Pontifícia Universidade Católica
do Rio de Janeiro

**Leonardo Alfredo Forero Mendoza**

# Coordenação Inteligente Para Multiagentes baseados em Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço

**Tese de Doutorado**

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio.

Orientadora: Profa. Marley Maria Bernardes Rebuzzi Vellasco
Co-orientador: Profa. Karla Tereza Figueiredo Leite

Rio de Janeiro
Agosto de 2013

Pontifícia Universidade Católica
do Rio de Janeiro

**Leonardo Alfredo Forero Mendoza**

# Coordenação Inteligente Para Multiagentes baseados em Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

**Profa. Marley Maria Bernardes Rebuzzi Vellasco**
**Orientadora**
Departamento de Engenharia Elétrica - PUC-Rio

**Profa. Karla Tereza Figueiredo Leite**
**Co-orientadora**
UEZO

**Profa. Bianca Zadrozny**
IBM Research Brasil

**Prof. Paulo Fernando Ferreira Rosas**
IME

**Prof. Ricardo Tanscheit**
Departamento de Engenharia Elétrica - PUC-Rio

**Prof. Marco Antonio Meggiolaro**
Departamento de Engenharia Mecânica - PUC-Rio

**Prof. Eugênio da Silva**
UEZO

**Prof. José Eugenio Leal**
Coordenador Setorial do Centro
Técnico Científico - PUC-Rio

Rio de Janeiro, 20 de agosto de 2013

**Leonardo Alfredo Forero Mendoza**

Graduou-se em Engenharia Eletrônica Universidade Pontificia Bolivariana (PUC-Rio) em 2003, Mestrado em Engenharia de Telecomunicações pela UFF (2004) e Doutorado em Inteligência Computacional (2013).

Ficha Catalográfica

Mendoza, Leonardo Alfredo Forero

Coordenação Inteligente para Multiagentes Baseados em Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço / Leonardo Alfredo Forero Mendoza; orientadora: Marley Maria Bernardes Rebuzzi Vellasco; co-orientadora: Karla Tereza Figueiredo Leite – 2013.

184 f.: il. (color.) ; 30 cm

Tese (Doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2013.

Inclui referências bibliográficas.

1. Engenharia elétrica – Teses. 2. Coordenação. 3. Neuro-Fuzzy Hierárquico. 4. Futebol Robótico. 5. Aprendizado por Reforço. I. Vellasco, Marley Maria Bernardes Rebuzzi. II. Leite, Karla Tereza Figueiredo. III. Pontifícia Universidade Católica do Rio de Janeiro.

CDD: 621.3

## Agradecimentos

À Deus que me ilumino o caminho.

À minha orientadora Professora Marley, pelo apoio, amizade e paciência.

Ao minha co-orientadora Karla, pela confiança, estímulo e importantes contribuições.

A minha esposa Marcella e a João por dar mais razões para fazer as coisas melhor.

A minha Mãe por sempre apoiar-me e inculcar-me o amor pelo estudo.

A minha irmã Alexandra mi tia Sofia meus sobrinhos.

Meu pai e minha Tia Noemi que estão no céu

Aos meus amigos Álvaro, Manoela, Alan, Eugenio, Marco.

À todos os amigos da PUC-Rio.

Aos professores que participaram da Comissão examinadora.

Ao CNPq, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

## Resumo

Mendoza, Leonardo Alfredo Forero; Vellasco, Marley Maria Bernardes Rebuzzi (Orientadora); Leite, Karla Tereza Figueiredo (Co-orientadora). **Coordenação Inteligente para Multiagentes baseados em Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço.** Rio de Janeiro, 2013. 184p. Tese de Doutorado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Esta tese consiste na investigação e no desenvolvimento de estratégias de coordenação inteligente que possam ser integradas a modelos neuro-fuzzy hierárquicos para sistemas de múltiplos agentes em ambientes complexos. Em ambientes dinâmicos ou complexos a organização dos agentes deve se adaptar a mudanças nos objetivos do sistema, na disponibilidade de recursos, nos relacionamentos entre os agentes, e assim por diante. Esta flexibilidade é um problema chave nos sistemas multiagente. O objetivo principal dos modelos propostos é fazer com que múltiplos agentes interajam de forma inteligente entre si em sistemas complexos. Neste trabalho foram desenvolvidos dois novos modelos inteligentes neuro-fuzzy hierárquicos com mecanismo de coordenação para sistemas multiagentes, a saber: modelo Neuro-Fuzzy Hierárquico com Aprendizado por Reforço com mecanismo de coordenação Market-Driven (RL-NFHP-MA-MD); e o Modelo Neuro-Fuzzy Hierárquico com Aprendizado por Reforço com modelo de coordenação por grafos (RL-NFHP-MA-CG). A inclusão de modelos de coordenação ao modelo Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço (RL-NHFP-MA) foi motivada principalmente pela importância de otimizar o desempenho do trabalho em conjunto dos agentes, melhorando os resultados do modelo e visando aplicações mais complexas. Os modelos foram concebidos a partir do estudo das limitações existentes nos modelos atuais e das características desejáveis para sistemas de aprendizado baseados em RL, em particular quando aplicados a ambientes contínuos e/ou ambientes considerados de grande dimensão. Os modelos desenvolvidos foram testados através de basicamente dois estudos de caso: a aplicação benchmark do jogo da presa-predador (Pursuit- Game) e Futebol de robôs (simulado e com agentes robóticos). Os resultados obtidos tanto no jogo da presa-predador quanto no futebol de robô através dos novos modelos RL-NFHP-MA-MD e RL-NFHP-MA-CG para múltiplos agentes se mostraram bastante promissores. Os testes demonstraram que o novo sistema mostrou capacidade de coordenar as ações entre agentes com uma velocidade de convergência quase 30% maior que a versão original. Os resultados de futebol de robô foram obtidos com o modelo RL-NFHP-MA-MD e o modelo RL-NFHP-MA-CG, os resultados são bons em jogos completos como em jogadas específicas, ganhando de times desenvolvidos com outros modelos similares.

## Palavras-chave

Coordenação multiagente; aprendizado por reforço; neuro-fuzzy.

## Abstract

Mendoza, Leonardo Alfredo Forero; Vellasco, Marley Maria Bernardes Rebuzzi (Advisor); Figueiredo, Karla (Co-advisor). **Intelligent Coordination for Multiagent based Models Hierarchical Neuro-fuzzy with Reinforcement Learning.** Rio de Janeiro, 2013. 184p. DSc. Thesis - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

This thesis is the research and development of intelligent coordination strategies that can be integrated into models for hierarchical neuro -fuzzy systems of multiple agents in complex environments . In dynamic environments or complex organization of agents must adapt to changes in the objectives of the system , availability of resources , relationships between agents , and so on . This flexibility is a key problem in multiagent systems . The main objective of the proposed models is to make multiple agents interact intelligently with each other in complex systems . In this work we developed two new intelligent neuro- fuzzy models with hierarchical coordination mechanism for multi-agent systems , namely Neuro -Fuzzy Model with Hierarchical Reinforcement Learning with coordination mechanism Market - Driven (RL - NFHP - MA - MD), and Neuro - Fuzzy model with Hierarchical Reinforcement Learning with coordination model for graphs (RL - NFHP - MA - CG) . The inclusion of coordination models to model with Neuro -Fuzzy Hierarchical Reinforcement Learning (RL - NHFP - MA) was primarily motivated by the importance of optimizing the performance of the work in all players , improving the model results and targeting more complex applications . The models were designed based on the study of existing limitations in current models and desirable features for learning systems based RL , in particular when applied to continuous environments and / or environments considered large . The developed models were tested primarily through two case studies : application benchmark game of predator-prey ( Pursuit - Game) and Soccer robots (simulated and robotic agents) . The results obtained both in the game of predator-prey as in soccer robot through new models RL - NFHP - MA - MD and RL - NFHP - MA - CG for multiple agents proved promising . The tests showed that the new system showed ability to coordinate actions among agents with a convergence rate nearly 30 % higher than the original version . Results soccer robot were obtained with model RL - NFHP - MA - MD – NFHP-RL and model - CG - MA , the results are good in games played in full as specific winning teams developed with other similar models .

## Keywords

Multiagent coordination; reinforcement learning; neuro-fuzzy.

## Agradecimentos

À Deus que me ilumino o caminho.

À minha orientadora Professora Marley, pelo apoio, amizade e paciência.

À minha co-orientadora Karla, pela confiança, estímulo e importantes contribuições.

A minha esposa Marcella e a João por dar mais razões para fazer as coisas melhor.

A minha Mãe por sempre apoiar-me e inculcar-me o amor pelo estudo.

A minha irmã Alexandra mi tia Sofia meus sobrinhos.

Meu pai e minha Tia Noemi que estão no céu

Aos meus amigos Álvaro, Manoela, Alan, Eugenio, Marco.

À todos os amigos da PUC-Rio.

Aos professores que participaram da Comissão examinadora.

Ao CNPq, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

## Sumário

## Lista de Figuras

## Lista de tabelas

# 1
# Introdução

## 1.1
## Motivação

Atualmente, em várias aplicações, há a necessidade de inserir autonomia nos sistemas computacionais, isto é, fazer com que esses sistemas incorporem a capacidade de decidir, sem intervenção humana, o que deve ser executado para satisfazer determinados objetivos previamente especificados pelo operador. Os agentes representam o paradigma utilizado no desenvolvimento de aplicações de software que visam a atender a essa necessidade, uma vez que percebem o ambiente através de sensores e agem nesse ambiente de forma independente, através de atuadores.

Entre as características que um agente pode possuir, a autonomia é aquela que determina que programas (códigos ou algoritmos) possam ser classificados como agentes. A autonomia é a capacidade de o agente seguir seus objetivos autonomamente, isto é, sem comandos que venham do ambiente. Outra característica de grande importância, por aumentar a autonomia de um agente, é a inteligência. (Brenner et. al. 1998) são enfáticos ao afirmar que o agente deve ser capaz de aprender para ser considerado autônomo. O aprendizado juntamente com a base de conhecimento e o raciocínio integram a característica da inteligência.

Um agente inteligente deve interagir com o ambiente para atingir seus objetivos: deve ser capaz de reunir informações sobre o ambiente, tomar decisões baseadas nestas informações e iniciar uma ação específica baseada nessas decisões. Recentemente, um novo campo de pesquisa na área de Inteligência Computacional vem emergindo, com o objetivo de desenvolver sistemas complexos envolvendo múltiplos agentes. Nesse caso, dependendo da aplicação, pode ser necessário o desenvolvimento de mecanismos de coordenação entre eles: os chamados Sistemas Multiagentes (SMA). Um SMA pode ser definido como um grupo de agentes autônomos, interagindo entre si e compartilhando um mesmo ambiente, que é percebido através de sensores, e onde eles agem realizando ações

através dos atuadores (Vlassis, 2003). Esse tipo de sistema vem encontrando uma grande variedade de campos de aplicação, como controle de processos (Stephan ET al., 2000), controle de sinais de trânsito (Wiering, 2000; Bakker et al., 2005), controle de redes de energia elétrica (Riedmiller et al., 2000), times robóticos (Ishiwaka et al., 2003; Kok et al., 2005; Xiao e Tan, 2007; Hladek, 2007), controle de navegação (Mataric, 1997; Veloso, 2002; Hu e Wellman, 2003; Calvo e Romero, 2007), definição de estratégias de negociação e *trading* (Lee, 2002; Lee et al., 2007; Wellman et al., 2003), entre outros.

Os benefícios trazidos pela aplicação de múltiplos agentes são diversos. Em primeiro lugar, através da computação paralela, vários agentes podem trabalhar em conjunto para melhor explorar a estrutura descentralizada de uma determinada tarefa e acelerar sua conclusão (Lagoudakis, 2002; Kok et al., 2005; Fitch et al., 2005). Além disso, agentes podem trocar experiências se comunicando (Tan, 1993), podem observar os mais habilidosos e aprender (Price e Boutilier, 2003), e os mais inteligentes podem servir de professores para outros agentes (Clouse, 1995). Os SMA também podem fornecer alto grau de escalabilidade, através da inclusão de novos agentes quando necessário, e ainda fazer com que agentes assumam as atividades de outros agentes em casos de falha (Busoniu, 2008). Os SMA também se destacam no estudo e compreensão da inteligência (Weiss e Sen, 1996). Como a inteligência está fortemente associada à interação, a melhor forma de criar máquinas inteligentes pode ser através da construção de redes sociais de máquinas.

A coordenação é uma característica fundamental de um sistema de múltiplos agentes que executam alguma atividade em um ambiente compartilhado (Goodwine, 2010). Ela está intimamente relacionada ao compartilhamento de conhecimento entre os agentes, e o seu principal objetivo é coordenar as ações individuais de cada agente para atingir o objetivo final do sistema multiagente (Berndt, 2011). Os mecanismos de coordenação da maioria dos sistemas multiagentes podem ser classificados como implícitos (centralizados) ou explícitos (distribuídos) (França, 2011). Esses mecanismos são úteis em ambientes simples, com um único objetivo (Eguchi, 2006; França, 2011; Vlassis, 2003). Entretanto, existem ambientes nos quais o objetivo final é alcançado cumprindo-se uma série de subobjetivos e condições, tendo os agentes diferentes

papéis. Estes ambientes demandam mecanismos de coordenação bem mais elaborados e um conhecimento maior do ambiente por parte do operador (Goodwine, 2010; Reis, 2007). A ausência de um mecanismo de coordenação mais elaborado nestes ambientes complexos acarreta o desaparecimento dos benefícios da execução distribuída de tarefas, e o grupo de agentes pode degenerar numa coleção caótica e incoerente de comportamentos individuais (Berndt, 2011). Diversos problemas podem ocorrer em um sistema multiagente sem mecanismo de coordenação, tais como: conflitos no acesso aos recursos disponíveis ou uso indevido; redundância na realização das tarefas; e aumento do tempo de espera, quando a atividade de um agente depende da realização de atividades por outros agentes (Reis, 2007); entre outros. Assim, um bom sistema de coordenação deve evitar ou minimizar a ocorrência desses problemas, otimizando os recursos e o tempo para alcançar os objetivos (Sellner, 2006). Uma coordenação eficaz entre agentes, que operam em um ambiente compartilhado, contribui para o aumento da qualidade das soluções atingidas, e para a melhora do desempenho na resolução de tarefas (Karray, 2008). Os benefícios trazidos por uma boa estratégia de coordenação são difíceis de perceber, sendo mais fácil a percepção da ausência ou falha no sistema de coordenação, por exemplo, quando ocorrem problemas de logística nos aeroportos, ou quando jogadores de uma equipe de futebol estão mal posicionados no campo ou erram muitos passes (Karray, 2008).

Atualmente, muitos dos sistemas multiagentes que tratam cenários complexos focam na coordenação entre eles, encontrando aplicações em áreas como: distribuição de energia elétrica (Hui, 2010; Hao, 2012;), navegação de aviões e veículos não tripulados (Savkin, 2010; Tumer, 2009), sistemas de telecomunicações (Jianxin, 2008; Naeini 2008) e futebol de robô (Hwang, 2007; Kose, 2005).

A motivação principal para a realização deste trabalho está na importância crescente dos Sistemas Multiagente e das metodologias de coordenação para melhorar seu desempenho. De fato, sob a designação geral de coordenação assentam metodologias como a cooperação, a negociação, a resolução de conflitos e a alocação de recursos em SMA, que constituem uma parcela muito relevante da investigação global realizada em Inteligência Artificial Distribuída.

O modelo neuro-fuzzy hierárquico multiagente RL-NFH-MA (França, 2010) permite a extração de conhecimento a partir da interação direta dos agentes com o ambiente, sem utilizar algoritmos supervisionados, de forma que eles aprendam quais ações devem ser executadas em ambientes grandes e/ou contínuos. Esse algoritmo tem como vantagem a superação de certas limitações presentes em algoritmos tradicionais de aprendizado por reforços (França, 2010). Este modelo, entretanto, está restrito a ambientes simples, com um único objetivo. Deste modo, o objetivo deste trabalho é estender o modelo RL-NFH-MA, unindo as vantagens já existentes às estratégias de coordenação, de forma a permitir aplicá-lo a ambientes complexos e com múltiplos objetivos, melhorando seu desempenho e aperfeiçoando a comunicação entre os agentes.

O xadrez foi durante muitos anos um dos domínios da aplicação das metodologias de Inteligência Artificial. No entanto, após a vitória da máquina (Deep Blue) sobre o campeão do mundo humano (Gary Kasparov), novos domínios mais estimulantes e complexos tornaram-se necessários. É nesse contexto que surge o futebol robótico (Reis, 2007; Karray, 2008). A importância crescente do RoboCup como domínio de teste e problema padrão para a Inteligência Artificial Distribuída e Robótica Inteligente constitui outra motivação evidente deste trabalho (Reis, 2007).

O futebol robótico é um domínio bem mais complexo que o xadrez, uma vez que o ambiente é multi objetivo, dinâmico, contínuo e não determinístico. Além disso, o domínio é inerentemente distribuído, descentralizado, parcialmente cooperativo e parcialmente adverso (Reis, 2007). Os problemas de pesquisa motivados pelo futebol robótico cobrem uma área muito ampla, mas onde se destacam claramente os problemas da coordenação e cooperação entre agentes. Desse modo, o futebol robótico foi selecionado como principal domínio de teste das metodologias de coordenação de agentes cooperativos desenvolvidas no âmbito desta tese.

## 1.2
## Objetivos

O objetivo principal deste trabalho foi o desenvolvimento, implementação e avaliação de estratégias de coordenação multiagente, integradas aos modelos neuro-fuzzy (híbridos) da família RL-NFHP, de forma a permitir a coordenação eficiente e ótima de agentes em ambientes complexos, mistos e com múltiplos objetivos.

Os objetivos secundários deste trabalho são:

- Estudo dos conceitos de Agente, Sistemas Multiagentes e de metodologias de coordenação, negociação e cooperação entre agentes inteligentes autônomos;
- Integração de métodos inspirados no livre mercado em coordenação de SMA;
- Criação e expansão da estrutura de regras dos múltiplos agentes inteligentes, sem o conhecimento prévio do número de regras;
- Extração de conhecimento a partir da interação direta de multiagentes com o ambiente, sem utilizar algoritmos supervisionados, de forma a aprender quais ações devem ser executadas em ambientes grandes e/ou contínuos de forma coordenada;
- Produção de resultados linguisticamente interpretáveis, sob a forma de regras fuzzy, que constituam o raciocínio que os múltiplos agentes devem construir para atingir suas metas;
- Capacidade de realizar o aprendizado eficiente de múltiplos agentes, fazendo com que estes possam explorar o espaço de estados - ações de forma simultânea;
- Estudo do domínio do futebol robótico nas suas variadas vertentes, incluindo todas as ligas robóticas e desafios associados, com ênfase no futebol robótico simulado;
- Implementação de uma equipe de futebol robótico capaz de participar na liga RoboCup.

## 1.3
## Organização da Tese

O restante desta tese está organizado em mais seis capítulos. No capítulo 2 faz-se uma breve introdução às áreas de Agentes e de Modelos de Aprendizado, destacando-se o aprendizado por reforço. Nesse capítulo, são explorados os principais conceitos, aspectos teóricos, aplicações e desafios relacionados aos Sistemas Multiagentes.

O capítulo 3 introduz o modelo multiagente RL-NFH (RL-NFH-MA), explorando aspectos como o aprendizado de diferentes tarefas de forma simultânea e o compartilhamento do conhecimento dos agentes em uma mesma estrutura.

O capítulo 4 explica a importância da coordenação de sistemas multiagentes e suas características. O capítulo também descreve a coordenação competitiva e a cooperativa, suas estruturas e características. O capítulo foca na coordenação estratégica e nas abordagens da coordenação *market driven* e coordenação por grafos.

O capítulo 5 apresenta detalhadamente os dois modelos propostos nesta tese: os modelos RL-NFHP-MA-MD e RL-NFHP-MA-CG. As estruturas de aprendizado dos modelos são explicadas, descrevendo suas particularidades, funcionamento e objetivos. Também são apresentadas a dinâmica da comunicação entre os agentes e a utilização do modelo em ambientes competitivos e cooperativos.

O capítulo 6 apresenta os resultados do trabalho, obtidos a partir da aplicação do modelo proposto no problema presa-predador e no futebol de robô, e a comparação com outros modelos existentes.

As conclusões, comentários finais e trabalhos futuros são apresentados no capítulo 7.

# 2
# Agentes Inteligentes e Sistemas Multiagentes

## 2.1
## Agentes Inteligentes

O processamento baseado em agentes pode ser caracterizado, em um nível alto de abstração, por um estado de entrada que, após processado, produz uma saída. O estado de entrada é determinado pelo ambiente em que o agente atua e a saída é determinada pelos efeitos que o agente causa em seu ambiente. A despeito de existirem muitas similaridades entre a teoria de agentes e a teoria de controle, agentes encontram sua melhor aplicação em situações específicas, como ambientes complexos e dinâmicos ou quando as informações sobre o ambiente são incompletas (Brenner et. al, 1998; Müller, 1996).

Agentes podem ser classificados segundo várias características. As principais são autonomia e inteligência.

**Autonomia:** uma das mais importantes diferenças entre agentes e programas tradicionais é a capacidade que um agente possui de seguir seus objetivos automaticamente, isto é, sem comandos externos. Para permitir este comportamento autônomo, o agente deve ter controle sobre suas ações e percepção sobre seu estado e o ambiente. O grau de autonomia deve ser especificado pelo usuário/programador, uma vez que podem existir situações nas quais não se deseja que o agente tenha completa autonomia no processo de decisão, como, por exemplo, em sistemas relacionados à área financeira ou em operações que envolvam a vida humana (Brenner et. al. 1998).

**Inteligência**: a característica da inteligência de um agente é formada por um ou mais dos seguintes componentes:

- a base de conhecimento;
- a capacidade de raciocínio baseado em sua base de conhecimento;
- a capacidade de aprender ou de se adaptar às mudanças ocorridas no ambiente.

A capacidade de raciocínio permite ao agente observar o ambiente e tomar decisões específicas quando mudanças ocorrem no mesmo. A capacidade de aprendizado e adaptação ao ambiente amplia a capacidade autônoma do agente. O aprendizado é, portanto, o componente mais importante, e também mais complexo, da inteligência de um agente (Nwana, 1996; Brenner et. al, 1998; Russel e Norvig, 2010).

Além dos aspectos de autonomia e inteligência, os agentes podem ser também denominados: a) **estáticos:** todo o seu processamento está baseado em um único computador; b) **móveis**: têm habilidade de se mover através de algum tipo de rede de computadores; c) **reativos:** não têm qualquer tipo de modelo do ambiente, agindo em resposta a estímulos do ambiente; d) **deliberativos:** possuem internamente um modelo de ambiente, planejando e negociando com o objetivo de coordenarem suas atividades; e) **sociáveis**: é a capacidade de um agente interagir com outros agentes (ou usuários) para satisfazer seus objetivos; f) **pró-ativos:** o agente não reage apenas a mudanças no ambiente, mas possui iniciativa própria sob certas circunstâncias (Müller, 1996; Nwana e Azarmi, 1997; Brenner et. al, 1998; Russel e Norvig, 2010).

Os agentes podem acumular duas ou mais características citadas anteriormente. Muitas dessas características ainda apresentam um conceito bastante vago, permitindo muitas interpretações, porém a inteligência é a que mais apresenta conceitos abstratos.

## 2.2
## Inteligência

Um agente autônomo deve interagir com o ambiente para atingir seus objetivos. Isso é possível quando o agente é capaz de reunir informações sobre o seu ambiente, tomar decisões baseadas nestas informações, e iniciar ações específicas baseadas nas decisões.

O agente inteligente deve ser capaz de se adaptar ao ambiente em que está inserido e de aprender, sem que seja necessário dar ao agente qualquer conhecimento explícito sobre o ambiente (Monsieurs, 2002).

Alguns autores abordam a característica inteligência de formas diferentes. Brooks (1991), por exemplo, diz que a inteligência não existe em ambientes

fechados, como no caso dos agentes deliberativos. Por outro lado, nos agentes reativos, a inteligência existe e cresce apenas através da interação contínua com o sistema. Agentes reativos são simples e podem se comunicar de forma básica, contudo, padrões de comportamento complexos podem emergir da interação entre os diversos agentes e/ou entre este(s) e o ambiente.

Segundo Nwana e Azarmi (1997), a capacidade do agente de se adaptar ao ambiente (aprendizado), pré-requisito para a inteligência, é uma forma de aumentar o desempenho de sistemas baseados em agentes. Maes (1995) também afirma que, muitas vezes, o uso do termo “agentes inteligentes” não é adequado a uma aplicação, porque agentes verdadeiramente inteligentes ainda não existem. Nwana e Azarmi (1997) e Brenner et. al. (1998) concordam com o fato de que, para os agentes serem autônomos, eles devem ser capazes de aprender.

Sendo assim, pode-se concluir que, para que um programa possa ser classificado como agente, é essencial que ele seja autônomo. Para que seja autônomo, deve possuir, segundo alguns pesquisadores, a característica inteligência, especificamente a capacidade de aprender.

Nas próximas seções serão abordadas as partes que compõem a inteligência, isto é, base de conhecimento, raciocínio e aprendizado.

### 2.2.1
### Base de Conhecimento

Existem muitas formas de conhecimento e diferentes maneiras de se armazenar e representar o conhecimento. Os programadores usam a forma simbólica (números ou letras) chamada de representação interna, pois esta é facilmente processada por computadores. A forma mais confortável é a linguagem natural, que, no entanto, apresenta muitas ambiguidades. Por este motivo, linguagens formais matemáticas e lógicas foram desenvolvidas. Existem diferentes tipos de conhecimento e muitas maneiras diferentes de representá-los: fatos simples ou relacionamentos complexos, fórmulas matemáticas ou regras de sintaxe de linguagem natural, etc. A escolha da forma de representar o conhecimento está diretamente relacionada às necessidades da aplicação e do computador. A base de conhecimento é um repositório de informações, central ou

não, contendo fatos que são conhecidos sobre objetos e suas relações. O processo de mapear um conjunto de conhecimentos no domínio de um problema e convertê-lo em base de conhecimento é chamado de engenharia do conhecimento ou aquisição do conhecimento. Maiores detalhes podem ser encontrados em (Russel e Norvig, 2010).

### 2.2.2
### Raciocínio

Regras do tipo *if-then-else* tornaram-se uma forma muito popular para representar o conhecimento. Isto é motivado pelo fato de ser mais confortável ler regras na forma em que normalmente as concebemos do que na forma de lógica de predicados (linguagem formal). Cada regra é vista como uma unidade do conhecimento que pode ser alterada pela modificação ou criação de novas regras, sem gerar problemas para o sistema.

Normalmente, as regras são compostas por dois componentes: antecedentes e consequentes. Se todas as cláusulas dos antecedentes forem verdadeiras, a regra é disparada. Quando isto acontece, o consequente da regra é tomado como verdade e um novo fato pode ser adicionado à base de conhecimento. O raciocínio é classificado como monotônico quando novos fatos podem ser adicionados à *work memory*, ou como não-monotônico quando fatos podem ser cancelados (Bigus e Bigus, 1998; Brenner et. al, 1998).

### 2.2.3
### Aprendizado

Para alguns pesquisadores (Brenner et. al., 1998), a autonomia está diretamente relacionada à capacidade de se aprender um comportamento.

Embora muitos pesquisadores explorem ainda hoje as áreas relativas à base de conhecimento e raciocínio, o aprendizado é, no entanto, o que pode dotar os agentes de verdadeira capacidade de desenvolver sua autonomia, pois com o aprendizado o agente está sempre apto a se adaptar aos ambientes.

O processo de aprendizado pode ser classificado segundo a quantidade de esforço empregado, como pode ser visto abaixo (Weiss, 1999):

- Aprendizado simples: implantação do conhecimento e da capacidade de aprendizado de forma direta, sem necessitar de inferência ou transformação por parte do agente;
- Aprendizado a partir de instrução ou por conselho: operacionalização de novas informações que não são diretamente executadas pelo agente, isto é, são transformadas para uma representação interna e integradas ao conhecimento ou capacidade existente;
- Aprendizado a partir de exemplos e pela prática: extração e refinamento do conhecimento; unificação de padrões a partir de exemplos positivos e negativos ou a partir de exemplos práticos;
- Aprendizado por analogia: transformação do conhecimento a partir de um problema resolvido ou similar;
- Aprendizado através de descobertas: novos conhecimentos podem ser adquiridos através de observações e/ou experimentos, gerando e testando hipóteses ou teorias.

Outra forma de classificar o aprendizado é segundo a forma de realimentação (Weiss, 1999):

- Aprendizado supervisionado: a realimentação específica a atividade desejada pelo agente e o objetivo do aprendizado é tornar a ação executada e a ação desejada tão próximas quanto possível;
- Aprendizado não supervisionado: nenhuma realimentação explícita é fornecida e o objetivo é descobrir atividades úteis e desejadas através de tentativa-e-erro e processos auto-organizáveis;
- Aprendizado por reforço: a realimentação é um valor ou sinal que especifica a utilidade da atividade ou ação realizada pelo agente quando ele se encontra em um determinado estado. O objetivo do aprendizado é maximizar esta utilidade para todos os estados.

Agentes veem uma ação de realimentação como um mestre no caso de aprendizado supervisionado, como um crítico no caso de aprendizado por reforço ou como um observador passivo no caso de aprendizado não supervisionado.

A próxima seção é destinada ao detalhamento do funcionamento de métodos de aprendizado por reforço, que serve de base para os algoritmos utilizados neste trabalho.

## 2.3
## Aprendizado por Reforço (*Reinforcement Learning* - RL)

O aprendizado por reforço é geralmente usado com o objetivo de controlar processos. Quando aplicado a agentes autônomos, possui a finalidade de fazer com que os agentes aprendam um determinado comportamento. De uma maneira geral, o algoritmo de aprendizado por reforço recebe do ambiente um sinal referente à avaliação do desempenho atual (reforço); baseado neste reforço, o algoritmo modifica o controlador tentando evoluir ações que melhor determinem o comportamento esperado.

Este tipo de modelo de aprendizado tem inspiração biológica: quando se realiza algo corretamente ou benéfico, recebe-se uma recompensa, e quando se realiza algo incorretamente ou prejudicial, recebe-se uma punição. A tendência é repetir a ações que estão associadas a recompensas positivas e evitar ações associadas a recompensas negativas, sempre que as condições forem semelhantes.

*Reinforcement Learning* consiste, portanto, em determinar o comportamento de um agente através da interação com o seu ambiente. As ações executadas têm como consequência um sinal de reforço, porém não existe conhecimento prévio da ação correta para cada possível estado. Cabe ao agente descobrir quais ações levam a um melhor desempenho do sistema. O modelo de *Reinforcement Learning* padrão pode ser visto na Figura 1.

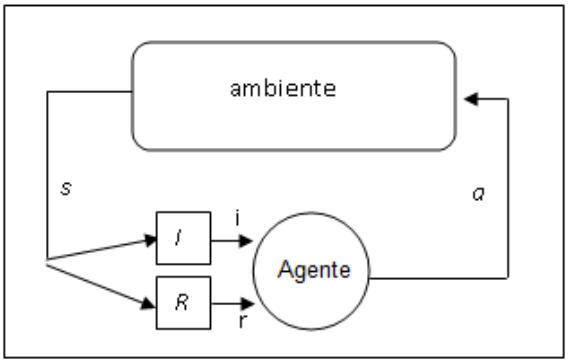


Figura 1: Esquema do modelo *Reinforcement Learning*.

A cada instante de tempo, o agente examina, através da sua percepção, (sensores) o estado *S(I).* A função *I* define a observabilidade do ambiente. Esta função será considerada como a função identidade, ou seja, os estados apresentam observabilidade total. O agente em um estado *s* deve escolher, baseado em suas observações, uma ação *a* a partir de um conjunto *A* discreto de ações disponíveis, isto é $a \in A$. Após a execução desta ação, o ambiente passa a um estado *s´* com probabilidade que depende da ação *a*, e um sinal de reforço *r* é recebido do ambiente. Este sinal, denominado de reforço primário, depende do estado anterior *s* e da ação *a*. O agente deve determinar qual ação executar quando está em um estado específico. A função de reforço qualifica uma tarefa realizada pelo agente através de recompensas e punições.

Para sistemas que não recebem sinais de reforços primários a cada iteração, premiar ou punir ações passadas que tenham contribuído para o sucesso ou falha do sistema torna-se um problema (*delayed reinforcement)*. Neste tipo de aplicação o sinal de reforço não é conhecido até que o agente atinja o objetivo ou chegue ao final da tarefa. Algumas vezes pode ser difícil chegar ou identificar o final da tarefa, além de consumir grande quantidade de memória. Este problema é denominado de associação temporal de crédito (*temporal credit-assignment)* (Sutton & Barto, 1998; Kaelbling et. al., 1996; Jouffe, 1998). Para contornar este problema, o modelo matemático usado é o Processo de Decisão de Markov (*Markovian Decision Problems* – MDP).

Dado um completo e preciso modelo MDP, a Programação Dinâmica oferece uma ferramenta eficiente para o aprendizado *off-line* do comportamento do agente. Programação Dinâmica (Bellman, 1957), para ser computacionalmente

eficiente, exige uma modelagem precisa do ambiente, como um Processo de Markov.

Como as necessidades de Programação Dinâmica muitas vezes são inviáveis, como, por exemplo, a necessidade do conhecimento do modelo do ambiente, a função de Programação Dinâmica tem sido a de servir de inspiração para outros modelos que visam a adaptar estas necessidades.

O método de Monte Carlo possibilita o aprendizado a partir da experiência obtida através de sequências de estados, ações e reforços recebidos pela interação agente-ambiente. Esse processo de aprendizagem a partir de experiência *on-line* é interessante, pois não requer conhecimento a priori da dinâmica do ambiente, mesmo assim consegue encontrar uma política ótima (Sutton e Barto, 1998).

Dessa forma, os métodos de Monte Carlo (Rubinstein, 1981) não precisam da modelagem do ambiente e se apresentam de forma simples em termos conceituais. Não são viáveis quando a solução do problema é possível apenas de forma incremental (utilizando somente o estado atual e os estados imediatos), pois para se atualizar a função de valor os métodos de Monte Carlo exigem que o estado final seja alcançado no processo.

Os métodos de Diferença Temporal são uma combinação de características dos métodos de Monte Carlo com as ideias da Programação Dinâmica, no sentido de que buscam estimar valores de utilidade para cada estado do ambiente através dos ganhos de transições e de valores de estados sucessivos. Nos métodos de Diferença Temporal o ambiente não precisa ser modelado completamente (é vantagem sobre Programação Dinâmica), não é necessário chegar ao estado final (é vantagem sobre o Monte Carlo). Os dois métodos mais representativos na literatura são *Q-Learning* (Watkins, 1992) e SARSA (Rummery e Niranjan, 1994; Sutton, 1998).

Vale ressaltar que a principal ideia de *Reinforcement Learning* é o uso de função de valores para a descoberta de boas políticas. Maiores detalhes podem ser encontrados no em Kaelbling et. al. (1996), Sutton & Barto (1998) e Bellman (1957).

## 2.4
## Sistemas Multiagentes (SMA)

### 2.4.1
### Introdução

Os Sistemas Multiagentes formam uma área emergente da Inteligência Artificial que fornece os princípios para construção de sistemas complexos envolvendo múltiplos agentes e os mecanismos de coordenação entre eles (Stone, Veloso, 2000). Um SMA pode ser definido como um grupo de agentes autônomos, interagindo entre si e compartilhando um mesmo ambiente, que é percebido através de sensores, onde eles agem realizando ações (Vlassis, 2003). Este tipo de sistema tem uma grande variedade de aplicações, como controle distribuído (Riedmiller et al., 2000; Stephan et al., 2000; Wiering, 2000; Bakker et al., 2005), times robóticos (Neri 2012; Luke et al., 1997; Luke, 1998; Fernandez e Parker, 2001; Stone e Veloso, 2000; Hladek, 2007), controle de navegação (Hu e Wellman, 2003; Calvo e Romero, 2007), controle e planejamento de rotas (Boyan e Littman, 1993; Choi e Yeung, 1995; Tillotson et al., 2004), programação de elevadores (Crites e Barto, 1998), balanceamento de carga (Schaerf et al., 1995), negociação (*trading*) automática entre agentes (Wellman et al., 2003; Hsu e Soo, 2001; Lee e Oo, 2002; Oo et al., 2002; Tesauro e Kephart, 2002; Raju et al., 2003), precificação dinâmica de produtos de varejo (Raju et al. 2003), etc.

Os benefícios trazidos pela aplicação de SMA são diversos. Em primeiro lugar, através da computação paralela, vários agentes podem trabalhar em conjunto para melhor explorar a estrutura descentralizada de uma determinada tarefa e acelerar sua conclusão (Lagoudakis, 2002; Kok et al., 2005; Fitch et al., 2005). Além disso, agentes podem trocar experiências se comunicando (Tan, 1993), podem apenas observar os mais habilidosos e aprender (Price e Boutilier, 2003), os mais inteligentes podem servir de professores para outros agentes (Clouse, 1995), etc. Os SMA também podem fornecer alto grau de escalabilidade, através da inclusão de novos agentes quando necessário, e ainda fazer com que agentes assumam as atividades de outros agentes em casos de falha (Busoniu et al., 2008). Gerhard e Sen (1996) também destacam a utilização de SMA no estudo

e elucidação da inteligência. Como a inteligência está profundamente acoplada à interação, a melhor forma de criar máquinas inteligentes pode ser através da construção de “redes sociais” de máquinas.

Shoham et al. (2003) apresentam quatro grandes subáreas de pesquisa dentro do aprendizado de sistemas Multiagentes: uma descritiva e 3 prescritivas. A descritiva, inspirada na Psicologia, explica como humanos aprendem no contexto de outros humanos. As três linhas de pesquisa prescritivas trabalham como os agentes devem aprender. A primeira delas, Inteligência Artificial Distribuída (“*Distributed AI*”), estuda problemas de controle distribuído. Geralmente, um agente central controla múltiplos agentes, mas não define a “política ótima” para eles individualmente. Define apenas um procedimento adaptativo que converge para uma “política ótima”. A segunda (“*Equilibrium Agenda*”) analisa o equilíbrio entre as estratégias de diferentes agentes. Na Teoria dos Jogos não-cooperativos, a noção de “estratégia ótima” perde o sentido, sendo substituída pelo conceito de “melhor resposta” individual, que gera um equilíbrio e faz o sistema convergir. A terceira e última (“*AI Agenda*”) estuda a escolha da melhor estratégia para um agente, considerando uma classe fixa de agentes no jogo, ou a definição de um agente ótimo para um determinado ambiente. Alguns não consideram esse tipo de aplicação como um problema verdadeiramente Multiagente, pois não utiliza totalmente alguns de seus conceitos básicos, como a coordenação entre os agentes.

Já Stone e Veloso (2000) colocam os SMA como um sub-campo da Inteligência Artificial Distribuída (DAI), que por sua vez, nasce da interseção de Computação Distribuída e Inteligência Artificial. Na Computação Distribuída, diversos processadores compartilham dados (mas não o controle) com foco em uma paralelização de baixo-nível ou em aspectos de sincronização. Na DAI, além dos dados, o controle também é distribuído, com foco na solução de problemas, coordenação e comunicação. Ainda, a DAI consiste em duas grandes linhas de pesquisa, havendo uma interseção entre elas. Na primeira, *Solução de Problemas Distribuídos* (SPD), a ênfase é na decomposição, compartilhamento e combinação de tarefas e dos resultados. Na segunda, os *Sistemas Multiagentes* (SMA), o foco é na coordenação do comportamento dos agentes. De uma forma resumida, enquanto SPD lida com o gerenciamento da informação, os SMA estudam como

gerenciar o comportamento. A interseção ocorre em problemas envolvendo múltiplos agentes que gerenciam informações.

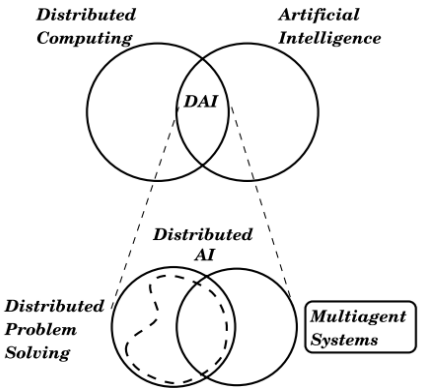


Figura 2: Taxonomia das áreas de pesquisa dentro da Inteligência Artificial Distribuída (DAI), de acordo com Stone e Veloso (2000).



