização dos recursos. Essa quantia abstrata, denominada Padrão de Balanceamento de Aeroporto (PBA), é uma função do tempo de atraso de cada vôo e do peso (fator de importância) a ele associado. A equação de PBA é:

$$PBA(t_1, t_2) = \sum_{i=1}^{n} p_a(f_i) \cdot m_a(f_i) \tag{5.1}$$

Onde:

PBA($\mathbf{t}_1, t_2$) : Padrão de Balanceamento de Aeroporto entre os instantes t_1 e t_2;

n : número de vôos entre os instantes t_1 e t_2;

$\mathbf{p}_a$: peso, fator de importância, atribuído para cada atraso;

$\mathbf{m}_a$: número de minutos de atraso;

$\mathbf{f}_i$: vôo em análise.

Em Dib (14), o PBA foi escolhido como uma função da quantidade de atraso porque quando um aeroporto está congestionado, alguns dos vôos são atrasados e a extensão do tempo de atraso é proporcional à severidade do congestionamento. Assim, combinando-se o tempo de atraso com o peso (fator de importância) do vôo atrasado, tem-se a indicação da demanda do aeroporto. O PBA utiliza um peso (fator de importância) diferente para aterrissagem e decolagem das aeronaves.

Considerando que o objetivo do ATFM é minimizar a retenção de aeronaves no ar, o peso *(p_a)* foi distribuído (na razão de $70\% \div 30\%$), para pousos e decolagens. Esse valor foi escolhido depois de várias simulações, adotou-se a um peso igual a 7 para aterrissagem e peso igual a 3 para a decolagem, como fatores que produzem uma avaliação compatível com a observação de um especialista.

A tabela 5.1, Tempo de Execução e Número de Mensagens por Aeroporto e por PBA Aceitável, apresenta os resultados de PBA obtidos para os aeroportos de Garulhos - GRU, Congonhas - CGH, Brasília - BSB e Galeão - GIG, respectivamente.

Na tabela, as siglas SAD, PAP, SAP, PAD e RPA têm o seguinte significado:

- **Solicitação de Alteração de Decolagem, SAD** - mensagem enviada pelo aeroporto responsável pela decolagem, solicitando ao aeroporto de destino concordância para alteração do horário de pouso de um determinado vôo;

- **Pedido de Alteração de Pouso, PAP** - pedido de alteração de pouso recebido pelo aeroporto de destino;

- **Solicitação de Alteração de Pouso, SAP** - mensagem enviada pelo aeroporto responsável pelo pouso, solicitando ao aeroporto de origem concordância para alteração do horário de decolagem de um determinado vôo;

- **Pedido de Alteração de Decolagem, PAD** - pedido de alteração de decolagem recebido pelo aeroporto de origem;
- **Resposta de Pedido de Alteraçao, RPA** - mensagem enviada pelo aeroporto solicitado acatando um PAP ou PAD.

PBA	**Base**	**SAD**	**SAP**	**PAP**	**PAD**	**RPA**	**Sub Total**	**T. de Exec.**	**Total Msgs.**
0	GRU	28	1	17	2	19	67	6:53,408	421
	CGH	133	53	9	7	16	218	8:07,354	
	BSB	64	23	6	4	10	107	8:24,220	
	GIG	9	2	8	1	9	29	7:27,628	
0,75	GRU	11	1	1	1	2	16	0:47,715	98
	CGH	21	5	6	1	7	40	1:37,587	
	BSB	18	9	1	1	2	31	0:53,843	
	GIG	7	2	0	1	1	11	0:13,329	
1	GRU	0	1	1	0	1	3	0:11,896	33
	CGH	0	0	6	1	7	14	1:21,377	
	BSB	5	1	1	1	2	10	0:45,431	
	GIG	3	1	0	1	1	6	0:09,794	
1,25	GRU	0	1	0	0	0	1	0:00,000	21
	CGH	0	0	4	1	5	10	0:48,478	
	BSB	3	1	1	0	1	6	0:27,106	
	GIG	2	0	0	1	1	4	0:06,800	
2,5	GRU	0	0	0	0	0	0	0:00,000	17
	CGH	4	1	2	0	2	9	0:19,453	
	BSB	3	2	0	0	0	5	0:04,366	
	GIG	2	1	0	0	0	3	0:00,360	

Tabela 5.1: Tempo de Execução e Número de Mensagens por Aeroporto e por PBA Aceitável, Dib (14).

A grande vantagem da sincronização do tráfego aéreo em tempo real é predizer com mais exatidão o horário de pico do tráfego. Perigos operacionais nos aeroportos, movimentos de partidas de vôos, desvios de aeronaves de seus planos de vôos, dentre outros eventos, são capturados pelos sistemas de monitoramento de ATFM em tempo real e modificarão as cargas de tráfego previstas, permitindo identificar picos de sobrecarga, cada vez mais corretos.

No caso de se estar analisando um aeroporto, a importância da sincronização do tráfego em tempo real é evidente para suavização da demanda nos minutos finais do pousos, até 30 minutos antes do pouso.

O objetivo do ATFMGS foi atuar no planejamento tático, isto é, no dia da operação. Atualmente a atuação das unidades de controle de fluxo de tráfego encerram suas atividades no máximo até duas horas antes da partida de um vôo. Utilizando as propriedades de sincronização em tempo real, pretende-se estender o tempo de atuação de 30 minutos a 45 minutos antes das decolagens.

Embora a plataforma construída para o protótipo permitisse atuação em tempo real no controle de tráfego aéreo, o modelo de simulação foi complexo, pois foram considerados dados gerados por estações de rastreamento em terra, satélites de posicionamento global, aeronaves, estações meteorológicas e criação de modelos matemáticos para considerar a influência dos ventos nos vôos "em-rota". Muitos esforços e consideráveis somas têm sido aplicados ao problema de tráfego aéreo em tempo real, que, sem dúvida alguma, é o mais importante. Porém, pouco se tem trabalhado no planejamento do fluxo de tráfego, Dib (14).

A proposta deste modelo é auxiliar o processo de tomada de decisão em sistemas de informações distribuídas, no caso específico, tráfego aéreo. Este capítulo apresentou, portanto, as características que despertaram maior interesse para esse ambiente.

Pelo fato de que a avaliação do modelo tomou como referência o ATFMGS, (14), como sistema distribuído de informações entre os aeroportos, uma parte considerável deste capítulo foi dedicada à explicação de aspectos referentes a ele.

Capítulo 6

Uma Abordagem de Aprendizagem por Reforço e Gerência em Nível Meta aplicada em Troca de Mensagens

No modelo meta gerente de mensagens, proposto neste trabalho, a aprendizagem por reforço se mostra favorável, no sentido que ocorre através da experiência adquirida pelo agente, com sua atuação no ambiente. Uma experiência boa adquirida por ele serve de exemplo para decisões futuras a serem tomadas. As mudanças que ocorrem nesse ambiente estocástico são aprendidas pelo agente, por meio de sua atuação no ambiente, que é refletida nas recompensas recebidas por ele.

Com intuito de acelerar a aprendizagem, três estratégias foram propostas: heurística inicial, epsilon adaptativo e heurística baseada em performance.

O controle em nível meta faz uma análise acerca dos metadados das mensagens e, dessa forma, em ambientes complexos, tende a apresentar melhor desempenho com o seu uso, possibilitando a melhoria da performance em sistemas de informações distribuídas. A análise, no controle em nível meta, busca estabelecer uma hierarquia de atendimento às mensagens.

6.1 O Modelo Meta Gerente Mensagens Proposto

Os componentes do modelo são três principais, conforme pode ser observado na figura 6.1:

- *o ambiente*, no modelo meta gerente de mensagens, é representado pelo Módulo de Decisão e Controle (MODEC);

- *o controle em nível meta*, no modelo corresponde ao meta gerente de mensagens, isto é, a consideração do MODEC e Módulo de Aprendizagem por Reforço (MAR) juntos;

- *a execução*, no modelo meta gerente de mensagens é representada pelo sistema hospedeiro e especificamente pela *Pronto Lista*, onde ficam guardadas as mensagens encaminhadas ao sistema, podendo ser qualquer sistema que se comunica via troca de mensagens.

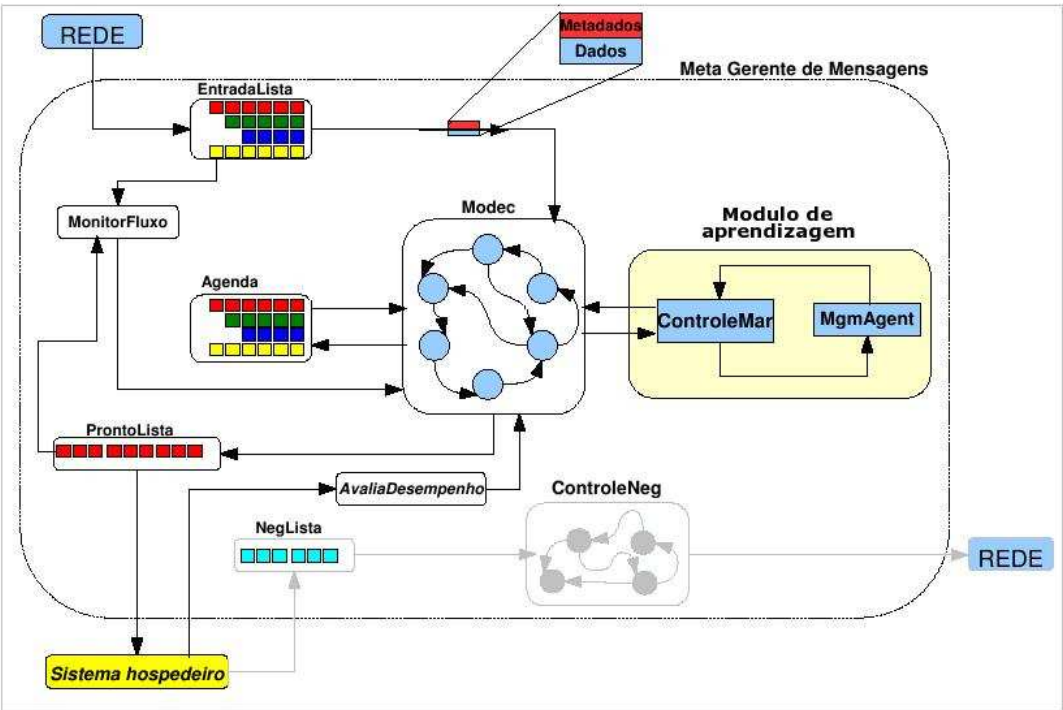

Figura 6.1: O modelo meta gerente de mensagens.

A figura 6.1 mostra a visão geral da arquitetura proposta. Nela, existem várias estruturas de dados que auxiliam no processo de gerência e tomada de decisão: uma *Entrada Lista*, que recebe as mensagens que chegam da Rede; uma fila intermediária, denominada *Agenda*, que guarda aquelas mensagens armazenadas para processamento futuro, e uma de *Pronto*, onde ficam as mensagens encaminhadas para o sistema hospedeiro.

O modelo proposto, nesta pesquisa, baseia-se no desempenho do sistema hospedeiro e também no desempenho do meta gerente de mensagens, buscando, quando é possível, que a execução ocorra de forma rápida, para auxiliar no processo de tomada de decisão. A abordagem adotada preocupou-se em apresentar uma proposta de aprendizagem por reforço,

que usa heurísticas no intuito de se acelerar o aprendizado, para o tratamento do problema. Em linhas gerais, o modelo proposto é composto de dois módulos principais: MODEC e MAR.

Para entendimento do modelo, os conceitos ambiente, estado, ação, política e recompensa fazem-se necessários. De forma resumida, um estado é uma representação do ambiente no qual o agente opera. Uma ação é aquilo que o agente faz, para afetar seu ambiente e mudar seu estado, Russel e Norvig (44). O mapeamento de um estado para uma ação é chamado política, Sutton e Barto (50). O reforço é um valor escalar associado a um par estado-ação, podendo ser direto ou indireto.

6.2 Módulo de Decisão e Controle - MODEC

O módulo de decisão e controle corresponde a uma máquina de estados e tem como formalismo matemático o processo decisório de Markov, Clarke e Disney (12). Assim, para estimar o estado futuro, basta as informações do estado atual. O tema processo decisório de Markov é apresentado no apêndice A deste texto.

É no módulo MODEC que é feito o controle da mudança de estados. É ele que retorna a recompensa para o módulo de aprendizagem. Ele recebe os metadados da mensagem e monta a nova mensagem com os parâmetros recebidos (prazo e utilidade da mensagem) e com os parâmetros derivados dos parâmetros recebidos (demais parâmetros). Comunica-se com as Listas que compõem o sistema: *Lista de Entrada*, *Agenda* e *Lista de Pronto*.

6.2.1 Ambiente no meta gerente de mensagens

O ambiente, no modelo MGM, consiste de um gerenciador de mensagens que recebe as mensagens, para que agentes individuais, baseados no modelo de aprendizagem definido, decidam qual a melhor ação a ser tomada. O controle em nível meta é invocado toda vez que uma nova mensagem chega à *Lista de Entrada*, ou quando existem mensagens para serem avaliadas na *Agenda*. As ações tomadas pelo MODEC podem ou não ter recompensas imediatas. Quando uma ação é tomada, a recompensa desta ação é armazenada no módulo de aprendizagem, para que o agente possa aprender a partir da experiência, com a atuação no ambiente.

O ambiente meta gerente de mensagens satisfaz a propriedade de Markov, porque o seu estado atual resume o passado de forma compacta, sem perder a habilidade de prever o futuro, ou seja, pode-se estimar qual será o próximo estado e a próxima recompensa esperada, dados estado e ação atuais, Sutton e Barto (50).

O objetivo do agente de aprendizagem meta gerente de mensagens é escolher ações, de modo a maximizar a recompensa de um valor-estado, seguindo uma política que busca a otimização das ações.

O ambiente depende diretamente dos critérios que determinarão uma instância de ambiente, isto é, um estado. Quanto maior a quantidade de critérios considerados, melhor é a descrição do ambiente. No modelo proposto, os critérios que influenciam o ambiente para que ocorram as mudanças de estado foram:

$\mathbf{P}_1$ - boa utilidade da nova tarefa;

$\mathbf{P}_2$ - prazo de execução da nova tarefa;

$\mathbf{P}_3$ - probabilidade de chegada de uma tarefa de alta utilidade, na *lista de entrada*;

$\mathbf{P}_4$ - boa utilidade do conjunto agendado, na *lista de agenda*;

$\mathbf{P}_5$ - prazo de execução do conjunto agendado, *na lista de agenda*;

$\mathbf{P}_6$ - razão de fluxo no meta gerente de mensagens.

P_1 e P_2 são parâmetros que a mensagem traz com a sua chegada. E de P_3 a P_6 são parâmetros que envolvem a situação atual do MGM, ou ainda parâmetros derivados dos dois primeiros. A cada um dos seis parâmetros, pode ser atribuído um dos três valores: ALTO, MÉDIO ou BAIXO. E o parâmetro poderá ou não ser informado.

Os valores de cada parâmetro têm a representação em dois *bits* e significa:

00 - parâmetro NÃO INFORMADO;

01 - parâmetro apresenta valor BAIXO;

10 - parâmetro apresenta valor MÉDIO;

11 - parâmetro apresenta valor ALTO.

6.2.2 Estado no meta gerente de mensagens

Um estado descreve a situação em que se encontra o ambiente, em determinado momento. No modelo, o estado atual é representado por um algarismo binário de 12 *bits* e, a cada dois *bits*, descreve a situação atual, de determinado parâmetro. Cada parâmetro é descrito, em detalhe, no Apêndice B deste trabalho.

O estado *s* e ação *a* atuais determinam o próximo estado *s'*, de acordo com a probabilidade de transição e a recompensa *r(s,a)* associada. Como um exemplo de estado, tem-se a instância de estado 111101011011_2, que corresponde ao estado 3931_{10}.

O estado 111101011011 ($b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12}$) pode ser verificado por meio de seus seis parâmetros, conforme a seguir:

$p_1(b_1, b_2) = p_1(1, 1) = ALTO$
$p_2(b_3, b_4) = p_2(1, 1) = ALTO$
$p_3(b_5, b_6) = p_3(0, 1) = BAIXO$
$p_4(b_7, b_8) = p_4(0, 1) = BAIXO$
$p_5(b_9, b_{10}) = p_5(1, 0) = MÉDIO$
$p_6(b_{11}, b_{12}) = p_6(1, 1) = ALTO$

6.3 Mensagens no Meta Gerente de Mensagens

Uma mensagem é composta por dados, que é o próprio conteúdo da mensagem, e também por metadados, que são parâmetros das mensagens que serão utilizados na avaliação das características e tratamento da mensagem. Como um exemplo típico de metadado, tem-se a utilidade de uma mensagem e o seu prazo para execução.

No contexto de tráfego aéreo, os dados de uma mensagem são informações que um aeroporto origem pode informar ao seu aeroporto destino durante um processo de negociação. A negociação ocorre no intuito de se conseguir estabelecer o escalonamento dos vôos. O estudo de caso apresenta um exemplo típico de tráfego aéreo, conforme pode ser verificado, no capítulo seguinte a este. As informações de negociação, ao final, influenciarão a tabela de escalonamento dos vôos. O conteúdo da mensagem também terá relevância, porque exerce influência nos metadados e, portanto, na decisão a ser tomada com a mensagem. Um exemplo é a urgência de escalonamento de um vôo que apresente baixa quantidade de combustível exercendo influência no prazo de execução desta mensagem.

Nos dados da mensagem, tem-se o conteúdo dela. Um exemplo de campo *dado*, no contexto de tráfego aéreo, é priorizar, durante a etapa de escalona-

mento de vôos, o pouso de uma aeronave *Boeing* 747 recentemente inserida, em preferência a uma aeronave *Boeing* 737, anteriormente pertencente ao conjunto escalonado, conforme indicam as regras *Air Traffic Control* (ATC), que regem a área.

6.4 Módulo Aprendizagem por Reforço - MAR

O módulo de aprendizagem usa, em determinado instante, um algoritmo de aprendizagem por reforço, que pode ser SARSA ou *Q-learning*. E, além do algoritmo, algumas heurísticas e adaptações foram propostas, com o intuito de se acelerar o aprendizado: heurística inicial, epsilon adaptativo e heurística baseada em performance.

Para a definição das heurísticas e adaptações, foi utilizada como base a pesquisa feita por Bianchi e Costa (6). As estratégias propostas serão explicadas nos parágrafos seguintes.

As heurísticas propostas buscam acelerar o aprendizado por reforço do agente. Elas são consideradas, durante o usufruto do agente. Em contrapartida, a exploração do agente ocorre de forma semelhante a qualquer algoritmo tradicional de aprendizado por reforço.

Heurística inicial

A primeira estratégia proposta, heurística inicial (HI), combina as sugestões presentes em Raja e Lesser (38). Assim, uma estrutura de dados armazena uma ação mais indicada a ser tomada pelo MODEC, de acordo com o estado atual do ambiente em que ele está atuando.

As regras, na heurística inicial, seguem a lógica de priorizar as mensagens com utilidade alta e prazo curto, em seguida, utilidade alta e prazo médio, e assim sucessivamente são atendidas. Depois, são tratadas as mensagens, com utilidade baixa e prazo curto, que podem, às vezes, ser descartadas, sem prejuízos ao sistema, porque uma análise criteriosa foi realizada.

A heurística inicial permite ao agente contar com uma base de conhecimento, nos instantes iniciais, quando ele ainda não tem experiência, no ambiente novo para ele. Esta inteligência inicial é empregada em todos os oito agentes que compõem o modelo.

Em síntese, o algoritmo da heurística inicial atribui um reforço alto para ação mais indicada, um reforço intermediário para a segunda ação mais indicada

e, por fim, um reforço baixo para a ação menos indicada. No meta gerente de mensagens, a ação mais indicada, de acordo com o estado atual, recebeu reforço igual a 1, a segunda na ordem, recebeu 0, e a menos indicada recebeu valor negativo, ou seja, valor -1. A seguir, três exemplos que mostram a idéia adotada serão apresentados.

Exemplo 1: Mensagem com utilidade alta e prazo curto

Uma mensagem com utilidade alta e prazo curto teria as seguintes regras iniciais. Se decidir realizar a ação de *encaminhar*, receberá recompensa alta, ou seja, 1. Se decidir *agendar*, receberá recompensa intermediária, ou seja, 0. E, por fim, se decidir *descartá-la*, receberá recompensa ruim, ou seja, -1. Então, existe inicialmente maior chance de o agente *encaminhá-la*, o que se mostra mais interessante.

Exemplo 2: Mensagem com utilidade alta e prazo longo

Uma mensagem com utilidade alta e prazo longo teria as seguintes regras iniciais. Se decidir realizar a ação de *agendar*, receberá recompensa alta, ou seja, 1. Se decidir *encaminhar*, receberá recompensa intermediária, ou seja, 0. E, por fim, se decidir *descartá-la*, receberá recompensa ruim, ou seja, -1. Então, existe inicialmente maior chance de o agente *encaminhá-la*, o que se mostra mais interessante.

Exemplo 3: Mensagem com utilidade baixa e prazo curto

Uma mensagem com utilidade baixa e prazo curto teria as seguintes regras iniciais. Se decidir realizar a ação de *descartar*, receberá recompensa alta, ou seja, 1. Se decidir *encaminhar*, receberá recompensa intermediária, ou seja, 0. E, por fim, se decidir *agendá-la*, receberá recompensa ruim, ou seja, -1. Então, existe inicialmente maior chance de o agente *encaminhá-la*, o que se mostra mais interessante.

Tanto os agentes que usam os algoritmos SARSA, quanto os que usam *Q-learning* empregam a heurística inicial, logo que eles comecem atuar no ambiente.

Epsilon adaptativo

A segunda estratégia proposta, epsilon adaptativo (EA), procura adequar o *trade-off* existente em algoritmos de aprendizagem por reforço, que é a escolha entre usufruir o conhecimento que se tem até o momento, ou explorar novas ações,

com o intuito de atingir melhores resultados. A idéia é que, se bons resultados estão sendo atingidos, deverá ser aumentada a distribuição de probabilidades para usufruto, mas, se resultados ruins estão sendo atingidos, deverá ser aumentada a distribuição de probabilidades para exploração.

Para decidir entre usufruir o conhecimento adquirido ou explorar novos valores, um valor de epsilon é sorteado. Tal valor está compreendido na faixa de valores entre 0 e 1. Nesta proposta, a porcentagem de valores a ser considerada na decisão, entre usufruir e explorar, está sujeita ao desempenho do agente.

A exploração no agente ocorre da seguinte forma: uma ação, dentre o conjunto de ações (*encaminhar*, *agendar* ou *descartar*), é escolhida por sorteio e é executada. A exploração ocorre no intuito de sair dos máximos locais, buscando atingir máximos globais, Sutton e Barto (50).

Na implementação do modelo para o epsilon adaptativo, o valor considerado foi de 5%. E, assim, se bons resultados estão sendo atingidos, no instante posterior existirá uma porcentagem 5% maior que a anterior de o agente usufruir e 5% menor de explorar. Em contrapartida, se resultados ruins estão sendo obtidos, a porcentagem será aumentada em 5%, para exploração. A seguir, será apresentada, em pseudocódigo, a estratégia do epsilon adaptativo.

Algoritmo: Epsilon Adaptativo

$epsilon \Leftarrow random$
$fatorReducao \Leftarrow 0.995$
$qtdeEstados \Leftarrow 100$
$contEstados \Leftarrow 0$
$fatorDesempenho \Leftarrow 0.5$
repeat
if contEstados $\leq qtdeEstados$ **then**
if fatorDesempenho $\geq avaliaDesempenho()$ **then**
$epsilon \Leftarrow epsilon * fatorReducao$
end if
elsedefault $epsilon \Leftarrow$ epsilon $*$ $(2 - fatorReducao)$
Critério de parada ser atingido

Onde:

epsilon - fator que determina a porcentagem a ser considerada no sorteio entre usufruir e explorar;

fatorReducao - define o fator de redução a ser considerado no epsilon adaptativo proposto;

qtdeEstados - define a quantidade total de estados a serem avaliados;

contEstados - contador que acumula a quantidade atual de estados;

fatorDesempenho - fator que define um desempenho padrão;

avaliaDesempenho() - recebe um valor baseado no cálculo da equação 6.1, que será mostrada na seção seguinte.

Um valor epsilon indicado ao *Q-learning* tende a ser diferente daquele indicado ao SARSA. O motivo é a natureza do algoritmo. Conforme já mencionado, o *Q-learning* busca a maximização, e o SARSA sorteia um valor, dentro da distribuição de probabilidades das ações tomadas. Por exemplo, ao rodar o algoritmo SARSA, se a ação *encaminhar* ocorre mais vezes, ela terá mais chance de ser sorteada.

Heurística baseada em performance

A terceira estratégia proposta é a inserção de uma heurística baseada em performance (HP) do sistema. Uma função avalia o desempenho do sistema e retorna uma nota que interfere no usufruto do agente de aprendizado por reforço. Segundo equação 6.1 proposta, um valor de desempenho atual é medido e posteriormente comparado com o valor de desempenho ideal.

A função que avalia o desempenho do meta gerente de mensagens, em determinado instante, obedece à equação:

$$AvD = \frac{(3 * SDS + 2 * RF + SEL + SAL + SPL)}{8} \tag{6.1}$$

Onde:

SDS , significa a situação do sistema hospedeiro;

RF , razão de fluxo e representa a diferença entre a quantidade de mensagens encaminhadas para o sistema hospedeiro em relação às mensagens que chegaram;

SEL , situação da *lista de entrada*;

SAL , situação da agenda;

SPL , situação da *lista de pronto*.

Por definição, foi considerado como desempenho ideal um valor que representa 100% do intervalo disponível ao agente. O desempenho atual reflete o desempenho do agente, em um instante de análise. O desempenho inicial reflete o desempenho atingido pelo sistema no início da análise. Para tanto, o desempenho do sistema é constantemente avaliado.

O objetivo dessa heurística é minimizar a distância entre o desempenho atual avaliado por AvD, equação 6.1, em determinado instante de análise, em relação ao desempenho tomado como referência.

A função heurística H que compara AvD, equação 6.1, funciona como um avaliador que observa a situação atual em que o ambiente se encontra e a compara com a situação ideal, em que o agente deveria estar.

Q-learning acelerado por heurística baseada em desempenho

No *Q-learning*, a idéia é combinar o vetor *qEst* com a função heurística ligada ao desempenho do MGM, por meio de um fator eta, que define a influência de HP, no usufruto do agente. A função heurística compara o desempenho atual do sistema em relação ao desempenho ideal, definido com um valor padrão ideal, ou seja, 100%. E isso é refletido pelo retorno a ser recebido pelo agente.

A heurística do sistema foi ajustada para contemplar o desempenho do sistema. Para tanto, foi criado o vetor qHeu, que armazena o desempenho no momento da ação a ser tomada, soma esse valor com qEst e a próxima decisão é tomada. O valor padrão eta foi definido como 0,5. E, de acordo com o agente adotado, um valor *eta* se mostra mais apropriado, sendo, portanto, diferente para o *Q-learning* e para o SARSA.