Embora o comportamento dos agentes em um ambiente Multiagente possa ser pré-programado, na grande maioria das situações é necessário algum tipo de aprendizado on-line, que faça com que o desempenho dos agentes melhore de maneira gradativa (Weiss e Sen, 1999; Stone e Veloso, 2000), a taxonomia do modelo é mostrado na Figura 2.

O aprendizado por reforço pode ser usado para aprender um determinado processo de controle, que pode ser aplicado a agentes com o objetivo de obter o comportamento desejado. Diversos algoritmos de aprendizado por reforço, que apresentam propriedades consistentes, já vêm sendo usados em aplicações que funcionam através de um único agente. Porém, existem novos desafios na utilização de algoritmos de aprendizado por reforço em problemas englobando múltiplos agentes. Entre as principais dificuldades estão: a definição de um objetivo para vários agentes, a coordenação entre os comportamentos dos agentes e a escalabilidade dos algoritmos, que já é inclusive um problema em aplicações com apenas um agente.

Esta seção teve como objetivo fornecer uma revisão dos pontos mais importantes na área de SMA, com ênfase no funcionamento de técnicas e algoritmos baseados em aprendizado por reforço. As próximas seções estão organizadas da seguinte forma: a seção 2.4.2 explora os principais conceitos e taxonomias ligados aos SMA; a seção 2.4.3 explica o funcionamento de alguns

dos algoritmos de *Reinforcement Learning* para múltiplos agentes mais usados e seus desafios; e, finalmente, na seção 2.4.4, aplicações com múltiplos agentes são descritas.

### 2.4.2
### Conceitos e Taxonomia

Existem diversos fatores que determinam a categoria de uma aplicação como SMA. Eles podem variar de acordo com o tipo da tarefa, ambiente, grau de homogeneidade do aprendizado, conhecimento a priori dos agentes, política de divisão do reforço, etc. O ambiente de uma aplicação SMA pode ser estático, quando o ambiente permanece inalterado, ou dinâmico, onde mudanças no estado podem ocorrer ao longo do processo de interação entre os agentes e dos agentes com o estado.

Quanto ao tipo da tarefa envolvida no problema, existem três categorias principais (Busoniu, 2008). A primeira é a das aplicações *totalmente competitivas*, que sempre têm um vencedor e um perdedor no final. Neste tipo de problema, um dos princípios que pode ser aplicado é o do *minimax* (Von Neumann, 1959). O *minimax* é um método da teoria da decisão tradicional para "minimizar a máxima perda possível". Também pode ser visto como a estratégia de punir o outro jogador, minimizando o máximo que ele pode obter. Littman (1994) criou o algoritmo *minimax-Q*, que aplica o princípio do *minimax* para calcular as estratégias a cada estágio do aprendizado, e propaga os valores aos pares de estado-ação usando o algoritmo de *Reinforcement Learning* denominado *Q-learning (Watkins, 1989).*

A segunda categoria é composta pelas aplicações *totalmente cooperativas*. Neste caso, o objetivo é maximizar o retorno comum dos múltiplos agentes. Neste tipo de aplicação, os retornos são positivamente correlatos. Ou seja, quando um agente recebe reforço positivo, os demais também recebem.

Finalmente, a terceira categoria é composta de problemas com tarefas mistas, onde nenhuma restrição é imposta aos retornos dos agentes. Este grupo ainda pode ser subdividido em algoritmos que não implementam a coordenação entre os agentes (*coordination-free*), como o *Team Q-learning* (Littman, 2001) e o

*Distributed Q-learning* (Lauer e Riedmiller, 2000); os que são baseados em um mecanismo de coordenação; e os que possuem uma forma de coordenação indireta.

Um dos pontos mais importantes dos SMA é a homogeneidade do aprendizado. O aprendizado dos múltiplos agentes pode ocorrer de três formas (Panait e Luke, 2005): homogêneo, com todos aprendendo da mesma forma; heterogêneo, onde cada agente aprende de uma forma distinta, aumentando o espaço de busca (Good, 2000); e híbrido, onde cada grupo de agentes aprende de uma forma homogênea, por exemplo, um time de futebol de robôs (Luke et al., 1997 ; Luke, 1998), sendo cada grupo de jogadores com uma especialização. Existem também diferenças em relação às observações de um agente em relação aos demais: um agente pode precisar observar as ações de outros agentes; pode precisar observar as ações e recompensas de outros agentes; ou não precisar observar nem as ações nem as recompensas de outros agentes. Mesmo não havendo nenhum tipo de comunicação ou compartilhamento entre os agentes, é possível haver coordenação entre eles (Weiss e Sen, 1996).

Após a avaliação das necessidades de observação, outro fator de extrema relevância a ser definido é o grau de comunicação entre os agentes. De uma maneira geral, existem duas diferentes formas de comunicação entre os agentes: direta e indireta. Sendo realizada indiretamente, um agente central pode ter acesso a tudo e repassar aos demais; as informações podem ser colocadas em um "*blackboard*" que todos acessem; ou pode haver uma transferência implícita de informação, por exemplo, um agente deixando "pegadas" ou sinais que outro agente pode usar no aprendizado. As próprias ações realizadas pelos agentes também podem ser utilizadas para uma comunicação implícita de informações e a coordenação entre os agentes, como proposto por Tummolini et al. (2005) no modelo BIC (*Behavioural Implicit Communication*). Ocorrendo comunicação direta, os agentes trocam mensagens entre si, não havendo a necessidade de troca de informações implícitas ou através de "*blackboards*". Combinando o grau de comunicação e a homogeneidade do aprendizado, Stone e Veloso (2000) sugerem uma definição dos principais grupos e o nível de complexidade de um SMA. Conforme mostra a Figura 3, quanto maior a heterogeneidade (ou maior especialização) e o grau de comunicação, maior é a complexidade do sistema.

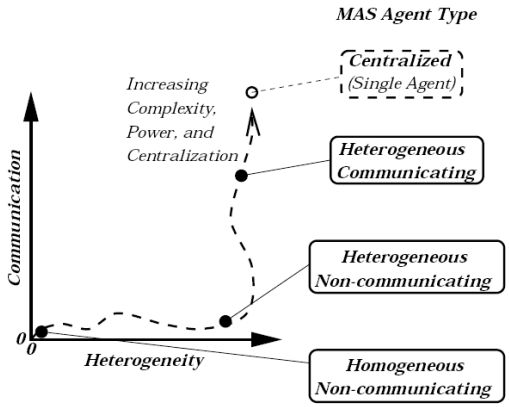


Figura 3: Taxonomia das áreas de pesquisa dentro da Inteligência Artificial Distribuída (DAI), de acordo com Stone e Veloso (2000).

Em relação ao nível de homogeneidade do aprendizado, o aumento da complexidade com a maior especialização ocorre porque cada agente ou grupo de agentes aprenderá utilizando uma dinâmica diferente. Já, na parte de comunicação, quanto maior a quantidade de informações que o agente recebe sobre os demais agentes, maior é o seu espaço de busca. Para evitar o crescimento exponencial do estado de busca, a comunicação pode ser simplificada, por exemplo, através da criação de níveis de percepção sobre as ações e localizações dos demais agentes (Touzet, 2000), de aproximações de funções (Guestrin, 2002), etc.

Havendo a comunicação total das informações, um SMA pode chegar a ser equivalente a um sistema de grande complexidade, onde um agente único, ao invés de vários, toma todas as decisões e controla os demais como se fossem "escravos". Com isso, não há nenhum tipo de distribuição ou decomposição de tarefas, visando simplificar o trabalho dos agentes.

Para cada categoria, existem diferentes definições e técnicas de aprendizado que devem ser escolhidas na construção de um sistema com múltiplos agentes. Nos parágrafos seguintes, alguns aspectos gerais de sistemas Multiagentes serão destacados. Em seguida, as características de cada tipo de sistema, de acordo com o grau de comunicação e homogeneidade do aprendizado, serão analisadas separadamente.

### 2.4.2.1
### Aspectos Gerais a todos os Sistemas Multiagentes

- Benevolência/Cooperação x Competitividade: um dos pontos básicos de definição do comportamento dos agentes é se eles irão ajudar um ao outro para chegar ao objetivo (benevolentes ou cooperativos) ou se irão competir entre si, considerando apenas seus próprios objetivos (competitivos). No extremo, podem ocorrer situações de "soma-zero", onde os agentes deverão se opor às estratégias dos demais para conquistar seus objetivos.

- Estabilidade x Evolução: também é necessário definir se os agentes são estáveis ou se evoluem suas estratégias. A evolução deve ocorrer apenas na presença de ambientes dinâmicos. Em aplicações onde há a competição com evolução, podem ocorrer uma corrida sem fim pela melhor estratégia, nunca permitindo a estabilização em um bom comportamento, ou ainda o problema do "*credit/blame assignment*", onde não é possível afirmar se o agente melhorou por causa de seu comportamento ou por causa do comportamento negativo de outro agente.

- Conhecimento a priori dos agentes: os agentes podem possuir ou não algum conhecimento a priori do modelo das tarefas que devem ser executadas. Considerando esse aspecto, Busoniu et al. (2008) classifica os algoritmos em *model-based learning*, quando o modelo de tarefa está disponível, e *model-free learning*, quando não há a disponibilização de modelo de tarefa a priori para os agentes.

- Divisão do Reforço (*Credit Assignment* ou *Reinforcement Sharing*): Em determinados problemas, geralmente de natureza cooperativa, o ambiente da aplicação não é suficientemente inteligente para definir individualmente o retorno de cada agente que faz parte de um mesmo time. Logo, se faz necessário a utilização de algum mecanismo de divisão do reforço (*Credit Assignment* ou *Reinforcement Sharing*). Entre as principais formas de divisão de reforço podem ser citados:

- *Global reinforcement* – reforço do "time" é dividido igualmente, devido às ações tomadas pelos agentes em conjunto, como se fosse uma inteligência coletiva (Wolpert e Tumer, 1999 e 2001).

- *Local reinforcement* – atualização de reforço feita somente com base no desempenho do próprio agente, com o objetivo de "desencorajar" os preguiçosos. Porém, isso faz com que os agentes não sejam incentivados a ajudar os demais.

– *Local reinforcement + social reinforcement*: reforço local pode ser melhorado com alguns tipos de "*reforço social*" (Mataric, 1994). Por exemplo:

→ *observational reinforcement* - pode ser obtido observando e imitando outros agentes;

→ *vicarious reinforcement* – recompensa indireta é obtida por um agente quando os outros são recompensados diretamente ("experiência de vida *indireta"*).

Harati et al. (2007) propõem o método KEBCA (*Knowledge evaluation-based credit assignment*), onde se utiliza a própria história e conhecimento do agente, através de medidas de "certeza", para a atualização do reforço. Essas medidas têm como objetivo avaliar e julgar as ações de cada agente para assinalar o crédito a elas da melhor maneira possível.

Abul et al. (2000) sugere uma simples combinação entre o retorno global e o retorno do agente em ambientes colaborativos. Na prática, cada agente recebe um retorno proporcional a sua contribuição para o retorno global.

$$r_{coop} = \alpha\, r + (1 - \alpha) \frac{\sum r_{\alpha}}{|\alpha|} \qquad (1)$$

Na equação 1, $r$ é o retorno do agente, $r_a$ é o retorno dos agentes observados e $\alpha$ ($0 < \alpha < 1$) é responsável pelo ajuste da relevância dos retornos.

Em Miyazaki e Kobayashi (2001), é proposto o modelo *Rationality Theorem of Profit Sharing*. Neste, quando um agente recebe uma recompensa direta, os demais agentes recebem uma recompensa indireta. Em Zhou e Hong

(2006), a preocupação maior é com a racionalização do uso de memória. Eles sugerem um modelo de *On-line profit sharing algorithm*, que consiste na atualização baseada no conceito de *eligibility trace* (Sutton e Barto, 1998), com o objetivo de consumir menos memória.

#### 2.4.2.2
#### Características dos Sistemas Homogêneos

##### 2.4.2.2.1
##### Sistemas Homogêneos com Comunicação (com "Agente Central")

As aplicações onde os agentes possuem as mesmas características (homogêneos) e podem se comunicar diretamente e sem restrições muitas vezes são consideradas sistemas de um agente único. Neste tipo de sistema, um agente central armazena todas as informações e coordena as ações dos demais agentes, que não possuem nenhuma inteligência. Embora possa parecer que sistemas com apenas um agente inteligente sejam mais fáceis de compreender, a distribuição do controle permite a criação de agentes simples, o que pode ajudar na solução de problemas complexos.
Existem dois aspectos relevantes a serem considerados neste tipo de SMA:

- Nível de paralelismo desejado: um dos principais objetivos a serem alcançados neste tipo de SMA é a exploração paralela do ambiente. O nível de exploração em paralelo é definido pela quantidade de agentes no ambiente. Quanto mais agentes, maior será o volume de tarefas executadas em paralelo.

- Grau de Controle do Agente Central: o grau de controle do agente central sobre os demais agentes também pode variar. Na maioria das vezes, neste tipo de SMA a relação dos agentes com o agente central é do tipo mestre-escravo. Ou seja, o agente central coordena todas as ações e informações dos demais agentes. Em algumas situações, apesar do controle do agente central, os demais agentes podem possuir algum tipo de inteligência própria.

#### 2.4.2.2.2
#### Sistemas Homogêneos sem Comunicação

Diversos fatores precisam ser estudados e definidos em Sistemas Multiagente homogêneos sem comunicação. Entre os principais, estão:

- Agentes reativos x deliberativos: é necessário definir o grau de "raciocínio" dos agentes. Quando reativos, os agentes se limitam apenas a recuperar comportamentos já existentes de acordo com seus reflexos. Agentes deste tipo são aqueles que precisam fugir de obstáculos e, para isso, convertem os dados obtidos através de seu sensor (posições dos obstáculos e de outros agentes) em vetores de movimento (Neri, 2012). Já, se deliberativos, os agentes assumem que cada um escolherá suas ações para concluir seus objetivos, podendo usar técnicas oriundas da teoria dos jogos para encontrar pontos de equilíbrio e decidir como agir (Levy e Rosenschein, 1992). Existe também a possibilidade de combinar "pensamentos" reativos e deliberativos.

- Perspectiva Local x Global: é importante definir o quanto de informação os agentes percebem do ambiente através de seus sensores. Em algumas situações, caso todos os agentes tenham uma perspectiva global e usem o mecanismo de aprendizagem, isso pode levá-los a escolher sempre as mesmas ações. Eventualmente, com menos informação, melhores resultados podem ser obtidos ("*A Ignorância é uma Benção*") (Durfee, 1995).

- Modelagem dos estados dos outros agentes: muitas vezes é necessário modelar o estado interno de outros agentes com objetivo de prever suas ações. Mesmo que os agentes conheçam os objetivos e estruturas dos demais, eles podem não conhecer suas futuras ações. A informação faltante é o estado interno (se deliberativo) ou os sensores de entrada dos outros agentes.

- Como afetar os outros agentes: Sem a possibilidade de comunicação, os agentes podem afetar os demais de forma direta ou indireta. Diretamente, eles podem ser percebidos pelos sensores alheios ou alterar o estado dos outros

agentes. Indiretamente, eles podem afetar o resto do grupo através de dois tipos de *Estigmergia* (Holland, 1996). Na Estigmergia ativa, o agente altera o ambiente de forma que as mudanças sejam captadas pelos outros sensores. Na passiva, a alteração no ambiente ocorre com a finalidade de mudar o efeito da ação do outro agente. Por exemplo, se um agente desliga um quadro central de energia de um prédio, o efeito da ação de outro agente ligar a luz de um quarto no prédio é alterado indiretamente, uma vez que a luz não acenderá, já que o quadro de luz, que faz parte do ambiente, está desligado.

#### 2.4.2.3
#### Características dos Sistemas Heterogêneos

De uma maneira geral, nos sistemas de natureza heterogênea, onde agentes podem possuir diferentes mecanismos de aprendizado, é importante a análise dos papéis e funções que os agentes deverão assumir, e de aspectos ligados à gestão de recursos.

- Papéis e Funções: quando os agentes possuem diferentes habilidades e mesmos objetivos, eles podem e devem ser organizados como um time, onde cada um desempenha suas funções específicas. Isso vale para sistemas com ou sem comunicação entre os agentes. Em ambientes dinâmicos, pode ainda ser necessário que cada agente tenha um determinado papel dependendo da situação em que se encontra. Uma das subáreas de pesquisa dentro de SMA é justamente como um agente pode assumir novas funções em ambientes em constante evolução.

- Gerenciamento de recursos: em um grande número de aplicações, agentes possuem recursos limitados para serem compartilhados. Por exemplo, em um problema de balanceamento de carga, vários computadores ou usuários possuem uma quantidade limitada de processamento que pode ser compartilhada (Schaerf et al., 1995). Havendo comunicação entre os agentes, também são possíveis a coordenação do tempo das atividades e o controle de cronogramas (Decker e Lesser, 1995).

#### 2.4.2.3.1
#### Sistemas Heterogêneos sem Comunicação

Não havendo a comunicação entre agentes em sistemas heterogêneos, algumas características precisam ser definidas durante a construção do SMA, como, por exemplo:

- Modelagem dos objetivos, ações e conhecimento dos outros agentes: como os agentes não são homogêneos, não basta modelar seus estados internos para prever suas ações. Com agentes heterogêneos, a previsão das ações só pode ser feita modelando os objetivos, ações e conhecimento dos demais, tornando o problema extremamente mais complexo e caro. A alternativa mais viável é tentar deduzir as ações dos agentes observando seus movimentos (Huber e Durfee, 1995; Wang, 1996).

- "Convenções sociais": mesmo não havendo comunicação entre os agentes, existem formas de se chegar a "acordos" ou escolhas coincidentes. O método do Ponto Focal (Fenster et al., 1995) explora o fenômeno cultural de preferências fazendo com que os agentes "se encontrem" sem nenhuma comunicação. Um exemplo prático foi dado por R. Vohra no "AAAI95 *Fall Symposium on Active Learning*". Ao perguntar para 40 pessoas onde elas se encontrariam se estivessem em Paris a procura de um amigo, 75% dos entrevistados disseram que iriam para a Torre Eiffel ao meio-dia.

#### 2.4.2.3.2
#### Sistemas Heterogêneos com Comunicação

Por sua vez, nos sistemas heterogêneos com comunicação entre os agentes, precisam ser analisados os seguintes fatores:

- Padrões de Comunicação: obviamente, existe a necessidade de definição do protocolo que será usada na comunicação entre os agentes. Esses padrões para troca de informações podem ser criados especificamente para uma aplicação ou

ser pré-definidos. Um determinado protocolo deve contemplar o conteúdo da informação (Genesereth e Fikes, 1992), formato das mensagens (Finin et al., 1994) e mecanismos de coordenação (Reis, 2007).

- Planejamento das ações de comunicação: é possível considerar que uma comunicação seja feita da mesma forma que uma ação qualquer. Ou seja, cada transmissão de informação pode possuir pré-condições e efeitos como uma ação teria. Em um SMA, o efeito da comunicação pode ter como objetivo a alteração da crença de um agente em relação ao estado de outro(s) agente(s). Além disso, Stone e Veloso (2000) também destacam que um agente pode fazer intencionalmente uma comunicação errada a outro agente, com o objetivo de atingir um objetivo. Ou seja, um agente pode simplesmente estar "blefando". Nestes casos, também é importante que o receptor da informação defina quando irá ou não assumir que uma determinada informação é verdadeira.

- Acordos e Compromissos: havendo comunicação entre os agentes, eles podem realizar acordos de cooperação com o objetivo de chegar a objetivos comuns. Os acordos e compromissos podem ser de vários tipos. Castelfranchi (1995) sugere 3 tipos: compromissos internos (o próprio agente se compromete a fazer algo), compromisso com outro agente e compromissos coletivos, onde o agente se dispõe a desempenhar uma determinada função. Agentes também podem usar os meios para se chegar a certos fins, de forma a planejar oportunidades de compromissos. Isso é contemplado no modelo BDI ("*Belief/Desire/Intention*") (Haddadi, 1995), que trata os possíveis "estados de comprometimento" como estados de planejamento, podendo ser uma cooperação potencial, um pré-compromisso ou um compromisso. Outro possível tipo de acordo é entre grupos de agentes, ao invés de acordos gerais ou entre dois agentes. Nestes casos, os agentes agem como se estivessem formando diversas coalizões (Zlotkin e Rosenschein, 1994).

A seção seguinte aborda os principais desafios e as famílias de algoritmos dentro da área de SMA, que funcionam a partir de técnicas de aprendizado por reforço.

### 2.4.3
### Sistemas Multiagentes baseados em *Reinforcement Learning*

#### 2.4.3.1
#### Principais Desafios

A aplicação de métodos de *Reinforcement Learning* tradicionais em "grandes ambientes", utilizando *lookup table* (Tabela de armazenamento das funções de valor para espaço de estados pequeno ou discreto) é muitas vezes inviável, devido à grande dimensão do espaço de estados. Conforme já descrito na seção 1.1, esta limitação é conhecida como "*curse of dimensionality*". Em SMA, o problema do armazenamento dos valores para cada possível estado-ação é ainda maior, já que a memória necessária passa a crescer também exponencialmente com a quantidade de agentes envolvidos na aplicação (devem ser representados os valores dos pares de estado-ação de todos os agentes em conjunto). A estabilidade na dinâmica de aprendizado do agente e a coordenação entre os agentes, enquanto ocorre a adaptação nas mudanças de outros agentes, também é um grande desafio. Ao longo da evolução do comportamento dos demais agentes, alguns valores de reforço de determinados pares de estados e ações podem ser alterados, gerando sinais confusos para o aprendizado do agente. Na prática, em um SMA, cada agente tenta aprender um objetivo "móvel", já que sua melhor política é alterada, conforme a política dos demais agentes muda. Isso pode fazer com que o aprendizado se torne não estacionário.

#### 2.4.3.2
#### Técnicas de RL usadas em SMA

As seções seguintes exploram os métodos de Diferença Temporal e o Gradiente Decrescente, que deram origem a diversos algoritmos de aprendizado para Multiagentes. Em seguida, são relacionados os principais algoritmos SMA já desenvolvidos, suas características e suas origens.

#### 2.4.3.2.1
#### Diferença Temporal

Um grande número de algoritmos de RL para Multiagentes deriva de métodos de Diferença Temporal (TD) (Sutton e Barto, 1998), sendo a grande maioria do algoritmo *Q-learning* (Busoniu, 2008). O *Q-learning* foi o primeiro método RL a possuir provas de convergência (Jaakkola et. al., 1994). É uma técnica muito simples que calcula diretamente as ações sem avaliações intermediárias e sem a necessidade de definições prévias. Nele, cada par estado-ação <*s, a*> é relacionado a um valor-Q, *Q(s, a),* que é o valor estimado para o estado *s* quando a ação *a* é executada, assumindo que a ação executada será uma ação ótima para todos os estados subsequentes.

A função de valor-Q pode ser estimada on-line sem necessidade de aprender a política. O *Q-learning* é um método iterativo para aprendizado de ações. Ele é baseado no valor da ação (ou Q) e é definido por:

$$Q(s_t, a_t) = E[r(s_t, a_t) + \gamma V^*(s_{t+1})] \tag{2}$$

$$= r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} | s_t, a) V^*(s_{t+1}) \tag{3}$$

Na equação acima, *r* é o retorno instantâneo recebido do ambiente, $V^*(s_{t+1})$ é o valor estimado para a função de valor do próximo estado, e γ é o fator de desconto que define a influência do valor associado ao estado $s_{t+1}$ no valor atual estimado.

O valor Q da ação representa o valor esperado ao se executar a ação $a_t$ quando o agente está no estado $s_t$ (Ribeiro, 1999). A partir desta definição e como uma consequência do princípio de optimalidade de Bellman (1957), tem-se:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t [r(s_t, a_t) + \gamma \hat{V}(s_{t+1}) - Q(s_t, a_t)] \tag{4}$$

$\alpha$ é a taxa de aprendizado variando entre 0 e 1. A forma mais simples e intuitiva de transformar o algoritmo *Q-learning* em uma versão Multiagentes é evoluindo a equação acima para considerar *n* agentes, que conhecem as ações dos

demais agentes, agindo ao mesmo tempo com as ações $a_1$, ... $a_n$. Considerando isso, a equação de atualização dos valores de Q para o i-ésimo agente passa a ser definida por:

$$Q^i(s_t, a_t^1, ..., a_t^n) = Q^i(s_t, a_t^1, ..., a_t^n + \alpha_t[r^i(s_t, a_t^1, ..., a_t^n) + \gamma \hat{V}(s_{t+1}) - Q^i(s_t, a_t^1, ..., a_t^n)] \quad (5)$$

Um problema que surge, neste caso, é a forma como *V* deve ser atualizado, dada a complexidade de um SMA (Claus e Boutilier, 1998; Yang e Gu, 2004). Para solucionar isso, deve ser escolhido um algoritmo que busque equilíbrio e convergência do sistema, entre eles, Nash-Q (Hu e Wellman, 1998), Minimax-Q (Littman, 1994), Hyper-Q (Tesauro, 2003), Team-Q (Littman, 2001), Distributed-Q (Lauer e Riedmiller, 2000), CE-Q (Greenwald e Hall, 2003), Asymetric-Q (Kononen, 2003), etc.

Por exemplo, supondo um jogo competitivo de soma-zero com dois agentes A e B, executando passos alternados, o objetivo de A é aprender uma política que maximize o valor esperado dos reforços. Conforme dito no parágrafo anterior, esse aprendizado é difícil, pois depende das ações do agente B. No caso do algoritmo Minimax-Q (Littman, 1994), a solução proposta é avaliar cada política em relação à ação do outro jogador que a torna pior. Para viabilizar isso, a atualização do valor ótimo para o jogador A ocorre de acordo com as equações abaixo:

$$V^*(s) = \max_\pi \min_b Q(s, a, b) \quad (6)$$

$$\Delta Q_t(s_t, a_t, b_t) = \alpha[r_t + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t, b_t)] \quad (7)$$

$$\Delta Q_t(s_t, a_t, b_t) = Q_{t+1}(s_t, a_t, b_t) - Q_t(s_t, a_t, b_t) \quad (8)$$

Nas equações, *r* é a recompensa resultante do jogador A tomar a ação *a* e B tomar a ação *b* no estado *s*, e $\gamma$ é o fator de desconto.

Uma das estratégias mais usadas na maioria dos algoritmos para convergência é o Equilíbrio de Nash (Nash, 1950). Estratégias que fazem parte desse equilíbrio são simultaneamente as melhores respostas para todos os jogadores. Logo, não há incentivo para nenhum jogador desviar do equilíbrio,

unilateralmente. Para saber se é equilíbrio de Nash, basta perguntar a cada jogador separadamente: mudando a sua estratégia você ficaria melhor? Quando todas as respostas de todos os jogadores forem negativas, então é um Equilíbrio de Nash. O grande desafio dos algoritmos é encontrar estratégias que estejam em Equilíbrio ao longo do jogo.

Um dos algoritmos que usa o conceito do Equilíbrio de Nash para a convergência é o Nash-Q (Hu e Wellman, 1998). Em seu trabalho, Hu e Wellman (1998) utilizam programação quadrática com este objetivo. Alguns pesquisadores questionam a usabilidade do Equilíbrio de Nash na convergência do aprendizado de múltiplos agentes. Por exemplo, Shoham et al. (2003) destacam a falta de clareza na ligação entre o estágio de conhecimento da convergência para equilíbrio de Nash e o desempenho na dinâmica do jogo. Já Bowling e Veloso (2001 e 2002) apresentam outra forma de convergência entres os agentes. Nela, um agente converge para a melhor resposta enquanto os outros agentes permanecem estacionários. Com isso, a convergência para Equilíbrio de Nash não é explícita. Porém, ela surge naturalmente se todos os agentes são 'racionais'.

Em Bowling (2004), é proposto o conceito de "no-regret" (sem arrependimento), que é uma forma de convergência alternativa à racionalidade. Usando este conceito, o agente que está aprendendo deve buscar retorno pelo menos tão bom quanto o retorno de qualquer estratégia estacionária. Assim, ele previne ser explorado por outros agentes.

Outra alternativa na busca pela convergência é o uso da predição e da racionalidade relacionadas à estabilidade e adaptação, respectivamente (Chalkiadakis, 2003). Predição pode ser considerada como a capacidade de aprender com certa precisão os modelos dos demais agentes. E um agente é considerado racional quando ele busca sempre maximizar seus retornos esperados, dado os modelos dos demais agentes. Ou seja, aprendendo o modelo usado pelos demais agentes, o agente ganha estabilidade. E sendo 'racional' ou buscando maximizar seus retornos, o agente se adapta ao ambiente.

#### 2.4.3.2.2
#### Busca Direta – Métodos de Gradiente

Métodos que funcionam através da atualização de gradientes também vêm sendo usados em SMA. Um dos principais algoritmos, que serviu como base para o desenvolvimento de vários outros, é o IGA - *Infinitesimal Gradient Ascent* (Singh et al. 2000). No IGA, os agentes adaptam suas estratégias através da atualização de um gradiente ascendente, com uma taxa de aprendizado $\eta$ decrescente. O grande mérito do trabalho de Singh et al. (2000) foi mostrar que, se 2 jogadores jogam com o algoritmo IGA, com $\eta \rightarrow 0$, as estratégias convergem para o Equilíbrio de Nash ou a média dos *payoffs* no tempo converge para os *payoffs* esperados de um Equilíbrio de Nash.

Um dos vários algoritmos derivados do IGA é WoLF-IGA – *Win or Learn Fast IGA* . Este algoritmo introduz o conceito de *Variable Learning Rate*, fazendo com que a taxa de aprendizado varie para aumentar ou reduzir a velocidade do aprendizado. De forma resumida, o objetivo é fazer com que o aprendizado seja mais rápido quando o agente está perdendo o jogo, e mais lento e cauteloso quando o agente está ganhando.

#### 2.4.3.2.3
#### Resumo dos Algoritmos de RL para SMA

No quadro seguinte, alguns dos principais algoritmos Multiagentes RL estão relacionados ao tipo de tarefa que abordam e o ambiente.

Tabela 1: Principais algoritmos Multiagentes de RL por Tipo de Tarefa e Ambiente.

| Algoritmo | Referência | Tipo da Tarefa | Ambiente |
|---|---|---|---|
| JAL | Claus e Boutilier (1998) | Cooperativa | Estático |
| FMQ | Kapetanakis e Kudenko (2002) | Cooperativa | Estático |
| Team-Q | Littman (2001) | Cooperativa | Dinâmico |
| Distributed-Q | Lauer e Riedmiller (2000) | Cooperativa | Dinâmico |
| OAL | Wang e Sandholm (2002) | Cooperativa | Dinâmico |
| Minimax-Q | Littman (1994) | Competitivo | Dinâmico |
| Ficticious Play | Brown (1951) | Misto | Estático |
| MetaStrategy | Powers e Shoham (2004) | Misto | Estático |
| IGA | Singh et al. (2000) | Misto | Estático |
| WoLF-IGA | Bowling e Veloso (2002) | Misto | Estático |
| GIGA | Zinkevich (2003) | Misto | Estático |
| GIGA-WoLF | Bowling (2004) | Misto | Estático |
| AWESOME | Conitzer e Sandholm (2003) | Misto | Estático |
| Hyper-Q | Tesauro (2003) | Misto | Estático |
| Sigle-Agent RL | Sen et al. (1994) | Misto | Dinâmico |
| Nash-Q | Hu e Wellman (1998) | Misto | Dinâmico |
| CE-Q | Greenwald e Hall (2003) | Misto | Dinâmico |
| Asymetric-Q | Kononen (2003) | Misto | Dinâmico |
| NSCP | Weinberg e Rosenschein (2004) | Misto | Dinâmico |
| WoLF-PHC | Bowling e Veloso (2002) | Misto | Dinâmico |
| PD-WoLF | Banerjee e Peng (2003) | Misto | Dinâmico |
| EXORL | Suematsu e Hayashi (2002) | Misto | Dinâmico |

De uma maneira geral, os algoritmos podem utilizar técnicas de Diferença Temporal, busca direta de políticas ótimas e Teoria dos Jogos, podendo também funcionar através da combinação de mais de uma técnica. Por exemplo, os algoritmos Nash-Q (Hu e Wellman, 1998) e Minimax-Q (Littman, 1994) se originaram do algoritmo *Q-learning*, e incorporaram os conceitos relacionados à Teoria dos Jogos de Equilíbrio de Nash (Nash, 1950) e *minimax* (Von Neumann, 1959), respectivamente.

A Figura 4 abaixo mostra os algoritmos agrupados de acordo com o método utilizado e técnicas utilizadas por cada um (Busoniu, 2008).

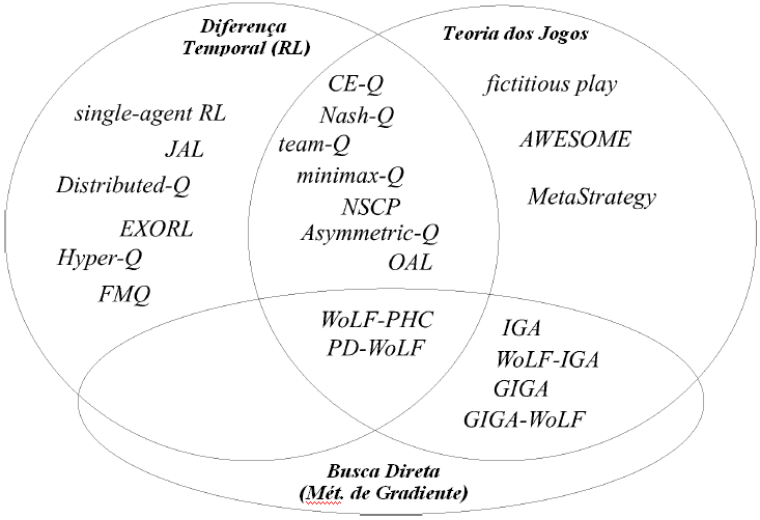


Figura 4: Resumo de algoritmos Multiagentes de RL e suas técnicas (Busoniu, 2008).



### 2.4.4
### Aplicações de Sistemas Multiagentes

Conforme já apresentado na introdução deste capítulo, os sistemas Multiagentes vêm encontrando uma grande variedade de aplicações. A seguir, os principais campos de pesquisa são explorados com o objetivo de fornecer uma visão geral do potencial dos SMA.

#### 2.4.4.1
#### Controle Distribuído

Em aplicações de controle distribuído, um conjunto de agentes autônomos interage em paralelo, através de ações de controle, com um mesmo objetivo. De maneira geral, sempre que o ambiente é um processo de controle e os agentes são os controladores deste processo, tem-se um problema de controle distribuído. São tipicamente aplicações que envolvem a cooperação entre os agentes, dado que o objetivo é o mesmo. Entre os principais problemas abordados estão: controle de processos (Stephan et al., 2000), controle de sinais de trânsito (Wiering, 2000; Bakker et al., 2005) e controle elétrico de redes de energia (Riedmiller et al., 2000).

### 2.4.4.2
### Times Robóticos

A criação de times robóticos é considerada uma das aplicações mais exploradas dos SMA, por ser talvez a mais natural, instintiva e de compreensão geral. Além disso, vários pesquisadores que atuam na área de SMA também são ativos no campo da robótica. De maneira geral, o ambiente de jogos para os times robóticos possuem duas dimensões, podendo ser real ou simulado. Os agentes podem usar qualquer tipo de algoritmo para aprender a melhor forma de se comportar e adquirir habilidades.

Uma aplicação largamente explorada como benchmark é o jogo da presa/predador ou o "*pursuit-game*". Neste jogo, vários agentes predadores devem capturar a presa convergindo ao seu redor (Ishiwaka et al., 2003; Kok et al., 2005; Xiao e Tan, 2007). Este jogo será mais detalhado posteriormente no capítulo de Estudo de Casos.

Há também problemas de controle de navegação, onde cada robô deve encontrar seu caminho a partir de um determinado ponto de origem, até o objetivo de destino fixo ou móvel, passando por obstáculos e evitando que a interferência de outros robôs o prejudique (Bowling e Veloso, 2002; Hu e Wellman, 2003; Calvo e Romero, 2007). Entre outras aplicações relacionadas à navegação estão: a localização e busca de objetos, cobertura de superfícies da forma mais ampla possível, e exploração de ambientes com o objetivo de captar o máximo de imagens através de sensores.

Um dos jogos mais populares para SMAs e que merece uma atenção especial é o futebol robótico. Com o objetivo de incentivar e acelerar a pesquisa em SMA, foi criada a RoboCup (Kitano, 1995), ou Copa do Mundo de Futebol de Robôs, onde os jogos ocorrem através do *SoccerServer*, desenvolvido originalmente por Noda (1995). O grande desafio lançado por Kitano (1998) foi a "criação de um time de robôs capaz de derrotar um time real campeão do mundo" até o ano de 2050. Após algumas evoluções na versão original do SoccerServer, foi realizada em 1996, a preRoboCup-96, em Osaka, Japão, e, posteriormente, a RoboCup-97 em Nagoya, também no Japão. Desde então, a competição ocorre todos os anos com crescente interesse de pesquisadores e do público em geral.

Atualmente, a competição possui diferentes categorias que permitem jogos em ambientes simulados, através do *SoccerServer*, ou reais, através de robôs. A Figura 5 ilustra o ambiente usado para jogos simulados entre times de robôs.

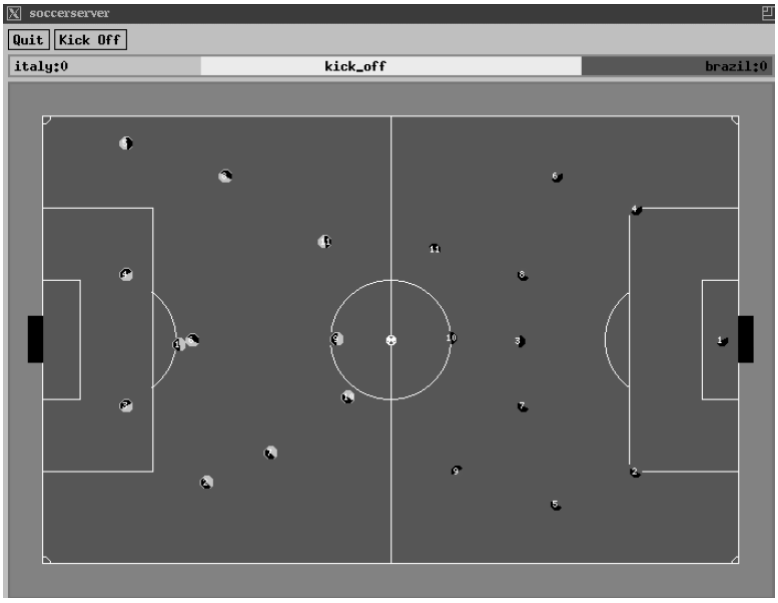


Figura 5: SoccerServer: servidor usado para jogos simulados entre times de robôs.

O simulador dos jogos é realista em diversos conceitos: a visão dos jogadores é limitada, os jogadores podem se comunicar através de mensagens colocadas em um "*blackboard*" visível a todos os demais jogadores, cada jogador é controlado por um processo distinto, cada time possui 11 jogadores, os jogadores possuem quantidades limitadas de energia, os sensores e ações têm ruídos, e o jogo ocorre em tempo real. As principais vantagens desta aplicação na avaliação de SMAs são (Stone e Veloso; 2000):

- Alta complexidade, o que o deixa bem próximo da realidade;
- Fácil acesso;
- Contempla a grande maioria dos conceitos envolvidos nos SMA, tanto para agentes homogêneos ou heterogêneos;
- Permite comparação direta entre times de robôs, independente da técnica utilizada por cada time.

### 2.4.4.3
### Sistemas de Negociação e Trading

Existem diversos mercados eletrônicos que permitem a comercialização de produtos entre os agentes, seja através de negociações ou leilões. Um exemplo de *benchmark* para este tipo de aplicação é o *Trading Agent Competition*, concurso onde os agentes devem organizar viagens negociando bilhetes de passagens e reservas de hotel (Wellman et al., 2003).