SARSA acelerado por heurística baseada em desempenho

Quando o agente SARSA for usufruir, ele irá usar uma proposta combinada à heurística baseada em desempenho.

A heurística do sistema foi ajustada para contemplar o desempenho do sistema. Assim, quando o desempenho estiver baixo, a contagem de vezes que a ação foi tomada é decrementada, ou, em caso contrário, se o desempenho estiver

muito bom, a contagem é incrementada. Isso influenciará as decisões estatísticas do SARSA. Nesse caso, o valor de eta indicado é nulo. Ou seja, o parâmetro eta não influencia a decisão do agente.

Consideração das heurísticas e do epsilon adaptativo ao mesmo tempo

Nesta abordagem, aplicam-se todas as propostas ao mesmo tempo e assim a heurística inicial, o epsilon adaptativo e a heurística baseada em performance exercem influência no *Q-learning* e SARSA.

Nesta proposta, todas as estratégias aqui definidas são combinadas para operar nos algoritmos ao mesmo tempo, levando em consideração que a heurística baseada em performance opera de forma diferente tanto para o *Q-learning*, quanto para o SARSA.

6.5 Política no Meta Gerente Mensagens

A política faz o mapeamento de estados em ações de valor $\pi(s, a)$. No modelo, a política adotada foi a política $\epsilon - Greedy$, adaptada às seguintes estratégias: heurística inicial, epsilon adaptativo e heurística baseada em performance.

Um valor q é escolhido de maneira aleatória, com distribuição de probabilidade uniforme sobre [0,1]. E $(0 \leq \epsilon \leq 1)$ é o parâmetro que define a taxa de exploração e usufruto. Quanto menor é o valor de ϵ, menor é a probabilidade de se fazer uma escolha aleatória.

A exploração do agente não sofreu alteração no modelo em relação aos algoritmos que propõem. Assim, uma vez que a ação do agente foi escolhida para ocorrer, segundo a exploração, uma ação aleatória é selecionada entre as ações possíveis de serem executas para o estado s_t.

Entretanto, se, segundo a política, a ação deverá ser selecionada de acordo com a experiência do agente, ou usufruto, as melhorias propostas são combinadas aos algoritmos de AR QL e SARSA, de acordo com qual um dos oito agentes está em operação.

6.6 Função Valor-estado, Reforço e Retorno no Meta Gerente de Mensagens

Define uma *função valor-estado* como mapeamento do estado, ou par estado-ação, em um valor que é obtido a partir do reforço atual e dos reforços futuros,

Sutton e Barto (50).

O reforço é um valor que descreve a satisfação do ambiente em relação à ação sugerida à execução. Tal valor é retornado pelo MODEC ao MAR. O MAR usa esses valores para adquirir experiência, no ambiente. Em outras palavras, é o termo direto que descreve a satisfação de um estado. Agentes de aprendizagem por reforço apresentam como objetivo maximizar a recompensa obtida, ao longo do tempo. Ele verifica, por meio de uma função matemática, o quanto, em termos de valores, a ação executada contribuiu para atender aos objetivos.

O *reforço* está relacionado à satisfação dos objetivos no MGM. Sendo que os objetivos são:

- melhorar a eficiência dos sistemas acoplados ao gerente em nível meta, que é refletido nele, via um parâmetro externo;
- administrar o recebimento e envio de mensagens, que é refletido nele, via parâmetros internos: forma de atuação do MGM, ou seja, processo de tomada de decisão;
- diminuir gargalos, relacionados com a intensa chegada de mensagem que não são processadas, imediatamente.

Dentro das opções de reforço prováveis, a abordagem adotada pelo modelo irá, a cada intervalo de tempo, exprimir uma função de retorno.

6.7 Funcionamento do Meta Gerente de Mensagens

A chegada de uma nova mensagem gera uma consulta do MODEC ao MAR, que deverá indicar uma ação a ser tomada. No módulo MAR, está definida, em determinado momento, a atuação de um dos seus oito agentes de aprendizagem, podendo ser:

- algoritmo *Q-learning* com a heurística inicial;
- algoritmo *Q-learning* com a heurística inicial e epsilon adaptativo;
- algoritmo *Q-learning* com a heurística inicial e heurística baseada em performance;
- algoritmo *Q-learning* com a heurística inicial, epsilon adaptativo e heurística baseada em performance;

- algoritmo SARSA com a heurística inicial;
- algoritmo SARSA com a heurística inicial e epsilon adaptativo;
- algoritmo SARSA com a heurística inicial e heurística baseada em performance;
- algoritmo SARSA com a heurística inicial, epsilon adaptativo e heurística baseada em performance.

O agente roda o seu algoritmo de aprendizagem e, ao final, indica uma ação a ser executada, devolvendo a resposta para o MODEC. O módulo MODEC do MGM é o responsável pela execução desta ação.

Uma mensagem, que chega ao controle em nível meta, é colocada na fila de prioridade de entrada. A hierarquia, dentro da fila, obedece ao grau de utilidade e ao prazo de resposta da mensagem, e os demais metadados são considerados pelo MGM. O MODEC retira a primeira mensagem da fila de prioridade de entrada e decide se a mensagem deve ser *agendada*, *encaminhada* diretamente ao sistema hospedeiro, ou simplesmente *descartada*. O processo decisório no MGM envolve um conjunto de estados markovianos, Clarke e Disney (12), e a aprendizagem por reforço, Sutton e Barto (50). O módulo de aprendizagem garante ao meta gerente de mensagens, figura 6.1, a capacidade de se adaptar às alterações no ambiente que exijam uma mudança no processo decisório atual.

Um ciclo completo, no modelo meta gerente de mensagens, pode ser resumido da seguinte forma: uma mensagem chega ao MODEC e, a seguir, é gerada uma consulta ao Módulo de Aprendizagem, para que este indique uma ação a ser tomada. Depois que uma ação é escolhida pelo MAR, via controle MAR, figura 6.1, ela é informada ao MODEC, que verifica a situação do ambiente, a partir da ação, e retorna um reforço atrasado ao MAR, para que este adquira experiência.

6.8 Fundamentos Técnicos

A idéia empregada buscou considerar como objetivos principais conceitos de aprendizagem por reforço. Assim, tanto os ambientes, quanto os estados, as ações e os resultados das ações foram modelados como objetos, e o paradigma de programação empregado foi a Orientação a Objetos. E, por conseguinte, optou-se pelo uso da linguagem JAVA. A ferramenta de programação JAVA empregada foi o *Eclipse* versão 3.1.

O uso da linguagem JAVA, Deitel (13), foi motivado porque é uma linguagem multiplataforma, que facilita o uso da solução proposta, em diferentes plataformas

(*Linux* e *Windows*). Outra vantagem de JAVA é poder utilizar conceitos de programação orientada a objetos, o que contribuiu para que a modelagem do sistema pudesse ser realizada de maneira natural e facilitada.

Foi utilizado um *framework* JAVA que implementa as classes básicas de qualquer sistema de aprendizagem por reforço, por meio dos seguintes conceitos: agente, ambiente, simulação, ação, resultado da ação e estado, encontrado em Kerr e Neller (27). No *framework*, as classes principais são implementadas de maneira abstrata, ficando a cargo de quem as usa prever, definir e programar a complexidade do modelo. O *framework* oferece as seguintes classes: *Action*, *ActionResult*, *Environment*, *Agent*, *Simulation* e *State*. O Apêndice C desta dissertação apresenta, em detalhe, as quatro classes e as duas interfaces.

A modelagem do sistema foi feita utilizando *Unified Modeling Language* - UML, Larman e Rumbaugh et al. (28; 43). Em especial, foram utilizados, para a visão estática, diagramas de classe e, para a visão dinâmica, diagramas de seqüência e diagramas de colaboração. A ferramenta para uso de UML foi a *Poseidon* versão não comercial 3.1.

Para plotar os gráficos do estudo de caso, no capítulo 7, foi escolhido o *software* MATLAB versão 7.0.

O uso do controle em nível meta junto à aprendizagem por reforço mostrou-se interessante para ser empregado em ambientes de troca de mensagens. De forma específica, em tráfego aéreo, por causa da conformidade à natureza estocástica deste ambiente.

Ao definir a representação de estados utilizando a concatenação de parâmetros binários, tornou o processo de tomada de decisão mais rápido do que se tivesse feita a representação somente em decimal. As operações são realizadas em binário, o que contribuiu para otimizar o funcionamento do MGM.

As idéias apresentadas no *framework AR* foram interessantes no início da implementação, contudo, à medida que o projeto foi avançando, várias alterações foram feitas até mesmo nas classes bases e verificou-se que o *framework* era sobremaneira generalizado. Assim, houve a necessidade de se especializar para o problema em análise, o que demandou mais esforço do que o esperado inicialmente.

Capítulo 7

Estudo de Caso - Avaliação da Aprendizagem

O estudo de caso deste trabalho avalia a meta gerência de mensagens, segundo o interesse de aprender e, como conseqüência, melhorar a performance da comunicação em gerência de tráfego aéreo.

De maneira específica, a intenção é avaliar a aprendizagem no ambiente estocástico de tráfego aéreo. Para tanto, o sistema ATFMGS foi estudado como referência para a criação da simulação. As principais características desse sistema foram apresentadas no capítulo 5, Gerência de Tráfego Aéreo, e outros aspectos relevantes apareceram no capítulo 6.

Para avaliar o modelo proposto, um conjunto de mensagens que chegam à Lista de Entrada do Meta Gerente de Mensagens é analisado, e, a partir do Módulo de Aprendizagem por Reforço (MAR), são decididas quais ações devem ser tomadas pelo Módulo de Decisão e Controle (MODEC).

A análise, neste estudo de caso, observa o modelo meta gerente de mensagem proposto em relação aos seguintes aspectos:

1. **Performance do algoritmo** - Nesta análise, o interesse está em avaliar o processo de tomada de decisões do agente quanto à rapidez.

2. **Qualidade da decisão tomada** - Nesta análise, a curva de aprendizagem é avaliada quanto à convergência. Ela serve para observar como o agente de aprendizagem evolui com o tempo, que, por conseguinte, reflete na experiência adquirida por ele.

3. **Situação do aeroporto em análise** - Esta característica avalia a aprendizagem, considerando a situação atual do aeroporto. O aeroporto pode ou não está congestionado.

4. **Alteração de parâmetros dos algoritmos *Q-learning* e SARSA** - Nesta análise, os parâmetros que influenciam diretamente o algoritmo são alterados para que se possa avaliar quais são os parâmetros mais indicados.

O problema de auxiliar a gerência de tráfego aéreo lida com dois aspectos que se contrapõem. De um lado, a necessidade de tomar decisões rápidas e, de outro, a qualidade das decisões que são tomadas. Acontece que boas decisões sobre um conjunto finito de opções disponíveis exigem um tempo de raciocínio relativamente grande, dentro do contexto de *Air Traffic Flow Management* (ATFM). Além disso, não existe a necessidade de demandar tempo imediato para ações que ocorrerão em um momento posterior.

Outros aspectos que também foram avaliados estão relacionados aos algoritmos *Q-learning* e SARSA e, portanto, a como eles se comportam quando os parâmetros α e γ são alterados.

7.1 Simulação da Meta Gerência de Tráfego Aéreo

Nesta seção, a intenção é mostrar como funciona a simulação das trocas de mensagens entre os controladores em nível meta. O foco da análise é o aprendizado por reforço proposto no controlador em nível meta. Com as considerações apresentadas, é possível classificar o processo, no sentido de priorizar algumas mensagens mais importantes, em detrimento de outras menos importantes. A desconsideração de algumas mensagens não apresenta prejuízo grave ao sistema, porque é feita uma análise cuidadosa por parte do controlador em nível meta.

A simulação do meta gerente de mensagem, assim como no ATFMGS, considerou quatro aeroportos, que aqui foram chamados de Aeroporto A, Aeroporto B, Aeroporto C e Aeroporto D. Fazendo um paralelo à situação real apresentada pelo ATFMGS, (59), está sendo avaliada a comunicação via troca de mensagens entre os aeroportos Brasília (BSB), Garulhos (GRU), Congonhas (CGH) e Galeão (GIG), respectivamente. Levou-se em consideração a visão de Brasília, ou ainda do Aeroporto A. Essa situação é representada graficamente pela figura 7.1.

Cada aeroporto que consta na figura 7.1 foi representado na simulação como uma fonte geradora de mensagem. Portanto, existem três fontes geradoras de mensagens: Aeroporto B, Aeroporto C e Aeroporto D. Os três aeroportos enviam mensagens ao quarto aeroporto em análise, que é o Aeroporto A. Em geral, não existem diferenças entre a lógica de cada fonte geradora de mensagem. A intenção é elas representarem características importantes de cada aeroporto que precisam ser tratadas, tais como congestionamento, mal tempo e adequação

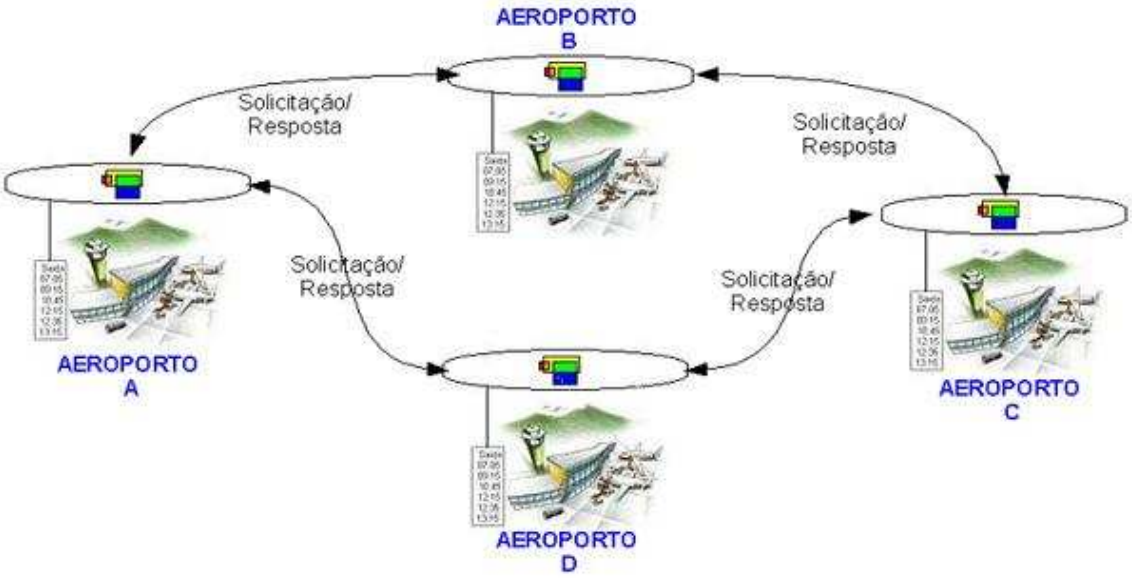

Figura 7.1: Simulação da comunicação envolvendo quatro aeroportos.

do escalonamento dos vôos, entre outros aspectos.

As adversidades enfrentadas pelos aeroportos B, C e D, que interferem no aeroporto A, podem ser simuladas por meio do intervalo de geração entre as mensagens. Portanto, para representar um processo de negociação intensa, entre os aeroportos envolvidos, as mensagens são geradas com espaçamento curto, entre uma e outra mensagem. De outra forma, um processo de negociação leve é representado com um espaçamento longo entre as mensagens.

Características das mensagens no contexto de tráfego aéreo

O simulador reproduz a fila de entrada do modelo meta gerente de mensagens, conforme pode ser observado no capítulo 6. Cada mensagem, nesse modelo, tem o significado que uma tarefa tem no modelo definido por Lesser e Raja (39).

Quando uma mensagem é gerada, dois metadados surgem a partir de sua geração: o prazo e a utilidade, que são específicos para cada mensagem. E, a partir deles, o modelo meta gerente, aqui proposto, cria outros quatro que o auxiliarão no processo de tomada de decisão. Os outros quatro parâmetros são: probabilidade de chegada de uma tarefa de alta utilidade, na *lista de entrada*; boa utilidade do conjunto agendado; prazo de execução do conjunto agendado; razão de fluxo no meta gerente de mensagens. Cada um dos parâmetros é descrito em detalhe no Apêndice B.

Uma mensagem é composta por dados, que é o próprio conteúdo da men-

sagem, e também por metadados, que são parâmetros das mensagens e serão utilizados na avaliação de suas características, para tratamento da mensagem.

Os *metadados* de uma mensagem informam características de uma determinada mensagem, em um dado instante. Como um exemplo típico de metadado, têm-se a utilidade de uma mensagem e o seu prazo para execução.

Os *dados* de uma mensagem são informações que um aeroporto origem pode informar ao seu aeroporto destino, durante um processo de negociação. E essas informações, ao final, influenciarão a tabela de escalonamento dos vôos. O conteúdo da mensagem terá relevância, porque exerce influência nos metadados. Um exemplo é a urgência de escalonamento de um vôo que apresente baixa quantidade de combustível exercendo influência no prazo de execução da mensagem.

Cada período significa um ciclo completo da mensagem, no controlador em nível meta, e uma quantidade aleatória de mensagens foi considerada por período.

Sobre o parâmetro *probabilidade de chegada de uma tarefa de alta utilidade na Lista de Entrada*, o estudo de controle em nível meta tomado como referência, Raja e Lesser (38), afirma que, se uma mensagem com alta utilidade chegar no tempo presente, existe grande chance de que a próxima mensagem chegue com alta utilidade. No modelo, tal valor foi considerado como a estimativa de 60%.

Na simulação, uma quantidade alta de mensagens foi considerada como sendo 200 mensagens por período de avaliação. E o prazo das mensagens, como um fator importante no processo de tomada de decisão do MODEC e tem seu valor avaliado segundo as regras:

- se ele compreende um período até 6 horas, é considerado curto;
- se o prazo é maior que 6 horas e menor ou igual que 12 horas, é considerado médio;
- se o prazo é maior que 12 horas e menor ou igual 48 horas, é considerado longo.

A freqüência do tipo de mensagem que chega ao controlador em nível meta é levada em consideração durante todo o processo de tomada de decisão.

Cada agente controlador em nível meta tem como objetivo tratar as mensagens que chegam mediante de um processo de tomada de decisão. O agente de aprendizado está subordinado aos interesses do agente controlador em nível meta.

Características da aprendizagem no estudo de caso

Pode ocorrer que, em algumas situações, os agentes tendem a piorar à medida que aprendem. Isso pode acontecer por causa da exploração do agente, que, na tentativa de atingir melhores resultados que os já alcançados, sorteia uma ação a ser executada, inserindo um fator randômico no resultado. Outro fato para que isso ocorra é o ambiente ser estocástico, e, desta maneira, o que pode ser uma boa ação no momento atual poderá não ser nos instantes que se seguem.

Os dados obtidos durante o uso de cada agente, em determinado momento, são exibidos em gráficos, permitindo análise comparativa entre eles.

7.2 Avaliação do Desempenho no Meta Gerente de Mensagens

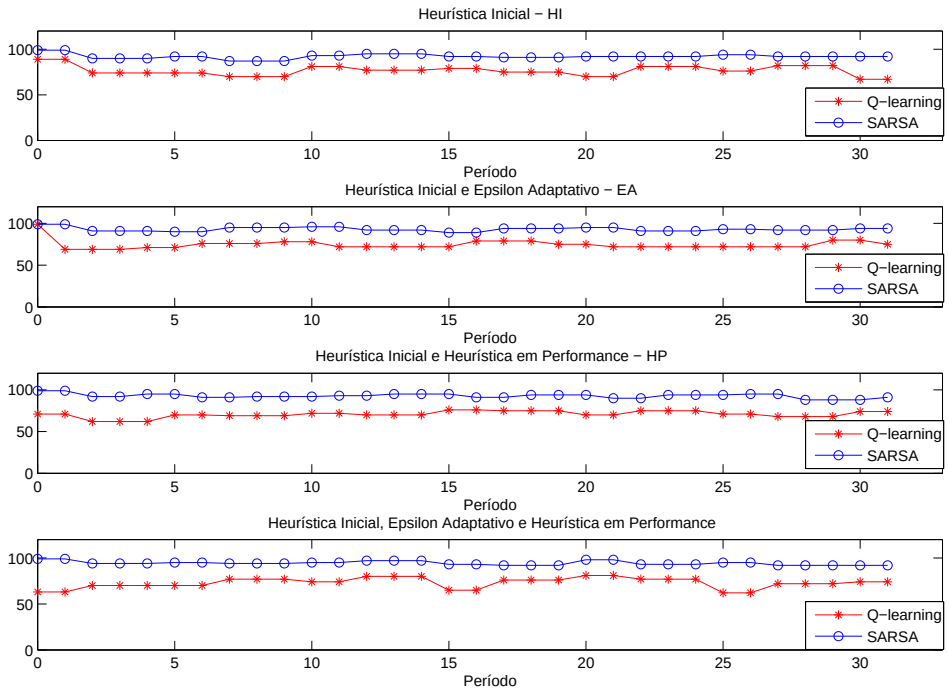

Figura 7.2: Avaliação de desempenho para os algoritmos *Q-learning* e SARSA.

A figura 7.2, Avaliação de Desempenho dos Algoritmos *Q-learning* e SARSA, mostra o desempenho do meta gerente de mensagens obtido com a implementação do algoritmo *Q-learning* e SARSA, quanto às estratégias propostas: heurística inicial, epsilon adaptativo e heurística baseada em performance.

Para avaliar, nesta situação, verifica-se a razão de fluxo do sistema. Assim, a quantidade das mensagens que chegam à *lista de entrada* é comparada à que sai

para a *lista de pronto*, definindo, assim, uma razão de fluxo para o meta gerente de mensagens.

No primeiro caso, o agente conta com a heurística inicial, que guia o agente *Q-learning* nos primeiros momentos, que ele ainda não tem conhecimento do ambiente onde está atuando. No segundo caso, a proposta do epsilon adaptativo combinada com a heurística inicial é avaliada. No terceiro caso, a proposta da heurística baseada em desempenho, combinada com a heurística inicial, é avaliada. E, por último, no quarto caso, todas as melhorias propostas - heurística inicial, epsilon adaptativo e heurística baseada em desempenho - são avaliadas.

Considera-se uma distribuição de probabilidade uniforme, no início da simulação, para a melhoria do epsilon adaptativo.

A seguir, os gráficos são comentados em três visões diferentes:

- **Avaliação do desempenho do algoritmo *Q-learning*** - epsilon adaptativo sobressaiu-se em relação aos demais. Heurística em performance e heurística inicial tiveram desempenho parecidos. E, por fim, a aplicação das três estratégias propostas obteve o pior desempenho, para o algoritmo *Q-learning* porque gerou a pior curva.

- **Avaliação do desempenho do algoritmo SARSA** - O algoritmo SARSA obteve bons resultados em todas as estratégias propostas. Iniciou com bons valores e permaneceu bem até o período 32.

- **Comparação entre os dois algoritmos** - Ao comparar os gráficos, infere-se que as implementações do SARSA obtêm melhores resultados, em relação ao *Q-learning*, quanto à medida de desempenho proposta e avaliada.

A literatura justifica, em Sutton e Barto, (50), que os resultados foram alcançados em razão da natureza dos dois algoritmos. Enquanto *Q-learning* busca maximizar a recompensa do agente, a partir da consulta de ações que podem ser tomadas em função do estado atual, o SARSA simplesmente sorteia a ação a ser executada, baseando-se na distribuição das probabilidades das ações tomadas. Por conseguinte, o *Q-learning* é muito mais caro computacionalmente, porque a escolha da ação exige mais operações que o algoritmo SARSA.

7.3 Avaliação da Qualidade da Aprendizagem dos Agentes

Neste segundo estudo de caso, o propósito inicial é avaliar o nível de aprendizado do meta gerente de mensagens. Para tanto, foi definido um vetor de triplas (es-

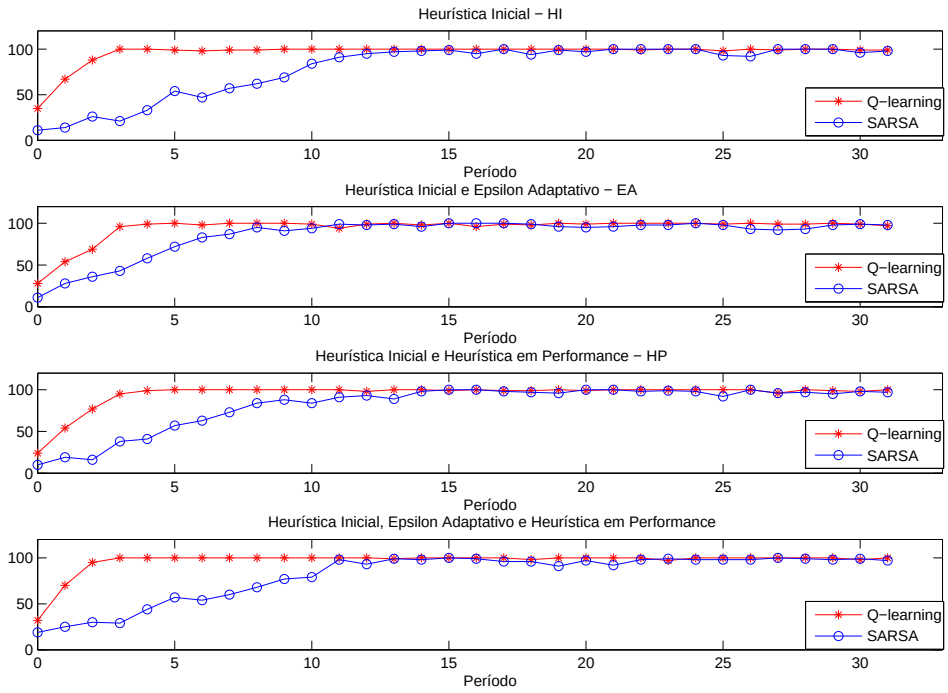

Figura 7.3: Avaliação da aprendizagem para os algoritmos *Q-learning* e SARSA.

tado, ação, recompensa), que é considerado aceitável, dependendo das condições em que o ambiente se apresente.

Nesta avaliação, o intervalo de geração entre as mensagens foi considerado constante e 800 milésimos de segundo. Os parâmetros estabelecidos para *alpha* e *gamma* foram 0,01 e 1, respectivamente.

Assim como na avaliação de desempenho, no primeiro caso, o agente conta com a heurística inicial que guia o agente *Q-learning* no começo, quando ele ainda não tem conhecimento do ambiente onde está atuando. No segundo caso, a proposta do epsilon adaptativo, combinada com a heurística inicial é avaliada. No terceiro caso, a proposta da heurística baseada em desempenho, combinada com a heurística inicial, é avaliada. E, por último, no quarto caso, todas as propostas - heurística inicial, epsilon adaptativo e heurística baseada em desempenho - são avaliadas quanto à qualidade da aprendizagem.

Considera-se uma distribuição de probabilidade uniforme, no início da simulação, para a estratégia do epsilon adaptativo. Para avaliar a qualidade da aprendizagem, é estabelecida uma nota ao agente, que define, em determinado instante, o quão boa está a aprendizagem em relação ao padrão tido como referência.