Em algumas aplicações, os agentes, representando uma determinada empresa ou indivíduo, apenas cooperam em tarefas distintas do processo de negociação (Lee e Oo, 2002; Oo et al., 2002; Lee et al., 2007). A Figura 6 ilustra um sistema de negociação Multiagente proposto por Lee et al. (2007), com quatro agentes, que desenvolvem capacidades específicas – gerar o sinal de compra, definir o preço de compra, gerar o sinal de venda e definir o preço de venda.

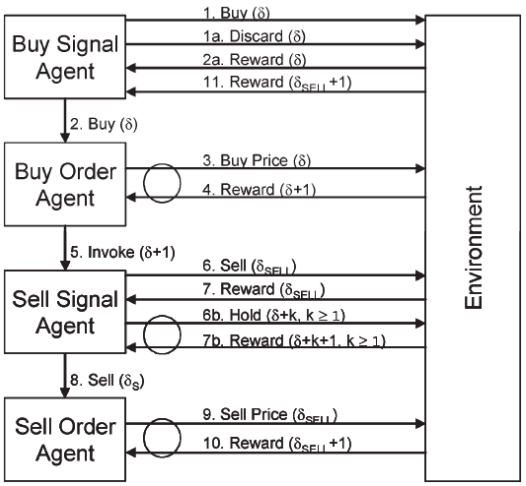


Figura 6: SMA de Negociação, onde cada agente possui uma função (Lee et al., 2007).

Em outros casos, os agentes autônomos buscam maximizar seus retornos interagindo em paralelo com os mercados (Hsu e Soo, 2001; Tesauro e Kephart, 2002; Raju et al., 2003).

#### 2.4.4.4
#### Gerenciamento de Recursos

Quando possuem o objetivo de gerenciar recursos coletivos, os agentes sempre agem de forma cooperativa e possuem basicamente dois tipos de comportamento (Busoniu, 2008):

- *gerentes dos recursos* - cada agente gerencia um recurso e aprende a melhor forma de responder a solicitações de uso deste recurso, sempre buscando otimizar uma determinada medida de desempenho, como em problemas de programação de elevadores (Crites e Barto, 1998);

- *clientes dos recursos:* os agentes aprendem a selecionar os recursos de maneira ótima, dada uma determinada medida de desempenho. Como, por exemplo, em aplicações de balanceamento de carga (Schaerf et al., 1995).

Exemplos de medidas de desempenho são: tempo de processamento de rotinas, tempo de espera por recursos, utilização e disponibilidade de recursos, distribuição justa de recursos aos clientes, etc. Além das aplicações de programação de elevadores e balanceamento de carga, um dos campos de pesquisa mais popular no gerenciamento de recursos através de múltiplos agentes é o controle e planejamento de rotas (Boyan e Littman, 1993; Choi e Yeung, 1995; Tillotson et al., 2004).

No próximo capítulo, será apresentado o modelo *Reinforcement Learning Neuro-Fuzzy Hierárquico Politree* (RL-NFHP), proposto originalmente em Figueiredo (2003), que foi usado como base para o desenvolvimento deste trabalho. Serão explorados a célula básica, a arquitetura, as regras hierárquicas associadas e o algoritmo de aprendizado utilizado pelo modelo.

# 3
# Reinforcement Learning Neuro-Fuzzy Hierárquico Politree (RL-NFHP)

## 3.1
## Introdução

Este capítulo apresenta formalmente o modelo *Reinforcement Learning* Neuro-Fuzzy Hierárquico Politree (RL-NFHP) proposto em Figueiredo (2003). O particionamento Politree é uma generalização sobre a forma de particionamento Quadtree, que por sua vez pode ser visto como uma extensão do particionamento binário do espaço (BSP).

## 3.2
## Particionamento Quadtree/Politree

No particionamento Quadtree o espaço é sucessivamente dividido em quadrantes, que, por sua vez podem ser novamente subdivididos em 4 regiões (quadrantes) em uma operação recursiva.

A Figura 7(a) ilustra este tipo de particionamento para o caso de duas dimensões, e a Figura 7(b) mostra a representação da árvore Quadtree.

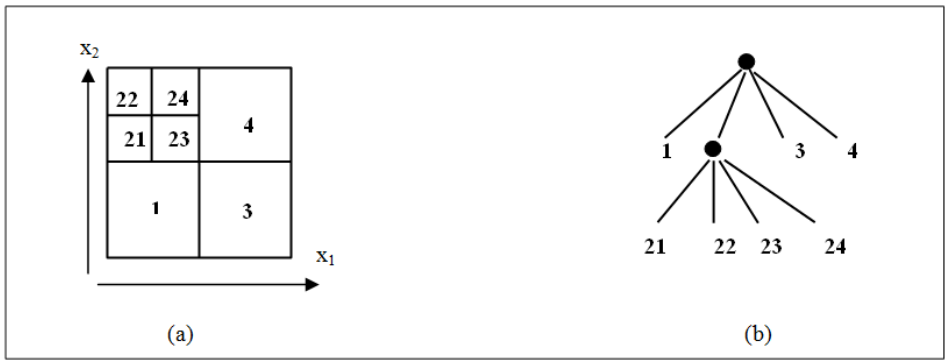


Figura 7: (a) exemplo de particionamento Quadtree; (b) árvore representativa do particionamento Quadtree referente a Figura 7ª.

O particionamento Quadtree pode ser fixo, como mostrado na Figura 7a, ou adaptativo. Neste último caso, as regiões geradas em cada subdivisão são retangulares e não mais quadradas, como ocorre no caso do particionamento fixo. A limitação do particionamento Quadtree (fixo ou adaptativo) está no fato de este trabalhar apenas em espaços bidimensionais. Isto pode ser contornado pela extensão para casos n-dimensionais. Por exemplo, no caso de dimensão n=3, temos o particionamento "Oct-tree" que divide o espaço em 8 ($2^3$ ) subespaços. Para n = 4 o espaço (hiperespaço) seria subdividido em 16 ($2^4$ ) subespaços e assim por diante. O particionamento utilizado no segundo modelo proposto nesta tese denominado *Reinforcement Learning* – Neuro-Fuzzy Hierárquico Politree (RL-NFHP) é uma generalização desta ideia. Nele, a subdivisão do espaço dimensional é realizada em n subespaços.

O modelo RL-NFHP é composto de uma ou várias células padrão chamadas *células RL-neuro-fuzzy politree* (RL-NFP). Estas células são dispostas numa estrutura hierárquica em forma de árvore. A célula de maior hierarquia gera a saída. As de menor hierarquia trabalham como consequentes das células de maior hierarquia. Estas células são descritas em detalhes na seção 3.3, a seguir.



## 3.3
## Célula Básica RL-Neuro-Fuzzy Politree

Uma célula RL-NFP é um mini-sistema neuro-fuzzy que realiza um particionamento politree em um determinado espaço, utilizando, para cada variável de entrada, as funções de pertinência descritas na seção 3.3. A célula RL-NFP gera uma saída precisa (*crisp*) após um processo de defuzzificação, conforme será mostrado posteriormente.

Apenas para efeito de ilustração, na representação da célula serão apresentadas duas entradas (Quadtree) tornando o desenho mais simples do que a forma n-dimensional proposta para o Politree.

As Figuras 8 e 9, a seguir, foram criadas para facilitar a compreensão do processo de defuzzificação da célula e o encadeamento dos consequentes. As entradas $x_1$ e $x_2$ geram os antecedentes das quatro regras fuzzy após serem computados os graus de pertinência $\rho_1(x_1)$, $\mu_1( x_1)$, $\rho_2(x_2)$ e $\mu_2(x_2)$, onde: $\rho_1$ é o

conjunto nebuloso *baixo e* $\mu_1$ é o conjunto nebuloso *alto* relativos à entrada $x_1$; e $\rho_2$ é o conjunto nebuloso *baixo e* $\mu_2$ é o conjunto nebuloso *alto* relativos à entrada $x_2$. Os valores definidos como consequentes são conjuntos de ações ($a_1$, $a_2$, ... $a_t$), onde cada ação está associada a uma função de valor-Q. Através do método de aprendizado baseado em RL, uma ação de cada polipartição ($a_i$, $a_j$, $a_p$ e $a_q$) será definida como aquela que representa o comportamento desejado do sistema quando o mesmo se encontra em um determinado estado. A Figura 8 ilustra a representação desta célula de forma simplificada e a Figura 9 apresenta-a sob o formato de rede neuro-fuzzy.

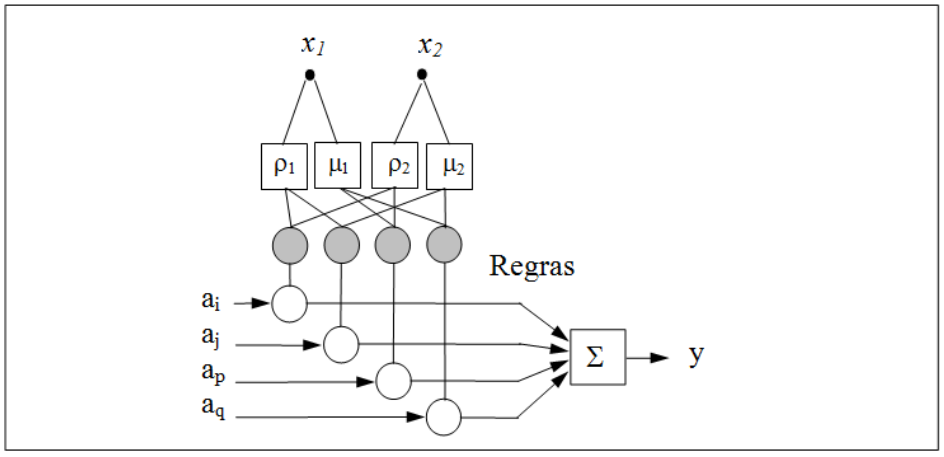


Figura 8: Célula Reinforcement Learning Neuro-Fuzzy Quadtree (Politree com n=2).

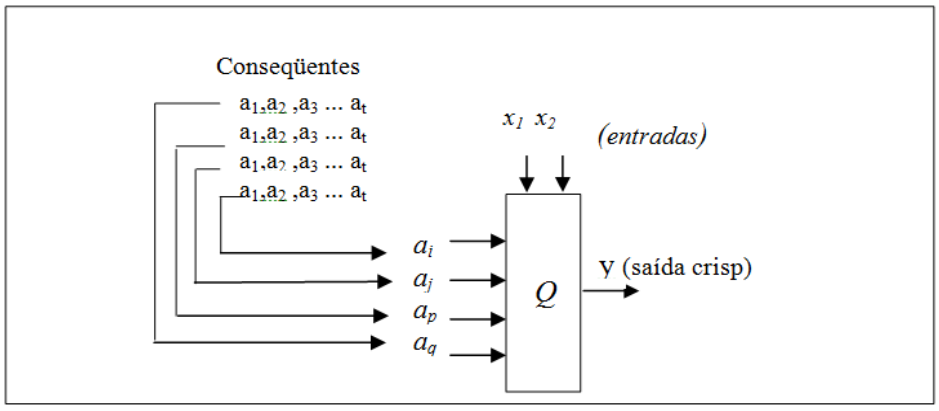


Figura 9: Diagrama simplificado da célula Reinforcement Learning Neuro-Fuzzy Quadtree relativo à Figura 8.

Na Figura 10, estão mostradas as camadas de fuzzificação, de regras e de defuzzificação. As expressões analíticas das funções de pertinência *alto* e *baixo* são dadas por sigmóides e seus complementos de 1. Estas funções de pertinência (FPs) possuem 2 parâmetros, 'a' e 'b', que definem os perfis das funções alto ($\mu$) e baixo ($\rho$) de cada variável de entrada. Os $\alpha_i$ (na Figura 10) simbolizam os graus de ativação das regras. Estes graus de ativação são calculados usando-se uma operação AND (T-norma) sobre os graus de pertinência de $\rho_1$, $\mu_1$, $\rho_2$ e $\mu_2$, conforme descrito a seguir:

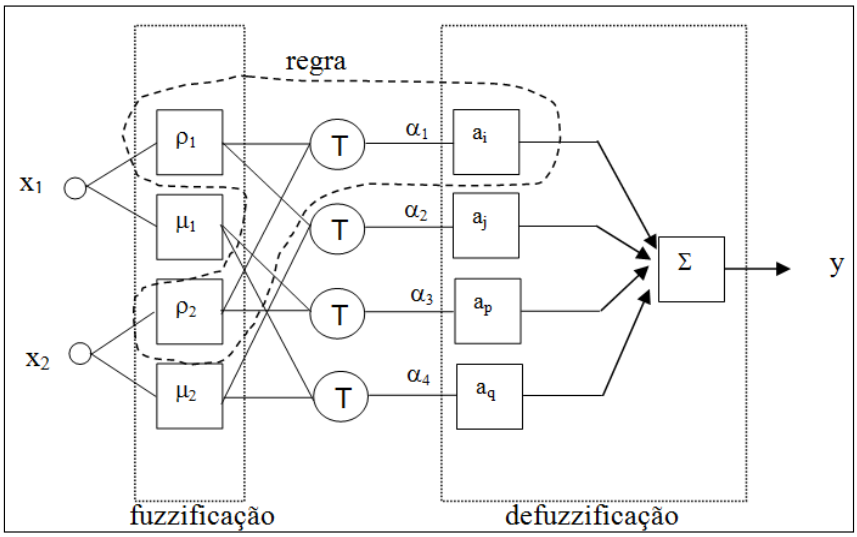


Figura 10: Célula RL-NFP representada sob o formato de rede neuro-fuzzy.

$$\begin{aligned} \alpha_1 &= \rho_1(x_1) * \rho_2(x_2); \\ \alpha_2 &= \rho_1(x_1) * \mu_2(x_2); \\ \alpha_3 &= \mu_1(x_1) * \rho_2(x_2); \\ \alpha_4 &= \mu_1(x_1) * \mu_2(x_2). \end{aligned} \tag{9}$$

O símbolo '*' representa a operação AND, que pode ser realizada, por exemplo, pela multiplicação ou pela operação de mínimo entre os dois valores. A interpretação linguística do mapeamento implementado pela célula RL-NFP da Figura 10 é dada pelo seguinte conjunto de regras:

$$
\begin{aligned}
&\text{regra}_1\text{: Se } x_1 \in \rho_1 \text{ e } x_2 \in \rho_2 \text{ então } y = a_i \\
&\text{regra}_2\text{: Se } x_1 \in \rho_1 \text{ e } x_2 \in \mu_2 \text{ então } y = a_j \\
&\text{regra}_3\text{: Se } x_1 \in \mu_1 \text{ e } x_2 \in \rho_2 \text{ então } y = a_p \\
&\text{regra}_4\text{: Se } x_1 \in \mu_1 \text{ e } x_2 \in \mu_2 \text{ então } y = a_q
\end{aligned}
\tag{10}
$$

Cada regra corresponde a um quadrante da Figura 11. Quando os valores das entradas incidem sobre o quadrante 1, é a regra 1 que tem maior grau de ativação. Quando a incidência é sobre o quadrante 2, é a regra 2 que tem maior grau de ativação. No caso das entradas caírem no quadrante 3, é a regra 3 que tem o maior grau de ativação e, finalmente, quando a incidência é sobre o quadrante 4, é a regra 4 que tem maior grau de ativação. Cada quadrante por sua vez pode ser subdividido em quatro partes, através de outra célula RL-NFP. É muito importante lembrar que os consequentes $a_i$ não são valores predeterminados, eles fazem parte de um conjunto de ações que deve ser explorado para que se possa determinar, através de aprendizado por reforço, a ação mais adequada para cada regra.

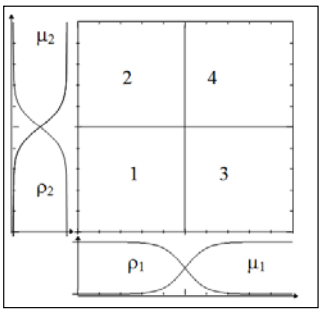


Figura 11: Divisão em quadrantes realizada pelas FPs alto e baixo.

A célula *RL-neuro-fuzzy politree* pode ser representada por uma estrutura em árvore como mostra a Figura 12(a). Ela corresponde ao particionamento da Figura 11 com duas entradas. Este formato proporciona a representação da célula RL-NFP genérica (com *n* entradas) que é exibida na Figura 12(b).

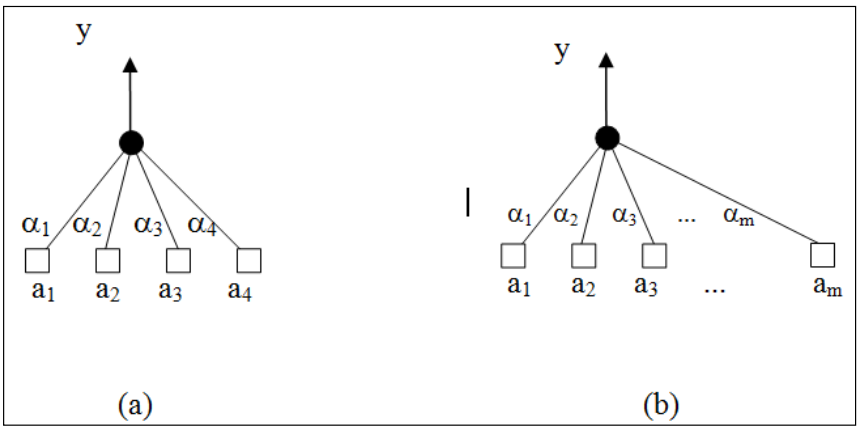


Figura 12: (a) Representação em árvore da célula RL-NFP com duas entradas; (b) representação genérica em árvore da célula RL-NFP com n entradas, onde m = $2^n$.

As saídas 'y' das células RL-NFP das Figuras 12(a) e 12(b) são dadas pelas médias ponderadas mostradas nas equações 11 e 12, respectivamente:

$$y = \left( \frac{\sum_{i=1}^{4} \alpha_i \times a_i}{\sum_{i=1}^{4} \alpha_i} \right) \tag{11}$$

$$y = \left( \frac{\sum_{i=1}^{2^n} \alpha_i \times a_i}{\sum_{i=1}^{2^n} \alpha_i} \right) \tag{12}$$

onde n é o número de entradas e $a_i$ corresponde a um dos dois consequentes possíveis abaixo:

- *um singleton* (consequente *fuzzy singleton*, ou Sugeno de ordem zero), caso em que $a_i$ = constante;
- *Saída de um estágio de nível anterio*r, caso em que $a_i = y_j$, onde $y_j$ representa a saída de uma célula genérica 'j', cujo valor é calculado, também, pela eq. 12.

Apesar do consequente *singleton* ser simples, este não é conhecido previamente. É através do algoritmo RL que será possível determinar o melhor valor *singleton* (ação) para esta regra. Ou seja, os consequentes de cada regra são representados por um conjunto de ações relacionadas àquele estado. O estado atual do agente é definido pelos valores das variáveis de entrada, que tornam ativas as células cujos domínios das funções de pertinência delimitam um estado.

Como mencionado, na célula RL-NFP básica as funções de pertinência são implementadas por *sigmóides* ($\rho$ e $\mu$) e por seu complemento de um [1- $\mu(x)$]. A utilização dos complementos a um leva a uma simplificação no procedimento de defuzzificação realizado pelo processo de média ponderada, pois o somatório dado pela equação abaixo, é igual a 1 para quaisquer valores de entrada $x_i$.

$$\sum_{i=1}^{2^n} \alpha_i = 1 \tag{13}$$

Desta forma, a saída da célula básica (eq. 12) fica simplificada, como mostra a eq. 14, a seguir.

$$y = \sum_{i=1}^{2^n} \alpha_i . a_i \tag{14}$$

As células RL-NFP formam uma estrutura hierárquica que resulta nas regras que compõem o raciocínio do agente (seção 2.2.2). Para este modelo, os antecedentes das regras são definidos pelas variáveis de entrada, às quais estão associados dois conjuntos fuzzy. No caso deste modelo, todas as variáveis de entrada do sistema compõem os antecedentes das células. Os valores das variáveis de entrada são lidos pelos ***sensores*** do agente e são os respectivos graus de pertinência são determinados pelos antecedentes. Se o grau de ativação não é nulo (resultado da aplicação do T-norma é diferente de zero), a regra é disparada. Os consequentes são as ações que o agente deve aprender ao longo do processo e são realizadas pelo seu ***atuador***. Sendo assim, o modelo RL-NFHP também cria e determina sua estrutura mapeando estados em ações.

## 3.4
## Arquitetura RL-NFHP

A arquitetura do modelo RL-NFHP é composta pela interligação entre as células básicas descritas acima. Isto é exemplificado na Figura 13, abaixo. A árvore *Politree* referente ao particionamento da Figura 13 é mostrada na Figura 14. Cada partição não subdividida é chamada de *polipartiçã*o.

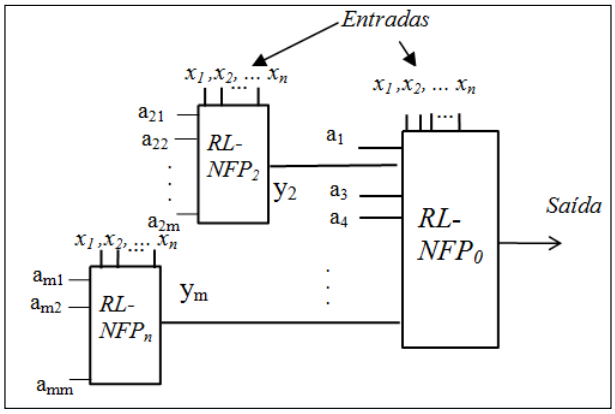


Figura 13: Exemplo de arquitetura RL-NFHP.

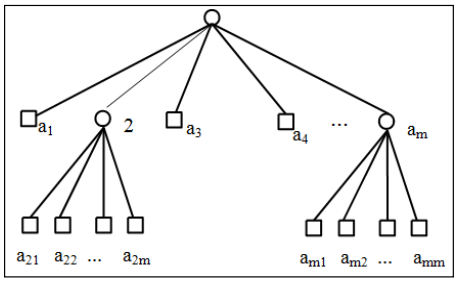


Figura 14: Árvore *Politree* referente ao particionamento da Figura 13.

Na árvore da Figura 14, os nós simbolizados com pequenos círculos são nós interiores e representam regiões que foram subdivididas. Os nós simbolizados por pequenos quadrados são nós terminais e representam as polipartições, isto é, as regiões que não sofreram subdivisões. A raiz da árvore simboliza todo o espaço a ser particionado.

No exemplo das Figuras 13 e 14 as polipartições 1, 3, 4 não foram subdivididas, portanto os consequentes de suas respectivas regras são os valores

$a_1$, $a_3$ e $a_4$. As partições 2 e m foram subdivididas e os consequentes de suas regras são as saídas ($y_2$ e $y_m$) dos subsistemas 2 e n. Estes por sua vez têm, como consequentes, os valores $a_{21}$, $a_{22}$, ..., $a_{2m}$, e $a_{m1}$, $a_{m2}$, ..., $a_{mm}$, respectivamente. Cada '$a_i$' corresponde a um consequente de *Sugeno* de ordem 0 (*singleton*), representando a ação que será identificada (dentre as ações possíveis), através de aprendizado por reforço, como sendo a mais favorável para um determinado estado do ambiente. A saída do sistema da Figura 13 é dada pela eq. 15, a seguir.

$$y = \alpha_1.a_1 + \alpha_2 \sum_{i=1}^{2^n} \alpha_{2i}.a_{2i} + \alpha_3.a_3 + \alpha_4.a_4 + \ldots + \alpha_m \sum_{i=1}^{2^n} \alpha_{mi}.a_{mi} \qquad (15)$$

De uma forma genérica, a equação de saída de um sistema RL-NFHP de dois níveis completos é dada pela eq. 16 abaixo. Neste caso houve necessidade de se incluir as variáveis $k_i$ e $k_{ij}$. Essas variáveis assumem apenas valores iguais a '0' ou '1', indicando a existência ou não das polipartições de ordem '*i*' e '*ij*', respectivamente.

$$y = \sum_{i=1}^{2^n} \alpha_i k_i a_i + \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \alpha_i \alpha_{ij} k_{ij} a_{ij} \qquad (16)$$

Expandindo a equação 16 para um sistema RL-NFHP de quatro níveis de hierarquia tem-se a seguinte fórmula:

$$\begin{aligned} y = & \sum_{i=1}^{2^n} \alpha_i k_i a_i + \\ & \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \alpha_i \alpha_{ij} k_{ij} a_{ij} + \\ & \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \sum_{p=1}^{2^n} \alpha_i \alpha_{ij} \alpha_{ijp} k_{ijp} a_{ijp} + \\ & \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \sum_{p=1}^{2^n} \sum_{q=1}^{2^n} \alpha_i \alpha_{ij} \alpha_{ijp} \alpha_{ijpq} k_{ijpq} a_{ijpq} \end{aligned} \qquad (17)$$

Na equação 17:

- $\alpha_i$ , $\alpha_{ij}$ , $\alpha_{ijp}$ , $\alpha_{ijpq}$ , são os níveis de disparo das regras de cada polipartição i, ij, ijk, ou ijkl, respectivamente;

- $k_i$ ($k_{ij}$ , $k_{ijp}$ , $k_{ijpq}$), é igual a “1” se a partição i (ou ij, ou ijp ou ijpq) existe e “0” caso contrário;
- $a_i$ , $a_{ij}$ , $a_{ijp}$ , $a_{ijpq}$ , são os consequentes (*singletons*) das regras existentes.

Na equação da expressão geral de saída do modelo RL-NFHP, descrita acima, já se levou em consideração a simplificação causada pelo uso das funções de pertinência complementares ($\rho + \mu = 1$) no método de defuzzificação das saídas de cada subsistema neuro-fuzzy.

O conjunto de regras que traduz o conhecimento linguístico do exemplo da Figura 13 é:

Se $x_1 \in \rho_1$ e $x_2 \in \rho_2 \ldots x_n \in \rho_n$ então $y = a_1$
Se $x_1 \in \mu_1$ e $x_2 \in \rho_2 \ldots x_n \in \rho_n$ então
{
Se $x_1 \in \rho_{21}$ e $x_2 \in \rho_{22} \ldots x_n \in \rho_{2n}$ então $y = a_{21}$
Se $x_1 \in \mu_{21}$ e $x_2 \in \rho_{22} \ldots x_n \in \rho_{2n}$ então $y = a_{22}$
Se $x_1 \in \mu_{21}$ e $x_2 \in \mu_{22} \ldots x_n \in \rho_{2n}$ então $y = a_{23}$
.
.
Se $x_1 \in \mu_{21}$ e $x_2 \in \mu_{22} \ldots x_n \in \mu_{2n}$ então $y = a_{2m}$
}
Se $x_1 \in \mu_1$ e $x_2 \in \mu_2 \ldots x_n \in \rho_n$ então $y = a_3$
Se $x_1 \in \mu_1$ e $x_2 \in \mu_2 \ldots x_n \in \rho_n$ então $y = a_4$
.
.
Se $x_1 \in \mu_1$ e $x_2 \in \mu_2 \ldots x_n \in \mu_n$ então
{
Se $x_1 \in \rho_{m1}$ e $x_2 \in \rho_{m2} \ldots x_n \in \rho_{mn}$ então $y = a_{m1}$
Se $x_1 \in \mu_{m1}$ e $x_2 \in \rho_{m2} \ldots x_n \in \rho_{mn}$ então $y = a_{m2}$
Se $x_1 \in \mu_{m1}$ e $x_2 \in \mu_{m2} \ldots x_n \in \rho_{mn}$ então $y = a_{m3}$
.
.
Se $x_1 \in \mu_{m1}$ e $x_2 \in \mu_{m2} \ldots x_n \in \mu_{mm}$ então $y = a_{mm}$
}

Onde :

- $\rho_1, \rho_2, \ldots, \rho_n$ e $\mu_1, \mu_2, \ldots, \mu_n$, são as funções de pertinência que definem a partição de nível 1.

- $\rho_{21}, \rho_{22}, \ldots, \rho_{2n}$ e $\mu_{21}, \mu_{22}, \ldots, \mu_{2n}$, são as funções de pertinência que definem as subdivisões da partição 2.

- $\rho_{m1}, \rho_{m2}, \ldots, \rho_{mn}, \mu_{m1}, \mu_{m2}, \ldots, \mu_{mn}$, são as funções de pertinência que definem as subdivisões da partição m.

### 3.4.1
### Antecedentes das Regras do Modelo RL-NFHP

A Figura 15 mostra a estrutura de aprendizado do agente. A leitura do ambiente ($s_1$, $s_2$, ..., $s_n$) é feita pelo agente através de seus sensores e estas leituras podem ser traduzidas em um ou mais valores de entrada ($x_1$, $x_{2, \ldots}$ $x_n$) das células; no entanto, cada célula tem associados a ela **todos** as entradas que serão considerados no sistema no momento de sua criação. Os valores $x_i$ são avaliados nas células, podendo disparar regras Sendo assim, toda vez que uma regra é disparada, ou, dito de outra forma, uma **célula** se torna **ativa**, o processo de aprendizado identifica que o agente está em um **estado definido** pelo domínio dos conjuntos fuzzy do antecedente da regra (domínio da entrada da célula).

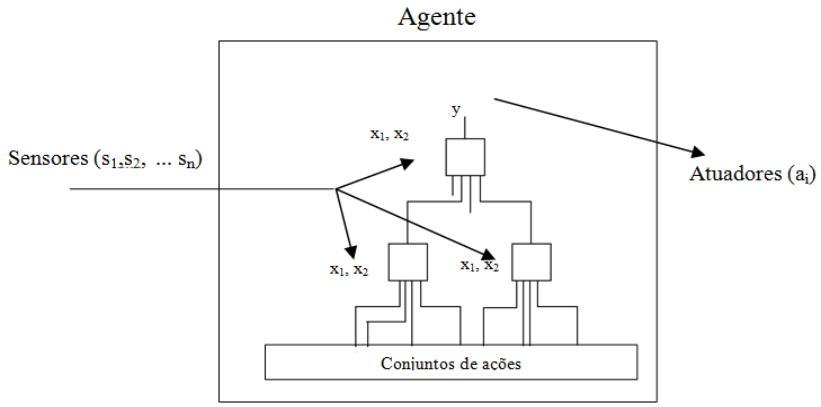


Figura 15: Esquema do processo de aprendizado do agente.

### 3.4.2
### Consequentes das Regras do Modelo RL-NFHP

Os consequentes das regras fuzzy nas células RL-NFHP são as ações que devem ser identificadas através de aprendizado por reforço. Quando as células se tornam ativas, as ações são selecionadas, cada uma relativa à combinação *baixo* e *alto* de cada uma das entradas da célula. A seleção ocorre em função de valores atribuídos a cada uma das ações que pertencem ao conjunto de ações disponíveis para cada polipartição *baixa* e *alta*. Novamente, para efeito de ilustração (Figura

16), serão apresentadas duas entradas (Quadtree), tornando o desenho mais simples do que a forma n-dimensional proposta para o Politree. A Figura 16 mostra a célula RL-NFHP com 4 conjuntos de ações (associados aos seus respectivos Q-valores), onde cada conjunto está relacionado às polipartições de cada célula. Cada conjunto pode possuir um número de ações t, independentemente do número de entradas.

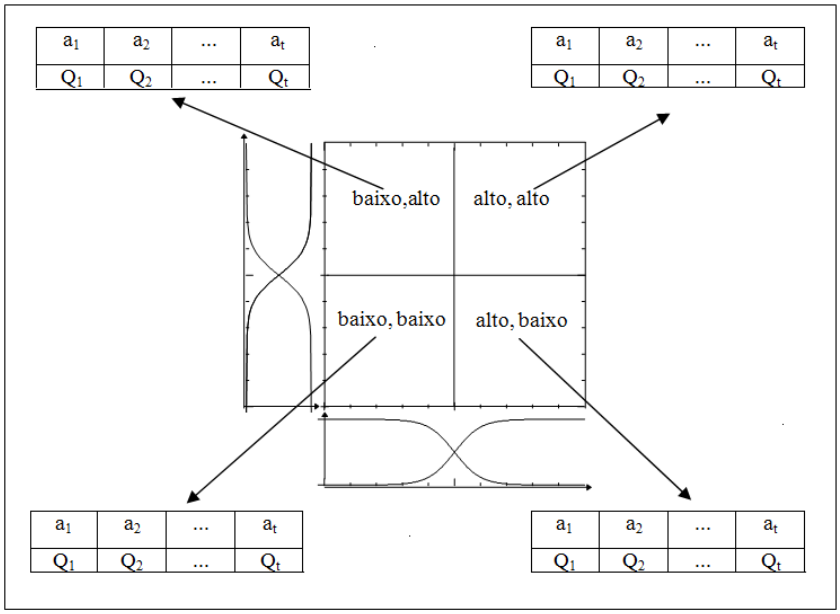


Figura 16: Interior da célula RL-NFHP com duas entradas (Quadtree).

Os consequentes serão do tipo 1 se a célula for uma folha da estrutura e serão do tipo 2 se forem células intermediárias.

**Consequente Tipo 1 – Singleton**

É o tipo mais simples de consequente de uma regra fuzzy. As regras fuzzy com este consequente são mostradas abaixo.

Se $x_1 \in \rho_1$ e $x_2 \in \rho_2$ então $y = a_i$

Se $x_1 \in \rho_1$ e $x_2 \in \mu_2$ então $y = a_j$

Se $x_1 \in \mu_1$ e $x_2 \in \rho_2$ então $y = a_p$

Se $x_1 \in \mu_1$ e $x_2 \in \mu_2$ então $y = a_q$

A vantagem do uso deste tipo de consequente está na facilidade do cálculo da saída que, neste caso, é geralmente efetuado através da média ponderada. Este tipo de consequente também é conhecido como consequente de *Sugeno* de ordem '0'.

Apesar de o método de cálculo para este tipo de consequente ser simples, o valor do consequente **não é conhecido a priori**. Um dos objetivos do modelo é, portanto, aprender, através do algoritmo SARSA (Sutton e Barto, 1998), a identificar as ações associadas aos conjuntos fuzzy baixo e alto da célula RL-NFP que melhor respondam ao estado atual do agente.

**Consequente Tipo 2 – Célula RL- NFP**

Este tipo de consequente é, na verdade, a saída de um outro mini-sistema RL- Neuro-Fuzzy Hierárquico Politree implementado por uma célula RL-NFP. Isto gera a hierarquia inerente aos sistemas neuro-fuzzy hierárquicos. As regras fuzzy com este consequente são como as que foram mostradas na seção 3.4- Arquitetura RL-NFHP.

## 3.5
## Algoritmo de Aprendizado

O processo de aprendizado começa com a definição das entradas relevantes, para o sistema/ambiente no qual o agente está inserido, e dos conjuntos de ações que ele pode dispor para atingir seus objetivos. As funções de valores-Q iniciais associadas às ações também devem ser definidas. Normalmente, as aplicações que utilizam SARSA ou *Q-Learning* iniciam suas funções de valores-Q com zero.

O algoritmo de aprendizado ocorre em seis fases, que são descritas a seguir.

**Passos do Aprendizado:**

**1 - Inicialização**

Uma célula raiz é criada, tendo como domínios dos seus conjuntos fuzzy (como definidos na seção 3.3), relativos a cada uma das entradas, os valores

mínimo e máximo destas entradas (Limite Inferior – LI e Limite Superior – LS). Com o objetivo de generalidade, os valores das variáveis de entrada são normalizados. Os valores correspondentes às variáveis de entrada da célula são lidos do ambiente, normalizados e podem ser aplicados diretamente às entradas da célula ou ser modificados segundo uma função que os torne adequados às variáveis das entradas da célula. Estes valores são avaliados nos conjuntos fuzzy baixo e alto, resultando nos graus de pertinência $\rho(x_1)\mu(x_1)$, ... $\rho(x_n)\mu(x_n)$, respectivamente, para cada variável de entrada. Cada uma das polipartições escolhe uma das ações de seu conjunto de ações baseando-se nos métodos descritos na seção 3.5 (passo 3 do algoritmo – Seleção das ações). A saída da célula é calculada pelo processo de defuzzificação e representa a ação que será executada pelos *atuadores* do agente.

Com a utilização de mais entradas, pode ocorrer que os valores $\alpha_i$ (resultado da aplicação da T-norma sobre os graus de pertinência relativos às entradas da célula) se tornem muito pequenos, o que faria o algoritmo ter um gasto maior de tempo computacional executando cálculos para a saída e atualizações das funções de valor que não teriam um peso significante para o algoritmo. Sendo assim, foi acrescentado o conceito de *alfa-cut* (limiar de corte) com o objetivo de inibir na saída ações relacionadas a polipartições cujos $\alpha_i$ sejam muito pequenos. Neste caso, o valor $\alpha_i$ desta polipartição torna-se igual a zero. Isso significa que, nesta iteração, esta polipartição não contribuirá na saída com sua ação, nem terá a sua função de valor-Q (associada à ação selecionada) atualizada.

**2 - Função de Avaliação/Retorno**

Após a execução da ação, uma nova leitura do ambiente é realizada. Esta leitura permite que seja calculado o valor de reforço do ambiente e se avalie a ação tomada pelo agente. Este valor deve ser calculado através de uma função de avaliação definida segundo os objetivos do agente, sendo fundamental para a orientação do agente ao longo do processo de aprendizado.

**3 - Retropropagar o retorno**

A cada passo, no processo de aprendizado, o retorno é calculado para cada partição de todas as células ativas, mediante a sua participação na ação resultante.

Dessa forma, o retorno do ambiente é retropropagado a partir da célula raiz até as células folhas, conforme o exemplo do sistema da Figura 17. A Figura 17(a) mostra duas células com duas entradas. Sendo assim, cada célula possui 4 polipartições, cada uma relativa à combinação baixo/alto dos graus de pertinência avaliados pelas entradas $x_1$ e $x_2$. A Figura 17(b) mostra o retorno global do sistema representado pela letra R no topo da árvore, que será definido conforme a aplicação ou estudo de caso em questão. Os valores correspondentes aos retornos de cada partição da célula RL-$NFP_0$ são $R_{0bb}$, $R_{0ba}$, $R_{0ab}$, $R_{0aa}$, e são calculados mediante os graus de pertinências correspondentes aos $\alpha_i$ relativos à célula RL-$NFP_0$.

$R_{0bb,}$ $R_{0ba}$, $R_{0ab}$, $R_{0aa}$ são os valores dos retornos locais da célula ‘0’ calculados para as polipartições bb (baixo/baixo), ba (baixo/alto), ab (alto/baixo) e aa (alto/alto) desta célula.

$R_{1bb,}$ $R_{1ba}$, $R_{1ab}$, $R_{1aa}$ são os valores dos retornos locais da célula ‘1’calculados para as polipartições bb (baixo/baixo), ba (baixo/alto), ab (alto/baixo) e aa (alto/alto) desta célula.

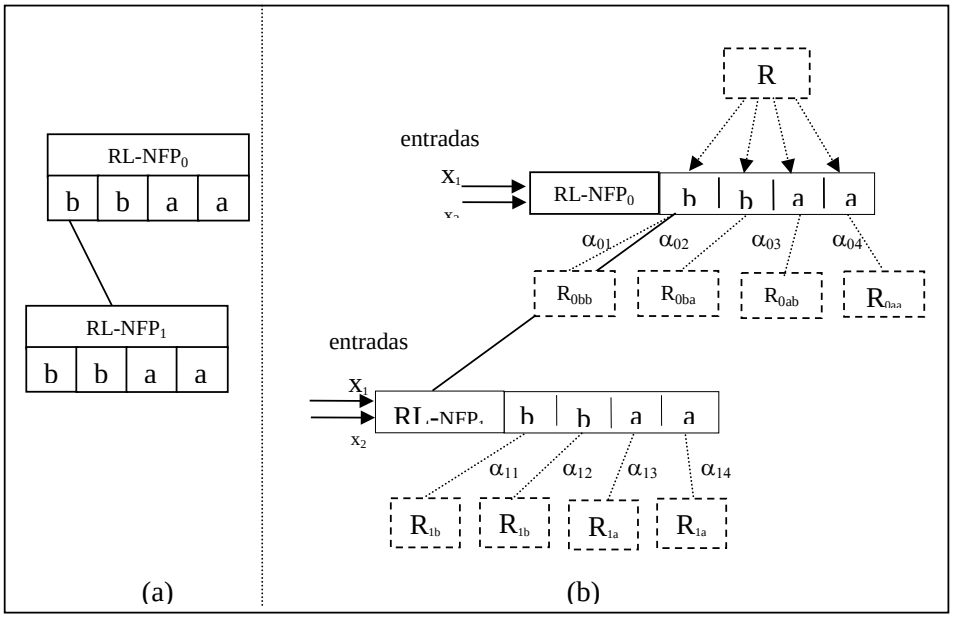


Figura 17: Retropropagação do retorno do ambiente para o modelo RL-NFHP.