Uma nota é gerada a partir da ação tomada em relação ao que é sugerido pela heurística inicial, também considerando-se o grau de congestionamento do meio externo. A nota é exibida no gráfico da figura 7.3. A figura permite, ao longo do tempo, avaliar a qualidade da aprendizagem, comparando os algoritmos implementados, uns com os outros.

Uma vez que a decisão é tomada, ela é comparada com o padrão aceitável. Estando essa decisão dentro do padrão aceitável, o parâmetro nota é incrementado e armazenado, no vetor de notas, que inicialmente tem cem posições. Depois disso, retira-se a média aritmética das notas e altera-se o valor dela. Isso garante que boas decisões conduzirão ao crescimento da nota e más decisões ao decrescimento. É conveniente perceber que a nota não interfere no processo de aprendizado, pois é externo ao módulo de aprendizado, apenas avalia a aprendizagem, pela comparação com um padrão considerado bom. Os dados da análise são gerados em períodos fixos.

A cada período, a nota atual é recuperada. Pode-se verificar, portanto, o gráfico da aprendizagem *versus* período. Na figura 7.3, observa-se que as decisões tendem a seguir o padrão esperado, sendo que, nos pontos iniciais do gráfico, algumas decisões ruins são tomadas. No início, é atribuída nota zero e, à medida que o módulo toma boas decisões, essa nota é aumentada. Para acelerar o processo e evitar um número excessivo de decisões ruins, o módulo de aprendizagem vem com a heurística inicial (HI).

A seguir,os gráficos são comentados sobre três visões diferentes: a avaliação da qualidade do algoritmo *Q-learning*, a avaliação da qualidade do algoritmo SARSA e a comparação do resultado obtido com os dois algoritmos.

- **Avaliação da qualidade do algoritmo *Q-learning*** - A estratégia que emprega a heurística inicial, combinada com a heurística baseada em performance e as três estratégias propostas ao mesmo tempo, obteve as melhores curvas de aprendizagem que convergiram por volta do período 3. Epsilon adaptivo obteve a pior curva, e a heurística inicial obteve resultados intermediários.

- **Avaliação da qualidade do algoritmo SARSA** - A melhor abordagem para o algoritmo SARSA envolve todas as estratégias propostas ao mesmo tempo.

- **Comparação entre os dois algoritmos** - Ao comparar os gráficos dos dois algoritmos, infere-se que as implementações do *Q-learning* obtiveram

melhores resultados do que o SARSA em relação a qualidade da aprendizagem.

7.4 Avaliação do Meta Gerente de Mensagens em Relação ao Congestionamento do Aeroporto

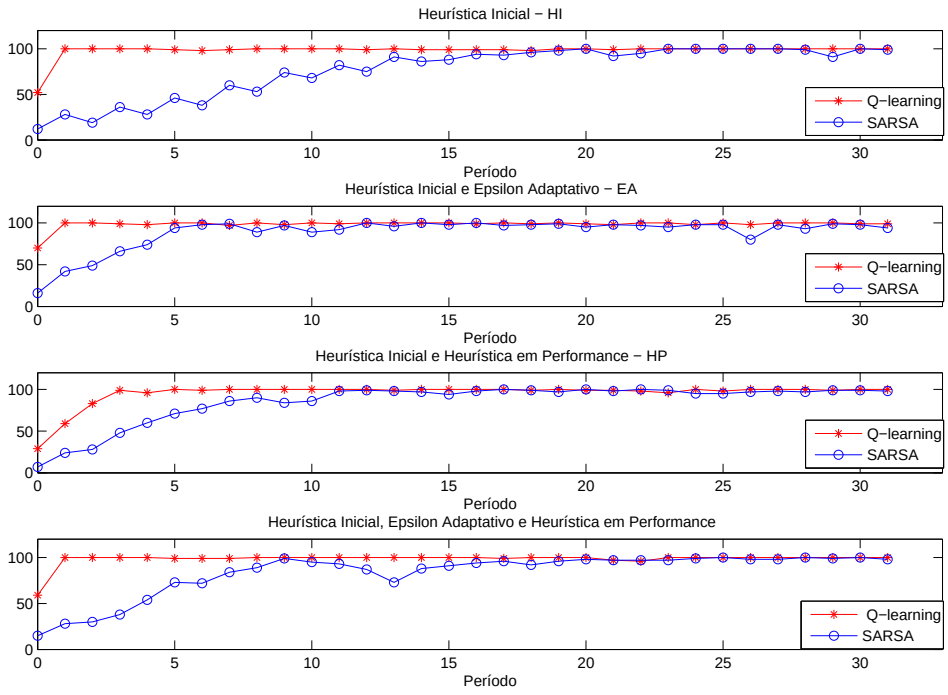

Figura 7.4: Avaliação da aprendizagem em congestionamento leve.

A simulação da intensidade de congestionamento do aeroporto foi realizada considerando o intervalo de geração entre uma e outra mensagem. Para o congestionamento intenso, o intervalo foi de 100 milésimos de segundo e, para o congestionamento leve, o espaçamento entre as mensagens foi de 1.600 milésimos de segundo.

Os resultados são mostrados no gráfico 7.4 e no gráfico 7.5. Em resumo, o *Q-learning* obteve melhores resultados em relação ao SARSA para todas as quatro avaliações.

Congestionamento leve

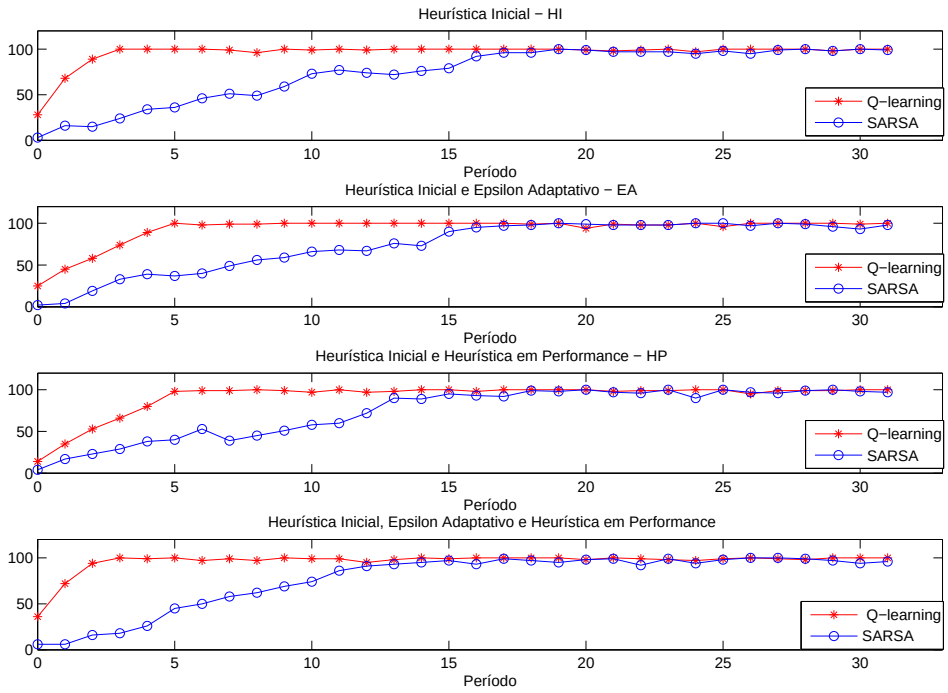

Figura 7.5: Avaliação da aprendizagem em congestionamento intenso.

De forma geral, os dois algoritmos, *Q-learning* e SARSA, operaram melhor no congestionamento leve. E, em relação ao aeroporto com congestionamento leve, os algoritmos *Q-learning* convergir nos momentos iniciais, antes do período 5, em todas as estratégias propostas. Contudo, o melhor resultado do SARSA, no aeroporto com congestionamento leve, foi com o uso da estratégia epsilon adaptativo. O pior resultado do SARSA foi usando somente a estratégia heurística inicial. Ele obteve resultados parecidos nas estratégias heurística baseada em performance e com as três estratégias ao mesmo tempo.

Congestionamento intenso

No congestionamento intenso, tanto o *Q-learning* quanto o SARSA demoraram mais períodos para convergirem. O melhor desempenho do *Q-learning* foi com a heurística inicial e com as três estratégias ao mesmo tempo, heurística inicial, epsilon adaptativo e heurística baseada em performance. O *Q-learning* obteve resultados semelhantes, ao empregar o epsilon adaptativo e a heurística em performance combinados à heurística inicial.

Os algoritmos SARSA demoraram ainda mais que o *Q-learning* para con-

vergir. E em geral, por volta do período 15 é que houve a convergência. Ele teve seus melhores resultados empregando as três estratégias ao mesmo tempo. Em seguida, HP e EA obtiveram resultados parecidos. O pior resultado, no congestionamento para o SARSA, foi com o uso somente da heurística inicial.

7.5 Avaliação dos Agentes quando o Parâmetro Alpha é Alterado

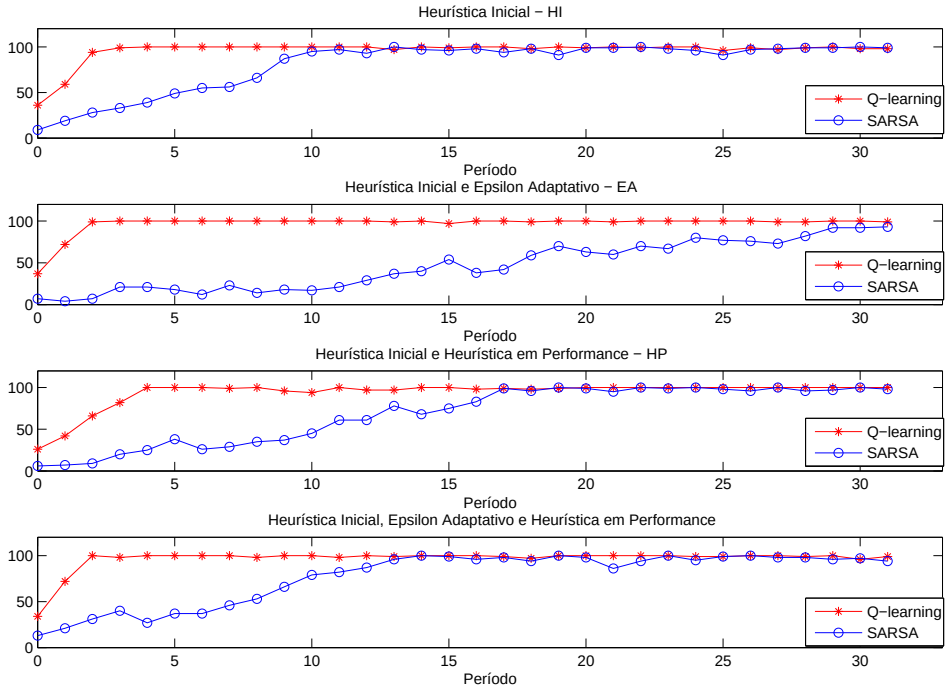

Figura 7.6: Avaliação dos agentes para *alpha* = 0,02.

As figuras 7.6 e 7.7 apresentam os resultados obtidos quando o parâmetro *alpha* é alterado. Ele foi alterado de 0,01, que foi considerado como um valor padrão para 0,02 e 0,04, respectivamente.

Os algoritmos *Q-learning*, combinado às estratégias propostas, operaram de forma semelhante, tanto para *alpha* igual a 0,02 quanto para 0,04. Sendo que o pior desempenho para 0,02 foi com a heurística baseada em performance.

Para heurística inicial, o SARSA foi um pouco melhor com *alpha* igual a 0,02. Para o epsilon adaptativo, ele foi pior com *alpha* igual a 0,02, se comparado a *alpha* igual a 0,04. Para heurística baseada em performance, ele operou melhor com *alpha* = 0,04. E, com as três estratégias propostas, ele operou um pouco

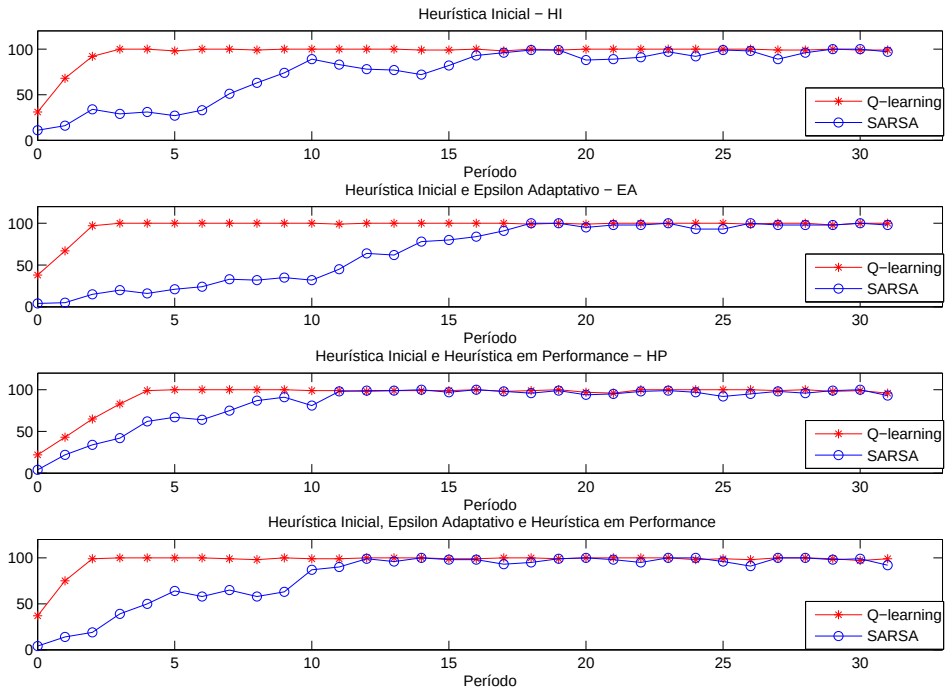

Figura 7.7: Avaliação dos agentes para *alpha* = 0,04.

melhor com 0,04. Então, no caso do SARSA, pode se afirmar que *alpha* igual 0,02 somente se mostra adequado à estratégia do epsilon adaptativo.