Os valores $\alpha_i$ são calculados usando-se uma operação AND (T-norma). A Figura 17 e a eq. 18 também exemplificam os cálculos dos $\alpha_i$:

$$\begin{aligned} \alpha_{01} &= \rho_0(x_1) \,.\, \rho_0\,(x_2) \\ \alpha_{02} &= \rho_0\,(x_1) \,.\, \mu_0\,(x_2) \\ \alpha_{03} &= \mu_0\,(x_1) \,.\, \rho_0\,(x_2) \\ \alpha_{04} &= \mu_0\,(x_1) \,.\, \mu_0\,(x_2) \end{aligned} \tag{18}$$

Os cálculos dos retornos das partições da célula RL-NFHP$_0$ são definidos pela eq. 19.

$$\begin{aligned} R_{0bb} &= \alpha_{01} \,.\, R \\ R_{0ba} &= \alpha_{02} \,.\, R \\ R_{0ab} &= \alpha_{03} \,.\, R \\ R_{0aa} &= \alpha_{04} \,.\, R \end{aligned} \tag{19}$$

Os valores correspondentes aos retornos de cada partição da célula RL-NFP$_1$ são: $R_{1bb,}\, R_{1ba}$ , $R_{1ab}$ , $R_{1aa}$ e são calculados mediante os graus de pertinências correspondentes aos $\alpha_i$ relativos à célula RL-NFP$_1$ onde:

$$\begin{aligned} \alpha_{11} &= \rho_1(x_1) \,.\, \rho_1\,(x_2) \\ \alpha_{12} &= \rho_1\,(x_1) \,.\, \mu_1\,(x_2) \\ \alpha_{13} &= \mu_1\,(x_1) \,.\, \rho_1\,(x_2) \\ \alpha_{14} &= \mu_1\,(x_1) \,.\, \mu_1\,(x_2) \end{aligned} \tag{20}$$

Os cálculos dos retornos da célula RL-NFHP$_1$ são definidos pelas eqs. 21.

$$\begin{aligned} R_{1bb} &= \alpha_{11} \,.\, R_{0bb} \\ R_{1ba} &= \alpha_{12} \,.\, R_{0bb} \\ R_{1ab} &= \alpha_{13} \,.\, R_{0bb} \\ R_{1aa} &= \alpha_{14} \,.\, R_{0bb} \end{aligned} \tag{21}$$

**4 - Seleção das ações**

As ações são associadas às funções de valores-Q e compõem um conjunto de ações que são selecionadas e experimentadas durante o aprendizado RL. Como já exposto nas seções anteriores deste capítulo, a exploração do espaço de estados é fundamental para a descoberta de ações que correspondam à melhor resposta do agente (que visa atingir um objetivo) quando este se encontra em um determinado estado do ambiente. Sendo assim, para o modelo proposto, dois métodos de seleção foram testados.

O primeiro método, denominado $\epsilon$-*greedy* (Sutton e Barto, 1998), seleciona a ação associada a maior função de valor-Q esperada com probabilidade (1-$\epsilon$); e com probabilidade $\epsilon$ seleciona aleatoriamente uma ação qualquer.

No segundo método, a ação escolhida com probabilidade (1-$\epsilon$) é também a que apresentar a maior função de valor-Q associada; e com um número de vezes proporcional a $\epsilon$ a seleção é baseada na distribuição de probabilidade dada pelas funções de valores-Q (eq. 22). Esta forma de seleção é mais conservativa, uma vez que as ações que apresentarem as maiores funções de valores-Q terão maiores chances de serem escolhidas. Este tipo de método de seleção de ações é conhecido como *softmax* (Sutton & Barto, 1998).

$$P(a_i|s) = \frac{Q(s, a_i)}{\sum_{a_k \in \{ações\}} Q(s, a_k)} \qquad (22)$$

$P(a_i|s)$ é a probabilidade de se escolher a ação $a_i$, dado que o agente está no estado *s*, $Q(s, a_i)$ é a função de valor da ação $a_i$ e $\sum_{a_k \in \{ações\}} Q(s, a_k)$ é a soma das funções de valores-Q relativas às ações disponíveis para o agente quando o mesmo se encontra no estado *s*.

O objetivo destes procedimentos é possibilitar a escolha de ações que resultem em um valor de reforço alto do ambiente, mas que ainda não tenham a função de valor-Q associada mais alta.

De uma forma geral, o segundo método de seleção de ações apresentado resultou em desempenho melhor do agente. Sendo assim, este foi o método adotado nos problemas definidos nos estudos de casos.

Idealmente, no início do processo de aprendizado, o parâmetro $\epsilon$ deveria ter um valor que permitisse mais *exploration* (diversificação na escolha das ações) e menos *exploitation* (intensificação na escolha de determinadas ações) e, à medida que o sistema aprende, permitir menos *exploration* e mais *exploitation*. No entanto, como ocorrem inclusões de células na estrutura ao longo do processo de aprendizado, o que seria indicado, nestas circunstâncias, é que estas novas células também tenham oportunidade de explorar suas ações. Sendo assim, foi definido, para cada partição da célula, um parâmetro denominado $\epsilon$*-greedy*, cujo valor deve estar em [0,1], possibilitando variar a política de escolha da ação deste modelo.

Com o objetivo de melhorar o desempenho do aprendizado do modelo, o parâmetro $\epsilon$*-greedy* foi definido de forma *adaptativa.* Este procedimento é usado no método de aprendizado por reforço AHC (Sutton 1998). Neste procedimento, quando a ação resultante é boa, o valor do parâmetro $\epsilon$*-greedy* desta partição é reduzido, diminuindo a probabilidade desta partição usar uma política *explorative* (seleção da ação associada a maior função de valor Q). Quando a ação resultante não apresenta um bom desempenho, as partições das células ativas têm os seus parâmetros $\epsilon$*-greedy* aumentados, permitindo que estas partições, nos passos seguintes, tenham maiores chances de usarem um método *exploitive* (seleção da ação aleatória da ação).

**5 - Atualização dos valores Q**

A partir dos valores de reforços calculados para cada célula da estrutura, as funções de valores-Q associadas às ações que tenham contribuído para a ação resultante executada pelo agente devem ser atualizadas. Esta atualização é feita a partir da avaliação entre os retornos globais, atual e anterior. A atualização das funções de valores-Q ocorre de duas formas distintas: para o caso do valor do retorno global atual ser maior que o retorno global anterior ($R_{t+1} > R_t$) e para o caso do retorno global atual ser menor ou igual ao retorno global anterior ($R_t \geq R_{t+1}$).

**Primeiro Caso: $R_{t+1} > R_t$**

Caso o retorno global atual seja maior que o retorno global anterior, então as ações atuais (ações que foram executadas quando o agente estava no estado $s_t$) têm maiores chances de fornecerem a melhor resposta do agente ao sistema quando o mesmo se encontrar neste estado. Assim, se $R_{t+1} > R_t$ deve-se premiar as ações selecionadas no passo (t), atualizando suas respectivas funções de valores-Q conforme a equação do algoritmo *SARSA* (Sutton, 1996), definida pela seguinte expressão:

$$Q(s_t, a_t) = (1 - \alpha_t) \cdot Q(s_t, a_t) + \alpha_t [\, r_{t+1} + \gamma\, Q(s_{t+1,}\, a_{t+1})] \qquad (23)$$

primeira parcela segunda parcela

onde: o valor $Q(s_t, a_t)$ é atualizado a partir do seu valor atual; $r_{t+1}$ é o reforço local imediato (este é o reforço que é definido no passo da retropropagação do retorno global); o $\gamma$ é um parâmetro que fixa um percentual da contribuição da função de valor-Q associada à próxima ação $a_{t+1}$ escolhida ($Q(s_{t+1,}\, a_{t+1},)$) quando o sistema está no estado $s_{t+1}$; e $\alpha$ é o parâmetro proporcional à contribuição relativa desta ação local na ação global. A seguir o parâmetro $\alpha$ e a definição do valor $Q(s_{t+1,}\, a_{t+1},)$ usados na eq. 28 serão detalhados.

**O parâmetro $\alpha$**

O parâmetro $\alpha$ está compreendido entre [0,1]. Na maioria das aplicações ele tem seu valor inicial igual a 1 e, à medida que o aprendizado evolui, seu valor é reduzido. No início do processo de aprendizado, sua função é estimular os novos valores aprendidos a partir do reforço ($r_{t+1}$) e de $Q(s_{t+1,}\, a_{t+1})$ (Sutton, 1998). À medida que o processo de aprendizado evolui, o valor de $\alpha$ é reduzido, aumentando o peso da primeira parcela, que é relativa aos valores $Q(s_t, a_t)$ já aprendidos.

No caso do modelo RL-NFHP (Figueiredo, 2003), baseado em aproximação de funções, a redução deste parâmetro ao longo do processo resultou em graves problemas de aprendizado. A explicação para este fato é que quando os graus de pertinência das bipartições são pequenos (podendo permanecer assim durante um número significante de passos), a contribuição da segunda parcela da eq. 28 não

era tão efetiva e o valor de $\alpha$ ainda alto não permitia que a função de valor-Q pudesse evoluir, ou seja, aumentar e se destacar dos demais. Quando o grau de pertinência desta bipartição aumentava, muitas vezes o parâmetro $\alpha$ tinha o seu valor já muito reduzido, não permitindo que a atualização da função de valor-Q desta partição fosse realizada adequadamente, nem para este valor Q, nem para outro que pudesse vir a ser escolhido por qualquer método *non-greedy*.

Após vários testes, a avaliação que gerou melhores resultados foi a que definiu o parâmetro $\alpha$ como sendo proporcional à contribuição relativa desta ação local na ação global. Como a punição e o prêmio são realizadas em condições distintas, a atualização da função de valor-Q da partição que estiver contribuindo mais naquele passo também terá seu valor atualizado segundo esta proporção e a que tiver participação minoritária terá seu valor alterado na proporção desta participação.

A ação de saída da célula é resultado da contribuição das ações de seus dois consequentes. Caso a bipartição que está sendo atualizada tenha um grau de pertinência muito pequeno, mesmo que a ação não seja uma ação ideal, essa influência é minimizada. À medida que o agente se desloca no espaço de estados e cresce o grau de pertinência desta partição, a ação "não ideal" que antes tinha seu peso reduzido graças ao grau de pertinência menor, passa a acarretar uma saída "ruim". Por isso a atualização da função de valor-Q (no que diz respeito ao retorno e ao valor de $Q(s_{t+1,} a_{t+1})$) deve depender do grau de importância que esta ação tem na saída.

Em outras aplicações, nas quais as modelagens diferem da modelagem tradicional de RL (como a *lookup table*), os autores também ajustaram este parâmetro segundo as necessidades de seus modelos e obtiveram bons resultados. Sutton (1998), para a aplicação do carro da montanha usando o modelo CMAC, define o $\alpha$ como uma constante. Jouffe (1998); também utilizou uma forma adaptativa para $\alpha$, na qual o parâmetro pode crescer ou decrescer segundo sua heurística definida para o aprendizado.

**O valor $Q(s_{t+1}, a_{t+1})$**

Após a execução da ação resultante, o agente passa ao estado $s_{t+1}$ do ambiente. Isso significa que pelo menos duas funções de valores-Q (correspondentes às ações selecionadas $a_{t+1}$ para as bipartições baixo e alto) devem ser consideradas para $Q(s_{t+1}, a_{t+1})$ da equação 23.

A Figura 18 exemplifica esta situação. No estado $s_t$ a célula ativa apresenta duas funções de valor a serem atualizadas: $Q_1$ no conjunto de ações relativo à bipartição baixo e $Q_2$ no conjunto de ações relativo a bipartição alto. Quando o agente passa ao estado seguinte, $s_{t+1}$, após a execução da ação resultante, a célula que é ativada também seleciona duas ações ($a_{t+1}$), cada uma associada a sua função de valor-Q (neste caso, $Q_2$ e $Q_3$). Para a atualização dos valores de $Q_1$ e $Q_2$ relativos ao estado $s_t$, duas propostas foram testadas:

- na primeira proposta o valor $Q(s_{t+1}, a_{t+1})$ do estado $s_{t+1}$ considerado na atualização é aquele que corresponder ao ramo da estrutura que apresentar maior peso na ação de saída; neste caso seria o valor de $Q_2$, se $\alpha_1 > \alpha_2$, ou $Q_3$, se $\alpha_2 > \alpha_1$;
- na segunda, o valor $Q(s_{t+1}, a_{t+1})$ é calculado a partir da soma ponderada de $Q_2$ e $Q_3$ com relação aos graus de pertinência da variável de entrada da célula ($\alpha_1 Q_2 + \alpha_2 Q_3$).

Apesar dos resultados não diferirem significativamente, o segundo método foi o adotado por apresentar resultados, em média, ligeiramente superiores ao primeiro.

O valor-$Q(s_{t+1}, a_{t+1})$ da ação escolhida para o estado $s_{t+1}$, que será usado para atualizar as funções de valores-Q (relativos aos conjuntos baixo e alto) quando o agente está no estado $s_t$ , também considera o peso que cada ação escolhida ($a_t$) no estado $s_t$ teve na ação resultante.

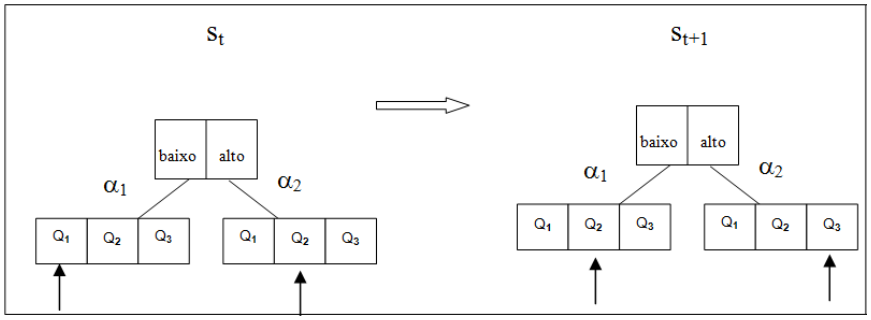


Figura 18: Caso exemplo de atualização da função de valor Qt+1 relativa à equação 19.

Neste passo do algoritmo também é atualizada a taxa de *exploration/exploitation* já mencionada anteriormente, $\epsilon$-greedy. Cada partição deve ter o seu parâmetro $\epsilon$-*greedy* reduzido/aumentado segundo um pequeno percentual (por exemplo, 5% do seu valor). Este procedimento segue as recomendações feitas por Sutton (1996). O parâmetro $\epsilon$-*greedy* não deve ser superior a 10%.

**Segundo Caso: $R_t \geq R_{t+1}$**

Caso o reforço global atual seja menor ou igual ao reforço global anterior, isto significa que as ações atuais não maximizam a função de reforço do estado em que o agente se encontra. Sendo assim, essas ações devem se tornar menos propensas a serem selecionadas nas próximas vezes em que as células, às quais elas pertencem, estiverem ativas.

Se $R_t \geq R_{t+1}$, as ações são punidas, reduzindo suas funções de valores-Q na proporção da contribuição do reforço local desta bipartição da célula e o reforço global. A explicação para esta medida é a mesma já descrita antes. Caso a influência de uma ação boa seja minimizada pelo seu grau de pertinência neste momento, não se deseja que sua função de valor-Q seja drasticamente reduzida, devido à má influência da ação da bipartição irmã. A função de valor-Q é atualizada como na equação 24.

$$Q(a_t, s_t) = (1 - fp) \,.\, Q(a_t, s_t) \qquad (24)$$

Na equação acima, fp é o fator de punição, que varia entre [0,1] e é definido como a relação entre o retorno local da partição e o retorno global.

Caso as ações escolhidas sejam mal sucedidas para um determinado estado, os parâmetros $\epsilon$-*greedy* das partições envolvidas terão suas taxas aumentadas, permitindo que, nas próximas vezes que esta partição estiver ativa, outras ações diferentes tenham mais chances de serem escolhidas. Isso se aplica a qualquer dos métodos de seleção descritos. Desta forma, é feito o aprendizado das ações que serão executadas quando o agente se encontra em um determinado estado. Este estado é definido pelas células que estão ativas a cada passo.

**6 - Particionamento das células**

Com relação ao crescimento da estrutura, algumas avaliações tornam-se necessárias para se garantir o aprendizado. Para isso, foram criados o parâmetro **variável de crescimento** e a **função de crescimento**. Essa variável e esta função têm como objetivo permitir ou limitar o crescimento da estrutura. À medida que as células são atualizadas, verifica-se o percentual de variação da função de valor Q ($\Delta Q$) das ações associadas às partições baixas e altas das células. Quando a variação da função de valor-Q associada à ação atualizada é maior que um percentual da maior variação já ocorrida para esta partição da célula, considera-se que esta partição apresenta potencial de crescimento, ou seja, esta variação pode indicar que as ações que estão sendo tomadas nesta bipartição não estão adequadas para o sub-domínio relativo a esta bipartição. Logo:

Se $\Delta Q > p * \Delta\vartheta$, então a **variável de crescimento** para esta bipartição da célula é incrementada. $\Delta Q$ é a variação percentual da função de valor-Q neste passo; $p \in [0,1]$ é um percentual que representa o percentual de atualização da função de valor-Q em relação a maior variação já ocorrida até o momento; $\Delta\vartheta$ registra a maior variação da função de valor-Q da partição de uma célula ao longo do aprendizado a cada ciclo.

Se $\Delta Q < p * \Delta\vartheta$, então a **variável de crescimento** para esta bipartição da célula é reduzida.

A definição do percentual $p$ está associada diretamente à taxa de crescimento desta estrutura: quanto maior for o valor de $p$, maior variação de $\Delta Q$ será permitida nesta bipartição.

O objetivo deste percentual é permitir que haja alguma variação na atualização da função de valor-Q desta bipartição, mas sem prejudicar o aprendizado. Pequenas variações ocorrem principalmente no início do aprendizado (já que a função de valor-Q inicial é zero) ou quando o processo de exploração escolhe uma ação diferente (com sua função de valor-Q associada) que passa a maximizar o valor de retorno para aquele estado. No entanto, variações muito grandes indicam que a ação tomada nesta bipartição não responde adequadamente ao comportamento desejado (definido através do retorno do ambiente) às exigências deste estado (domínio), ou seja, esta ação não consegue atender ao comportamento desejado para o agente em todo este domínio, indicando que ele deve ser particionado.

Associada à variável critério de aprendizado deve-se definir a função de crescimento, cujo objetivo é limitar o crescimento ao longo do processo de aprendizado. Idealmente, ela deve ser menos exigente com relação às células iniciais do sistema, ou seja, deve permitir que no início do aprendizado as células se multipliquem mais rapidamente e, à medida que o sistema evolui, deve crescer o grau de exigência da função, ou seja, aumentar o efeito de *exploration/exploitation* sobre as ações das células da estrutura. Isso se deve ao fato de que as ações das células criadas no início do aprendizado não são tão efetivas para domínios ainda muito abrangentes.

A cada ciclo avalia-se se a variável de crescimento da bipartição de uma célula tornou-se maior do que a **função de crescimento**. Caso isso seja verdade, então esta bipartição apresenta a primeira condição para gerar uma célula filha.

Esta função de crescimento, definida heuristicamente, pode ser uma constante (como em (Pyatt & Howe, 1998)) ou ser função do número de passos, tamanho da estrutura (profundidade da árvore), ou qualquer outra função relacionada ao processo ou ao objetivo do aprendizado. Nos testes apresentados neste trabalho a função de crescimento é função do número de passos e do número de ciclos, como mostra a Figura 19. O eixo x representa os ciclos e o eixo y representa o valor limite para variável de crescimento.

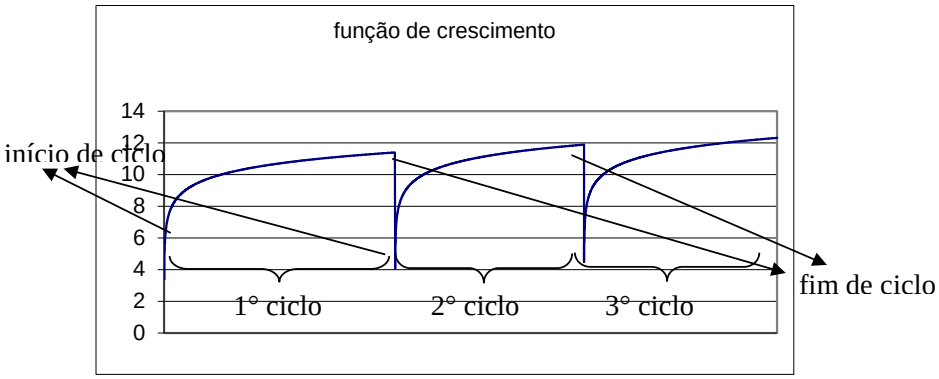


Figura 19: Função de crescimento: (log(n x número de passos x número de ciclos), onde n > 1.

Ao longo dos ciclos, a função de crescimento aumenta o valor que a variável critério de aprendizado deve atingir para que ocorra um novo particionamento. No entanto, a cada novo ciclo, a função de crescimento apresenta valor maior do que o valor do início do ciclo anterior e menor do que o valor do final do ciclo deste ciclo. Isso permite que as novas células criadas a cada ciclo também tenham chances de ser particionadas.

O valor $\Delta Q$ é armazenado em uma lista, a qual será posteriormente avaliada em relação à dispersão destes valores. Este é o segundo critério adotado usado em juntamente com o primeiro critério, para se determinar se deve ou não ocorrer o particionamento de uma célula. Então, o crescimento da estrutura acontece segundo estes dois critérios adotados:

1 – o valor do critério de aprendizado desta partição for maior do que a função de crescimento;

2 – $|\mu| < 2\,\sigma$, onde $\mu$ e $\sigma$ são a média e o desvio padrão de $\Delta Q$.

Caso o aprendizado das ações de cada uma das partições das células não seja efetivo, ou seja, caso a variável de crescimento de qualquer uma das partições (ou mesmo das duas) da célula atinja o valor definido pela função de crescimento, a partição apresenta o primeiro requisito para o particionamento.

O segundo requisito para o particionamento é que o valor da média da variação da função de valor-Q desta partição seja menor em módulo do que duas vezes o seu desvio padrão.

Estes mecanismos também foram usados em outros modelos por outros pesquisadores em sistemas de aprendizado baseados em árvores (Pyatt & Howe, 1998; Uther & Veloso, 1998). Sendo assim, quanto mais rapidamente a bipartição baixa e/ou alta da célula ultrapassar os limites de capacidade de aprendizado (por não conseguir aprender adequadamente), mais rapidamente ela será particionada, de forma a especializar o domínio da célula que não conseguiu atingir seus objetivos.

A função de crescimento apresenta um valor pequeno no início do aprendizado (reduzindo o grau de exigência para realizar o particionamento), quando os conjuntos fuzzy das células ainda são abrangentes, com relação ao domínio, e maior ao final do processo (aumentando o grau de exigência para realizar o particionamento), quando os conjuntos fuzzy já se especializaram o suficiente para garantir o aprendizado.

Quando uma polipartição possuir todos os requisitos necessários para o particionamento, uma célula filha é criada e conectada àquela polipartição. Seu domínio será o subdomínio correspondente à polipartição do seu ancestral. As células filhas também herdam da partição ancestral o conjunto de ações com seus respectivos valores Q.

Na Figura 20, a célula raiz (ou célula pai) possui os domínios definidos pelos intervalos ($x_{1LI}$, $x_{1LS}$) para a entrada $x_1$ e ($x_{2LI}$, $x_{2LS}$) para a entrada $x_2$. A célula filha 1 descende da polipartição referente à composição do conjunto baixo relativo à entrada $x_1$ e ao conjunto baixo da entrada $x_2$ da célula raiz (ou célula pai). Seus domínios são, portanto, diretamente relacionados aos subdomínios da partição baixo/baixo da célula pai (RL-$NFHP_0$) e são definidos por ($x_{1LI}$, $x_{1(LS+LI)/2)}$) relativo à entrada $x_1$ e por ($x_{2LI}$, $x_{2(LS+LI)/2)}$) relativo à entrada $x_2$.

Quando uma célula (raiz ou pai) possuir todas as células descendentes (no caso exemplo da Figura 20 quatro células), todas as ações da célula pai tornam-se as ações de saída das células filhas e neste caso têm suas funções de pertinência anuladas. Ou seja, os graus de pertinência $\rho$ e $\mu$ da célula pai passam a valer 1.

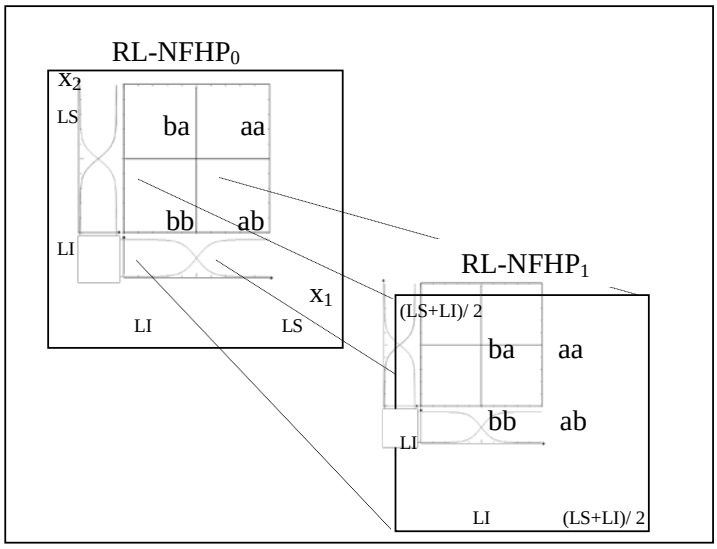


Figura 20: Particionamento da Célula RL-NFP.

Como já mencionado no modelo descrito no capítulo anterior, este procedimento contorna o problema de se ter na saída da estrutura valores de ações muito pequenos e não prejudica o mapeamento (estado-ação), uma vez que o domínio da célula pai já está representado nas células filhas com um grau de precisão maior (ver seção 3.5).

O próximo capítulo apresentará a versão Multiagentes dos algoritmos da família RL-NFH. A principal motivação para o desenvolvimento destes novos modelos é fazer com que cada agente possa explorar diferentes tarefas simultaneamente, acelerando o aprendizado e a convergência para uma política ótima. O capítulo 4 também apresentará os princípios de *satisfatoriedade* e *não-dominação* (Goodrich e Quigley, 2004), que serão incorporados ao modelo RL-NFHP (Figueiredo, 2003), com o objetivo de superar algumas limitações existentes.

## 3.6
## Modelo *Reinforcement Learning* Neuro-Fuzzy Hierárquico para Múltiplos Agentes – RL-NFHP-MA

### 3.6.1
### Introdução

A ideia central da versão Multiagentes dos algoritmos da família RL-NFH é fazer com que cada agente possa explorar diferentes tarefas ou diferentes pares de estado-ação de forma simultânea, acelerando assim o aprendizado e a convergência para uma política ótima.

Conforme descrito no capítulo 2, existem vários tipos de abordagens envolvendo Multiagentes, do ponto de vista do objetivo a ser aprendido, da coordenação entre os agentes e da homogeneidade do aprendizado. No desenvolvimento da proposta do modelo RL-NFHP-MA, buscou-se estender o modelo RL-NFHP de forma flexível para contemplar a maior gama possível de aplicações. Neste capítulo, serão explorados os diferentes tipos de sistemas RL-NFHP-MA criados a partir da versão original. Com este objetivo, os SMA propostos serão apresentados em detalhes, descrevendo seu funcionamento, particularidades e objetivos. Em seguida, para cada modelo proposto, serão exploradas as diferenças no modelo de aprendizado, na coordenação e na dinâmica do aprendizado de maneira detalhada.

Entretanto, antes de apresentar os novos SMA da família RL-NFH, este capítulo apresenta, na próxima seção, os princípios de *Satisfatoriedade* e *Não-Dominação* propostos por Goodrich e Quigley (2004), os quais foram incorporados ao modelo RL-NFHP (Figueiredo, 2003), com o objetivo de melhorar o desempenho do modelo.

## 3.7
## Princípios de *Satisfatoriedade* e *Não-Dominação*

Conforme descrito na seção *3.4.2-Consequentes das Regras do Modelo RL-NFHP*, os consequentes das regras fuzzy nas células RL-NFHP são as ações que devem ser identificadas através de aprendizado por reforço. Quando as células se

tornam ativas, as ações são selecionadas, cada uma relativa à combinação das partições *baixo* e *alto* de cada uma das entradas da célula. A seleção ocorre em função de valores atribuídos a cada uma das ações que pertencem ao conjunto de ações disponíveis para cada polipartição baixa e alta.

A atualização destes valores é realizada através do algoritmo SARSA (Sutton e Barto, 1998), conforme descrito na seção 3.4.2. Uma das principais limitações do algoritmo é a necessidade de explorar um grande número de vezes todos os possíveis pares de estado-ação para garantir a convergência para uma política ótima. Na prática, muitas vezes isso não é viável. Logo, o algoritmo se restringe a convergir com maior probabilidade para valores-Q que correspondam a políticas aceitáveis. Além disso, o método convencional do *Q-learning* também necessita que os valores de recompensa e punição sejam escolhidos de forma que um não mascare o outro, e vice-versa. Geralmente, isso demanda um longo processo de ajustes (Goodrich e Quigley, 2004).

Visando superar estas limitações e dificuldades do algoritmo, este trabalho propõe uma modificação nos consequentes da estrutura RL-NFHP, através do conceito de *satisfatoriedade*. O termo foi usado primeiramente por Simon (1996) no contexto de tomada de decisões para definir ações que sejam “boas o suficiente”. De uma maneira geral, são ações em que a recompensa esperada supera o custo. Obviamente, é necessário que as medidas de recompensa e custo esperados sejam avaliadas através de uma mesma métrica, para que possam ser comparadas.

Um conjunto de ações *satisfatórias* em um estado θ é dado por $S(\theta) = \{a : \mu_A(a, \theta) > \mu_R(a, \theta)\}$, onde $\mu_A$ indica o que é aceitável, ou o grau que uma ação tem de alcançar recompensas, e $\mu_R$ indica o que rejeitável, ou o grau que uma ação tem de alcançar penalidades. Supondo que seja possível aprender $\mu_A$ e $\mu_R$ de maneira independente e que ambos sejam comparáveis, o conjunto de ações satisfatórias S(θ) pode ser definido e usado como base para exploração do ambiente, evitando que escolhas ruins de ações sejam feitas.

Em paralelo, com o objetivo de maximizar o valor de utilidade na seleção das ações, o princípio da **não-dominação** pode ser usado. Uma ação é *dominada* se existir outra ação com uma maior *aceitabilidade* e menor *rejeitabilidade*. Os princípios de *satisfatoriedade* e *não-dominação* podem ser combinados para

refinar o conjunto de ações justificáveis. Ou seja, as ações satisfatórias (recompensa esperada maior que penalidade) e não-dominadas (nenhuma outra ação possui recompensa maior e penalidade menor) compõem o conjunto das ações fortemente satisfatórias, e que devem ser exploradas (Goodrich e Quigley, 2004).

Para viabilizar os princípios acima descritos, a estrutura de reforços deve ser definida de forma que $r(a, \theta) > 0$ indique que o agente atingiu algum objetivo, e $r(a, \theta) < 0$ indique que o agente realizou alguma ação indesejável, onde $r(a, \theta)$ é o reforço da ação a no estado $\theta$. O algoritmo de atualização dos valores-Q deve ser alterado de forma que seja criada uma estrutura separada para as recompensas e penalidades. A estrutura das recompensas, denominada função-G (*Goal*), utiliza a mesma equação do *Q-learning* para atualização, porém, apenas as recompensas afetam sua estimativa. Já, a estrutura das penalidades, função-L (*Loss*), é sensibilizada apenas pelas penalidades, mas também usa a mesma estrutura do *Q-learning*.

$$G(s_t, a_t) = (1 - \alpha_t)\, G(s_t, a_t) + \alpha_t [\, r_{t+1} + \gamma\, G(s_{t+1}, a_{t+1})] \quad (25)$$

$$L(s_t, a_t) = (1 - \alpha_t)\, L(s_t, a_t) + \alpha_t [\, - r_{t+1} + \gamma\, L(s_{t+1}, a_{t+1})] \quad (26)$$

Para calcular $\mu_A$ e $\mu_R$, divide-se o valor de $G(a, \theta)$ ou $L(a, \theta)$ de uma determinada ação em um determinado estado pela soma de todos os G e L de todas as ações no mesmo estado $\theta$.

$$\begin{aligned} \mu_A(a,\theta) &= \frac{G(a,\theta)}{\sum_a G(a,\theta)} \\ \mu_R(a,\theta) &= \frac{L(a,\theta)}{\sum_a L(a,\theta)} \end{aligned} \quad (27)$$

Para um melhor entendimento, a Tabela 2 mostra um exemplo numérico, onde, em um determinado estado, existem 4 ações disponíveis.

**[m1] Comentário:** A Tabela abaixo usa $\mu_a$ e $\mu_r$ ao invés de $\mu_A$ e $\mu_R$. Deixe uniforme, com as letras maiúsculas ou minúsculas em todos os lugares.

Tabela 2: Exemplo da forma de cálculo de $\mu_A$ e $\mu_R$.

| **Ações** | $\mathbf{a_1}$ | $\mathbf{a_2}$ | $\mathbf{a_3}$ | $\mathbf{a_4}$ | $\sum_a$ |
|---|---|---|---|---|---|
| **L** | 0,4 | 0,6 | 0,5 | 0,3 | **1,8** |
| **G** | 0,8 | 0,4 | 0,5 | 0,7 | **2,4** |
| $\mu_a$ | 0,33 | 0,17 | 0,21 | 0,29 | - |
| $\mu_r$ | 0,22 | 0,33 | 0,28 | 0,17 | - |

Dado o exemplo, as ações que podem ser consideradas *satisfatórias* são aquelas em que a o grau de *aceitabilidade* é maior que o grau de *rejeitabilidade* ou $\mu_A > \mu_R$. Neste caso, ações satisfatórias são 1, 3 e 4. Já, as ações *não-dominadas* são aquelas em que nenhuma outra ação possui, ao mesmo tempo, *aceitabilidade* maior e *rejeitabilidade* menor. Neste caso, seriam as ações 1 e 4. As ações 2 e 3 são dominadas tanto pela ação 1, como pela 4.

Com a introdução dos princípios da *satisfatoriedade* e *não-dominação*, os consequentes do modelo RL-NFHP passam a possuir uma nova estrutura, conforme mostrado na Figura 21.

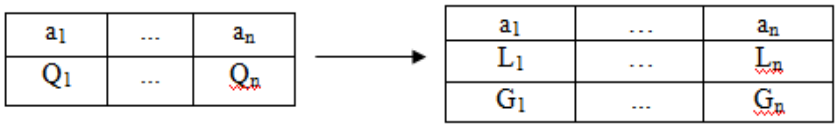


Figura 21: Subdivisão da Célula RL-NFP.

Devido aos benefícios sugeridos nesta seção, os princípios propostos por Goodrich e Quigley (2004) foram incorporados ao modelo RL-NFHP original e estendidos para os modelos RL-NFHP-MA, descritos a seguir.

## 3.8
## Sistemas RL-NFHP-MA

Os sistemas RL-NFHP-MA propostos diferem entre si em dois aspectos principais: a *dinâmica do aprendizado* e o *mecanismo de coordenação*.

### 3.8.1
### Dinâmica do Aprendizado

Em relação à dinâmica de aprendizado dos múltiplos agentes, ela pode ocorrer basicamente de duas maneiras: com ou sem o compartilhamento da estrutura de aprendizado RL-NFH.

Havendo o compartilhamento, a estrutura de aprendizado é única e igual para todos os agentes. Ou seja, cada agente pode executar uma determinada ação em um estado distinto do problema e alimentar a estrutura com o conhecimento

adquirido. Os agentes seguintes acessam a informação “depositada” pelos agentes anteriores na estrutura única, selecionam a ação a ser tomada e atualizam os valores de Q de acordo com o retorno recebido.

Vale ressaltar que este tipo de modelo, onde múltiplos agentes utilizam a mesma estrutura, é indicado para problemas em que os agentes precisam aprender uma mesma tarefa e buscam a cooperação. Se todos competissem utilizando exatamente a mesma “inteligência” não haveria perdedores ou vencedores. Além disso, em um ambiente de competição, não faz sentido um agente fornecer o conhecimento obtido aos demais competidores. A Figura 22 ilustra o funcionamento do aprendizado coletivo dos agentes através do modelo RL-NFHP para Multiagentes.

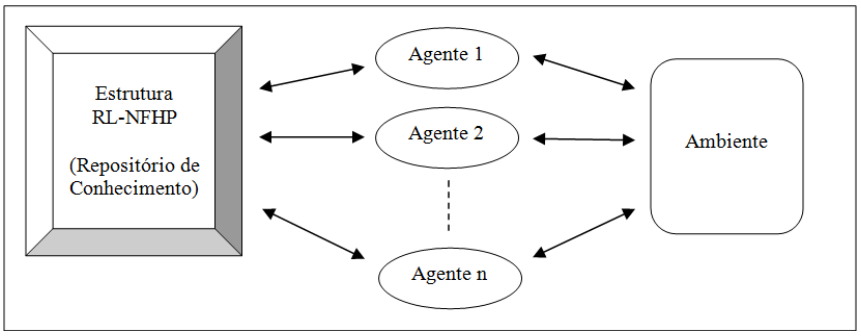


Figura 22: Compartilhamento do conhecimento e exploração do ambiente pelos Multiagentes.

A outra forma de aprendizado é quando cada agente mantém sua própria estrutura, independentemente dos demais agentes presentes no ambiente. Neste caso, cada agente se especializa em uma determinada tarefa, com o objetivo de cooperar ou competir. Esta abordagem visa contemplar problemas que envolvam tanto a cooperação, quanto a competição entre múltiplos agentes.

Para que ocorra a cooperação entre os agentes, deve existir um mecanismo de coordenação, ou até mesmo um agente central, recebendo e combinando as informações de cada estrutura RL-NFHP. A Figura 23 mostra um esquema de sistema onde cada agente mantém sua própria estrutura de aprendizado e um agente central recebe e combina as informações aprendidas.

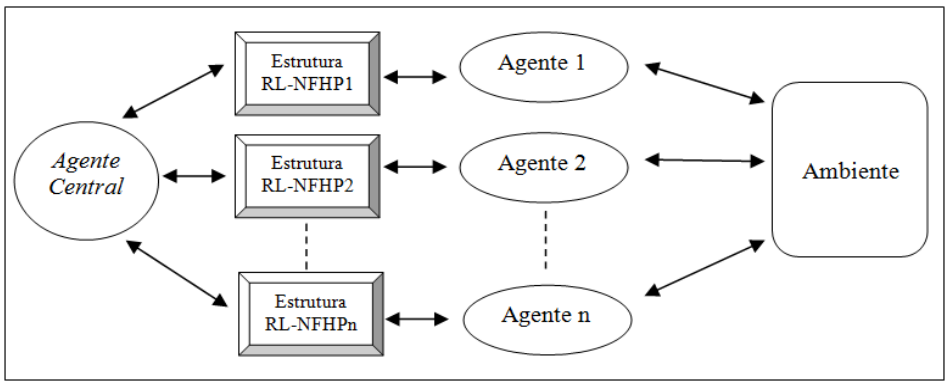


Figura 23: Sistema RL-NFHP-MA em que os agentes possuem diferentes estruturas de aprendizado.

A representação da arquitetura de um sistema RL-NFHP-MA com diferentes estruturas está ilustrada na Figura 24. No exemplo, um “agente central” recebe os valores de saída de dois agentes RL-NFHP com *n* entradas e as combina de acordo com as necessidades da aplicação. O “agente central” pode ser desde uma mera combinação das saídas, como no caso abaixo, até outra estrutura RL-NFHP, que use como entradas as saídas dos demais agentes.