Para o algoritmo *Q-learning*, pode-se afirmar que os parâmetros 0,01; 0,02 e 0,04 se mostraram indicados, e o modelo comportou de forma semelhante para cada um dos parâmetros testados, na amostra de 32 períodos.

7.6 Avaliação dos Agentes quando o Parâmetro Gamma é Alterado

As figuras 7.8 e 7.9 apresentam os resultados obtidos quando o parâmetro *gamma* é alterado. Ele foi alterado de 1, que foi considerado como um valor padrão para 0,2 e 0,5, respectivamente.

O algoritmo *Q-learning*, combinado às estratégias, operou de forma semelhante, tanto para *gamma* igual a 0,2 quanto para 0,5. Sendo que o pior desempenho foi para 0,2 com uso da heurística baseada em performance.

Para heurística inicial, o SARSA foi um pouco melhor, com *gamma* igual a 0,2. Para o epsilon adaptativo, ele foi melhor com *gamma* igual a 0,5, se comparado a *gamma* igual a 0,2. Para heurística baseada em performance, ele

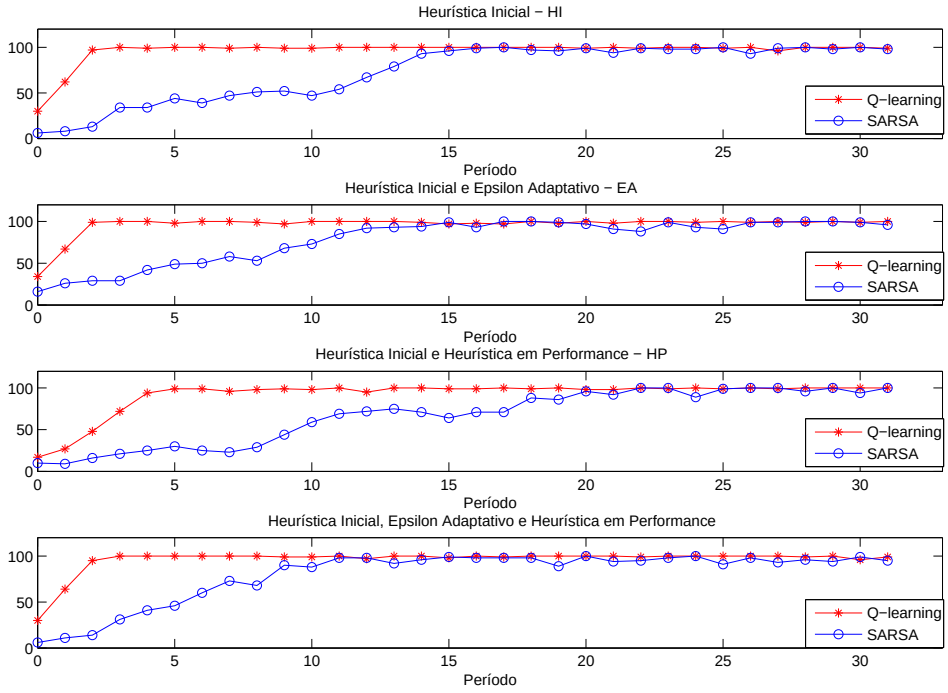

Figura 7.8: Avaliação dos agentes para $gamma = 0{,}2$.

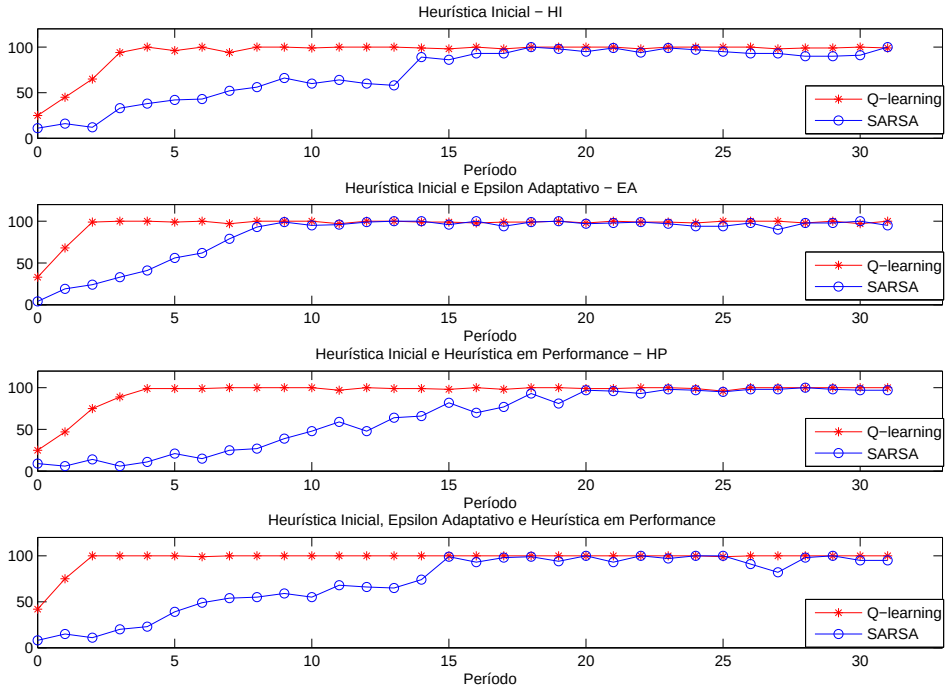

Figura 7.9: Avaliação dos agentes para $gamma = 0{,}5$.

operou pior com *gamma* igual 0,5. E, com as três estratégias propostas, ele operou um pouco melhor com 0,2. Portanto, pode se afirmar que *gamma* igual 0,2 se mostra mais adequado no caso do SARSA.

Para o algoritmo *Q-learning*, pode-se afirmar que os parâmetros 1, 0,2 ou 0,5 se mostraram indicados, e o modelo se comportou de forma semelhante para cada um desses parâmetros testados.

7.7 Considerações Finais

O algoritmo SARSA tem tempo de execução mais rápido do que o *Q-learning*, conforme pode ser verificado no estudo de caso 1 deste capítulo. Os experimentos realizados nos outros estudos de caso mostram, entretanto, que ele aprende mais lentamente, visto que precisa de mais períodos para atingir uma nota próxima de cem. Em contrapartida, o algoritmo *Q-learning*, que busca a otimização, leva uma quantidade bem menor de períodos para atingir uma nota próxima de cem.

Em geral, os agentes que utilizam o algoritmo SARSA apresentaram melhor desempenho em relação aos agentes que utilizavam *Q-learning*. Conforme mencionado na revisão de literatura, seção 3.5, os agentes que utilizavam SARSA estão mais sujeitos à randomicidade da distribuição de probabilidade na tomada de decisão. Os agentes *Q-learning*, porém, preocuparam-se em maximizar as recompensas obtidas.

Sobre os parâmetros α e $\_\gamma$, verifica-se que eles comportaram de forma diferente, para os algoritmos SARSA e *Q-learning*. No caso do *Q-learning*, os algoritmos combinados às estratégias propostas comportaram-se de forma semelhante, mesmo sendo alterados os parâmetros. Mas, no caso do algoritmo SARSA, houve diferença na curva de aprendizagem quando os parâmetros foram alterados. Para o *alpha* = 0,04, a maioria dos algoritmos SARSA operaram melhor.

Em relação ao parâmetro *gamma*, o algoritmo *Q-learning* também operou de forma semelhante, quando o parâmetro foi alterado. E o SARSA operou ligeiramente pior para *gamma* igual 0,5.

Capítulo 8

Conclusões e Trabalhos Futuros

A sincronização da escala de vôos para um conjunto de aeroportos motivou o trabalho realizado por Weigang nos anos de 1994 e 1997 (57; 58). Em seguida, a descentralização da informação do escalonamento foi proposta por Dib em 2004 (14). Esses trabalhos foram os precursores na elaboração de um projeto que visa à sincronização automática das escalas de vôos para um conjunto de aeronaves. Nesse sentido, o trabalho aqui proposto contribui ao apresentar uma camada para o gerenciamento das mensagens geradas pelos aeroportos.

A proposta do modelo meta gerente de mensagens se mostrou interessante de ser combinada à descentralização de informações entre os aeroportos por vários aspectos. Inicialmente, porque inseriu um processo de avaliação com intuito de escolher a melhor ação a ser tomada sob certos critérios (congestionamento de mensagens, grande tempo para avaliação, entre outros). Em seguida, porque usa aprendizagem no processo, permitindo adaptação ao longo do tempo. Por último, na aprendizagem foi observada a característica estocástica do ambiente. Não são consideradas somente regras estáticas que *apriori* foram previstas, mas uma combinação delas mediante a heurística inicial com as outras propostas: epsilon adaptativo e heurística baseada em performance.

Em tráfego aéreo, as situações de congestionamentos induzem à necessidade de tomar decisões rápidas, e o estudo aqui apresentado sugere o uso do algoritmo SARSA melhorado. Com ele, os resultados alcançados não são os melhores possíveis em termos da qualidade da aprendizagem. Todavia, se o congestionamento do aeroporto estiver leve, existe tempo para a busca da melhor ação a ser tomada, procurando-se maximizar a recompensa alcançada com uso do algoritmo *Q-learning* melhorado.

Outra contribuição deste trabalho foi a proposta de adaptações aos algoritmos *Q-learning* e SARSA originais, levando em consideração o desempenho do sistema hospedeiro em determinado instante. Assim, este modelo se mostra útil

em outros contextos em que é desejado o aprendizado em ambientes estocásticos que se comuniquem via troca de mensagens.

A pesquisa aqui apresentada e os resultados dos estudos de caso servem como insumos em decisões futuras de quais valores padrões utilizar pelo agente de aprendizagem por reforço. Escolher bem os parâmetros *alpha* e *gamma* que influenciam diretamente o usufruto do agente é importante para que o agente desempenhe bem a função de aprendizado.

A aprendizagem por reforço, ao ser combinada com as heurísticas, aumentou a velocidade de aprendizado. Em contrapartida, a exploração no agente inseriu fatores randômicos que, em alguns períodos, levou a resultados indesejáveis (conforme pode ser visto no estudo de caso). É o preço que o agente paga pelo aprendizado através da experiência.

Os estudos de caso apresentados no capítulo anterior mostram que uma proposta interessante é combinar os dois algoritmos de forma que inicialmente se atinja um nível de aprendizado satisfatório rapidamente, pelo uso do *Q-learning* com melhorias e, em seguida, passa-se a usar o SARSA sobre a base de conhecimento montada com o *Q-learning*. O SARSA garantirá um desempenho melhor no processo decisório. Vale ressaltar, porém, que esta proposta não funcionará bem em todos os casos, porque o ambiente é estocástico e muda ao longo de tempo.

Trabalhos futuros

Como trabalhos futuros, a sugestão é o uso de um modelo matemático da teoria de jogos conhecido como equilíbrio Nash. Nesta proposta, vários jogadores são empregados para estabelecer uma sociedade de agentes homogêneos que jogarão em um jogo simétrico. Nesse sentido, espera-se o estabelecimento de um jogo que corresponda ao benefício coletivo de todos os aeroportos envolvidos, sendo que cada aeroporto representa um jogador.

Outra sugestão de trabalho futuro é a alteração do modelo meta gerente de mensagens para a operação tanto na entrada da camada de gerência em nível meta de cada aeroporto, quanto na saída, fechando assim um ciclo completo da gerência das mensagens.

Para a tomada de decisão, outra possibilidade de pesquisa é o uso de redes bayesianas. De outra forma, elas também são indicadas para se lidar com incerteza.

Referências Bibliográficas

[1] A. ALONSO, L. F. ESCUDERO, and M. T. ORTUNO. A stochastic 0-1 program based approach for the air traffic flow managemment problem. *European Journal of Operational Reasearch*, 120:727–733, 2000.

[2] G. ANDREATTA, L. BRUNETTA, and G. GUASTALLA. The flow management problem: recent computational algorithms. *Frontiers of Interdisciplinary Research in the Life Sciences*, 6(6):727–733, 1998.

[3] L. BAIRD and A. MOORE. Gradient descedent for general reinforcement learning. *Advances in Neural Information Processing Systems*, 1999.

[4] R. BELLMAN. *Software Agents*. Press, United States, 2001.

[5] F. BIAN, E. K. BURKE, G. KENDALL, G. M. KOOLE, J. D. SILVA, J. MULDER, M. C. E. PAELINCK, C. REEVES, I RUSDI, and M. O. SULEMAN. Making airlane schedules more robust. *1st Multidisciplinary International Conference on Scheduling: Theory and Applications* ., pages 678–693, Ago 2003.

[6] R. A. C. BIANCHI and A. H. R. COSTA. Uso de heurísticas para aceleração do aprendizado por reforço. *Anais do Concurso de Teses e Dissertções do XXV Congresso da Sociedade Brasileira de Computação.*, Artificial Intelligence issue:130–139, Jul 2005.

[7] Y. BOCHI, T. OZONO, and T. SHINTANI. A direct-indirect reward sharing model in multiagent reinforcement learning. pages 940–941. ACM Press, 2003.