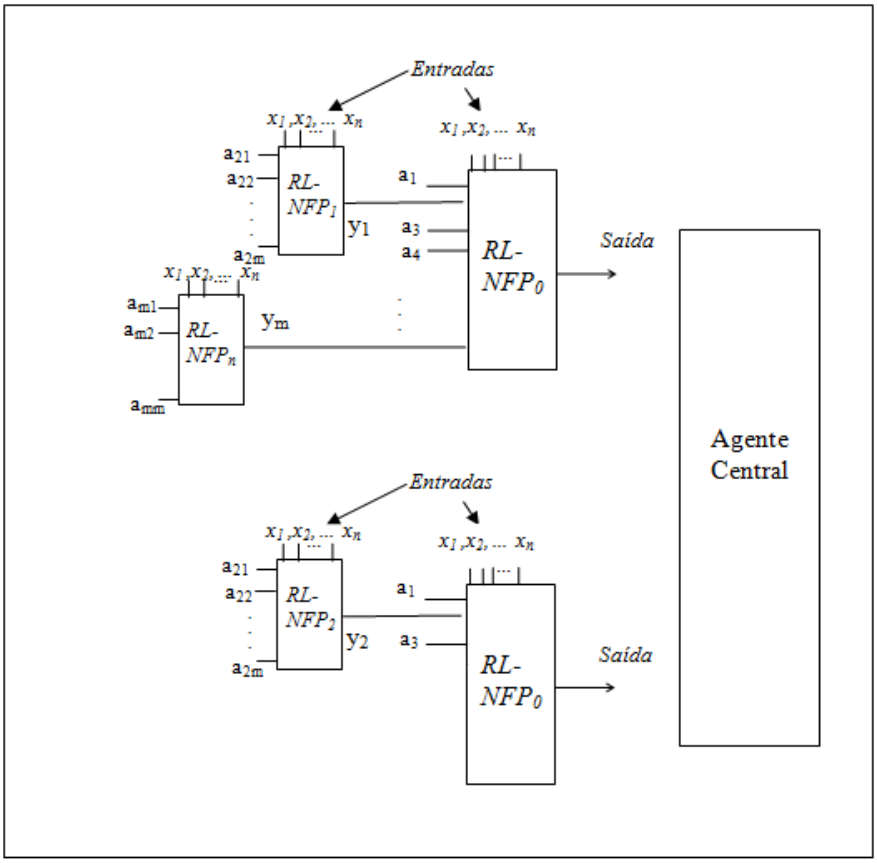


Figura 24: Arquitetura RL-NFHP-MA com diferentes estruturas e um Agente Central.

Nos casos de problemas competitivos, os agentes podem utilizar suas próprias estruturas de forma independente, com o objetivo de aprender uma tarefa específica, e competir uns contra os outros, sendo que cada agente está igualmente "equipado" com uma estrutura de aprendizado RL-NFHP.

O modelo permite também a competição entre um grupo de múltiplos agentes RL-NFHP-MA, coordenados por um agente central, e agentes individuais, ou outros grupos de múltiplos agentes.

## 3.9
## Algoritmo de Aprendizado

Nos casos em que os agentes utilizam estruturas independentes, o algoritmo de aprendizado funciona de maneira semelhante à versão voltada para um agente único. A grande diferença é que a saída obtida de cada modelo pode ser

combinada de diversas maneiras, dependendo do problema em questão. Ou ainda, podem ser utilizadas como entradas para uma nova estrutura RL-NFHP.

Já, nas situações em que os agentes compartilham uma mesma estrutura como repositório de conhecimento, a grande diferença é a alternância dos agentes, que ocorre de forma cíclica na exploração do ambiente. Inicialmente, os agentes acessam a estrutura para tomar uma decisão, exploram os pares de estado-ação do ambiente, recebem o retorno e repassam “o que aprenderam” à estrutura de conhecimento, que processa as informações do par estado-ação explorado e o retorno recebido. Em seguida, um novo ciclo é realizado para outro agente, que pode estar explorando um espaço completamente distinto do ambiente, em relação ao agente anterior.

O algoritmo de aprendizado da estrutura utiliza os mesmos mecanismos do modelo RL-NFHP para a seleção das ações, a atualização dos valores de Q, o particionamento da estrutura, etc. O processo completo de treinamento é dividido em sete etapas, exibidas na Figura 25.

Figura 25

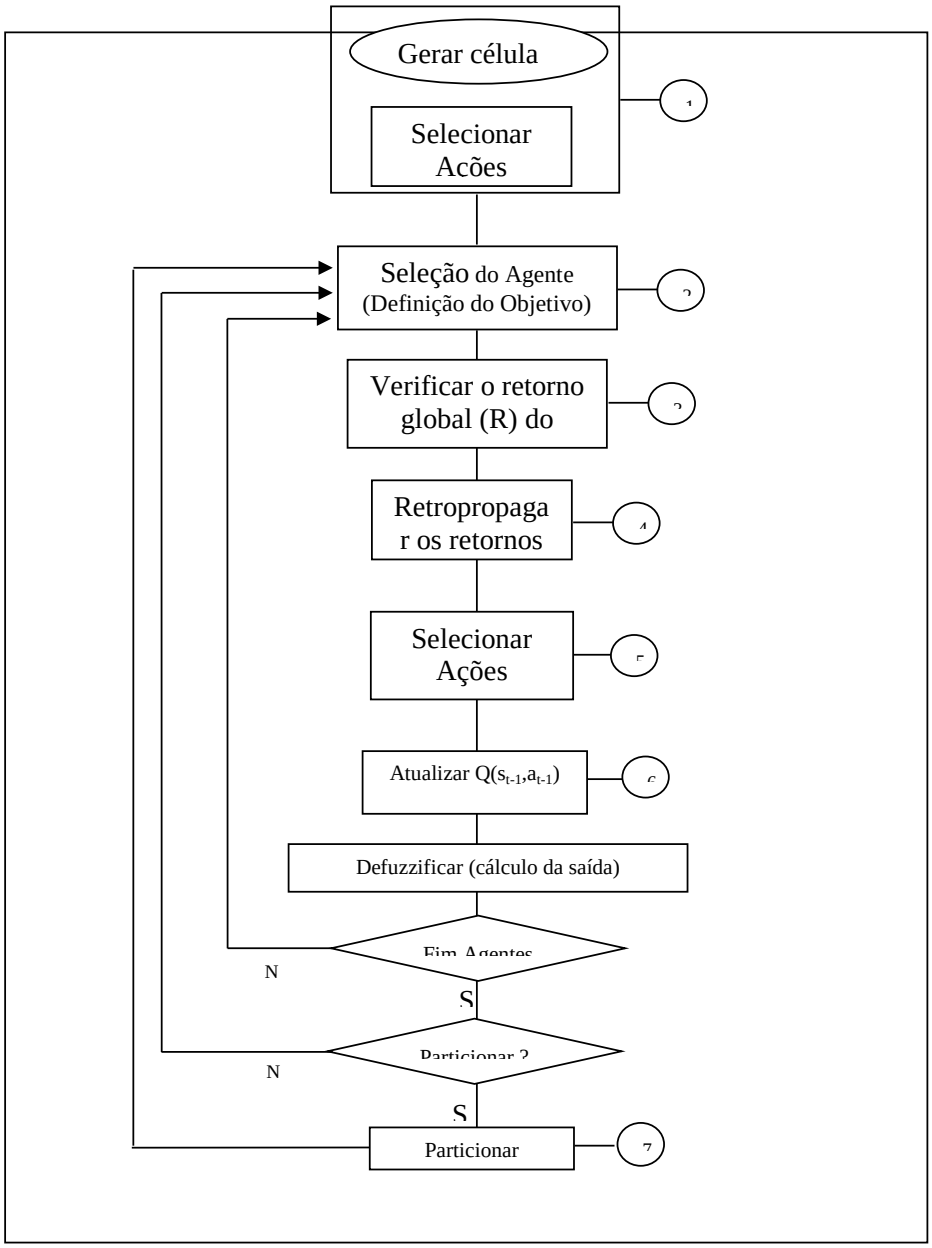


Figura 25: Algoritmo de aprendizado do modelo RL-NFHP-MA.

**1- Inicialização –** ocorre basicamente da mesma forma que na versão para um único agente. Uma célula raiz é criada, tendo como domínios dos seus conjuntos fuzzy, relativos a cada uma das entradas, os valores mínimo e máximo destas entradas (*Limite Inferior* e *Limite Superior*). Com o objetivo de generalidade, os valores das variáveis de entrada são normalizados. Os valores correspondentes às variáveis de entrada da célula são lidos do ambiente,

normalizados e podem ser aplicados diretamente às entradas da célula ou ser modificados segundo uma função que os torne adequados às variáveis das entradas da célula. Estes valores são avaliados nos conjuntos fuzzy *baixo* e *alto*, resultando em dois graus de pertinência, $\rho(x_1),\mu(x_1)$; ... $\rho(x_n),\mu(x_n)$; respectivamente, para cada variável de entrada. Cada uma das polipartições escolhe uma das ações de seu conjunto de ações baseando-se nos métodos descritos na seção 4.5 (passo 5 do algoritmo – Seleção das Ações). A saída da célula é calculada pelo processo de defuzzificação e representa a ação que será executada pelos *atuadores* do agente.

**2- Seleção do Agente –** nesta etapa, um dos agentes no ambiente é selecionado para tomar uma ação e ter seu retorno avaliado. Os agentes são numerados de forma aleatória. A cada ciclo, a seleção é realizada de forma sequencial, sempre do agente 1 ao agente n, para que todos possam interagir igualmente com o ambiente. Na grande maioria dos problemas em que os SMA são aplicados, os agentes possuem um objetivo a ser realizado. Quando o objetivo é alcançado ao longo do processo de aprendizado, os agentes podem ser reinicializados, voltando para um determinado estado de origem, ou para algum estado selecionado de forma aleatória. O aprendizado prossegue normalmente com a movimentação do próximo agente.

Além disso, nesta etapa, o agente selecionado também poderá definir/redefinir seu objetivo durante o aprendizado. Por exemplo, em caso de problemas em que cada agente possui um objetivo distinto, estes objetivos podem ser alterados constantemente ao longo da dinâmica do aprendizado, devido à movimentação dos agentes. Não há uma regra única para a determinação dos objetivos. Ela varia de acordo com o problema ou aplicação que está sendo abordada. Alguns exemplos serão vistos no capítulo 5.

**3- Função de Avaliação/Retorno -** Após a execução de uma ação por um determinado agente, uma nova leitura do ambiente é realizada. Esta leitura permite que seja calculado o valor de reforço do ambiente e se avalie a ação tomada pelo agente. Este valor deve ser calculado através de uma função de

avaliação definida segundo os objetivos do agente, ou do grupo de agentes, sendo fundamental para a orientação do agente ao longo do processo de aprendizado.

A grande diferença em relação à versão original do RL-NFHP é que o reforço de uma ação de um agente passa a depender também da situação dos demais agentes no ambiente (ou do seu objetivo naquele momento do jogo). Uma determinada ação em um estado pode ser boa ou ruim, de acordo com o posicionamento dos demais agentes no jogo. Logo, a função de avaliação, além de fundamental no aprendizado, passa a ter também extrema importância na coordenação dos agentes.

**4- Retropropagar o retorno** – ocorre da mesma forma que na versão para um único agente. Porém, ao invés da retropropagação do retorno ocorrer somente após o final de cada passo do aprendizado, ela ocorre ao final de cada interação de um agente com o ambiente. O retorno é calculado para cada partição de todas as células ativas, mediante a sua participação na ação resultante. Dessa forma, o retorno do ambiente é retropropagado a partir da célula raiz até as células folhas.

**5- Seleção das ações –** é realizada através dos mesmos métodos usados pelo algoritmo RL-NFHP. O primeiro, o $\epsilon$*-greedy* (Sutton, 1998), seleciona a ação associada à maior função de valor-Q esperada com probabilidade (1-$\epsilon$); e com probabilidade $\epsilon$ seleciona aleatoriamente uma ação qualquer.

No segundo método, a ação escolhida com probabilidade (1-$\epsilon$) é também a que apresentar a maior função de valor-Q associada; e com um número de vezes proporcional a $\epsilon$, a seleção é baseada na distribuição de probabilidade dada pelas funções de valores-Q. Esta forma de seleção é mais conservadora, uma vez que as ações que apresentarem as maiores funções de valores-Q terão maiores chances de serem escolhidas. Este tipo de método de seleção de ações é conhecido como *softmax* (Sutton & Barto, 1998).

**6- Atualização dos valores Q** – no caso de múltiplos agentes, para a atualização dos valores de Q devem ser considerados o retorno global atual e o retorno global anterior sempre do mesmo agente. Só assim é possível avaliar sua ação. Para que isso seja viável, os retornos dos agentes devem permanecer

armazenados até que o algoritmo selecione novamente o agente em questão, calcule seu novo retorno, e compare com o seu retorno anterior. Isso evita que retornos de diferentes agentes sejam misturados, causando problemas na atualização dos valores de Q.

Individualmente, cada atualização de valores-Q também ocorre de duas formas distintas: para o caso do valor do retorno global atual ser maior que o retorno global anterior ($R_{t+1} > R_t$) e para o caso do retorno global atual ser menor ou igual ao retorno global anterior ($R_t \geq R_{t+1}$). No primeiro caso, é utilizada a equação do algoritmo *SARSA* (Sutton, 1996), com o objetivo de premiar a ação escolhida. Na segunda situação, as ações escolhidas são punidas, para que se tornem menos propensas a serem selecionadas nas próximas vezes em que as células, às quais elas pertencem, estiverem ativas. A punição ocorre através da mesma equação utilizada pelo modelo RL-NFHP. Além deste método, também foram aplicados os princípios de Não-Dominação e Satisfatoriedade descritos na seção 4.2.

7- **Particionamento das células –** A decisão de particionar ou não, e o particionamento em si, também ocorre exatamente da mesma forma que na versão para um único agente. Porém, este passo ocorre sempre após uma rodada dos passos 2, 3, 4, 5 e 6 para todos os agentes, e não após cada passo de "Atualização de Valores de Q". Ou seja, para cada agente selecionado, o retorno da ação anterior escolhida é avaliado (passo 3), retropropagado (passo 4), uma nova ação é selecionada (passo 5), e os valores de Q são atualizados (passo 6). Somente após a realização destes cinco passos para todos os agentes é que o passo 7 é executado.

## 3.10
## Outras Considerações sobre os Sistemas RL-NFHP-MA

### 3.10.1
### Homogeneidade do Aprendizado

Nos sistemas RL-NFHP-MA, pode-se dizer que a dinâmica do aprendizado é homogênea, já que os agentes aprendem através do mesmo algoritmo. Mas vale

reforçar que, dependendo do tipo de problema, mesmo compartilhando a estrutura de conhecimento, nada impede que os agentes explorem diferentes pares de estados e ações. O potencial da utilização de múltiplos agentes neste tipo de estrutura é justamente a exploração paralela do ambiente, através de diferentes comportamentos.

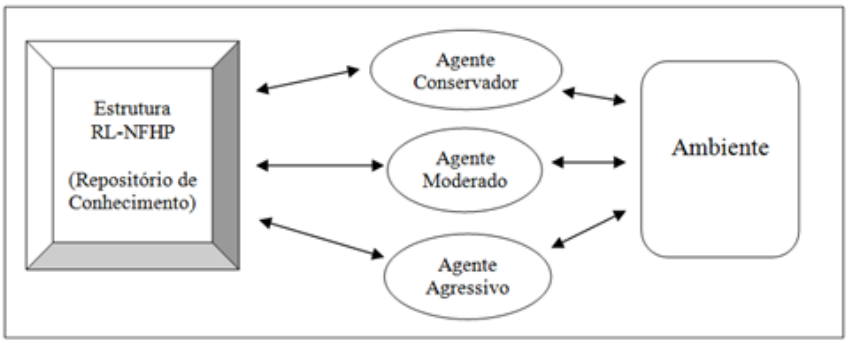


Figura 26: Exemplo de ambiente Multiagente em que agentes possuem comportamentos distintos.

Para definir que estratégias cada agente irá explorar (ou que ações serão escolhidas dado um determinado estado do ambiente), pode haver um agente central controlando os demais agentes e indicando que ações cada um deve realizar. Portanto, mesmo sendo um algoritmo com uma dinâmica de aprendizado homogênea, o RL-NFHP-MA permite que o comportamento dos agentes seja distinto, de acordo com as definições originais do problema.

# 4
# Coordenação de Sistemas Multiagentes

## 4.1
## Introdução

O interesse pela pesquisa sobre sistemas multiagentes (SMA) vem crescendo graças à utilização de múltiplos agentes nos mais diversos campos de aplicações como energia, robótica e militar (Davidson, 2006; Balaji, 2010). Essa pesquisa inclui o estudo de conceitos, linguagens, arquiteturas, teorias e ambientes computacionais, a fim de que seja alcançado um melhor desempenho na solução de problemas em que vários agentes autônomos convivem, cooperam ou competem entre si (Balaji, 2010). Atualmente, uma das características mais estudadas sobre os sistemas multiagentes é a coordenação.

Pode-se dizer que coordenação é "O ato de trabalhar em conjunto de forma harmoniosa no sentido de atingir um acordo e objetivo comum" (Reis, 2007). Trabalhar em conjunto, por sua vez, depende da aplicação ou do problema. A coordenação é importante em ambientes competitivos, cooperativos e mistos. No caso de um ambiente competitivo, a coordenação está centrada na resolução de conflitos e negociação. Já no ambiente colaborativo, a coordenação se concentra na cooperação entre agentes, incluindo maneiras de resolver problemas globais de forma distribuída, criando equipes de agentes e realizando trabalho de grupo para otimizar o cumprimento dos objetivos.

A coordenação em sistemas multiagentes pretende organizar de forma eficiente um comportamento inteligente de um conjunto de agentes autônomos. A ideia é definir estratégias ou formas de coordenação das diferentes habilidades e comportamentos dos agentes para que possam, em conjunto, realizar ações e resolver problemas de forma eficiente. A coordenação em SMA envolve a coordenação entre os agentes e a coordenação das ações de um agente.

Muitos problemas podem ocorrer em um sistema multiagente que não possua um mecanismo de coordenação ou não tenha um que seja adequado, tais como:

- Conflitos de recursos ou uso de recursos do sistema sem necessidade;
- Redundância na realização das tarefas pelos agentes;
- Aumento do tempo de espera, quando a atividade de um agente depende da realização de atividades de outros agentes.

Desta forma, a principal finalidade da coordenação é tentar evitar ou minimizar esses problemas, além de otimizar os recursos e o tempo para a realização de tarefas complexas, em que um único agente não possui todos os recursos necessários para completar a tarefa ou precisa cumprir vários subobjetivos para alcançar o objetivo global (Beaumont, 2007) (Liang, 2012).

A determinação do melhor método de coordenação envolve alguns critérios, como a especificação do que é importante e válido avaliar e a definição de parâmetros para realizar a avaliação. A resolução desses itens depende, basicamente, do conhecimento acerca do ambiente em que será inserido o sistema multiagente. Este conhecimento na maioria de casos práticos tem que ser proporcionado por um especialista na área da aplicação.

Na maioria dos casos práticos, é difícil perceber uma boa estratégia de coordenação, sendo mais fácil notar a ausência de coordenação ou uma coordenação precária como, por exemplo, nos atrasos em aeroportos por questões logísticas, passes errados em uma equipe de futebol ou jogadores mal posicionados no campo. A Figura 27(a) apresenta um exemplo de um ambiente sem coordenação em que uma esquina sem sinal (semáforo) torna o trânsito no local caótico. Já a Figura 27(b) mostra um conjunto de aviões não tripulados com formação coordenada que permite economizar energia e atingir maior campo de visão.

Figura 27: (a) - Sistema sem coordenação (b) Sistema com coordenação.

## 4.2
## Importância e Necessidade da Coordenação de SMA

A coordenação é um aspecto que influencia as atividades dos agentes que atuam em sistemas multiagentes. Ela tem sido abordada em diversas aplicações, pois, quando bem aplicada, contribui para uma execução eficiente das atividades pelos agentes. Deste modo, torna-se um ponto importante a ser analisado em uma sociedade de agentes.

A coordenação faz-se necessária, pois existem relações de dependência entre os agentes, ou seja, um agente precisa do outro para cumprir seus objetivos (Meng, 2007).

Existem três razões principais para coordenar as ações em SMA:

- Existência de dependências nas ações dos agentes: ocorre quando as ações necessárias para atingir os objetivos dos agentes individuais estão relacionadas.
- Necessidade de que o conjunto de agentes respeite restrições globais: as restrições podem ser por custo, tempo ou recurso. Se os agentes atuarem individualmente, eles não conseguem respeitar essas restrições.
- Incapacidade de um agente sozinho executar a tarefa ou resolver o problema completo em função da ausência de recursos ou informações.

Grande parte das aplicações de SMA necessita de conhecimentos distintos para ser resolvida e tais conhecimentos só podem ser obtidos com diferentes agentes. Além disso, para que o resultado desejado seja alcançado, o conhecimento dos agentes precisa ser combinado. Esses agentes também podem apresentar recursos distintos (capacidade de processamento, memória, etc.), que precisam ser utilizados de forma coordenada para resolver o problema. Além disso, os agentes podem ter acesso a diferentes informações, uma vez que podem utilizar sensores diferentes ou estar geograficamente separados. Podem ainda ter capacidades diferentes de posicionamento, o que significa que podem se posicionar em zonas distintas e, consequentemente, ter uma percepção diferente do ambiente.

As dependências entre agentes podem aumentar em consequência do número de atividades sendo executadas no mesmo ambiente e da quantidade de recursos compartilhados. A coordenação gerencia as dependências entre atividades e como os recursos do ambiente são compartilhados (Reis, 2007). Esta gerência traz grandes vantagens para os SMAs.



### 4.2.1
### Vantagens da Coordenação de SMA

Uma coordenação eficaz entre agentes autônomos, que operam em um ambiente multiagente, contribui para o aumento da qualidade das soluções alcançadas pelos agentes e para o aumento no desempenho da atuação dos agentes na resolução de tarefas.

A existência de coordenação torna-se vantajosa até em aplicações onde os agentes trabalham de forma independente, já que pode aumentar a eficiência do sistema. Pode-se, portanto, destacar as duas vantagens mais relevantes da coordenação:

- *Eficiência*: através da troca de informação ou divisão de tarefas, a coordenação pode aumentar a eficiência do sistema. Mesmo um agente sendo capaz de executar uma tarefa que lhe é atribuída, outro agente poderá colaborar na sua realização para executar de forma mais eficiente a nova tarefa.

- *Prevenção da redundância e caos:* a coordenação, por ter um planejamento, evita que agentes obtenham resultados já produzidos por outros agentes, ou façam tarefas já realizadas utilizando recursos do sistema de maneira ineficiente.

As vantagens podem ser maiores ou menores dependendo das características do mecanismo de coordenação utilizado, conforme discutido a seguir.

### 4.2.2
### Características da Coordenação de Ações e Agentes

Há vários tipos de características que podem ser avaliadas em sistemas coordenados e são reunidas em quatro grupos, a saber: temporais, organizacionais, e de realização (Reis, 2007).

#### 4.2.2.1
#### Características Temporais

Estas características envolvem os conceitos de:

- *Rapidez*: capacidade de um sistema agir prontamente a um evento (previsto ou imprevisto). O tempo de reação de um sistema pode depender de vários parâmetros, como a complexidade de cálculos, o número de agentes envolvidos, a quantidade de tarefas e outros;

- *Adaptabilidade*: capacidade do sistema de se adaptar a eventos ou a situações inesperados. Quanto mais um sistema for adaptável, mais poderá ser utilizado em ambientes complexos;

- *Preditividade*: sistema capaz de determinar, com certo grau de precisão, o estado futuro do ambiente e dos agentes. Os sistemas preditivos são eficazes em contextos que dispõem de muitas informações sobre eventos futuros.

#### 4.2.2.2
#### Características Organizacionais

Estas características tratam da maneira como é organizada a coordenação de ações e envolvem:

- *Local da Decisão*: A estrutura organizacional das ações pode ser centralizada ou distribuída. As organizações centralizadas são de simples utilização. As organizações distribuídas são mais facilmente adaptáveis a modificações não previstas no ambiente;
- *Forma de Comunicação entre os Agentes*: O modo de comunicação define a forma na qual os agentes tomam conhecimento das ações e posição dos outros agentes.
- *Grau de Liberdade de Ação do Agente*: caracteriza a independência do comportamento deste agente. Quanto mais liberdade o agente possuir para agir, mais adaptável ele será a novas situações.

#### 4.2.2.3
#### Características de Realização

Estas características referem-se aos meios necessários para se obter um sistema de coordenação, que são:

- A quantidade de informação que os agentes devem manipular e trocar para coordenar suas ações. Estas trocas de informações podem ocorrer durante a elaboração de planos de ação ou no curso da execução das ações;
- Informações sobre o ambiente e sobre os outros agentes, quanto mais informações o agente possuir sobre o ambiente e seus elementos, melhor será a previsão de evolução do sistema, que responderá de maneira adaptada às diferentes ações dos agentes;
- Diminuição das dificuldades, existem mecanismos de coordenação de difícil utilização e implementação.

## 4.3
## Mecanismos de Coordenação

A coordenação envolve a solução dos problemas de ordenação das atividades e a determinação de como estas serão executadas pelos vários agentes. Desta forma, a importância de uma coordenação eficaz e a correspondente necessidade de criação de mecanismos de coordenação sofisticados aumenta em situações onde há interdependência entre as atividades dos agentes. Pode-se citar como exemplo, agentes que trabalham sobre um mesmo problema onde existem maneiras alternativas que podem ser usadas para gerar uma solução. Neste caso, a coordenação envolve uma tomada de decisão em relação a se os agentes devem trabalhar ou não de forma concorrente e à alocação de recursos necessários para a execução das alternativas (Sabanovic, 2007).

Quando se quer aplicar coordenação em um SMA, é necessário levar em consideração dois aspectos. O primeiro é quanto esforço deve ser despendido para que seja encontrada uma sequência satisfatória de ações a ser executada. O segundo é determinar o escopo das decisões, ou seja, quais ações necessitam ser coordenadas e quais agentes devem executá-las.

Um mecanismo de coordenação é composto por três elementos, como mostrado na Figura 4 (Reis, 2007):

- **Agentes Coordenáveis:** define os agentes ou entidades cujas interações serão regidas pelo modelo de coordenação.
- **Meios de Coordenação:** específica os meios utilizados que permitem as interações entre os agentes, ou seja, os tipos de comunicação.
- **Leis de coordenação:** determina as leis (política, algoritmo) que definem a maneira de utilização dos meios de coordenação nos momentos de interação. Estas leis são passadas aos agentes através de uma linguagem de comunicação.


Figura 28: Mecanismo de coordenação.

Os mecanismos de coordenação dependem do fato de os agentes serem cooperativos ou competitivos. A seção a seguir apresenta, de forma resumida, os mecanismos de coordenação competitivos. Em seguida, apresenta-se em mais detalhe os mecanismos de coordenação cooperativos, foco principal deste trabalho de tese.

### 4.3.1
### Mecanismos de Coordenação para agentes competitivos

Os agentes competitivos agem considerando apenas seus próprios objetivos. Neste caso, os mecanismos de coordenação mais usados são os baseados em negociação para resolução de conflitos (Sabanovic, 2007). A negociação é um mecanismo de coordenação baseado em um processo de geração de acordos dentro de um grupo de agentes.

A negociação tem como principais objetivos:

- Permitir a modificação dos planos dos agentes;
- Efetuar a alocação de tarefas e recursos;
- Resolver conflitos e diferenças nos objetivos de agentes;
- Determinar a organização estrutural numa dada situação;

### 4.3.2
### Mecanismos de Coordenação de agentes cooperativos

O objetivo da coordenação em agentes cooperativos é otimizar o desempenho de um sistema multiagente, distribuindo as tarefas entre os agentes de maneira organizada e com uma estratégia específica dependendo do estado do ambiente (Srinivasan, 2008) (Balaji, 2010) (Olfati-Saber, 2006).

Sem coordenação os benefícios de distribuição de tarefas entre agentes desaparecem e o sistema multiagente pode degenerar em uma coleção caótica e incoerente de agentes individuais (Reis, 2007). No entanto, existem cenários de coordenação cooperativos onde os agentes têm objetivos diferentes mais precisam de regras de convivência para cumprir estes objetivos. Um exemplo clássico são os sinais de trânsito. Nesse caso, as pessoas (pedestres e motoristas) precisam conhecer e cumprir as regras dos sinais de forma a evitar o caos.

Os SMA devem ser capazes de suportar formas de coordenação, mas esta tarefa não é trivial. A dificuldade de criar um mecanismo de coordenação de agentes autônomos, em um ambiente multiagente, está na definição das ações individuais. Essas ações devem ser realizadas por cada um dos agentes envolvidos, de maneira que o conjunto de agentes atinja uma situação final coerente e eficiente. Por isso, os mecanismos de coordenação devem abranger todos os tipos de interações entre os agentes.

Os mecanismos de coordenação de agentes cooperativos têm duas características: a organização e o planejamento (Reis, 2007).

- *Organização*: organização em sistema multiagente é vista como um conjunto de relacionamentos estruturais entre as funções dos agentes na sociedade. Há muitas propostas de classificação de estruturas organizacionais, dependendo da forma como as ações são atribuídas aos agentes. Em geral, há algum conhecimento dentro de uma organização referente a tarefas e subtarefas. Ser responsável por uma tarefa composta significa gerenciar a coordenação de suas subtarefas por meio de ordens dadas aos agentes que as estão realizando. Por exemplo, em uma organização do tipo hierárquica, há um agente gerente que controla e coordena a realização de cada tarefa; já, em

uma organização do tipo lateral, não há agente gerente e as tarefas são realizadas de modo cooperativo (Reis, 2007).

- *Planejamento*: Os agentes formam planos que especificam suas ações e interações futuras, em relação a um objetivo. Neste contexto, todos os agentes envolvidos em um plano multiagente comprometem-se a comportar-se conforme este plano. Na abordagem tradicional de planejamento multiagente, os planos completos e detalhados são gerados para todos os agentes: ações e interações para serem realizadas durante o plano são especificadas antes que a execução comece. São considerados dois tipos de planejamento: centralizado (um único agente elabora o plano e distribui as atividades entre os demais agentes) e distribuído (vários agentes podem participar da elaboração de planos).

Existem vários mecanismos de coordenação para agentes cooperativos, sendo que, neste trabalho, adotou-se a classificação de Ferber (Goodwine, 2010), que descreve quatro mecanismos de coordenação básicos: sincronização, planejamento, reatividade e regulamentação. Cada um desses mecanismos é utilizado como base para mecanismos mais complexos, como os mecanismos de coordenação por troca de papéis e estratégico que são utilizados neste trabalho. Estes mecanismos serão explicados a seguir.



#### 4.3.2.1
#### Coordenação por Sincronização

Trata-se de uma forma elementar de coordenação. Toda coordenação deve descrever precisamente a sequência de ações que terá necessidade de uma sincronização para a sua execução. A sincronização pode gerar uma simultaneidade de várias ações e verificar se os resultados das operações são coerentes (Goodwine, 2010).

Sincronizar várias ações é definir a forma de encadeamento dessas ações para que sejam realizadas em uma determinada situação. Se há o objetivo de assegurar certa coerência no sistema e impedir que o resultado das ações de uns

agentes prejudique as dos outros, a sincronização também é relevante. A sincronização constitui o "baixo-nível" da coordenação de ações, onde são implementados os mecanismos de base que permitem às diferentes ações se articularem corretamente.

#### 4.3.2.2
#### Coordenação por Planejamento

Neste tipo de planejamento é necessário ter em consideração que as atividades dos diferentes agentes presentes no sistema podem interferir umas com as outras. Desta forma, é necessário coordenar tais atividades (Qingguo, 2003).

A coordenação por planejamento baseia-se em duas fases: determinação do conjunto de ações a serem realizadas para atingir um objetivo, produzindo um conjunto de planos; e escolha de um dos planos gerados para ser executado. Os planos escolhidos podem ser revisados durante a sua execução. Em sistemas multiagentes, os diferentes planos elaborados pelos agentes podem ocasionar conflitos de objetivos ou de acesso a recursos. Portanto, os planos devem ser coordenados de forma a resolver estes conflitos e satisfazer os objetivos dos agentes. Um plano organiza uma coleção de ações que podem ser executadas sequencial ou concorrentemente.

#### 4.3.2.3
#### Coordenação Reativa

É um mecanismo de coordenação baseado em agentes reativos. De uma maneira geral, este mecanismo utiliza a relação percepção-ação de um agente e a capacidade dos agentes reativos de reagirem às modificações do ambiente. Considera que é mais simples agir diretamente, sem planejar o que deve ser feito.

Nesta coordenação não há representação explícita do conhecimento, o conhecimento dos agentes é implícito (as suas regras de comportamento) e sua manifestação se externa através do seu comportamento e dos demais agentes.

As informações disponíveis, relativas a seus comportamentos e aos dos outros agentes, encontram-se no ambiente. Suas ações dependem unicamente da

percepção que possuem do ambiente. O problema de determinar "qual agente faz o quê e quando" é baseado na utilização da coordenação reativa.

Assim, a ação de um agente reativo consiste, então, na reação deste agente a modificações que ocorrem em seu ambiente local e a adaptação de suas ações as ações dos outros agentes.

#### 4.3.2.4
#### Coordenação por Regulamentação

Trata-se de um mecanismo posto em prática em sistemas que necessitam de uma coordenação limitada. Baseia-se em leis ou convenções sociais utilizadas para assegurar uma coordenação imediata. Sistemas com estas características são chamados de sistemas normativos. O princípio deste método é a utilização de regras de comportamento que visam eliminar possíveis conflitos. Por exemplo, atribuir regras de prioridade a veículos em um cruzamento, com o objetivo de evitar uma colisão (conflito).

Os agentes possuem conhecimento sobre as leis utilizadas em um ambiente multiagente e podem escolher se as obedecem ou não conforme a situação. Desta forma, as leis podem influenciar o comportamento dos agentes.

Em um ambiente computacional, as leis sociais garantem o sucesso da coexistência de múltiplos agentes (Shoham, 2003). Elas permitem um equilíbrio entre a liberdade dos agentes para atingirem seus objetivos e as restrições para que eles não interfiram demais uns nos outros.

#### 4.3.2.5
#### Coordenação por troca de papéis

A coordenação por troca de papéis parte do conceito de coordenação por decomposição, onde uma tarefa maior é decomposta em tarefas menores e estas são alocadas aos agentes. Baseia-se igualmente no princípio de efetuar a decisão de quem faz o quê. No entanto, é realizada em um nível bem mais elevado. Em vez de alocar uma ou mais tarefas a cada agente, se alocam papéis aos agentes. Esses papéis definem completamente o comportamento individual de cada agente.

Além da definição de papéis para cada um dos agentes que compõem a equipe, é útil realizar a troca de papéis entre esses agentes em determinadas circunstâncias.

Necessariamente, a troca de papéis entre dois agentes $A_i$ e $A_j$ deverá ser se realizada, num determinado instante (Reis, 2007):

- Um determinado agente $A_i$ tem atribuído o papel $P_i$ e um agente $A_j$ tem atribuído um papel $P_j$.
- O agente $A_i$ é capaz de desempenhar o papel $P_j$ melhor do que o agente $A_j$;
- O agente $A_j$ é capaz de desempenhar o papel $P_i$ melhor do que o agente $A_i$.

Esta estratégia de troca de papéis é integrada em vários modelos de coordenação (Reis, 2007), uma vez que tira proveito das condições atuais dos agentes dentro do ambiente para cumprir, de forma mais eficiente, uma determinada função. Os dois modelos propostos neste trabalho baseiam-se em coordenação por troca de papéis. Ele foi escolhido por ter uma estrutura especializada em aplicações multiobjetivos, onde os agentes precisam cumprir diferentes tarefas dependendo da situação. Existem modelos baseados em coordenação de troca de papéis para uma aplicação específica, como coordenação estratégica, que será explicado a seguir.



#### 4.3.2.6
#### Coordenação Estratégica

A coordenação estratégica é baseada em conhecimento a *priori* do ambiente ou cenário (Debenham, 2001). Baseia-se na definição de uma estratégia, que é composta por sua vez der um conjunto de táticas. Estas táticas implicam diferentes formas de atuação da equipe como um todo. Cada tática utiliza um conjunto de formações. Dependendo da situação global, a formação mais adequada é selecionada em cada instante e cada formação atribui a cada agente um papel (Reis, 2007). Esta coordenação é bastante complexa e utiliza elementos de várias metodologias de coordenação explicadas anteriormente, ela se inspira no futebol e é aplicada em ambientes dinâmicos e hierárquicos.

Para a aplicação da coordenação estratégica é necessário ter (Reis, 2007):

- A criação de uma biblioteca de táticas elaborada por um especialista;
- O conceito de situação e a definição de situações facilmente identificáveis pelos agentes que permitam efetuar a troca dinâmica de táticas e formações;
- O conceito de papel como forma de definição do comportamento individual de cada agente como parte de uma equipe;
- A definição de papéis adequados à tarefa cooperativa a realizar, e de mecanismos que permitam aos agentes uma troca dinâmica de papéis.
- A definição de mecanismos de posicionamento flexíveis, dinâmicos e adequados ao domínio.

A coordenação estratégica faz sentido em cenários onde os agentes devam possuir uma distribuição espacial, tais como resgate, salvamento, cenários de guerra e o tema de interesse desta tese, que é o futebol robótico. Estes cenários precisam de respostas rápidas, por isso os modelos têm que mapear o ambiente o mais resumido possível.

Assim, o objetivo da coordenação estratégica é ter um esquema para poder coordenar diferentes aspectos de um sistema dinâmico de alta complexidade (Debenham, 2001).

## 4.4
## Mecanismos de Coordenação Cooperativa Multiagente Para Ambientes Complexos

Os ambientes complexos são multiobjetivos e têm vários condicionamentos; como a grande maioria de aplicações dos SMA é nesta classe de ambientes, por exemplo, futebol de robôs, resgate e aplicações militares (Reis, 2007). As abordagens de coordenação de SMA para ambientes complexos são mais elaborados ou tentando tirar maior proveito do ambiente, para economizar recursos e se adaptar as mudanças (Dias, 2001). Dois dos mecanismos de coordenação mais utilizados e com melhores resultados práticos nos SMA são: o

mecanismo de coordenação *Market Driven* que é inspirado em economia de mercados (Kose, 2004) (Chun, 2010) e coordenação por grafos (Vlasis, 2006).

Estes dois modelos de coordenação são integrados ao modelo NFHP-RL, os dois mecanismos têm diversos elementos dos mecanismos de coordenação básicos e cada um têm características diferentes.

O mecanismo *Market Driven* baseia sua coordenação em escolher a ação com menos custo individual para cada agente, partindo do princípio que, se um agente de um sistema cooperativo consegue um ganho alto para ele, este ganho se verá refletido em todo o sistema (Chun, 2010) (Kose, 2004).

O mecanismo de coordenação por grafos tenta escolher a ação individual de cada agente que maximize o retorno conjunto do sistema (Vlasis, 2006). Estes dois mecanismos serão explicados nas próximas seções.



### 4.4.1
### Coordenação Cooperativa *Market-Driven*

Existem vários modelos de SMA que têm suas estratégias de coordenação inspiradas no livre mercado (Dias, 2001; Kose, 2004; Chun, 2010) com bons resultados. Nessas estratégias os agentes são individualistas dentro do domínio de suas capacidades ou dentro do papel que estão desempenhando em um dado momento. As abordagens econômicas de mercado para coordenação multiagente partem do pressuposto que uma economia é essencialmente uma população de agentes produzindo uma saída global (Dias, 2001). Os agentes interagem entre si para produzir um conjunto agregado de bens.

Os custos e ganhos ditarão o desempenho de uma abordagem baseada no mercado. Nestas abordagens usa-se uma função ganho ($f_g$), a qual fornece um valor aproximado de ganho segundo os resultados das tarefas realizadas, e uma função de custo ($f_c$), que mapeia os possíveis custos de realizar uma tarefa. Dessa forma, o objetivo dessa abordagem é executar um plano ($p$) tal que o lucro, definido como a diferença entre o ganho e o de se realizar certo plano $p$ ($f_{g(p)}$ - $f_{c(p)}$), seja maximizado. Essas funções podem ser simples ou muito complexas, dependendo do domínio da aplicação.

Em modelos econômicos aparece o conceito de leilão. O leilão é um mecanismo baseado em modelos econômicos de negociação definido por uma

série de regras para especificar a forma de determinação do vencedor e o quanto este deve pagar. O leiloeiro é o encarregado de dar o preço e autorizar a venda. Uma característica marcante dos leilões é a presença de assimetria de informações, o que faz com que a caracterização deste mecanismo se torne necessária, uma vez que diferentes tipos de leilões podem levar a resultados divergentes (Kose, 2004). Para um SMA este mecanismo é responsável por atribuir tarefas aos agentes em uma situação específica. A Figura do leiloeiro pode ser interpretada por algum agente do sistema ou por uma entidade central

Esta abordagem considera um grupo de agentes cujo objetivo é completar as tarefas com sucesso, tentando minimizar os custos totais, uma vez que cada agente terá como objetivo minimizar seu custo individual e maximizar seu lucro individual. A ideia do método orientado para sistemas multiagentes é baseada na interação dos agentes de uma forma distribuída para ter maior poder de negociação e informação. Dessa forma, os agentes devem ter um mecanismo de comunicação entre eles.

No momento em que o SMA tem um objetivo, este é descomposto em tarefas menores. Logo, um leilão é realizado para cada uma dessas tarefas e é associado um ganho (reforço) por essa tarefa. Em cada leilão os agentes participantes calculam o seu custo estimado para realizar a tarefa e oferecem um preço para o leiloeiro (Kose, 2004; Kose 2005).

No final do leilão, o licitante (agente) com o menor preço ofertado será dado o direito da execução da tarefa e receberá um ganho. Um agente pode abrir outro leilão para a venda de uma tarefa que ele ganhou. Dois ou mais agentes podem trabalhar em conjunto para realizar uma tarefa que seja difícil de ser completada por um único agente. Para desenvolver a estratégia *Market-Driven*, uma função de custo deve ser definida para todos os recursos necessários para realizar a tarefa e se transformar em um número real. Cada um destes recursos pode ter um peso diferente dependendo da importância de cada um para a realização da tarefa.

### 4.4.2
### Coordenação Cooperativa Baseada em Grafos

Esta seção apresenta outro sistema de coordenação inspirado na teoria dos grafos. Esta coordenação utiliza grafos para relacionar os agentes que coordenam

suas ações em uma situação específica. Os grafos são uma ferramenta gráfica útil para representar problemas em diversas áreas do conhecimento e foram utilizados para a coordenação de sistemas Multiagentes por Vlasis (Vlasis, 2006). Na maioria das aplicações, é rara a ocorrência de agentes com ações interdependentes, isto é, em poucas situações as ações de um agente necessitam que ações de outros agentes tenham sido previamente realizadas. O problema de coordenação global é agora substituído por problemas de coordenação locais com menos agentes. Esta decomposição pode ser representada por grafos, onde o nó é um agente e uma aresta indica que dois agentes devem coordenar as suas ações. Esta técnica é chamada de Coordenação por Grafos (CG) (Vlasis, 2006). Cada dependência corresponde a uma função local, que atribui um valor específico para cada uma das diferentes combinações de ações dos agentes envolvidos. Na Figura 29 é apresentada uma estrutura de CG para oito agentes (círculos numerados), onde é especificada, para cada ligação, uma função de dependência entre os agentes. Por exemplo, a aresta entre o agente 1 e o 2 é representado por $f_{12}$, que é uma função que contem todas as possíveis ações do agente um com o agente dois.

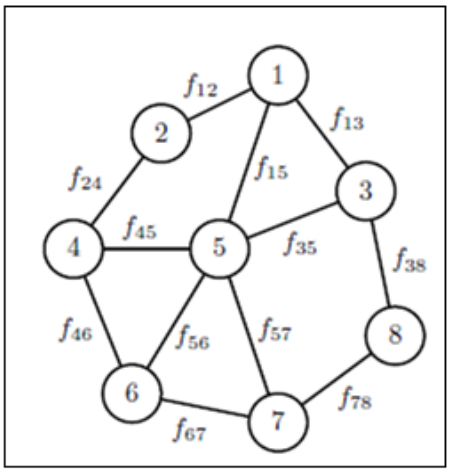


Figura 29: Coordenação por grafos para oito agentes.

A função global é igual à soma de todas as funções locais. Para calcular a ação conjunta que maximiza a função global, o algoritmo de eliminação de variável pode ser utilizado (Vlasis, 2006), o qual é apresentando brevemente na próxima seção.

## 4.5
## Eliminação de Variável

A representação gráfica das dependências das ações dos agentes é realizada através da coordenação por grafos, mas, para encontrar as ações individuais que tornem o desempenho do sistema ótimo, é utilizado o método de Eliminação de Variável (Vlasis, 2006). Este método descreve matematicamente por uma função as relações mostradas na coordenação por grafos.

A ideia do método é calcular uma ação coordenada que envolva um grupo de *n* agentes com o propósito de maximizar uma função de retorno. O problema pode ser descrito como: cada agente *i* seleciona uma ação individual $a_i$ a partir de um conjunto de ações $A_i$ e a ação conjunta resultante $a= (a_1,\dots a_n)$ gera uma função de retorno *R(a)*. O problema é encontrar a ação ótima que maximize a função de retorno $a^* = argmax_a R(a)$.

A abordagem aparentemente simples para resolver o problema da coordenação é enumerar todas as possíveis ações conjuntas e escolher aquela que maximiza *R(a).* Na maioria das aplicações reais isso se torna impraticável computacionalmente, entretanto, em diversos problemas a ação de um agente $i$ depende unicamente de um grupo pequeno de agentes $\Upsilon(i)$ e não de todos os agentes. Deste modo, pode-se decompor a função global de retornos *R(a)* em uma combinação linear de funções locais de retorno, como apresentado na equação 28.

$$R(a) = \sum_{i=1}^{n} f_{ij}(a_i, a_j) \tag{28}$$

A representação por grafos foi escolhida para facilitar a modelagem do problema, além de permitir representar qualquer dependência (Vlasis, 2006). Para resolver o problema da coordenação, que é encontrar a melhor ação conjunta $a^* = argmax_a R(a)$, se utiliza a estratégia de eliminação de variável (Vlasis, 2006).

Quando um agente é selecionado para eliminação, primeiro são retiradas todas as funções de retorno ou dependência associado com os seus vértices. Em seguida, calcula-se uma função de retorno condicional ($\emptyset(a)$), que retorna o valor máximo que o agente é capaz de contribuir para o sistema em função de cada combinação de ação com seus vizinhos e uma função de melhor resposta ($B(a)$),

que retorna a ação correspondente para este valor maximizado. O agente comunica esta função de recompensa condicional para um dos seus vizinhos e é eliminado.

O agente vizinho cria uma nova dependência (ligação) entre si e os agentes envolvidos na função condicional e em seguida, o agente seguinte na ordenação é selecionado para eliminação. Este processo é repetido até que um único agente permaneça. Este agente fixa sua ação para maximizar a função final de retornos. Esta ação individual é parte da ação conjunta ótima, e o valor associado das funções condicionais de retorno é igual ao valor desejado $a^* = argmax_a R(a)$. Uma segunda passagem na ordem inversa é então realizada, em que cada agente calcula a sua ação ótima com base na sua estratégia condicional e as ações fixas dos seus vizinhos (Vlasis, 2006). Para entender melhor o método, utiliza se o exemplo mostrado na Figura 30, onde são representados quatro agentes com suas dependências. Cada nó do grafo é um agente, as ações do agente 1 dependem das ações dos agentes 2 e 3. As ações do agente 2 dependem unicamente das ações do agente 1. As ações do agente 3 dependem das ações do agente 1 e 4

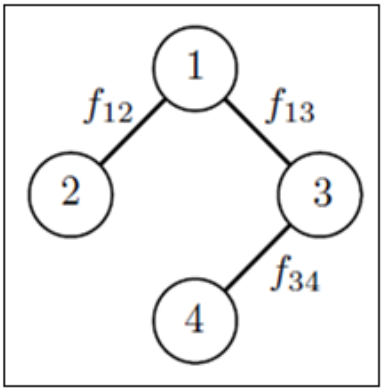


Figura 30: Exemplo de CG par eliminação de variável.

A função de reforço para o exemplo é mostrado na Equação 29:

$$\mathrm{R(a)} = f_{12}(a_1, a_2) + f_{13}(a_1, a_3) + f_{34}(a_3, a_4) \tag{29}$$

Para este exemplo se eliminará primeiro o agente 1 depois o 2 e assim sucessivamente. Quando se elimina o agente número1, a função de retorno é reescrita conforme mostrado na Equação (30):

$$max_a R(a) = max_{a_2 a_3 a_4} = f_{34}(a_3, a_4) + max_{a_1}[f_{12}(a_1, a_2) + f_{13}(a_1, a_3)] \quad (30)$$

Dado que a maximização depende das ações do agente 1 com os agentes 2 e 3, a função de retorno condicional fica como na Equação (31):

$$\emptyset_{23}(a_2, a_3) = max_{a_1}[f_{12}(a_1, a_2) + f_{13}(a_1, a_3)] \quad (31)$$

A função de melhor resposta apresenta a melhor ação que o agente 1 é capaz de realizar dadas as ações dos agentes 2 e 3, como mostrado na Equação (32).

$$B_1(a_1, a_2) = argmax_{a_1}[f_{12}(a_1, a_2) + f_{13}(a_1, a_3)] \quad (32)$$

A função final $\emptyset_{23}(a_2, a_3)$ é independente de $a_1$ e cria uma nova dependência entre os agentes 2 e 3, conforme mostrado na Figura 31.

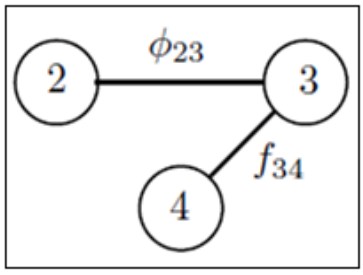


Figura 31: Depois de eliminar o agente 1.

$$\mathrm{R(a)} = max_{a_2 a_3 a_4}[f_{34}(a_3, a_4) + \emptyset_{23}(a_2, a_3)] \quad (33)$$

Em seguida, repete-se o procedimento para o agente dois, onde suas ações dependem unicamente do agente 3 *Φ23* se define.

$$B_2(a_3) = argmax_{a_2} \emptyset_{23}(a_2, a_3) \quad (34)$$

Agora $\Phi_{23}$ é substituído por $\Phi_3(a_3)=max a_2 \Phi_{23}$

$$max_a R(a) = max_{a_3 a_4}[f_{34}(a_3, a_4) + \emptyset_3(a_3)] \quad (35)$$

Seguindo essa lógica, o agente quatro é o último agente a fazer a eliminação da variável e fixa sua opção ótima conforme a Equação (36).

$$a_4^* = argmax_{a_4} \emptyset_4(a_4) \quad (36)$$

Depois vem um segundo passo na ordem inversa de eliminação em que cada agente calcula a sua ação (incondicional) ótima desde suas funções de melhor retorno e suas ações fixas a partir de seus vizinhos. Um exemplo é mostrado na Equação (37)

$$a_3^* = B_3(a_4^*) \qquad a_2^* = B_2(a_3^*) \qquad a_1^* = B_1(a_2^*, a_3^*) \quad (37)$$

No caso de um agente ter mais de uma resposta que maximiza o reforço o sistema escolhe aleatoriamente. A escolha sempre resulta em uma ação coordenada, onde cada agente baseia a sua decisão sobre as ações feitas pelos seus vizinhos.

Este método tenta obter a ação ótima para um grupo de agentes procurando a melhor ação par a par. Esta heurística não garante a melhor ação para o grupo, mas reduz o esforço computacional.

# 5
# Coordenação Inteligente para Multiagentes Baseados em Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço-RL-NFHP-MA

## 5.1
## Introdução

A ideia central deste trabalho é propor e avaliar a integração de dois mecanismos de coordenação ao modelo RL-NFHP-MA (França, 2010), para que os agentes aprendam a realizar suas ações de maneira coordenada em qualquer situação que possa ser observada pelo agente em um dado ambiente. Dessa forma, pretende-se, através da coordenação inteligente dos agentes envolvidos, melhorar o processo emergente, minimizando os recursos utilizados no sistema e acelerando o aprendizado de uma política ótima. Além disso, a inserção dos mecanismos de coordenação permitirão ao modelo RL-NFHP-MA contemplar uma nova gama de aplicações que envolvem ambientes complexos multiobjetivos com restrições (como, por exemplo, futebol de robôs), resultando em estruturas mais robustas para responder melhor à dinâmica do ambiente.

O modelo RL-NFHP-MA original tem uma coordenação deficiente, pois não foi concebido visando soluções para problemas caracterizados como multiagente misto (cooperativos e competitivos). Esse modelo compartilha a sua estrutura com todos os agentes, ou seja, a estrutura aprende um único papel para o um determinado agente. O modelo apresenta uma metodologia limitada que não visa melhorar a vantagem de ter vários agentes trabalhando no mesmo ambiente. Para problemas reais como o futebol de robô, o número de estados e ações para que o modelo RL-NFHP-MA aprenda se torna impraticável. Muitos dos problemas reais são complexos e têm múltiplos objetivos. Nestes casos, dependendo das localizações dos agentes em um dado cenário ou situação (espaço de estados), a troca de papéis, com o intuito de otimizar os recursos do sistema, é conveniente. Por exemplo, em um jogo de futebol, quando se está em uma situação de ataque, um jogador de meio de campo pode cumprir um papel de

atacante, mas em uma jogada defensiva esse mesmo jogador pode desempenhar o papel de um zagueiro.

Conforme o mostrado no capítulo 4, existem vários tipos de abordagens envolvendo coordenação multiagente, dependendo do tipo de problema a solucionar. Neste capítulo serão exploradas duas abordagens de coordenação cooperativa: uma inspirada em Coordenação *Market Driven* e outra em Coordenação por Grafos, para ser integradas com o modelo RL-NFHP-MA.

Esses dois modelos foram escolhidos por apresentar excelentes resultados em coordenação de sistemas multiagentes em diferentes aplicações (Vlasis, 2006) e ser atualmente os mais utilizados na literatura. Os modelos propostos serão apresentados nas próximas seções descrevendo seus funcionamentos, particularidades e objetivos.

## 5.2
## Modelo Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço com mecanismo de coordenação Market-Driven RL-NFHP-MA-MD

Este modelo propõe a integração do RL-NFHP-MA com o mecanismo de coordenação *Market Driven*. Esta nova abordagem se propõe a ser aplicada na resolução de problemas de múltiplos objetivos, que mudam quando as condições do ambiente mudam. Este modelo propõe utilizar RL-NFHP-MA para aprender o papel dos agentes e a coordenação *Market Driven* para decidir a ação de cada agente. O modelo é descrito na Figura 32.

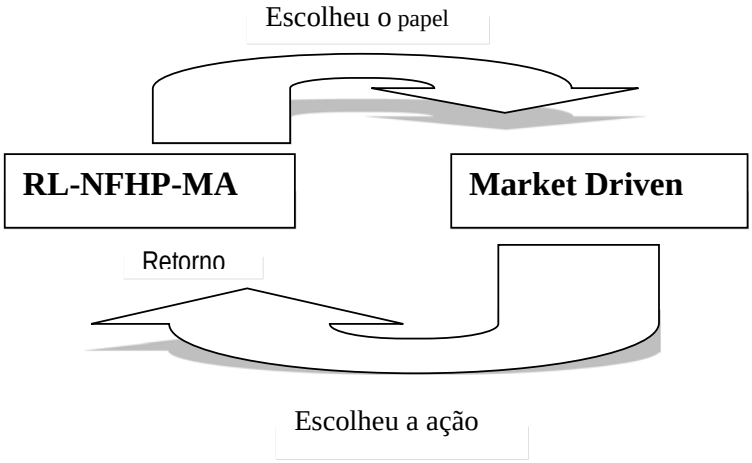


Figura 32: Modelo RL-NFHP-MA-MD.

No Capítulo 4 foi apresentado o modelo por troca de papéis onde, em vez de alocar tarefas a cada agente, alocam-se papéis aos agentes. Esses papéis definem o comportamento individual de cada agente. Por exemplo, no futebol, em uma situação de ataque, a função de atacante é dada ao jogador mais bem posicionado para que possa tirar proveito da situação.

Deste modo, no modelo RL-NFHP-MA-MD, primeiramente, através de um especialista, define-se uma série de possíveis situações para o ambiente (as situações são definidas no capitulo 4 na coordenação por troca de papéis). Em seguida, uma entidade central, que percebe as condições do ambiente, decide qual situação deve ser considerada. Por exemplo, no futebol existem situações como ataque, defesa e marcação. Ao encontrar-se em uma situação específica, cada agente tem um papel específico a cumprir e cada papel tem uma série de ações associadas. Este modelo é, portanto, hierárquico, onde a hierarquia de maior nível contém as situações, a segunda hierarquia contém os papéis e terceiro as ações dos agentes. Isso é descrito na Figura 33.

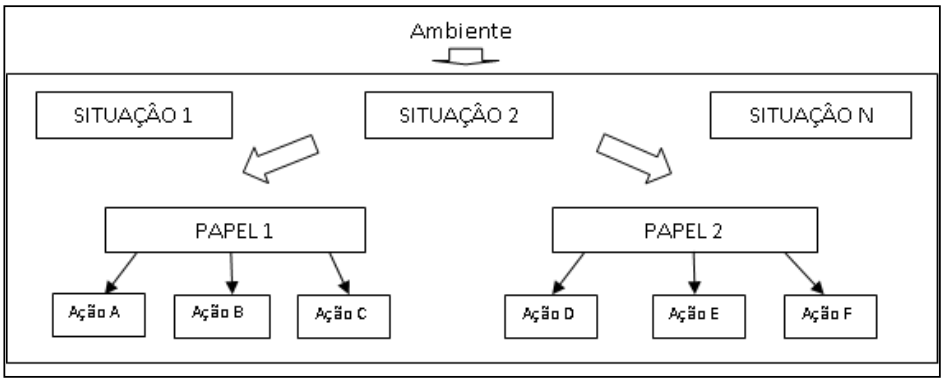


Figura 33: Descreve um ambiente com uma situação associada.

O modelo é dividido em duas etapas: a primeira utiliza o modelo NFHP-MA para aprender o melhor papel de cada agente; a segunda etapa utiliza o conceito de MD para escolher a melhor ação para ser realizada pelo agente, conforme mostrado na Figura 33. Esta ação é escolhida entre as ações que pertencem ao conjunto de ações associadas ao papel.

A ideia da abordagem proposta é utilizar o modelo NFHP-MA para especializar-se em identificar o melhor papel a ser desempenhado por cada agente em uma situação específica do ambiente. A célula criada para este modelo é similar a original. A Figura 34 apresenta um exemplo de uma célula com duas entradas.

As entradas $x_1$ e $x_2$ geram os antecedentes das quatro regras *fuzzy* após serem computados os graus de pertinência $\rho_1(x_1)$, $\mu_1(x_1)$, $\rho_2(x_2)$ e $\mu_2(x_2)$, onde: $\rho_1$ é o conjunto nebuloso baixo e $\mu_1$ é o conjunto nebuloso alto relativo à entrada $x_1$; e $\rho_2$ é o conjunto nebuloso baixo e $\mu_2$ é o conjunto nebuloso alto relativo à entrada $x_2$. Os valores definidos como consequentes são conjuntos de papéis para a situação atual $(p_1, p_2, \ldots p_t)$, onde cada papel está associado a uma função de valor-Q. Através do método de aprendizado baseado em RL, um papel de cada polipartição $(p_i, p_j, p_p$ e $p_q)$ será definido como aquele que representa o comportamento desejado do sistema quando o mesmo se encontra em um determinado estado.

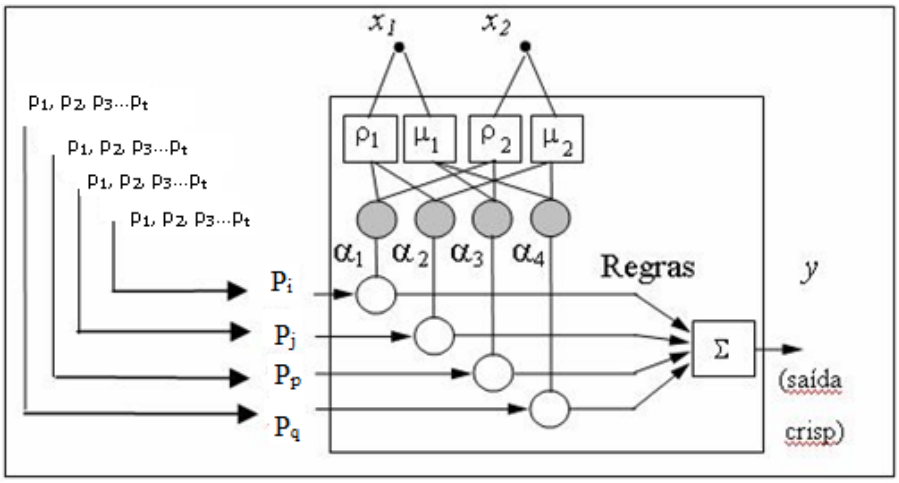


Figura 34: Célula Reinforcement Learning Neuro-Fuzzy Quadtree coordenada (Politree com n=2).

Logo após a escolha do papel do agente, seleciona-se, por meio de leilão, a melhor ação de acordo com o mecanismo *Market-Driven*. As ações têm uma função custo para entrar no leilão. Conforme apresentado no capítulo anterior, a coordenação por *Market-Driven* segue com a mesma filosofia, maximizando o lucro individual de cada agente para aumentar o lucro global. Parte do êxito desta coordenação depende do conhecimento do especialista, por que ele vai definir as características necessárias para realizar uma tarefa e o peso de cada característica, conforme descrito na próxima seção.

A Figura 35 ilustra o esquema de blocos do modelo RL-NFHP-MA-MD, onde a estrutura RL-NFHP decide o papel que cada agente vai ter em uma determinada situação do ambiente e o modelo *Market-Driven* toma a decisão de que ação o agente deve executar. Neste modelo escolheu-se uma única estrutura de aprendizado (repositório do conhecimento) para os agentes do sistema, uma vez que qualquer agente pode assumir um dos papéis associados a uma situação do ambiente.

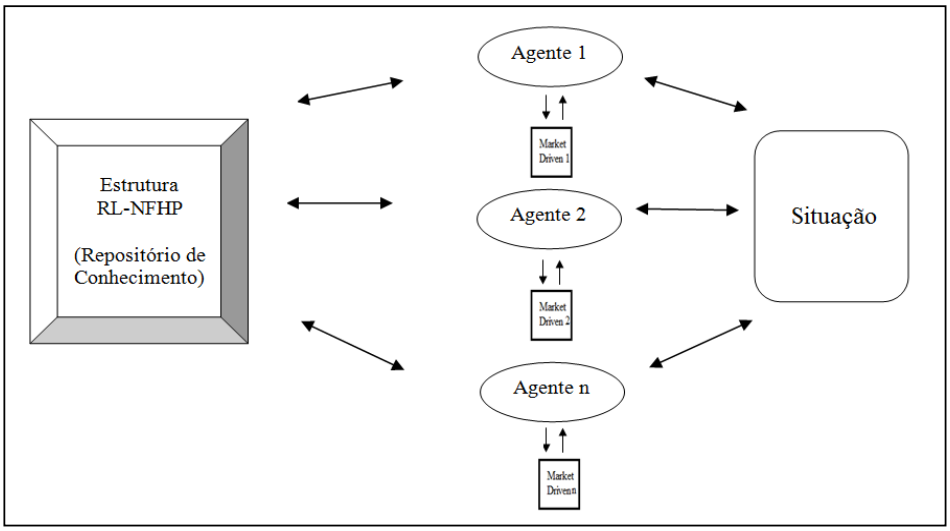


Figura 35: Modelo RL-NFHP-MA-MD.

### 5.2.1
### Algoritmo de Aprendizado do Modelo RL-NFHP-MA-MD

O algoritmo de aprendizado da estrutura utiliza os mesmos mecanismos do modelo RL-NFHP-MA original para a seleção do papel a ser desempenhado por cada agente. No modelo proposto, apenas a escolha da ação é feita por *Market-Driven*. Na grande maioria dos problemas em que os SMA são aplicados, os agentes possuem um objetivo a ser realizado, neste algoritmo se parte desta suposição.

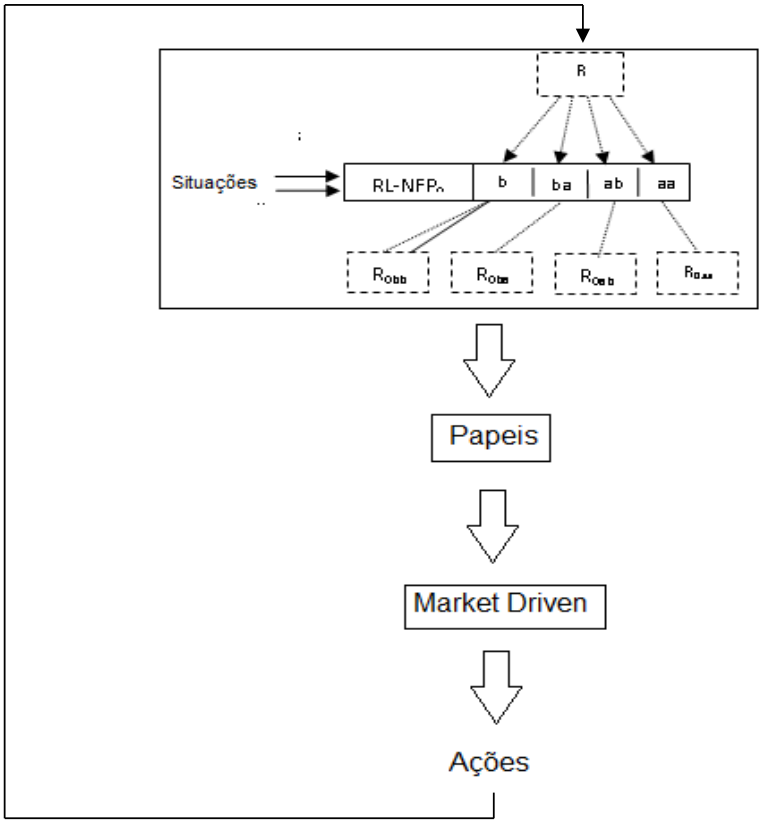


Figura 36: Funcionamento modelo RL-NFHP-MA-MD.

Na Figura 36 é descrito o funcionamento do modelo, onde em uma situação definida por uma entidade central, o RL-NFHP escolhe os papéis dos agentes e, em seguida, o *Market Driven* decide as ações e cada agente. Após a execução da ação selecionada pelo *Market Driven*, retorna-se um valor de reforço. A seguir detalha-se o algoritmo RL-NFHP-MA-MD, descrevendo seu funcionamento e a integração dos dois principais modelos. O processo completo de treinamento é dividido em oito etapas, exibidas na Figura 37. O modelo é iniciado a partir da identificação de uma situação s, a qual está associada um conjunto de papéis *p*.

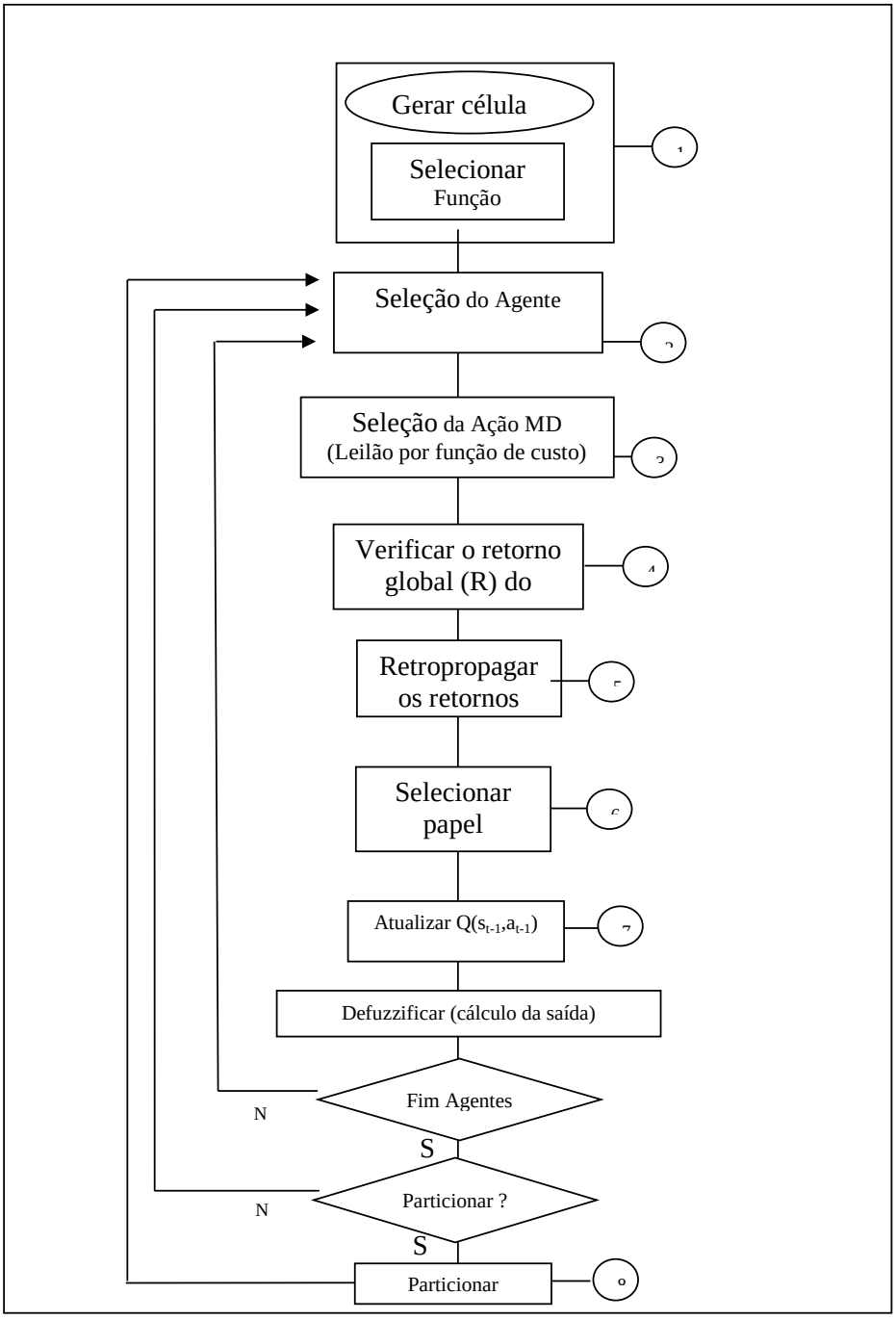


Figura 37: Algoritmo de aprendizado do modelo RL-NFHP-MA-MD Dentro de uma situação.

**1- Inicialização –** Ocorre basicamente da mesma forma que na versão para um único agente. Assim, uma célula raiz é criada, tendo como domínios dos seus conjuntos fuzzy os valores mínimo e máximo das variáveis de entradas (*Limite Inferior* e *Limite Superior*). Com o objetivo de generalidade, os valores das variáveis de entrada são normalizados. Os valores correspondentes às variáveis de entrada da célula são lidos do ambiente, normalizados e são aplicados diretamente às entradas da célula. Estes valores são avaliados nos conjuntos fuzzy *baixo* e *alto*, resultando em dois graus de pertinência, $\rho(x_1)$, $\mu(x_1)$;... $\rho(x_n)$, $\mu(x_n)$; respectivamente, para cada variável de entrada. Cada uma das polipartições escolhe uma dos papéis de seu conjunto de papéis baseando-se nos métodos descritos no passo 6 do algoritmo. A saída da célula é calculada pelo processo de defuzzificação e representa a ação que será executada pelos *atuadores* do agente.

**2- Seleção do Agente –** Nesta etapa, um dos agentes no ambiente é selecionado para exercer um papel. Os agentes são numerados de forma aleatória. A cada ciclo, a seleção é realizada de forma sequencial, sempre do agente 1 ao agente n, para que todos possam interagir igualmente com o ambiente. O aprendizado prossegue normalmente com a movimentação do próximo agente. Para esta abordagem, ao agente é atribuído um papel e como foi explicado, cada papel é composto por várias ações possíveis. Além disso, nesta etapa, o agente selecionado também poderá definir/redefinir seu objetivo durante o aprendizado. Por exemplo, em caso de problemas em que cada agente possui um objetivo distinto, estes objetivos podem ser alterados constantemente ao longo da dinâmica do aprendizado, devido à movimentação dos agentes. Não há uma regra única para a determinação dos objetivos. Ela varia de acordo com o problema ou aplicação que está sendo abordada. Alguns exemplos serão vistos no capítulo 6.

**3- Seleção das ações individuais –** Depois de escolher os papéis para os agentes, através da estrutura RL-NFHP-MA, o modelo entra na etapa de decidir as ações, a qual é realizada pela coordenação *Market Driven*. Cada ação possível tem uma função custo associada, este custo pode ser calculado pelo agente ou por uma entidade central. O leiloeiro escolherá a ação relacionada ao papel que tenha o

menor custo. Cada uma das ações tem um custo diferente para o sistema; escolhendo a de menor custo se está beneficiando o sistema como um todo.

### 3 a- Função de custo

Para implementar o mecanismo *Market-Driven* é necessário que seja definida uma função de custo (*FC*) ela define qual é o custo do agente executar determinada tarefa ou ação. Esta função de custo inclui um conjunto de recursos (*R*), que são necessários para levar a cabo a tarefa (*T*), e um peso específico relacionado a cada um destes recursos (μ). Dependendo da importância dentro da função de custo, que é definida pelo arquiteto. A função de custo é apresentada na equação (38).

$$FC(T) = (\mu_1 R_1 + \mu_2 R_2 + \cdots + \mu_n R_n) \tag{38}$$

O valor da função custo pode ser calculado pelo agente, por uma entidade central ou pelo leiloeiro. Em caso de ser calculado pelo agente, ele calculará a função custo para cada uma das ações envolvidas, e passará unicamente os resultados à entidade central. Em caso de ser calculado pela entidade central, o agente passará as métricas necessárias para calcular cada função custo à entidade central.

### 3 b-Leiloeiro

O leiloeiro é uma entidade externa que tem como função principal escolher a oferta com menor custo e entregar a tarefa. Este custo pode ser dado pelo agente ou calculado pelo leiloeiro.

**4- Função de Avaliação/Retorno -** Após a execução de uma ação por um determinado agente, uma nova leitura do ambiente é realizada. Esta leitura permite que seja calculado o valor de reforço do ambiente e se avalie a ação tomada pelo agente. Este valor deve ser calculado através de uma função de avaliação definida segundo os objetivos do agente, ou do grupo de agentes, sendo fundamental para a orientação do agente ao longo do processo de aprendizado.

A grande diferença em relação à versão original do RL-NFHP é que o reforço de uma ação de um agente passa a depender também da situação dos demais agentes

no ambiente (ou do seu objetivo naquele momento do jogo). Uma determinada ação em um estado pode ser boa ou ruim, de acordo com o posicionamento dos demais agentes no jogo. Logo, a função de avaliação, além de fundamental no aprendizado, passa a ter também extrema importância na coordenação dos agentes.

**5- Retropropagar o retorno** – ocorre da mesma forma que na versão para um único agente. A cada atuação dos agentes no ambiente, um valor de retorno é observado e este é utilizado para atualizar as funções de valor das células ativas para uma dada situação. Essa atualização é feita a partir do valor de retorno global, que é retropropagado para cada partição de todas as células ativas, mediante a sua participação na ação resultante (para maiores detalhes ver seção 4.2). Dessa forma, o retorno do ambiente é retropropagado a partir da célula raiz até as células folhas (gerando valores de reforços locais).

**6- Seleção do papel –** são realizadas através dos mesmos métodos usados pelo algoritmo RL-NFHP. O primeiro, o $\epsilon$*-greedy* (Sutton, 1998), seleciona o papel associado à maior função de valor-Q esperada com probabilidade (1-$\epsilon$); e com probabilidade $\epsilon$ seleciona aleatoriamente uma ação qualquer.

No segundo método, a ação escolhida com probabilidade (1-$\epsilon$) é também a que apresentar a maior função de valor-Q associada; e com um número de vezes proporcional a $\epsilon$, a seleção é baseada na distribuição de probabilidade dada pelas funções de valores-Q. Esta forma de seleção é mais conservadora, uma vez que as ações que apresentarem as maiores funções de valores-Q, terão maiores chances de serem escolhidas. Este tipo de método de seleção de ações é conhecido como *softmax* (Sutton & Barto, 1998).

**7- Atualização dos valores Q** – no caso de múltiplos agentes, para a atualização dos valores de Q, devem ser considerados o retorno global atual e o retorno global anterior sempre do mesmo agente (papel). Só assim é possível avaliar sua ação. Para que isso seja viável, os reforços locais dos agentes devem permanecer armazenados até que o algoritmo selecione novamente o agente em

questão, calcule seu novo reforço, e compare com o seu reforço anterior. Isso evita que retornos de diferentes agentes sejam misturados, causando problemas na atualização dos valores de Q.

Individualmente, cada atualização de valores-Q também ocorre de duas formas distintas: para o caso do valor do retorno global atual ser maior que o retorno global anterior ($R_{t+1} > R_t$) e para o caso do retorno global atual ser menor ou igual ao retorno global anterior ($R_t \geq R_{t+1}$). No primeiro caso, é utilizada a equação do algoritmo *SARSA* (Sutton, 1996), com o objetivo de premiar a ação escolhida. Na segunda situação, as ações escolhidas são punidas, para que se tornem menos propensas a serem selecionadas nas próximas vezes em que as células, as quais elas pertencem, estiverem ativas. A punição ocorre através da mesma equação utilizada pelo modelo RL-NFHP. Além deste método, também foram aplicados os princípios de Não dominação e Satisfatoriedade descritos na seção 4.2.

**8- Particionamento das células –** A decisão de particionar ou não, e o particionamento em si, também ocorre exatamente da mesma forma que na versão para um único agente. Porém, este passo ocorre sempre após uma rodada dos passos 2, 3, 4, 5, 6 e 7 para todos os agentes, e não após cada passo de "Atualização de Valores de Q". Ou seja, para cada agente selecionado, o retorno da ação anterior escolhida é avaliado (passo 4), retropropagado (passo 5), uma nova papel é selecionado (passo 6), e os valores de Q são atualizados (passo 7). Somente após a realização destes cinco passos para todos os agentes é que o passo 7 é executado.

## 5.3
## Modelo Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço com modelo de coordenação por grafos RL-NFHP-MA-CG

Este modelo propõe a integração do modelo RL-NFHP-MA com a Coordenação por Grafos (CG). O modelo envolve uma metodologia central cujo objetivo é coordenar os agentes, onde as ações de um agente dependem das ações de outro. Nos casos onde as ações dos agentes não dependem de outros agentes, os agentes aprendem de maneira individual. As situações em que dois agentes ou mais coordenam suas ações são decididas por um especialista dependendo das condições do ambiente. As dependências dos agentes são desenhadas com grafos, como foi explicado no Capítulo 4 na coordenação por grafos. Na Figura 38 é mostrado um diagrama geral do modelo.

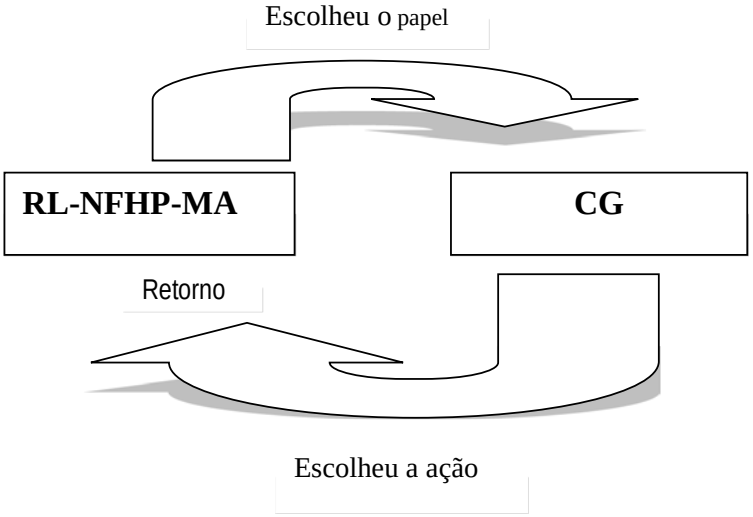


Figura 38: Modelo RL-NFHP-MA-CG.