[8] A. BONZANO, P. CUNNINGHAM, and C. MECKIFF. Isac: a cbr system for decision support in air traffic control. *EWCBR, Advances in Case-Based Reasoning.*, pages 44–57, 1996.

[9] J. M. BRADSHAW. *Software Agents*. AAAI Press/The MIT Press, United States, 1997.

[10] T. J. CALLANTINE. Agents for analysis and design of complex systems. pages 567–573, 2001.

[11] G. CHALKIADAKIS and C. BOUTILIER. Coordination in multi-agent reinforcement learning: A bayesian approach. pages 709–716, 2003.

[12] A. B. CLARKE and R. L. DISNEY. *Probability and Random Process for Engineers and Scientists.* Wiley, United States, 1970.

[13] H. M. DEITEL and P. J. DEITEL. *Java: Como Programar. Terceira Edição.* Bookman, United States, 2001.

[14] M. V. P. DIB. Sistemas multi-agentes para sincronização e gerenciamento de fluxo de tráfego aéreo em tempo real., 2004. Dissertação de Mestrado - Universidade de Brasília.

[15] T. G. DIETTRICH. Hierarquical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[16] J. DOWLING and V. CAHILL. Self-managed descentralized systems using k-components and collaborative reinforcement learning. *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems TCD-CS*, pages 39–43, 2004.

[17] S. ELFWING, E. UCHIBE, K. DOYA, and H. I. CHRISTENSEN. Multi-agent reinforcement learning: Using macro actions to learn a mating task. 4:3164–3169, 2004.

[18] D. ERNST, P. GEURTS, and L. WEHENKEL. Iteratively extending time horizon reinforcement learning. In N. Lavra, L. Gamberger, and L. Todorovski, editors, *Proceedings of the 14th European Conference on Machine Learning*, pages 96–107. Springer-Verlag Heidelberg, September 2003.

[19] G. FARIA and R. F. ROMERO. Explorando o potencial de algoritmos de aprendizado com reforço em robôs móveis. *Apresentado no IV Congresso Brasileiro de Redes Neurais.*, pages 237–242, 1999.

[20] A. M. R. FERNANDES. *Inteligência Artificial: Noções Gerais.* Visual Books, 2003.

[21] C. N. FIETCHER. Efficient reinforcement learning. pages 88–97. ACM Press, 1994.

[22] I. GHORI. Reinforcement learning in board games. *Techinal Report: CSTR-04-004 of Departament of Computer Science*, May 2004.

[23] C. GOODCHILD, M. A. VILAPLANA, and S. ELEFANTE. A decision tool for atfm using a sthocastic approach for a ground hold strategy. *Fourth USA/Europe Air Traffic Management RD Seminar*, 2001.

[24] C. GOODCHILD, M. A. VILAPLANA, and S. ELEFANTE. A strategic and tactical tool for planning based and probability theory. *Fourth USA/Europe Air Traffic Management RD Seminar*, 2001.

[25] A. GOSAVI. A tutorial for reinforcement learning. `Disponível em: http://www.eng.buffalo.edu/ agosavi/tutorial.pdf`, 2004. Acesso em: 22 de mar. 2006.

[26] T ISHIDA, Y. SASAKI, and Y. FUKUHARA. A meta-level control architeture for production system. *Presenting in IEEE Transactions on Knowledge and Data Engineering.*, 7(1), 1995.

[27] A. J. KERR, T. W. NELLER, and M. D. LA PILLA, C. J.and SCHOMPERT. Java resources for teaching reinforcement learning. In *PDPTA*, pages 1497–1501, 2003.

[28] C. LARMAN. *Utilizando UML e Padrões: Uma Introdução à Análise e ao Projeto Orientados a Objetos.* Bookman, 2000.

[29] M. L. LITTMAN. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163. Morgan Kaufmann, 1994.

[30] G. F. LUGER. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* Addison Wesley, 4th edition, 2002.

[31] S. MANNOR. Dynamic abstration in reinforcement learning via clustering. *Proceedings of the* $21^{th} International Conference of Machine Learning.$, 2004.

[32] K. MORIYAMA and M. NUMAO. Construction of a learning agent handling its rewards according to environmental situations. *AAMAS*, 2002.

[33] L. NAVAZIO and G. ROMANIN-JACUR. The multiple connections multi-airport ground holding problem: Models and algorithms. *Transportation Science,*, 32:268–276, 1998.

[34] M. NGUYEN-DUC, J. P. BRIOT, A. DROGOUL, and V. DOUG. An application of multi-agent coordination techniques in air traffic management. *Proceedings of the International Conference on Human-Computer Interaction in Aeronautics (HCI-Aero 2002).*, pages 149–154, 2003.

[35] A. R. ODONI. The flow management problem in air traffic control. flow control of congested networks,. pages 269–288, 1987.

[36] T. PREVOT. Exploring the many perspectives of distributed air traffic management: The multi aircraft control systems macs. *Proceedings of the International Conference on Human-Computer Interaction in Aeronautics (HCI-Aero 2002).*, pages 149–154, 2002.

[37] A. RAJA. *Meta-Level Control in Multi-Agent System.* PhD thesis, University of Massachusetts, 2003.

[38] A. RAJA and V. LESSER. Efficient meta-level control reasoning in bounded-rational agents. pages 1104–1105. ACM Press, 2003.

[39] A. RAJA and V. LESSER. Meta-level reasoning in deliberative agents. *Proceedings IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-2004)IEEE*, pages 141–147, September 2004.

[40] S. O. REZENDE. *Sistemas Inteligentes: Fundamentos e Aplicações.* Manole Editora, 2003.

[41] C. H. C. RIBEIRO. A tutorial on reinforcement learning techniques. `Disponível em: http://student.ulb.ac.be/ aackerma/rlearn2.pdf`, 2005. Acesso em: 22 de mar. 2006.

[42] J. RIZZI and C. MULLER. Um modelo matemático de auxílio para o problema de controle de tráfego aéreo. *XVII Congresso em Pesquisa em Transportes*, 2003.

[43] J. RUMBAUGH, I. JACOBSON, and G. BOOCH. *The Unified Modeling Language Reference Manual.* Addison Wesley, United States, 1999.

[44] S. RUSSEL and P. NORVIG. *Artificial Intelligence: A Modern Approach.* Prentice Hall, New Jersey, 2002.

[45] M. R. G. SERRA. Aplicações de aprendizado por reforço em controle de trafego veicular urbano. `Disponível em: http://www.das.ufsc.br/camponog/Disciplinas //`

/DAS-5341/AprendizagemporReforco/dissertacao.pdf, 2004. Dissertação de Mestrado - Universidade Federal de Santa Catarina.

[46] C. SERVÉ, J. M. YEN, Z. B. ZANBINSKY, and L. GRIGNON. Incorporating weather uncertainty in airport arrival rate decisions. *FAA-NEXTOR-INFORMS Conference on Air Traffic Management and Control*, page 37, 2003.

[47] B. B. SOUZA. Um meta-controle aplicado ao fluxo de tráfego aéreo., 2004. Trabalho Final de Curso.

[48] S. STOLTZ and P. KY. Reducing traffic bunching more flexible in air traffic flow management. *4th USA/EUROPE Air Traffic Management RD Seminar.*, 2001.

[49] M. J. A. STRENS. Learning multi-agent search strategies. *Artificial Intelligence and the Simulation of Behavior.*, 3394/2005:245–259, 2005.

[50] R. S. SUTTON and A. G. BARTO. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[51] C. SZEPESVÁRI and W. D. SMART. Interpolation-based q-learning. *Presenting at Proceedings of the* $21^{st} International Conference on Machine Learning$, 2004.

[52] A. S. TANENBAUM and STEEN M. V. *Distributed Systems - Principles and Paradigms.* Prentice Hall, New Jersey, 2002.

[53] D. TARGA, C. J. P. ALVES, and M. J. MAHLER. Uma ferramenta automatizada no auxílio à alocação de slots para o problema de gerenciamento de fluxo de tráfego aéreo brasileiro. 2001.

[54] B. TETHER and S. METCALFE. Horndal at heatrow?. *CRIC Discussion Paper*, (46), 2001.

[55] G. TIDHAR, A. RAO, and M. LJUNBERG. Distributed air traffic management system. *Technical Report*, (25), 1992.

[56] C. WATKINS. *Learning from Delayed Rewards.* PhD thesis, The University of Cambridge, 1989.

[57] L. WEIGANG. *Knowledge-Based System for Air Traffic Flow Management:Time Table Rescheduling and Centralized Flow Control.* PhD thesis, Instituto Tecnológico de Aeronáutica - ITA, 1994.

[58] L. WEIGANG, C. J. P. ALVES, and N. OMAR. An expert system for air traffic flow management. *Advanced Transportation*, 31(3):343–361, 1997.

[59] L. WEIGANG, M. V. P. DIB, and A. C. M. A. MELO. Balanceamento de números de negociação entre agentes de sincronização de tráfego aéreo em grids computacionais. IEEE, 2006.

[60] G. WEISS. *Multi-agent Systems: A Modern Approach to Distributed Artificial Inteligence.* The MIT Press, United States, 2000.

Apêndice A

Processo Decisório de Markov

Algumas situações do cotidiano, na tentativa da compreensão de seu comportamento, podem ser divididas em elementos chamados estados. Nessas condições, um estado determina a situação de alguma coisa no sistema, em um dado instante. Assim, os possíveis estados futuros sucedem o estado corrente, que, por sua vez, é antecedido pela seqüência de estados anteriores, ou passados, Clarke e Disney (12).

Um exemplo de situação do cotidiano para os conceitos acima pode ser o contexto de um reservatório de água. Neste contexto, o nível de água no reservatório pode ser definido de forma simples com o nível de complexidade desejado através dos estados "emergencial", "normal"e "cheio". De forma mais elaborada, outras divisões discretas, como o percentual de capacidade, podem constituir estados. Nota-se que muitos estados (por exemplo, volume do reservatório) sugerem uma distribuição contínua.

Um processo de mudança de estados governado por probabilidades é chamado um processo estocástico, (12). Nele, as observações devem ser feitas sobre um período de tempo [0, ∞) e são influenciadas por efeitos que ocorrem, de maneira randômica, não considerando apenas um instante único, mas o intervalo inteiro de tempo ou uma seqüência inteira definida.

A.1 Processo Estocástico

Dado um processo estocástico que envolve o comportamento de um sistema ao longo do tempo, deve-se primeiramente especificar o conjunto de tempo T envolvido, (12). Isso será o intervalo de tempo na situação em que a medição será continuamente feita. Nesse caso, o processo é definido com parâmetro contínuo. Em outras situações, T poderá ser uma seqüência discreta de tempos e, nesse outro caso, será chamado processo estocástico com parâmetros discretos, consistindo

de uma seqüência de Inteiros consecutivos:

$$T = \{0, 1, 2, 3, ...\} \tag{A.1}$$

Por questão de elucidação, o T contínuo pode ser representado por:

$$T = \{t : 0t\} \tag{A.2}$$

Por exemplo, pode ser considerada uma lista de tempo t. De tempos em tempos, esta lista é verificada. E, em cada ponto de verificação, t não obterá um valor inteiro único, mas uma função inteira X_t.

Cada ponto da amostra ou saída experimental é denotado por s (estado), então a função pode-se denotar por $X_t(t)$. E X_t corresponde a um ponto de amostra único s, e é a realização de um processo estocástico.

A faixa de valores de X_t é uma coleção de valores numéricos que as variáveis randômicas X_t podem assumir e são chamados de espaço de estados. Eles devem ser um intervalo de número real.

Observa-se que, para um valor fixo de t, X_t é uma variável randômica que descreve o estado do processo no tempo t.

Dada uma coleção finita de funções $X_{t_1}, \mathrm{X}_{t_2}, ..., X_{t_n}$ como um conjunto de n variáveis randômicas com distribuição de probabilidade conjunta, a estrutura da probabilidade do processo X_t é completamente determinada e fornecida pela distribuição conjunta ou função de densidade para cada conjunto de variáveis randômicas. Basicamente, a análise do processo estocástico envolve determinar esta distribuição conjunta e usá-la para prever o comportamento do processo no futuro, dado um certo comportamento no passado.

Os antecedentes históricos não têm importância e não são considerados diretamente por esse método, mas indiretamente por meio das somas dos estados anteriores.

Seqüências independentes

Uma variável definida randomicamente de maneira idêntica e independente é uma seqüência que pode ser considerada resultado de repetições independentes de algum experimento randômico similar para cada repetição, (12). A ela é aplicada uma medida associada ou variável randômica X.

Dada uma seqüência de eventos independentes $X_1, \mathrm{X}_2, X_3, ...$ de variáveis randômicas, outra seqüência ou processo estocástico pode ser construído por meio da seqüência de somas parciais: $\mathrm{S}_1, S_2, S_3, ...$ de eventos dependentes, onde:

$$S_n = X_1 + X_2 + ... + X_n = \sum_{i=1} X_i \tag{A.3}$$

Se a variável original X_i forma uma seqüência independente, a seqüência de somas parciais não será independente, mas irá formar uma seqüência dependente de Markov. Vale lembrar que o processo decisório de Markov precisa que as variáveis sejam dependentes. A função randômica de t é chamada de processo estocástico ou processo randômico.

A noção de processo estocástico é útil para descrever matematicamente o comportamento de um sistema e sua evolução no tempo, ou para gerar um modelo probabilístico e aplicá-lo a outros problemas de escopo semelhante.

A.2 Fundamentos Matemáticos

Existem dois conceitos que devem ser conhecidos para facilitar a modelagem de um problema como um sistema de aprendizagem por reforço: a propriedade de Markov e o processo decisório markoviano.