O modelo se divide em duas etapas. Na primeira etapa o modelo RL-NFHP-MA, como no modelo anterior, encarrega-se de aprender o papel do agente em cada situação, como é mostrado na Figura 6. Esse aprendizado é inspirado na coordenação por troca de papéis conforme explicado no Capítulo 4, onde cada papel possui umas ações associadas. Após a especificação do papel de cada agente, passa-se à segunda etapa, onde é escolhida a ação individual de cada agente por eliminação de variável. O modelo aprende os papéis e situações e observa o retorno dos agentes.

A escolha das ações se faz através do método de eliminação de variável explicado no capítulo 4, o qual tenta escolher a ação individual que forneça o maior retorno nessa situação em particular. A diferença principal deste modelo para o baseado em *Markey Driven* é que, este último, a ação individual de cada agente é escolhida dependendo do papel selecionado. Já neste modelo, escolhe-se a ação individual dependo das ações dos outros agentes, visando maximizar o retorno do sistema através do método de eliminação de variável. Neste modelo escolheu-se um único repositório do conhecimento para os agentes do sistema, conforme mostrado na Figura 39.

Este modelo tem como objetivo diminuir o número de situações em que os agentes coordenam as ações, reduzindo ao máximo o consumo de recursos do sistema. Quando se encontra em uma situação em que os agentes tenham que coordenar as ações, os agentes aprendem a ação individual que maximizam o retorno do sistema. Essas ações ótimas individuais são determinadas a partir da análise entre pares de agentes. Embora não se possa afirmar que com o método de eliminação de variável se encontre a ação individual ótima do sistema, é uma heurística que pretende economizar recursos do sistema.

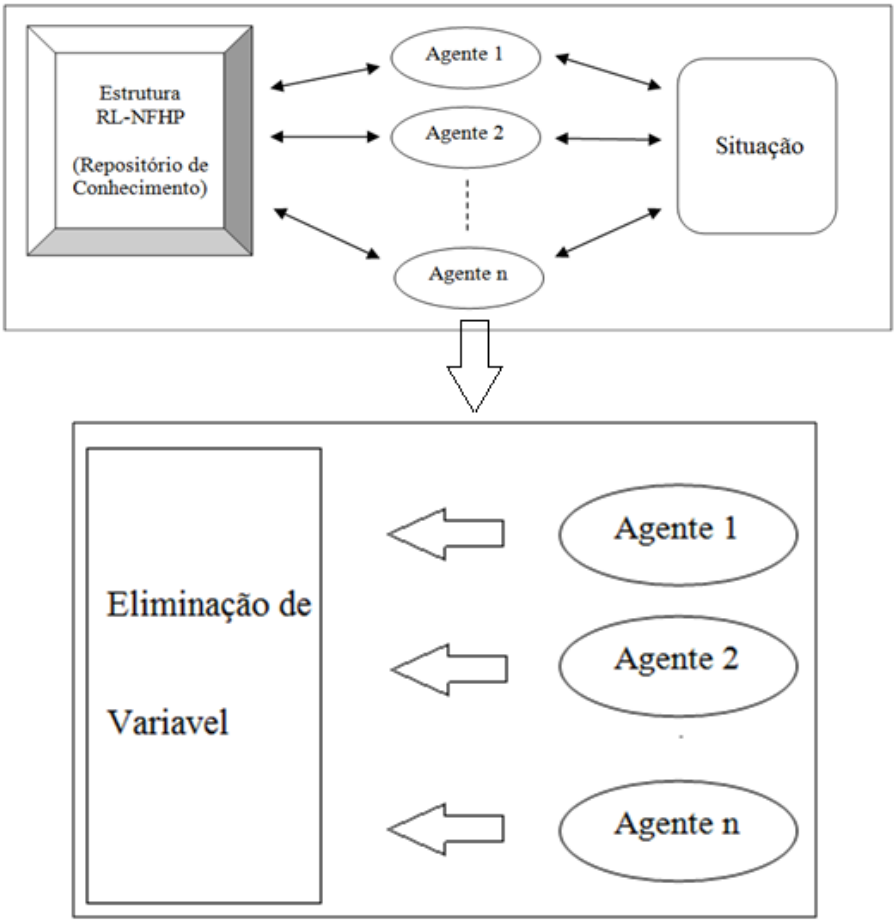


Figura 39: Modelo RL-NFHP-MA-CG.

### 5.3.1
### Algoritmo do Modelo RL-NFHP-MA-CG

O algoritmo começa com a definição das situações que os agentes começam a coordenar suas ações, as entradas para saber qual é a situação atual, são definidas por um especialista. Cada situação tem papéis associados, e cada papel tem várias ações associadas, da mesma forma ao modelo anterior. Em cada situação os agentes aprendem as dependências entre si.

Ao se encontrar em uma situação específica, o algoritmo RL-NFHP-MA aprende o papel de cada agente O algoritmo é idêntico ao apresentado na Seção 5.2.1, exceto pela escolha das ações pelo método de eliminação de variável. A Figura 40 é mostrada um diagrama do modelo completo.

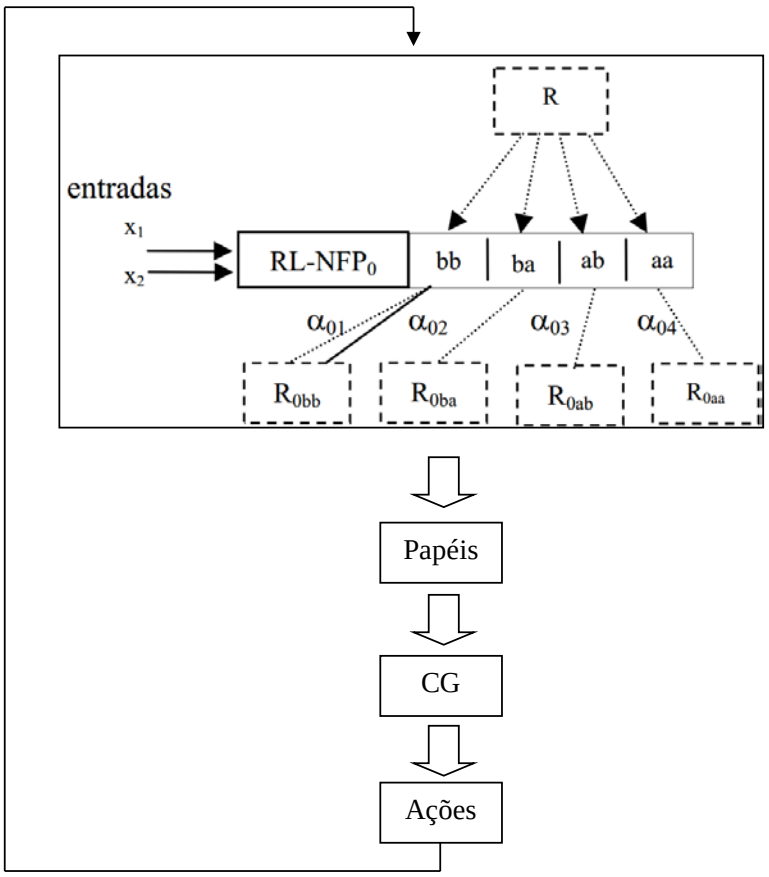


Figura 40: Funcionamento modelo RL-NFHP-MA-MD Dentro de uma situação.

# 6
# Estudo de Caso

## 6.1
## Introdução

Os casos de estudo apresentados neste capítulo foram desenvolvidos com o objetivo de avaliar as estratégias de coordenação integradas ao modelo RL-NFHP-MA com estrutura compartilhada.

Primeiramente, foram desenvolvidos dois sistemas para avaliar os dois modelos propostos: RL-NFHP-MA-MD e RL-NFHP-MA–CG. Em ambos os sistemas os agentes utilizam uma estrutura de conhecimento compartilhada, com o objetivo de acelerar o processo de aprendizado cooperativo dos agentes na realização de uma determinada tarefa, aprendendo o melhor papel a ser representado por cada um destes agentes.

Foram escolhidos dois casos de estudo para a avaliação dos modelos desenvolvidos: o problema clássico para sistemas baseados em multiagentes, denominado presa-predador (Benda et al., 1986); e o futebol de robô na liga *small size*, o qual foi criado para incentivar a realização de pesquisas na área de robótica autônoma multiagente (Reis, 2007). A seção seguinte introduz os casos de estudo selecionados.

## 6.2
## Seleção e Apresentação dos casos de estudo

Para avaliar os dois modelos propostos foram selecionadas duas aplicações. A primeira delas é o Jogo da Presa e Predador, também conhecido como "*Pursuit Game*" (Benda et al., 1986). Através desta aplicação foram explorados sistemas em que os agentes possuem uma estrutura única de conhecimento. A aplicação pode ser considerada de natureza colaborativa, já que ocorre a interação entre os Predadores com o objetivo único de capturar a Presa.

A segunda aplicação foi o futebol de robôs na liga *small size*,. Foi desenvolvido um time de futebol utilizando os dois modelos propostos neste trabalho. A ideia é aprender o melhor papel para cada agente em determinada situação do jogo e, em seguida, escolher a melhor ação a ser executada pelo agente. Este ambiente é bastante complexo, multiobjetivo, com várias situações e restrições, sendo o ambiente ideal para avaliar os modelos propostos.

Os agentes atuam de duas formas, de forma cooperativa, para aprender a melhor estratégia de coordenação, e de interação entre eles para alcançar um objetivo. Os agentes também atuam de forma competitiva já que disputam com outro time de agentes com o objetivo de ganhar o jogo.

## 6.3
## Desenvolvimento dos Casos de estudo

### 6.3.1
### Jogo da Presa e Predador

O Jogo da Presa e Predador, também conhecido como "*Pursuit Game*" (Benda et al., 1986), é utilizado como primeiro caso de estudo. Em sua forma mais tradicional, participam do jogo 4 predadores e 1 presa, que estão dispostos em um *grid* ortogonal, dividido em linhas e colunas, tal que cada célula do grid não pode ser ocupada por mais de um participante ao mesmo tempo. O objetivo do jogo é fazer com que os predadores capturem a presa da forma mais rápida possível. A Figura 41 mostra o funcionamento do jogo e como uma captura ocorre.

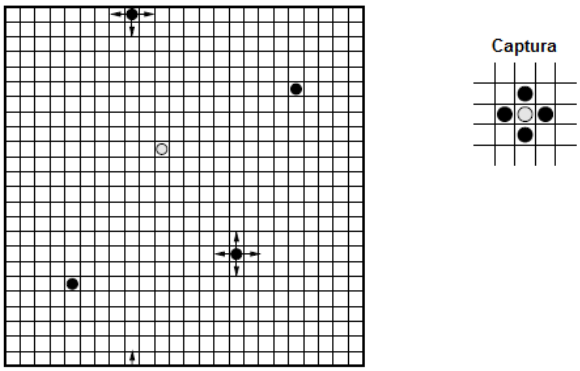


Figura 41: Grid ortogonal mostrando o esquema do Jogo da Presa/Predador e a captura da presa.

Para concretizar a captura, é importante que cada agente faça o cerco à presa em uma posição distinta: por cima, por baixo, pela direita e pela esquerda. No trabalho desenvolvido por França (França, 2010), os predadores tinham um objetivo predefinido: desde o início do jogo conhecia-se o lado pelo qual cada predador ia cercar a presa. Já neste trabalho, cada predador terá um papel, que pode variar com base nas informações gerais do ambiente, como a distância de cada agente em relação à presa. Então, de acordo com a percepção da localização da presa e dos demais agentes, o predador aprende o melhor papel para cercar a presa. Como esta e os demais agentes se movimentam ao longo do jogo, o papel do agente pode ser redefinido a cada rodada. Esse tipo de coordenação está ilustrado na Figura 42, onde os agentes (predadores) partem da origem e cercam a presa em função de sua posição.

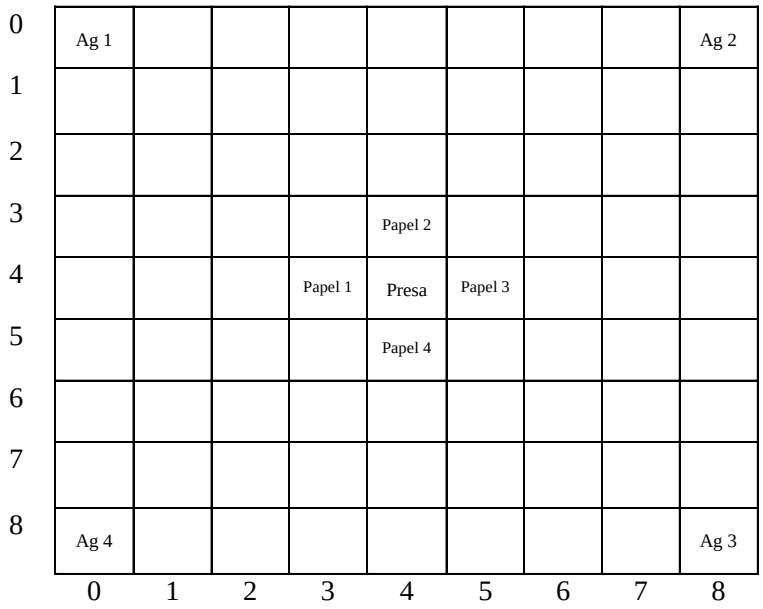


Figura 42: Jogo pressa predador onde Agentes (Ag) são os agentes nas posições inicias, e os papéis é modo de capturar da presa.

A cada papel é associado um conjunto de ações; após aprender o melhor papel, o predador escolhe a melhor ação associada a esse papel.

### 6.3.2
### Metodologia

Nesta aplicação foram explorados sistemas em que os agentes possuem uma estrutura única de conhecimento. Os sistemas RL-NFHP-MA foram construídos seguindo as especificações, premissas e restrições abaixo:

- Espaço composto por um *grid* ortogonal de 9x9 posições;
- Os 4 agentes predadores aprendem um papel RL-NFHP Multiagente, com o objetivo de capturar a presa. Considerando a posição (x, y) da presa como ($P_r$x, $P_r$y):
  - Papel 1: agente que captura pela esquerda; seu objetivo é ($P_r$x - 1, $P_r$y).

- Papel 2: agente que captura pela direita; seu objetivo é ($P_r x + 1, P_r y$).
- Papel 3: agente que captura por cima; seu objetivo é ($P_r x, P_r y + 1$).
- Papel 4: Agente que captura por baixo; seu objetivo é ($P_r x, P_r y - 1$).

- Cada papel tem as mesmas 4 ações associadas que serão escolhidas por *Market-Driven* ou Coordenação por Grafos;
- A presa e os predadores só podem se movimentar em 4 direções; os movimentos diagonais não são permitidos;
- Cada rodada do jogo é composta de uma movimentação de cada participante;
- Não há o conceito de aceleração. Cada movimentação é sempre feita para uma posição imediatamente acima, abaixo, à esquerda o à direita;
- Os participantes se movem de forma alternada, e não simultânea;
- Cada agente predador consegue visualizar a posição dos demais agentes e da presa;
- Os agentes predadores não conhecem os objetivos dos demais agentes;
- Os agentes podem compartilhar o 'conhecimento' obtido ao longo do aprendizado;
- Em caso de colisão após uma movimentação, o agente e a presa retornam para a posição em que estavam, e termina a rodada;
- Em caso de um participante tentar se mover para fora do *grid*, ele também retorna para a posição em que estava.

A Figura 42 apresenta um espaço de estados relacionado às posições para o problema da presa-predador. Nesse caso de estudo, o espaço do caso do estudo foi modelado de forma discreta, por meio de um grid de 9x9 células (França, 2010). Como a maioria dos trabalhos relacionados também usou esse formato, procurou-se efetuar comparações em igualdade de condições para os diferentes modelos.

### 6.3.3
### Treinamento com o modelo RL-NFHP-MA-MD

Esta seção apresenta o processo de treinamento para este caso de estudo do modelo RL-NFHP-MA-MD detalhado no Capítulo 5. O treinamento é dividido em duas etapas: o mapeamento das posições para aprender o papel do agente (predador) e o mapeamento do papel para escolher a melhor ação do agente.

A primeira etapa do treinamento para o estudo do caso presa-predador consiste em aprender o melhor papel de cada predador dependendo de sua posição com respeito à presa e aos outros predadores. Este aprendizado se faz através do modelo NFHP-MA. A segunda etapa do treinamento consiste em, uma vez escolhido o papel do predador, escolher a melhor ação. Neste caso, foi usado o mecanismo *Market-Driven*. Como foi explicado no Capítulo 5, o mecanismo *Market-Driven* baseia-se em leilão para escolher a ação de menor custo. Este é medido por funções de avaliação. Neste caso, como cada agente consegue visualizar tanto a posição da presa como as dos demais agentes, a função de avaliação usada busca encontrar sempre a formação em que a soma das distâncias do agente e do objetivo é a menor possível ou a mais econômica.

O reforço recebido pelo modelo após a escolha do papel e da ação foi calculado de maneira inversamente proporcional à distância entre o agente e a presa, conforme mostram as Equações (39) e (40).

$$\text{Distância entre o agente e a presa} \rightarrow d = |(A_x - P_x)| + |(A_y - P_y)| \quad (39)$$

$$\text{Reforço} \rightarrow r = 1 - d_{norm} \quad (40)$$

$A_x$ e $P_x$ são as posições do agente e da presa no eixo x, e $A_y$ e $P_y$ são as posições do agente e da presa no eixo y, respectivamente; $d_{norm}$ é a distância entre o agente e a presa normalizada de forma linear entre 0 e 1.

O processo de treinamento foi realizado com a presa sempre fixa na posição (4, 4). A posição de cada agente no início do jogo, ou após cada captura, também foi inicializada de maneira fixa, sempre nos cantos do grid – posições (0, 0), (8, 8), (8, 0), e (0, 8). Ou seja, os agentes saem dos cantos e devem capturar uma presa fixa exatamente do centro do *grid*.

Eventualmente, ocorria colisão de objetivos, já que um agente não sabe o objetivo dos demais. Esse tipo de colisão deve ser solucionado naturalmente ao longo da movimentação dos agentes, conforme eles se aproximem da presa e, consequentemente, de seus objetivos.

O treinamento foi realizado durante 100 ciclos. Um ciclo compreende desde o início da perseguição até a captura da pressa. O número ótimo de passos para a captura da presa em um ciclo é 7 para cada agente. Conforme esperado, nos primeiros ciclos do treinamento são necessários mais passos para capturar a presa; entretanto, com o aumento no número de ciclos de treinamento, o tempo e o número de passos para captura diminui, embora poucas capturas tenham sido efetivadas em 7 passos (ver Figura 43). Isso ocorre porque o método de seleção das ações do algoritmo RL-NFHP-MA, durante o processo de aprendizagem, nem sempre escolhe o papel que possui o maior valor de Q. Porém, pode-se ter uma ideia da velocidade de convergência do método.

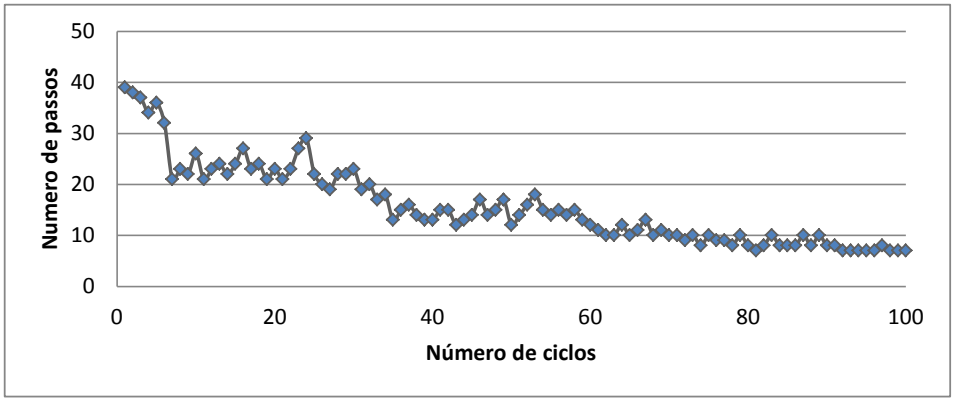


Figura 43: Treinamento RL-NFHP-MA-MD número de passos por ciclo.

Após o período de treinamento, o conhecimento adquirido e armazenado na estrutura foi avaliado em novas situações (fase de teste), cujos resultados são mostrados na Seção 6.3.5.

### 6.3.4
### Treinamento com o modelo RL-NFHP-MA-CG

O treinamento para este segundo modelo é similar ao primeiro modelo, sendo também dividido em duas etapas. A primeira etapa é idêntica a do primeiro modelo, onde se aprende o melhor papel de cada predador, dependendo de sua posição (estado) com respeito à presa e aos outros predadores.

Já na segunda etapa do treinamento, onde uma vez escolhido o papel do predador deve-se escolher a melhor ação, o processo se faz através de coordenação por grafos. Em coordenação por grafos todos os predadores escolhem a ação com base na teoria da eliminação de variável apresentada no Capítulo 5.

O treinamento é realizado durante 100 ciclos. Da mesma forma que no caso anterior, o número ótimo de passos para capturar a presa é 7 para cada agente. Conforme apresentado na Figura 44, no início do treinamento o número de passos necessário para recuperar a presa é bem maior do que ao final do treinamento. Comparando com o modelo RL-NFHP-MA-MD, percebe-se que o modelo RL-NFHP-MA-CG demora mais tempo para convergir.

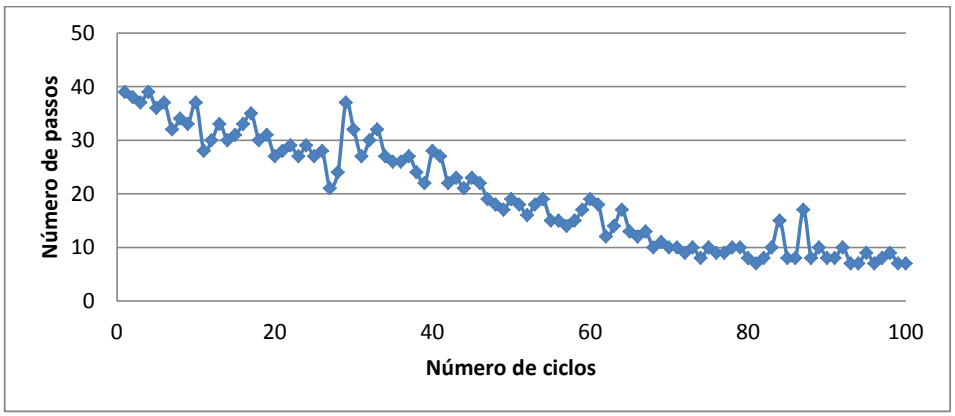


Figura 44: Treinamento RL-NFHP-MA-MD número de passos por ciclo.

Após o período de treinamento, o conhecimento adquirido e armazenado na estrutura foi também avaliado em novas situações, de forma a medir o desempenho de teste. Os resultados são mostrados na próxima seção.

### 6.3.5
### Resultados dos Testes com o Jogo Presa Predador

Esta seção apresenta os testes para mostrar o desempenho dos modelos RL-NFHP-MA-MD e RL-NFHP-MA-CG no caso de estudo presa-predador. Estes testes foram feitos para três configurações de jogo diferentes. Na primeira bateria de testes, foi realizado um total de 1.000 “perseguições”, com os predadores largando sempre dos cantos do *grid*, com a presa fixa. Deve ser destacado que o número total ideal de passos para a captura da presa é 7000 no total, já que o caminho ótimo consiste em 7 passos.

Na segunda bateria de testes, foi realizado um total de 1000 “perseguições”, com os predadores sempre aleatórios e a presa sempre fixa. A matriz das posições aleatórias iniciais é guardada para todos os testes, com o intuito de dar as mesmas condições iniciais a todos. No terceiro teste a presa e os predadores são aleatórios. Os resultados são apresentados nas Tabelas 3, 4 e 5, onde são comparados com os resultados do modelo RL-NFHP-MA original apresentado por França (França, 2010).

Tabela 3: Resultados do primeiro teste, predadores fixos e presa fixa.

| Método | Largada presa | Largada predadores | Coordenação explícita | Redução do numero de passos |
|---|---|---|---|---|
| **RL-NFHP-MA coordenação implícita** | Fixa | Fixa | 13161 | 0% |
| **RL-NFHP-MA central** | Fixa | Fixa | 11294 | 15% |
| **RL-SMA-NFHP-CG** | Fixa | Fixa | 9200 | 30% |
| **RL-SMA-NFHP-Market driven** | Fixa | Fixa | 7620 | 43% |

Tabela 4: Resultados do segundo teste, predadores sempre aleatórios e a presa sempre fixa.

| Método | Largada presa | Largada predadores | Coordenação explícita | Redução do número de passos |
|---|---|---|---|---|
| **RL-NFHP-MA coordenação implícita** | Fixa | Aleatória | 9548 | 0% |
| **RL-NFHP-MA central** | Fixa | Aleatória | 7794 | 19% |
| **RL-SMA-NFHP-CG** | Fixa | Aleatória | 7355 | 23% |
| **RL-SMA-NFHP-Market driven** | Fixa | Aleatória | 6750 | 30% |

Tabela 5: Resultados do terceiro teste, predadores e presa aleatórios.

| Método | Largada presa | Largada predadores | Coordenação explícita | Redução do número de passos |
|---|---|---|---|---|
| **RL-NFHP-MA coordenação implícita** | Aleatória | Aleatória | 9476 | 0% |
| **RL-NFHP-MA central** | Aleatória | Aleatória | 7695 | 19% |
| **RL-SMA-NFHP-CG** | Aleatória | Aleatória | 7210 | 23% |
| **RL-SMA-NFHP-Market driven** | Aleatória | Aleatória | 5940 | 38% |



### 6.3.6
### Discussão dos resultados presa predador

Neste primeiro estudo de caso foram testados os modelos propostos - RL-NFHP-MA-MD e RL-NFHP-MA–CG - no *pursuit game*. O objetivo foi verificar como um mecanismo de coordenação permite melhorar o desempenho de SMA e aumentar a velocidade da convergência do aprendizado.

Os testes realizados comparando o desempenho dos modelos propostos com o modelo original (França, 2010) mostraram que os primeiros são superiores em todos os testes, embora o aprendizado tenha um alto custo computacional. Apesar

do *pursuit game* ser uma aplicação simples, com um único objetivo (capturar a presa), já se pode verificar uma diferença no desempenho do modelo RL-NFHP-MA quando é incluído um modelo de coordenação estruturado e hierárquico.

O treinamento dos modelos apresentados neste trabalho demorou mais em convergir aos passos ótimos se comparado ao modelo original (França, 2010). Isso acontece por que nos modelos de coordenação propostos o agente aprende o melhor papel e em seguida escolhe a ação, tornando muito mais complexo o aprendizado. No modelo original ele unicamente aprendia a melhor ação para a captura da presa.

No primeiro teste, os dois modelos apresentados neste trabalho mostraram um desempenho muito bom. Já no segundo e no terceiro testes, onde as posições dos predadores são aleatórias, a coordenação RL-NFHP-MA-CG apresentou uma redução no desempenho quando comparado ao RL-NFHP-MA-MD. Em todos os testes, o resultado apresentado pelo modelo de coordenação *Market-Driven* foi muito melhor do que o de outros sistemas.

Os testes foram realizados em computadores com processador intel i7 e 8 Gigas de memória RAM, em plataforma Java. Para o treinamento do modelo RL-NFHP-MA-MD, foram necessárias aproximadamente 27 horas. Já para o modelo RL-NFHP-MA-CG, o tempo necessário para efetuar o treinamento foi de 98 horas. O modelo RL-NFHP-MA-MD é, em média, quatro vezes mais rápido do que o método RL-NFHP-MA-CG no treinamento, resultando em um menor custo computacional. Isso mostra que o modelo de coordenação baseado em *Market-Driven* é mais promissor para estudos de caso mais complexos.

O modelo com coordenação *Market-Driven* utiliza uma função de avaliação para a escolha das ações. Da escolha dessa função depende em parte o êxito do sistema. Para o estudo de caso mostrado, a escolha foi uma tarefa simples. Entretanto, para sistemas mais complexos, é necessário ter um conhecimento maior do ambiente.

A coordenação por troca de papéis permite tirar vantagem da posição do agente, otimizando o desempenho do modelo. Isso mostra que a coordenação não se limita a uma coordenação central e distribuída. É necessário aplicar uma metodologia dentro de um SMA e conhecer bem a aplicação. Este método é

hierárquico; primeiro aprende o papel do agente para, em seguida, escolher a ação, sendo ideal a ambientes multiagentes mais complexos como futebol de robô.

Percebe-se, assim, que com a introdução de coordenação nos modelos da família RL-NFHP-MA, foi possível obter resultados melhores que os modelos originais, mostrando serem promissórios em aplicações em ambientes mais complexos.

## 6.4
## Futebol Robótico

Dentre os possíveis domínios de aplicação para testar as estratégias desenvolvidas de coordenação de agentes, destaca-se o Futebol Robótico. O Futebol Robótico constitui um ambiente adequado à investigação de mecanismos de coordenação em SMA, com ênfase em agentes cooperativos. A escolha do futebol robótico como domínio principal de teste para os modelos desenvolvidos teve como objetivo observar o seu desempenho em um ambiente com as características complexas que envolvem o futebol robótico. Essas características complexas do futebol robótico são: ambiente dinâmico, não determinístico, contínuo e que impõe elevadas limitações às capacidades de comunicação dos agentes. Existem atualmente duas confederações mundiais de futebol robótico: a FIRA (*Federation of International Robot-soccer Association*) e *Robocup*. As duas confederações organizam todos os anos um mundial com diferentes ligas. Nesta tese se escolheu como aplicação a liga *small size*, que pertence à *Robocup*, por vários motivos, entre eles o fato da *Robocup* ser a confederação com maior divulgação no Brasil e a nível mundial. Além disso, a maioria de trabalhos da área de coordenação multiagente utiliza a liga *small size* da confederação *Robocup* como aplicação de seus modelos, sendo mais fácil a comparação de desempenho entre eles.

### 6.4.1
### Robocup (*small size league*)

A iniciativa *RoboCup* (Kitano, 1995) (Kitano, 1997) é um projeto internacional que tem o objetivo de promover a pesquisa e a educação em Inteligência Artificial e Robótica Inteligente (RoboCup, 2013).

O futebol foi a grande motivação para o *Robocup*. Além de ser um esporte bastante popular em quase todo o mundo, o futebol apresenta também desafios científicos importantes, uma vez que é um jogo coletivo com problemas individuais (identificação e localização de objetos, dribles, remates etc.) e, ao mesmo tempo, problemas coletivos (coordenação, estratégia etc.). A existência de um ambiente dinâmico, onde os agentes envolvidos precisam desenvolver características de cooperação e competição, exige uma grande sofisticação dos algoritmos envolvidos.

O *Robocup Soccer* engloba as competições de futebol robótico em variadas formas. Um elevado conjunto de tecnologias é necessário para possibilitar a construção de uma equipe de Robôs, reais ou virtuais, capaz de participar de um desafio de futebol que segue um determinado conjunto de regras (Plant, 2009). A competição envolve seis diferentes ligas:

- *Middle size league*
- *Small size league*
- *Four-legged league*
- *Simulation league*
- *Humanoid league*
- *E-league*

A liga de robôs pequenos (*small size*) é a categoria utilizada como aplicação neste trabalho. Na liga *small size* os robôs são bem menores, quando comparados com as outras ligas. Esses robôs são menos complexos estruturalmente e mais econômicos, por isso são os mais usados para medir o desempenho dos algoritmos de coordenação multiagente, sendo dessa forma os mais encontrados na literatura científica. Embora sejam independentes uns dos outros, todo o processamento é

efetuado em um computador central, baseado em informação (posição dos jogadores, da bola etc.) vinda de uma ou duas câmeras de vídeo posicionadas sobre o campo de futebol. Os comandos são enviados para os robôs via *wireless*, conforme mostrado na Figura 45.

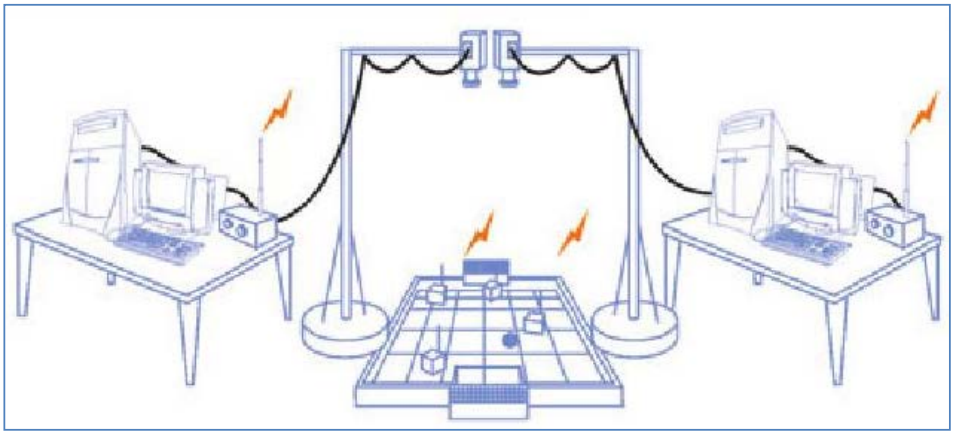

Figura 45: Descrição de *small size league.*

A intervenção humana resume-se à introdução e remoção dos robôs no campo. Cada jogo é disputado entre equipes compostas por, no máximo, seis robôs de dimensões reduzidas e padronizadas, que jogam num campo verde com marcações brancas e dimensões definidas. Mais detalhes sobre as regras do jogo podem ser obtidas da página oficial da liga *small size* (Robocup, 2013). As regras do jogo permitem a utilização de visão global e controle centralizado dos robôs. Desta forma, durante o jogo, o sistema de controle dos robôs, sempre externo a estes, recebe a informação proveniente da câmera localizada sobre o campo, processa esta informação, inferindo ou identificando as posições, orientações e velocidades dos robôs e da bola, decide o comando mais apropriado a ser executado por cada robô, e envia este comando por rádio a cada um dos robôs.

A necessidade de alta velocidade e de um controle preciso deu a esta liga a reputação de “a liga de engenharia” (Plant, 2009). Os desenvolvimentos científicos realizados no âmbito desta liga incluem novos sistemas eletromecânicos, de controle, de eletrônica digital e comunicações sem fios (Mackworth, 2012). No entanto, os avanços verificados ao longo dos últimos anos no hardware e a relativa estabilidade das regras do jogo contribuem para uma

necessidade urgente da realização de pesquisa em metodologias de coordenação, de forma a se obter melhores resultados nesta liga ao longo dos próximos anos (Mackworth, 2012).

Para a realização do estudo de caso dessa tese, conforme já mencionado acima, a liga escolhida foi a *small size* em ambiente virtual, já que o processo de aprendizado em ambiente real demandaria um tempo computacional muito grande, impossibilitando a sua aplicação. Assim, na próxima seção será descrito o ambiente do simulador escolhido para essa liga de futebol robótico.

### 6.4.2
### Simulador small size league (Soccerbots)

*SoccerBots* é um simulador do RoboCup da liga *small size* desenvolvido na plataforma Java (Ramani, 2007). *Soccerbots* simula a dinâmica, as dimensões do campo, robôs, bola e as regras do *RoboCup Small Size League*. A Figura 46 ilustra as dimensões do campo em mm. As regras, as dimensões do campo, robôs e bola podem ser alteradas caso seja do interesse do usuário.


Figura 46: Dimensões do Campo utilizado.

*Soccerbots* também funciona como uma plataforma benchmark onde diferentes pesquisadores, em sua maioria estudantes de mestrado e doutorado, criam times para testar seus algoritmos. No momento existem 40 times na

plataforma (Ramani, 2009). A seguir se apresentam as características mais relevantes da plataforma *Soccerbots*.

- Campo: As dimensões usadas nos testes foram as dimensões oficiais do regulamento de 2013, detalhadas na Figura 46.
- Robôs: Os robôs são circulares com 12 centímetros de diâmetro. O tamanho máximo permitido para os robôs é de 15 centímetros. Os robôs simulados podem mover-se a uma velocidade de 0,3 metros / segundos e viram 360°/seg.
- Equipes: Um time é composto por 5 jogadores.
- Bola: A bola simulada representa uma bola de golfe laranja de 40mm de diâmetro, que pode se mover a uma velocidade máxima de 0,5 m/seg e desaceleração de 0,1m/seg. As colisões com outros robôs e com as paredes são elásticas.
- Parede: A parede é colocada ao redor de todo o campo, com exceção dos gols. As colisões da bola com as paredes são perfeitamente elásticas.

A simulação é executada em tempo real, mas pode ser reproduzida mais rápida ou mais lentamente, dependendo do desejo do usuário. Na Figura 47 é mostrada a interface gráfica do simulador.

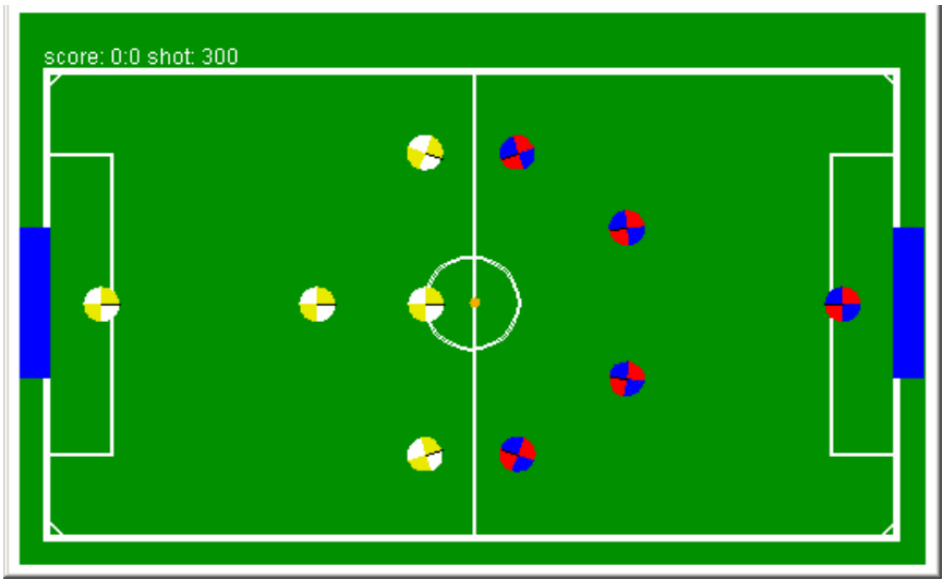


Figura 47: Interface gráfica do simulador *Soccerbots.*

**Informações obtidas pelos robôs**

As informações obtidas pelos robôs e usadas para mapear o ambiente são:

- Vetor de distância do robô para a bola.
- Vetor que aponta do robô para o gol do time.
- Vetor que aponta do robô para o gol do adversário.
- Conjunto de vetores que apontam do robô para os robôs companheiros de equipe.
- Conjunto de vetores que apontam do robô para os robôs adversários.
- A posição do robô no campo.
- Tempo em milissegundos desde o início do jogo.

**Ações**

As ações que o robô pode executar no simulador *soccerbots* são mostradas e explicadas na Tabela 6 a seguir.

Tabela 6: Ações disponíveis no simulador.

| Açoes | Descrição |
|---|---|
| ***Drive*** | O jogador conduz a bola. |
| ***MoveToBall*** | O jogador mais próximo move-se para a bola |
| ***Pass*** | O jogador passa a bola ao companheiro mais próximo. |
| ***TakePositionAttack:*** | O jogador assume posição de ataque. |
| ***TakePositionDefense*** | Dependendo da estratégia, o jogador assume posição de defesa. |
| ***PlayMidfield*** | Joga como um jogador de meio-campo |
| ***Playdefender*** | Joga como um jogador de defesa |
| ***KickBallToGoal*** | Chuta a bola para o gol |
| ***Stop*** | O jogador é selecionado para deter um robô |
| ***Block*** | Mantém a bola fora de alcance dos robôs oponentes |

A maioria dos times desenvolvidos no projeto *soccerbots* baseia sua estratégia em escolher um papel predefinido para cada robô com ações pré-estabelecidas. Um exemplo é o time *Market* desenvolvido por Kose (Kose, 2007), onde a cada robô é dado um papel predefinido, sendo que várias ações são associadas a esse papel. Essas ações são escolhidas por meio de uma função de avaliação.

A seguir será descrita a metodologia utilizada para os modelos desta tese nessa aplicação.