Propriedade de Markov

Um processo de aprendizagem por reforço que satisfaz a propriedade de Markov é chamado Processo Decisório de Markov (PDM), ou seja, *Markov Decision Process* (MDP). Um MDP é uma seqüência de estados, com a propriedade de que qualquer predição de valor de estado futuro dependerá apenas do estado e ação atuais, e não da seqüência de estados passados.

Quando a probabilidade de transição de um estado s para um estado $s^{'}$ depende apenas do estado s e da ação a adotada em s, isso significa que o estado corrente fornece informação suficiente para o sistema de aprendizado decidir que ação deve ser tomada. Quando o sistema possui essa característica, diz-se que ele satisfaz a propriedade de Markov.

No caso mais geral, se a resposta em *t+1* para uma ação efetuada em t depende de todo o histórico de ações até o momento atual, a dinâmica do ambiente é definida pela especificação completa da distribuição de probabilidade, como mostra a equação abaixo:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0\} \tag{A.4}$$

onde a probabilidade Pr do estado s_{t+1} igual r é uma função que depende de todos os estados, ações e reforços passados.

Se a resposta do ambiente em *t+1* depende apenas dos estados e reforços em *t*, a probabilidade de transição para o estado *s'* é dada pela expressão da equação a seguir:

$$P^a_{s,s'} = Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t = a\} \tag{A.5}$$

A probabilidade de transição satisfaz às seguintes condições:

1) $P_{s,s'} \geq 0, paratodos, s' \in S, paratodoa \in A(S)$;

2) $\sum^a_{s,s'} = 1$, para todo s $\in S$, para todo a $\in A(S)$.

A propriedade de Markov é de fundamental importância na aprendizagem por reforço, uma vez que tanto as decisões quanto os valores são funções apenas do estado atual, abrindo a possibilidade de métodos de soluções incrementais, com os quais se pode obter soluções a partir do estado atual e para cada um dos estados futuros, como é feito no método de programação dinâmica.

A.3 Processo Decisório de Markov

Um Processo Decisório de Markov é chamado finito, se o espaço de estados e ações forem finitos. Formalmente, um PDM finito é definido por um conjunto (S,A,P,R), onde:

S representa o espaço finito dos estados do sistema;

A representa o espaço finito das ações;

P representa o modelo ou matriz de transição de estados;

$\mathbf{P}_a(s, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a$ define a probabilidade de uma ação *a* no estado *s* no tempo *t* levar ao estado *s'* no tempo *t+1*;

R(s) representa a recompensa no estado *s*.

Quando o espaço de estados não é diretamente observável no instante em que a ação se dá, o problema é chamado de Processo Decisório de Markov Parcialmente Observável, ou seja, *Partially Obsevable Markov Decision Process* (POMDP), Sutton e Barto (50). Por exemplo, um jogo de xadrex é observável. *Poker*, por sua vez, é parcialmente observável: a informação sobre a carta que o oponente tem na mão pode ser inferida a partir das cartas da mão do jogador, das cartas da mesa e das apostas.

Na área de inteligência artificial, algoritmos de aprendizagem por reforço fazem uso dos MDP e POMDP, se as probabilidades não são conhecidas, ou seja, a matriz completa não está disponível ou é de difícil construção.

Modelagem de um problema de aprendizagem por reforço como um processo decisório de Markov

Um MDP é uma quíntupla: (S, A, s_0, σ, r)

S, um conjunto de estados, ou ainda espaço de estados.

A, conjunto de ações disponíveis para o agente decidir.

s_0, estado inicial

σ, função de transição

r, função de recompensa

σ e r dependem do estado corrente e ação (propriedade de Markov), e os eventos são determinísticos.

A.4 Cadeia Markoviana

Na matemática, uma cadeia de Markov de tempo discreto é um processo estocástico de tempo discreto que apresenta a propriedade de Markov, chamada assim em homenagem ao matemático russo Andrei Andreyevich Markov. Em tal processo, os estados anteriores são irrelevantes para a predição dos estados seguintes, dado que o estado atual é conhecido.

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

$$Pr(X_{t+1} = x|X_0, X_1, X_2, X_3, ..., X_t) = Pr(X_{t+1} = x|X_t = y) \tag{A.6}$$

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

Tipos de visualização para uma cadeia markoviana

Uma maneira simples de visualizar um tipo específico de cadeia de Markov é através de uma máquina de estados finitos. Estando no estado y no tempo n, então a probabilidade do movimento para o estado x no tempo $n+1$ não depende do estado atual y. Assim, em qualquer tempo n, uma cadeia de Markov finita pode ser caracterizada por uma matriz de probabilidades cujo elemento (x, y) é dado por Pr ($X_{t+1} = x|X_t = y$).

Estes tipos de cadeia de Markov finitas e discretas podem também ser descritas por meio de um grafo dirigido, no qual cada aresta é rotulada com as probabilidades de transição de um estado a outro, sendo os estados representados como os nós conectados pelas arestas. Uma das principais vantagens da utilização da cadeia de Markov é a simplicidade matemática e operacional, pois os modelos markovianos podem fornecer projeções convenientes com poucos dados.

Assim, para quase todos os problemas de aprendizagem por reforço é suposto que o ambiente tenha a forma de um processo decisório de Markov, desde que seja satisfeita a propriedade de Markov pelo ambiente. Nem todos os algoritmos de aprendizagem por reforço necessitam de uma modelagem PDM inteira do ambiente, mas é necessário ter-se pelo menos a visão do ambiente como um conjunto de estados e ações.

Classificação dos estados das cadeias de Markov

Cada tipo de estado em uma cadeia markoviana pode ser classificado de diversas formas: alcançável, comunicante, absorvente, transiente e recorrente:.

- **Alcançável** - Um estado j é dito alcançável a partir de i, se $P_{ij} > 0$ para algum $n \geq 0$. Isso implica que é possível o sistema entrar no estado j eventualmente, quando este começa em i.

- **Comunicante** - Estados acessíveis, a partir de um estado i, são aqueles em que há um caminho que liga o estado i ao estado j. Caso a recíproca seja verdadeira, é dito que esses estados são comunicáveis. Se dois estados comunicam entre si, eles pertecem a uma mesma classe. Se todos os estados de um sistema forem comunicantes entre si e pertencerem a uma única classe, a cadeia é irredutível.

- **Absorvente** - Um estado especial é o estado absorvente, em que caso o sistema entre nele, não há como sair dele. Após entrar em um estado i, o processo não consegue mais voltar a esse estado, por conseguinte o estado é classificado como transiente. Portanto, o estado i é transiente, quando, a partir de i alcança-se j, mas i não é alcançável a partir de j. Em resumo, um estado é dito absorvente, quando, chegando a esse estado, o processo nunca mais irá deixá-lo. Assim, se i é absorvente, $P_{ij} = 1$ em todos os casos. Um estado absorvente é um caso particular de estado recorrente.

- **Recorrente** - Essa classificação diz respeito à probabilidade do processo retornar a um dado estado i, se o processo partiu desse estado. Estados

recorrentes são aqueles que, partindo do estado i, o processo eventualmente retornará ao estado i. Se um estado ocorre em um processo e o processo definitivamente irá retornar após a ocorrência desse estado, o estado é classificado como recorrente. Conseqüentemente, um estado é recorrente, se somente se, não é um estado transiente.

- **Transitório** - Estados transitórios são aqueles que, partindo do estado i, há uma probabilidade positiva de que o processo retornará a esse estado. Os estados recorrentes e transitórios podem coexistir em uma cadeia de Markov.

Caso um estado seja periódico, tem-se que ele pode ser alcançado apenas de tempos em tempos, ou seja, em um tempo m, $2m$, $3m$, ... em que m é inteiro maior que 1.

Caso o estado possa ser alcançado a qualquer tempo, ele é aperiódico. E, por fim, se todos os estados em uma cadeia são recorrentes, aperiódicos e comunicáveis, tem-se que a cadeia é *ergódica*.

Apêndice B

Representação Abstrata de Estados do Agente MGM

A representação abstrata, adotada pelo modelo, baseia-se nas características oriundas das mensagens, características estas derivadas a partir de outras originais, e também em fatores do ambiente. As características originais, na verdade, são metaparâmetros que chegam ao Módulo de Decisão e Controle - MODEC, com seus valores fornecidos. Elas são a boa utilidade da mensagem e o prazo de execução para mensagem.

A característica derivada das características originais é a probabilidade de chegada de uma nova mensagem com alta utilidade na *Lista de Entrada*.

As características observadas através da mudança do ambiente são: boa utilidade do conjunto agendado na Lista Agenda, Prazo de Execução do conjunto agendado, na *Lista Agenda* e *Razão de fluxo* no MGM. A seguir, cada parâmetro será descrito, em detalhe, com o intuito de esclarecer por que foram adotados pelo modelo.

- **Parâmetro 1: Boa utilidade da nova mensagem** - Descreve a utilidade de uma nova mensagem, baseada em como a tarefa é valorada, podendo ser: alta, média, baixa; em relação às outras, sendo recebidas pelo agente. Este parâmetro representará para o ambiente o quanto a mensagem que chega à *Entrada Lista* é importante para o sistema.

- **Parâmetro 2: Prazo de execução da nova mensagem** - Descreve a tensão do prazo de resposta, de uma particular mensagem. Ela determina o tempo máximo que uma mensagem tem para ser processada, pelo controle em nível meta, através de um processo de tomada de decisão, considerado pelo MODEC.

- **Parâmetro 3: Probabilidade de chegada de uma tarefa de alta**

utilidade na lista de entrada - Provê a probabilidade de chegada de uma mensagem de alta utilidade e com prazo de resposta curto, em um futuro próximo, usando informações da mensagem, como tipo da mensagem, a freqüência de chegada e a tensão de prazo para execução.

- **Parâmetro 4: Boa utilidade do conjunto agendado** - Descreve a utilidade do conjunto de mensagens que estão na Lista de Agenda em determinado momento. Podendo ser valorado como alta, média ou baixa.

- **Parâmetro 5: Prazo de execução do conjunto agendado** - Descreve a tensão do prazo de resposta da *Lista de Agenda*. De acordo com o prazo de execução de cada mensagem, será possível identificar o prazo de execução do conjunto. O menor prazo de execução que a *Lista de Agenda* contém servirá como parâmetro de decisão, para o MODEC pegar uma nova mensagem da Entrada Lista, ou encaminhar uma mensagem da *Agenda* para *Lista de Pronto*.

- **Parâmetro 6: Razão de fluxo no MGM** - Dependerá tanto do fluxo de mensagens na *Lista de Entrada*, quanto do fluxo de mensagens, na *Lista de Pronto* (saída do MGM). À medida que o acúmulo de mensagens na *Lista de Entrada* cresça em demasia, ações que reduzam o fluxo devem ser tomadas.

Apêndice C

Descrição do Framework de Aprendizagem por Reforço

Neste apêndice, são descritas todas as classes e interfaces que foram usadas do *Framework* de Aprendizagem por Reforço, encontrado em Kerr e Neller (27).

Interface Action

Uma interface de saída do agente que controla as ações.

Interface State

Apresenta o método isTerminal(), que verifica se o estado atual é terminal.

Classe Environment

É uma classe abstrata que implementa todos os ambientes possíveis. Um ambiente define um problema a ser resolvido. Esta classe determina a dinâmica do ambiente, a recompensa e as tentativas de término. Uma instância de classe *Simulation* é usada pelo ambiente para acessar uma simulação do sistema. Esta classe apresenta os métodos init(), starTrial(), e step(). O método init() inicializa uma instância de ambiente e atribui valores a algumas estruturas de dados. Geralmente, este método é chamado uma vez, quando a simulação do sistema é montada e inicializada. O método startTrial() desempenha alguma inicialização necessária do ambiente preparando-o, para começar uma nova tentativa. Ele cria e retorna, o primeiro estado de uma tentativa. O método step() possibilita uma transição do estado atual para o próximo estado, dependendo da ação tomada. O método retorna uma instância de *ActionResult*, descrevendo uma recompensa

e o próximo estado, depois de tomar uma ação, que estava no estado corrente. Se o estado terminal foi atingido, então a instância do próximo estado deverá ser nula. O método é chamado uma vez, através da instância da simulação, em cada passo. Os parâmetros do método step() são: ação e retorno.

Classe ActionResult

É uma classe que estende a classe padrão Java.lang.object. Esta classe combina instâncias de estado e recompensa numérica, para formar um objeto único que descreverá o resultado de tomar uma ação específica estando em determinado estado. É uma classe envoltória, de maneira que o ambiente consiga retornar, de uma só vez, os dois valores (estado e recompensa). Os parâmetros desta classe são o próximo estado e a recompensa.

Classe Simulation

Esta classe implementa o objeto-base da Interface. Uma instância de *Simulation* gerencia a interação entre o agente e o ambiente. No construtor de *Simulation*, deve-se instanciar os filhos da classe *Agent*. Os métodos dessa classe são: collectData(), init(), startTrial(). O método collectData() é chamado regularmente através dos métodos steps() e trials() de uma instância de *Simulation*. Os parâmetros do método collectData() são: estado, ação, próximo estado e a recompensa. O parâmetro do método init() é um vetor de objetos. Os parâmetros desta classe são: agente e ambiente.

Classe Agent

Esta classe estende a classe padrão Java.lang.object. Inicializa os parâmetros (*alpha*, *gamma*, *epsilon* e *lambda*). Inicializa a quantidade de ações. É uma classe abstrata, para implementar todos os agentes. Um agente é uma entidade que interage com o ambiente, recebendo estados e selecionando ações. O agente pode, ou não aprender e construir um modelo do ambiente. Agentes específicos são instâncias de sub-classe *Agent*. Esta classe apresenta os métodos: init(), startTrial() e step().