### 6.4.3
### Metodologia

Para testar o desempenho dos modelos de coordenação apresentados neste trabalho, foi desenvolvido um time de futebol robótico no *small size league* da plataforma *Soccerbots*. O time desenvolvido utiliza os modelos propostos para fazer a coordenação entre os robôs, da seguinte maneira: a coordenação visa que o agente aprenda um papel e dentro do papel escolha uma ação. O modelo proporciona um processo de aprendizado da coordenação dos robôs, de qual papel o robô deve aprender e qual ação deve escolher em diversas situações. Dependendo da situação do jogo, o agente (robô) aprende o melhor papel para aquela situação e, em seguida, escolhe a melhor ação a ser executada para fazer o gol, ou evitar o gol contra.

Nesta abordagem, em vez de alocar uma ou mais tarefas a cada agente, aloca-se dinamicamente papéis aos agentes. Por exemplo, ao invés de predefinir o papel de atacante a um robô, ele vai receber o papel de atacante caso esteja em uma situação de ataque dependendo de sua posição; por outro lado, em uma jogada de defesa, o mesmo robô pode receber o papel de zagueiro. Essa abordagem tira proveito da posição do jogador em determinada situação.

Para desenvolver o time de futebol primeiro deve-se modelar uma estratégia de jogo. Esta estratégia, definida pelo criador de cada time, inclui a definição de quais papéis podem ser desempenhados pelos robôs em determinada situação do jogo, como situação de ataque ou defesa. Estas situações têm que ser facilmente identificáveis pelos agentes para permitir efetuar a troca dinâmica de papéis de maneira eficiente. Na Figura 48 é descrito o ambiente do futebol de robô.

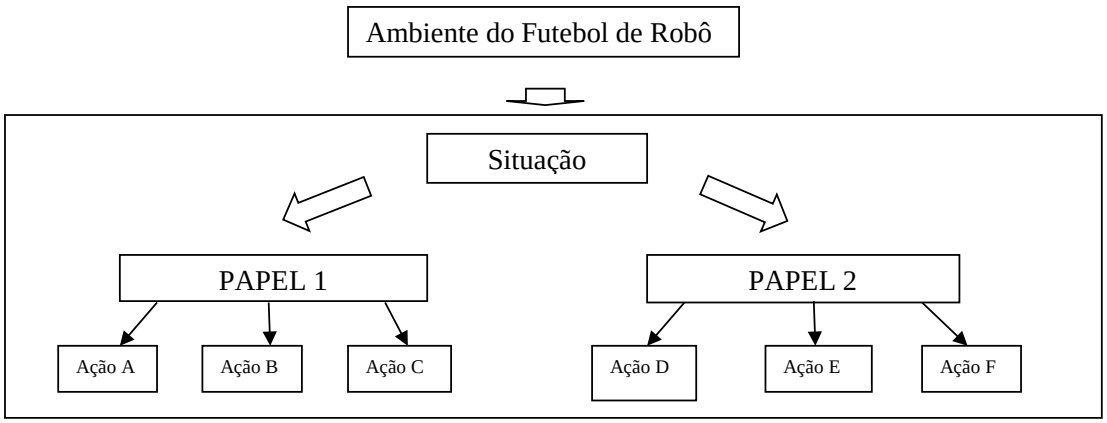


Figura 48: Modelo do ambiente futebol de robô.

Para testar o desempenho dos modelos de coordenação apresentados foram criados dos times LRI-MD e LRI-CG: o primeiro usando coordenação baseada no modelo RL-NFHP-MA-MD e o segundo coordenação RL-NFHP-MA-CG, ambos com uma única estratégia, que será apresentada a seguir.

### Estratégia Times LRI

O time desenvolvido para avaliar os métodos de coordenação apresentados neste trabalho no futebol robótico foi denominado LRI (time do Laboratório de Robótica Inteligente do Departamento de Engenharia Elétrica da PUC-Rio). O time contém 5 robôs, todos com as mesmas características físicas e de processamento. O time é baseado em troca de papéis (cinco no total), cada um com poucas ações associadas, conforme mostrado na Tabela 7. Cada papel tem algumas ações associadas já explicadas na Tabela 6. O time LRI foi avaliado no simulador *Soccerbots* com o objetivo de participar do campeonato *Robocup smallsize league*.

Tabela 7: Papéis e ações associadas do Time LRI.

| Papéis | Largada presa |
|---|---|
| Atacante | KickBallToGoal, Pass, Give ball. |
| Zagueiro | Stop, Block, MoveToBall. |
| Apoiador de ataque | GiveBall, TakePositionAttackright, TakePositionAttackleft, MoveToBall. |
| Apoiador de zaga | Block, TakePositionDefense, Stop. |
| Condutor | Drive, Pass, GiveBall, KickBallToGoal |

Foram investigadas e avaliadas diferentes configurações de possíveis papéis, com diferentes conjuntos de ações, utilizando como mecanismo de coordenação o RL-NFHP-MA-MD para uma avaliação mais rápida. Após a realização de diversos testes, chegou-se à configuração de papéis e ações apresentada na Tabela 7, a qual apresentou o melhor desempenho do time. O objetivo foi obter uma configuração onde cada papel tivesse características individuais dependendo das diferentes situações apresentadas em um jogo de futebol, tais como ataque, defesa e passagem de defesa a ataque.

O papel de goleiro é fixo. Os papéis descritos na tabela 7 são distribuídos entre os quatro jogadores de acordo com o método de cooperação utilizado (*market driven* ou coordenação por grafos).

Vários jogadores podem ter um mesmo papel em determinado estado. A seguir são apresentadas as características e funções de cada um dos papéis escolhidos para o Time LRI:

**- Goleiro**

Conforme mencionado, o papel de goleiro é fixo. Essa estratégia foi escolhida por ter apresentado um melhor desempenho nos diferentes testes preliminares e por ser usada pela maioria dos times. A estratégia do goleiro é igual à utilizada pelo time MattiHetero (Ramani, 2009). O goleiro segue a posição do jogador que está com a bola. Se a bola estiver longe, o goleiro fica perto da rede. Se a bola estiver dentro da área do gol, o goleiro vai a seu encontro. O goleiro

segue a bola para trás e para frente no campo, mantendo-se dentro do raio de sua defesa.

**- Atacante**

O atacante é o responsável por ser a referência no ataque e pelas finalizações, estando localizado no campo adversário. A estratégia geral do atacante é procurar sempre a melhor posição em campo para chutar a gol.

**- Apoiador do ataque**

O apoiador de ataque é o responsável por acompanhar de perto o atacante pelos lados, como opção de passe. Outra função é evitar a marcação do atacante e pegar a bola.

**- Zagueiro**

O zagueiro é responsável por bloquear outros jogadores nas proximidades da grande área, mesmo quando está em frente ao goleiro. Também tem como função roubar a bola do time contrário.

**- Apoiador de zaga**

O apoiador de zaga é responsável por defender os avanços adversários pelos lados do campo e roubar a bola do time contrário.

**- Condutor**

O condutor tem como função conduzir a bola o mais perto da área adversária e passar ao jogador melhor posicionado ou chutar a gol.

Para avaliar o desempenho do time, este foi testado jogando contra outros times com diversas estratégias, as quais são apresentadas a seguir.

### 6.4.3.1
### Times Adversários

De forma a comparar os resultados obtidos pelos modelos propostos, foram escolhidos quatro times já desenvolvidos e disponibilizados na plataforma Soccerbots. Esses times foram selecionados por serem os mais utilizados nos diferentes trabalhos existentes na literatura (Ramani, 2007), (Reis, 2009), (Kose, 2007). Esses times não foram alterados para esta aplicação, de modo a estar em igualdade de condições com outros trabalhos da área. Apesar de não se ter muitos detalhes sobre o desenvolvimento desses times, por razões de competitividade (Ramani, 2007), uma breve descrição de cada um desses times é apresentada abaixo:

**- AIKHomoG**

Esse time é baseado em aprendizado por reforço com campos de potencial para designação do papel (Fidan, 2006) (Saghir, 2013). Esta estratégia de aprendizado coloca campos de potencial negativos nas posições que o robô deve evitar e campos positivos nos objetivos a se alcançar. Estes campos, por meio de atração, indicam qual é o melhor papel para o robô em uma situação e funcionam como função de avaliação para o aprendizado por reforço. Este time apresenta quatro papéis: atacante, zagueiro, apoiador, e um goleiro com funções definidas através de regras (Ramani, 2008). Este time apresenta um bom desempenho, tendo ficado em segundo lugar em 2007 e terceiro em 2008 em campeonatos internos da plataforma Soccerbots (Ramani 2008). O algoritmo utilizado é considerado homogêneo já que todos os robôs usam o mesmo algoritmo de aprendizado. O campo para este time foi discretizado para um grid de 12.

**- MattiHetero**

Este time é baseado em aprendizado por reforço para aprender o papel desempenhado por cada robô (Ramani, 2008). Ele é composto por 3 papéis: ala esquerda, ala direita e volante. O papel do goleiro é atribuído a um robô de maneira fixa. Todos, exceto o goleiro, se comportam da mesma maneira: caso possuam a bola, tentam chegar perto o suficiente para obter uma visão do gol.

Caso não possam chutar, tentam passar a bola ao agente mais próximo do mesmo time. Este algoritmo é considerado heterogêneo já que o algoritmo de aprendizado é diferente para o goleiro. Este algoritmo é considerado o melhor da plataforma Soccerbots, sendo o vencedor de 2006 a 2009 e utilizado como benchmark na maioria de trabalhos de coordenação multiagente (White, 2006) (Ramani, 2008) (Kose, 2007). O campo para este time foi discretizado para um grid de 12.

**- GoToBall**

Neste time todos os jogadores se movem para trás da bola e tentam chutá-la na direção do gol. É o algoritmo mais simples da plataforma, mas é também bastante usado como benchmark. (Ramani, 2008).

**- PermHomoG**

Este time baseia-se em permutações de papéis. São quatro os papéis existentes: um bloqueador, dois atacantes e um zagueiro. O bloqueador tem como objetivo ficar na frente do oponente mais próximo à bola. O desempenho deste time é mediano.

### 6.4.4
### Treinamento do time LRI-MD com o modelo de coordenação RL-NFHP-MA-MD

Esta seção apresenta o processo de treinamento do modelo de coordenação RL-NFHP-MA-MD, detalhado no Capítulo 5, para o caso de estudo de futebol de robô. Nesta seção são apresentadas as duas etapas do processo de treinamento: o mapeamento das posições para aprender o papel do agente (jogador) e o mapeamento do papel para escolher a melhor ação do agente. Estas duas etapas culminam no aprendizado integral, tanto do aprendizado coletivo (coordenação dos agentes), como individual do agente de forma hierárquica.

A primeira etapa do treinamento consiste em aprender o melhor papel de cada jogador, dependendo da sua posição com respeito a seus companheiros, seus adversários e a bola. Este aprendizado se faz através do modelo RL-NFHP-MA. Nesta etapa as entradas da célula RL-NFHP-MA para cada jogador são:

- as coordenadas (x, y) de cada companheiro,
- as coordenadas (x, y) de cada adversário,
- as coordenadas (x, y) da bola.

A segunda etapa do treinamento consiste em, uma vez escolhido o papel do jogador, escolher a melhor ação; neste caso, foi usado o mecanismo *Market-Driven*. Como foi explicado no Capítulo 5, o mecanismo *Market-Driven* se baseia em teoria de leilão para escolher a ação de menor custo. Este custo, associado a cada ação, é medido através de funções de avaliação. As funções de avaliação foram definidas a partir das informações do ambiente (nesse caso a plataforma *Soccerbots*), tais como localização do jogador, dos companheiros dos adversários e da bola.

Na Tabela 8 são apresentadas algumas dessas informações do ambiente.

Tabela 8: Exemplo informações do ambiente da plataforma *Soccerbots.*

| Classe | Função |
|---|---|
| Cleargoal | Em um ângulo de 60 graus de visão não apresenta obstáculo na direção do gol contrário (ver Figura 50) |
| Clearball | Em ângulo de 60 graus de visão não apresenta obstáculo na direção da bola |
| Talignggol | Tempo para alinhar-se com o gol contrário |
| Distgol | Distância do jogador ao gol |

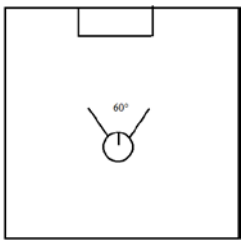


Figura 49: Função *clearball.*

Cada ação tem uma função de avaliação associada que informa o custo dessa ação para o leiloeiro, o qual escolhe a ação de menor custo. As funções de avaliação de cada ação são mostradas na Tabela 9. Estas ações foram explicadas na tabela 6.

Tabela 9: Funções de avaliação.

| Ação | Função de Avaliação |
|---|---|
| **KickBallToGoal** | $\mu_1$cleargoal+ $\mu_2$distgol+$\mu_3$talignggol |
| **Drive** | $\mu_4$clearball+$\mu_5$distgol+$\mu_6$talignggol+$\mu_7$distnearpartner |
| **Give ball** | $\mu_8$clearball+$\mu_9$distgol+$\mu_{10}$talignggol+$\mu_{11}$distnearpartner+$\mu_{12}$cleargoal |
| **Stop** | $\mu_{13}$distdriveballothertime+$\mu_{14}$taligdriveballothertime |
| **Block** | $\mu_{15}$distnearothertime+$\mu_{16}$talignearothertime |
| **MoveToBall** | $\mu_{17}$clearball+$\mu_{18}$distball+$\mu_{19}$aligball |

Depois de passar pelas duas etapas, a estratégia de reforço foi a seguinte: dava-se reforço positivo de 100 por um gol a favor e -80 por gol em contra.

O treinamento foi feito colocando como adversário o time *Basictime* também da plataforma *Soccerbots*, o qual é um time com uma configuração básica de um goleiro, dois atacantes e dois zagueiros. Este time foi escolhido por ser usado pelos times adversários para também treinar seus times. O treinamento foi feito em três configurações diferentes: um campo dividido em um grid de 12, como é mostrado na Figura 50; um campo dividido em um grid de 16, mostrado na Figura 51; e, por último, um campo com espaço contínuo.

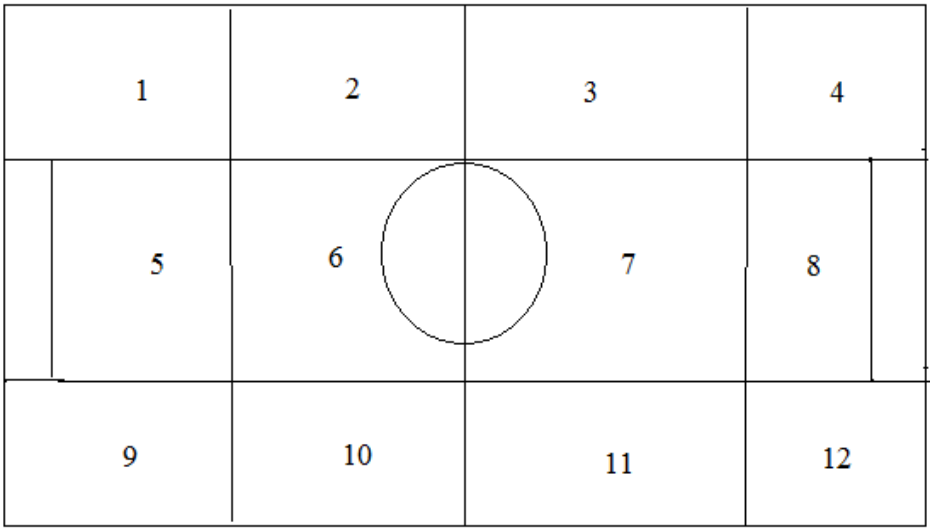


Figura 50: Campo de futebol dividido em um grid de 12.

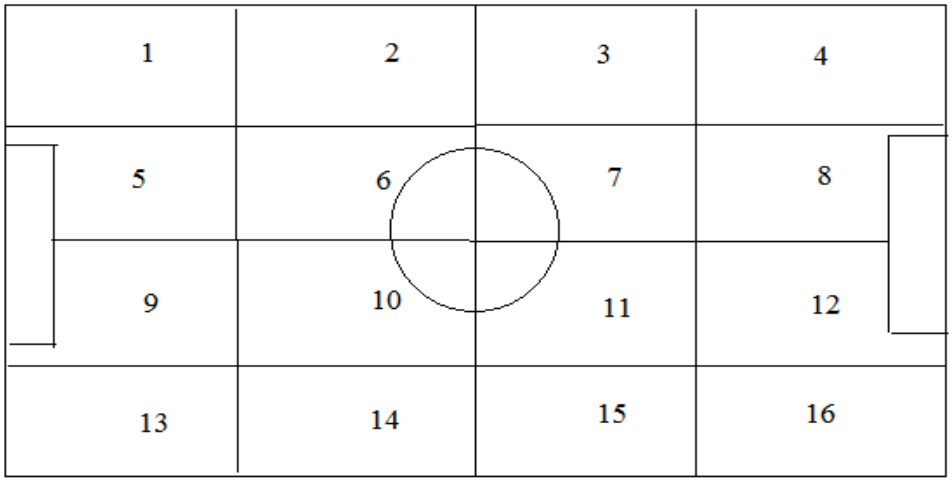


Figura 51: Campo de futebol dividido em um grid de 16.

Para o treinamento nos dois grid, foram feitos 200 ciclos, e cada ciclo terminava com um gol a favor ou um gol do adversário. Para a configuração contínua, foram feitos 1000 ciclos de treinamento. O modelo contínuo necessita de um número maior de ciclos, pois além de aprender os papéis e ações, o agente deve aprender a identificar estados similares do ambiente. Após o período de treinamento, o conhecimento adquirido e armazenado na estrutura foi avaliado em testes e os resultados são mostrados na seção 6.4.6.

### 6.4.5
### Treinamento do time LRI-CG com o modelo RL-NFHP-MA-CG

Esta seção apresenta o processo de treinamento do modelo de coordenação RL-NFHP-MA-CG (detalhado no Capítulo 5), para o estudo de caso do futebol de robôs. O treinamento para este segundo modelo é similar ao primeiro modelo, e também é dividido em duas etapas. A primeira etapa, igual ao caso anterior, se aprende o melhor papel de cada jogador, dependendo da sua posição (estado), da posição de seus companheiros, dos adversários e da bola. Já na segunda etapa do treinamento, onde uma vez escolhido o papel do jogador deve-se escolher a melhor ação dentro das ações associadas a este papel, considerando o estado do jogador, neste modelo o processo se faz através de coordenação por grafos. Em coordenação por grafos todos os jogadores devem escolher a ação baseados na teoria da eliminação de variável (detalhada no Capítulo 5).

Os testes foram os mesmos praticados no modelo RL-NFHP-MA-MD, a menos da configuração do espaço de estados contínuo, que será explicado na discussão dos resultados. Após o período de treinamento, o conhecimento adquirido e armazenado na estrutura foi avaliado em testes. Os resultados serão apresentados a seguir.

### 6.4.6
### Resultados dos testes de futebol robótico com modelo de coordenação RL-NFHP-MA-MD

#### 6.4.6.1
#### Testes com o time LRI

Esta seção apresenta os testes de desempenho do modelo RL-NFHP-MA-MD no caso de estudo futebol robótico. Em todos os testes foram realizados 20 jogos do time LRI contra os times AIKHomoG, MattiHetero, GoToBall e PermHomoG. Os testes foram feitos para as três configurações de treinamento: campo dividido em um grid de 12, grid de 16 e campo contínuo. Os jogos tiveram uma duração de 5 minutos e as regras usadas seguiram os regulamentos da

plataforma. Nestes testes o time LRI com coordenação RL-NFHP-MA-MD é chamado LRI-MD. Foram feitos 4 testes que serão detalhados a seguir:

### Teste 1

Neste teste os pesos das funções de avaliação foram deixados todos iguais. Os resultados para o treinamento do campo em grid de12 são mostrados na tabela 10, para o grid de 16 na tabela 11 e para o campo contínuo na tabela 12.

Tabela 10: Resultados do LRI-MD com grid de 12.

| Time adversário | Jogos ganhos pelo LRI-MD | Jogos perdidos pelo LRI-MD | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 14 | 6 | 87 | 41 |
| PermHomoG | 15 | 5 | 112 | 30 |
| GotoBall | 19 | 1 | 204 | 6 |
| MattiHetero | 13 | 7 | 50 | 35 |

Tabela 11: Resultados do LRI-MD com grid de 16.

| Time adversário | Jogos ganhos pelo LRI-MD | Jogos perdidos pelo LRI-MD | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 17 | 3 | 105 | 30 |
| PermHomoG | 18 | 2 | 136 | 36 |
| GotoBall | 20 | 0 | 274 | 12 |
| MattiHetero | 14 | 6 | 45 | 21 |

Tabela 12: Resultados do LRI-MD com grid contínuo.

| Time adversário | Jogos ganhos pelo LRI-MD | Jogos perdidos pelo LRI-MD | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 14 | 6 | 68 | 28 |
| PermHomoG | 12 | 8 | 33 | 25 |
| GotoBall | 20 | 0 | 95 | 1 |
| MattiHetero | 12 | 8 | 30 | 22 |

Os resultados dos modelos propostos nesta tese foram comparados com outros trabalhos de referência (Kose, 2007), (Vanik, 2008) (Ramani, 2007). Conforme pode ser verificado nas tabelas anteriores, o modelo LRI-MD ganhou dos times adversários em todas as configurações de grid, podendo ser considerado um modelo superior aos avaliados para a coordenação em futebol de robôs.

A configuração que apresentou os melhores resultados foi com o grid de 16, conseguindo, com 200 ciclos de treinamento, um bom desempenho. Foram feitos outros testes variando-se o número de ciclos, sendo que os resultados com 100 ciclos foram inferiores. Com 300 ciclos, os resultados ficaram um pouco inferiores, 10% em média, mas não muito distantes dos anteriores. Em termos de tempo computacional, obviamente o mais vantajoso a ser aplicado é o grid de 12, seguido pelo grid de 16. A configuração que teve a maior média de gols marcados também foi a com grid de 16; já a que sofreu menos gols contra, foi a configuração contínua.



### Teste 2

O treinamento do time LRI, para este teste, foi realizado com configuração de pesos da função de avaliação conforme mostra a Tabela 13. Basicamente, deu-se mais peso à função *cleargoal* para aumentar a agressividade das jogadas de ataque.

Tabela 13: Funções de avaliação do teste 2.

| Ação | Função de Avaliação |
|---|---|
| **KickBallToGoal** | 0,70cleargoal+ 0,15distgol+0,15talignggol |
| **Drive** | 0,50clearball+0,15distgol+0,15talignggol+0,2distnearpartner |
| **Give ball** | 0,50clearball+0,125distgol+0,125talignggol+0,125distnearpartner+0,125clearg oal |

Os resultados com esse novo teste são mostrados na Tabela 14 para o grid de 12, na Tabela 15 para o grid de 16 e na Tabela 16 para o campo contínuo. Neste teste, esta configuração do time enfrenta  a melhor configuração do teste 1.

Tabela 14: Resultados do LRI-MD com grid de 12.

| Time adversário | Jogos ganhos pelo LRI-MD | Jogos perdidos pelo LRI-MD | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 16 | 4 | 95 | 51 |
| PermHomoG | 14 | 6 | 101 | 27 |
| GotoBall | 20 | 0 | 243 | 4 |
| MattiHetero | 14 | 6 | 63 | 47 |
| LRI-teste1 | 8 | 12 | 18 | 24 |

Tabela 15: Resultados do LRI-MD com grid de 16.

| Time adversário | Jogos ganhos pelo LRI-MD | Jogos perdidos pelo LRI-MD | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 14 | 6 | 80 | 44 |
| PermHomoG | 18 | 2 | 110 | 20 |
| GotoBall | 20 | 0 | 242 | 4 |
| MattiHetero | 14 | 6 | 44 | 16 |
| LRI-teste1 | 11 | 9 | 28 | 26 |

Tabela 16: Resultados do LRI-MD com campo contínuo.

| Time adversário | Jogos ganhos pelo LRI-MD | Jogos perdidos pelo LRI-MD | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 13 | 7 | 41 | 30 |
| PermHomoG | 16 | 4 | 93 | 13 |
| GotoBall | 20 | 0 | 104 | 7 |
| MattiHetero | 12 | 8 | 37 | 21 |
| LRI-teste1 | 7 | 13 | 14 | 24 |

Os resultados foram parecidos com os apresentados no teste 1, mas para o grid de 12 melhoraram um pouco. O time subiu um pouco a média de gols a favor. Os gols realizados pelo time LRI-MD aumentaram, mas os gols feitos pelos adversários também. Essa questão da melhor escolha dos pesos das funções de avaliação é um ponto interessante para investigações futuras, pois através de pesos adequados pode-se melhorar ainda mais os resultados já obtidos.

### Teste 3

O teste 3 é um teste específico com foco em jogadas de defesa. Esta jogada começa na metade do campo do time adversário, e este está com a posse da bola. A jogada pode terminar de duas formas: com a recuperação da bola pelo time LRI-MD ou com gol do time adversário. Foram avaliadas 20 jogadas com cada um dos times (ver tabela 17) adversários, contabilizando o tempo que o time LRI-MD leva para recuperar a posse da bola do adversário. Nos casos de gol, a contabilidade de tempo é feita desde o início da jogada até receber o gol. Este teste foi realizado para as três configurações de espaço de estados: grid de 12, grid de 16 e contínuo. As tabelas 17, 18 e 19 apresentam os resultados das avaliações das jogadas de defesa, com os times adversários e com espaços de estados em grid de 12, grid de 16 e contínuo, respectivamente.

Tabela 17: Resultados do LRI-MD com grid de 12.

| Time adversário | Recuperação em 20 jogadas | Média de tempo | Gols contra | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 15 | 24 seg | 5 | 38 seg |
| PermHomoG | 16 | 18 seg | 4 | 27 seg |
| GotoBall | 20 | 7 seg | 0 | 0 seg |
| MattiHetero | 14 | 36 seg | 6 | 22 seg |

Tabela 18: Resultados do LRI-MD com grid de 16.

| Time adversário | Recuperação | Média de tempo | Gols contra | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 17 | 22 seg | 3 | 41 seg |
| PermHomoG | 17 | 12 seg | 3 | 30 seg |
| GotoBall | 20 | 9 seg | 0 | 0 seg |
| MattiHetero | 16 | 28seg | 4 | 32 seg |

Tabela 19: Resultados do LRI-MD com campo contínuo.

| Time adversário | Recuperação | Média de tempo | Gols contra | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 14 | 34 seg | 6 | 29 seg |
| PermHomoG | 16 | 18 seg | 4 | 22 seg |
| GotoBall | 20 | 18 seg | 0 | 0 seg |
| MattiHetero | 13 | 40seg | 7 | 19 seg |

No futebol pode-se dizer que o grande objetivo é ganhar o jogo, mas este objetivo tem dois subobjetivos fundamentais: fazer e não receber gols. Para cumprir estes subobjetivos são necessárias duas jogadas: ataque e defesa. Neste teste, portanto, quis-se medir a eficiência do time especificamente nas jogadas de defesa. Na jogada de defesa se verifica a eficiência da coordenação no objetivo de não receber gol. As melhores métricas são, portanto, a quantidade de recuperações de bola e o tempo gasto para a recuperação da bola. Se o tempo é menor para recuperar a bola, o time aprendeu bem esta tarefa, que é fundamental para não sofrer gols. A quantidade de tempo que o adversário leva para conseguir fazer gols também mostra o quão eficiente é a defesa de um time; se o adversário consegue fazer gol rapidamente, a defesa não é boa nesta tarefa.

Neste teste a melhor configuração de espaço de estados foi o grid de 16, obtendo o maior número de recuperações de bola e recebendo um menor número de gols. Nas outras configurações de espaço de estados o time LRI-MD teve um bom desempenho, tendo médias parecidas no tempo de recuperação e na recuperação da bola.

### Teste 4

O teste 4 é um teste específico com foco em jogadas de ataque. Esta jogada começa na metade do campo do time adversário, sendo que o time LRI-MD está com a posse da bola. A jogada pode terminar de duas formas: o time LRI-MD perde a posse da bola ou faz gol no time adversário. Foram feitas 20 jogadas, com diversos times adversários (ver tabelas 20, 21 e 22), contabilizando o tempo que demora para o time LRI-MD perder a bola para o adversário e para fazer o gol. Este teste foi realizado para as três configurações de espaço de estados: grid de 12,

grid de 16 e contínuo. As tabelas 20, 21 e 22 apresentam os resultados das avaliações das jogadas de ataque com os times adversários para os espaços de estados em grid de 12, grid de 16 e contínuo, respectivamente.

Tabela 20: Resultados do LRI-MD com grid de 12.

| Time adversário | Perda da bola | Média de tempo | Gols a favor | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 13 | 19 seg | 7 | 29seg |
| PermHomoG | 12 | 22 seg | 8 | 26 seg |
| GotoBall | 1 | 56 seg | 19 | 11 seg |
| MattiHetero | 15 | 17 seg | 5 | 34 seg |

Tabela 21: Resultados do LRI-MD com grid de 16.

| Time adversário | Perda da bola | Média de tempo | Gols a favor | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 11 | 24 seg | 9 | 27seg |
| PermHomoG | 10 | 27 seg | 10 | 23 seg |
| GotoBall | 0 | 0 seg | 20 | 9 seg |
| MattiHetero | 13 | 16 seg | 7 | 30 seg |

Tabela 22: Resultados do LRI-MD com campo contínuo.

| Time adversário | Perda da bola | Média de tempo | Gols a favor | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 15 | 18 seg | 5 | 30seg |
| PermHomoG | 11 | 28 seg | 9 | 30 seg |
| GotoBall | 0 | 0 seg | 20 | 12 seg |
| MattiHetero | 16 | 16 seg | 4 | 33 seg |

Neste teste se avaliou o desempenho do time LRI-MD em jogadas de ataque e o desempenho foi considerado pouco satisfatório em todas as configurações, pois o número de gols a favor é geralmente inferior ao número de vezes que o modelo perdeu a bola para o time adversário. A melhor configuração foi para o grid de 16, mostrando-se como a melhor configuração em todos os testes. A configuração do espaço de estados contínuo comportou-se de forma semelhante ao modelo com grid de 12, provavelmente por não ter tido um número de ciclos suficiente para aprender satisfatoriamente a complexidade do ambiente.

### 6.4.6.2
### Discussão dos resultados com o time LRI-MD

O time LRI-MD teve um bom desempenho em todas as configurações de espaço de estados (grids e contínuo) quando comparado a outros modelos cooperativos, em especial contra o adversário MattiHetero que aparece como o melhor da plataforma e campeão em vários anos.

Os melhores resultados alcançados com o modelo LRI-MD foram através da configuração do espaço de estados com o grid de 16.

Os espaços de estados divididos em grid têm uma grande vantagem sobre o modelo contínuo, porque seus estados são conhecidos previamente. No caso contínuo os estados não são conhecidos previamente. Por isso, os resultados do espaço contínuo são considerados bons, já que são similares aos do espaço discreto.

**[m2] Comentário:** CORRIGIR

O tempo de treinamento do grid de 12 foi menor quando comparado com o grid de 16, consumindo menos recurso computacional. Por esse ponto de vista, a configuração de espaço de estados com melhor custo/benefício, entre as duas configurações que usaram grids, é a com  o grid de 12. Isso faz sentido, pois nesse caso o número de espaço de estados é menor, mas não o suficiente para inviabilizar um aprendizado que atinja um resultado satisfatório.

Para a configuração de espaço de estados com espaço contínuo foram feitos vários treinamentos com diferentes números de ciclos. O primeiro teste usou 200 ciclos, mesmo número utilizado nos testes com grids, mas não foi suficiente para aprender de forma satisfatória. Em seguida, aumentou-se para 1000 o número de ciclos, possibilitando os resultados apresentados nesta tese. Devido ao grande tempo computacional consumido no ajuste dos modelos, não foi possível determinar o número de ciclos exato para ter um excelente resultado, sendo necessário para isso inserir um período de validação entre blocos de ciclos (a cada 100 ou 200 ciclos, por exemplo), e assim poder determinar o número de ciclos ideal de forma a interromper o treinamento.

A maioria dos modelos baseados em aprendizado por reforço, que se encontram na plataforma *Soccerbots*, treina os modelos no campo dividido em um grid de 12, visando economizar recurso e tempo de processamento.

#### 6.4.6.3
#### Testes com o time LRI com coordenação RL-NFHP-MA-CG

Para o time LRI com coordenação por grafos foram realizados 3 testes iguais ao do time LR com coordenação RL-NFHP-MA-MD, também jogando 20 vezes contra os times AIKHomoG, MattiHetero, GoToBall e PermHomoG. Os jogos tiveram uma duração de 5 minutos, as regras usadas seguiram os regulamentos da plataforma. Foram contabilizados os gols a favor e contra de cada time. Neste caso só foram realizados testes com grid de 12 e grid de 16, pois configuração contínua tornou-se extremamente custosa computacionalmente com o modelo RL-NHHP-MA-CG. A configuração com ambiente contínuo não convergiu para uma solução após várias semanas de treinamento, sendo considerado não viável para esta configuração. Nestes testes o time LRI com coordenação RL-NFHP-MA-CG é chamado LRI-CG.

#### Teste 1

No teste 1 o time LRI com coordenação por grafos enfrenta os times adversários em jogos completos. As tabelas 23 e 24 apresentam os resultados do time LRI-CG.

Tabela 23: Resultados do LRI-CG com grid de 12.

| Time adversário | Jogos ganhos pelo LRI-CG | Jogos perdidos pelo LRI-CG | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 13 | 7 | 61 | 22 |
| PermHomoG | 16 | 4 | 50 | 12 |
| GotoBall | 20 | 0 | 107 | 3 |
| MattiHetero | 12 | 8 | 38 | 28 |

Tabela 24: Resultados do LRI-CG com grid de 16.

| Time adversário | Jogos ganhos pelo LRI-CG | Jogos perdidos pelo LRI-CG | Gols a favor | Gols contra |
|---|---|---|---|---|
| AIKHomoG | 15 | 5 | 51 | 19 |
| PermHomoG | 18 | 2 | 74 | 12 |
| GotoBall | 20 | 0 | 113 | 0 |
| MattiHetero | 13 | 7 | 30 | 18 |

O time LRI-CG também foi treinado com 200 ciclos, visando deixar os dois modelos desta tese em igualdade de condições para comparar seus desempenhos. As duas configurações tiveram um desempenho satisfatório frente aos adversários. O grid de 16 apresentou os melhores resultados, da mesma forma que ocorreu com o modelo LRI-MD. Entretanto, o número de gols a favor são inferiores em todos os casos aos apresentados pelo modelo LRI-MD.

### Teste 2

O teste 2 é um teste com foco em jogadas de defesa, da mesma forma que foi feito no teste 3 do modelo LRI-MD. Foram avaliadas duas configurações de espaço de estados: com grid de 12 e grid de 16. As tabelas 25 e 26 apresentam os resultados obtidos nas duas configurações.

Tabela 25: Resultados do LRI-CG com grid de 12.

| Time adversário | Recuperação | Média de tempo | Gols contra | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 14 | 26 seg | 6 | 34 seg |
| PermHomoG | 17 | 15 seg | 3 | 22 seg |
| GotoBall | 20 | 9 seg | 0 | 0 seg |
| MattiHetero | 12 | 39 seg | 8 | 27 seg |

Tabela 26: Resultados do LRI-CG com grid de 16.

| Time adversário | Recuperação | Média de tempo | Gols contra | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 15 | 24 seg | 5 | 37seg |
| PermHomoG | 18 | 12 seg | 2 | 32 seg |
| GotoBall | 20 | 7 seg | 0 | 0 seg |
| MattiHetero | 14 | 31seg | 6 | 33 seg |

Os melhores resultados foram obtidos com a configuração do espaço de estados com o grid de 16, semelhante aos outros testes. Os resultados destes testes são melhores, se comparados com o time LRI-MD, o que mostra que este modelo se comporta melhor em jogadas de defesa.

### Teste 3

O teste 3 é um teste com foco em jogadas de ataque. Da mesma forma que o teste 4 do modelo LRI-MD, foram avaliadas as configurações de espaço de estados com grid de 12 e grid de 16. As tabelas 27 e 28 apresentam os resultados desse teste.

Tabela 27: Resultados do LRI-CG com grid de 12.

| Time adversário | Perda da bola | Média de tempo | Gols a favor | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 15 | 20 seg | 5 | **32seg** |
| PermHomoG | 13 | 23 seg | 7 | 31 seg |
| GotoBall | 0 | 0 seg | 20 | **15 seg** |
| MattiHetero | 17 | 16 seg | 3 | 39 seg |

Tabela 28: Resultados do LRI-CG com grid de 16.

| Time adversário | Perda da bola | Média de tempo | Gols a favor | Média de tempo |
|---|---|---|---|---|
| AIKHomoG | 14 | 30 seg | 6 | **30seg** |
| PermHomoG | 12 | 39 seg | 8 | 27 seg |
| GotoBall | 0 | 0 seg | 20 | **14 seg** |
| MattiHetero | 16 | 23 seg | 4 | 38 seg |

O mesmo comportamento apresentado testes anteriores se repete neste teste, obtendo os melhores resultados com o grid de 16. Para o teste com as jogadas de ataque os resultados observados para o modelo LRI-CG foram consideravelmente melhores do que o modelo LRI-MD.

#### 6.4.6.4
#### Discussão dos resultados com o time LRI-CG

A maioria dos resultados do time com modelo de coordenação RL-NFHP-MA-CG foram bons, pois manteve alta a média de partidas ganhas. No teste com jogadas de defesa seu desempenho foi ótimo quando comparado com o teste com jogadas de ataque. A média de gols a favor e contra caiu consideravelmente para todas as configurações de espaço de estados, para todos os testes. O time LRI-CG perdeu a bola com mais facilidade nas jogadas de ataque, tanto nos testes com partidas completas como nos testes de jogadas de ataque. A redução de desempenho em jogadas para realizar gols deve ser melhorado, alterando a estratégia por exemplo com a inclusão de novos papéis.

O treinamento com o modelo baseado em coordenação por grafos foi bem mais custoso que o baseado em *Market Driven*, para todas as configurações do ambiente. No caso do espaço de estados com o grid de 12, o treinamento foi bastante custoso, mas obteve bons resultados. Já para o grid de 16 foi necessário um cluster de computadores para acelerar o processamento, de forma a viabilizara obtenção dos resultados. Já no caso de ambiente contínuo, não se conseguiu obter bons resultados para o modelo RL-NFHP-MA-CG.

### 6.4.5
### Discussão dos resultados Futebol Robótico

O segundo estudo de caso apresentado nesta tese foi o futebol robótico baseado no *small size league* da *Robocup*. Foram desenvolvidos dois times de futebol robótico (baseado nas características da liga correspondente na *Robocup*), utilizando cada um dos modelos RL-NFHP-MA-MD e RL-NFHP-MA-CG para a coordenação dos agentes. Em ambos os modelos, os agentes utilizam uma

estrutura de conhecimento compartilhada, ou seja, a partir de uma mesma estrutura hierárquica foi feito o particionamento do espaço de estados e a identificação do melhor papel e da ação relacionada aos estados.

O objetivo deste estudo de caso foi mostrar como um mecanismo de coordenação permite melhorar o desempenho de um SMA, em um ambiente multiobjetivo e complexo como o futebol robótico. Os times desenvolvidos foram testados com diversos adversários, entre os quais estão dois campeões (AIKHomoG e MattiHetero) da *Robocup*, cujos modelos também se baseiam em aprendizado por reforço.

Os dois modelos apresentados se mostram superiores a quase todos os times usados como *benchmark*, perdendo em algumas métricas para o time MattiHetero. Isso mostra a efetividade dois modelos de coordenação propostos nesta tese.

O melhor desempenho foi alcançado pelo time LRI-MD, baseado no modelo RL-NFHP-MA-MD, que mostrou superioridade ao time LRI-CG em quase todos os testes. Além disso, o time LRI-MD necessitou de um treinamento menos custoso computacionalmente (tempo de processamento) em todas as configurações de espaço de estados, mostrando-se muito mais adequado para problemas complexos. Para o caso do espaço de estados contínuo, o time LRI-MD gastou muito tempo para atingir bons resultados e o time LRI-CG não teve o mesmo desempenho, sendo considerado não viável pela demanda de recurso computacional.

O time LRI-CG desenvolvido com o modelo RL-NFHP-MA-CG, detalhado no Capítulo 5, escolhe uma ação comparando todas as ações uma a uma, selecionando a melhor combinação. Isso é uma desvantagem em termos de custos computacionais em relação à coordenação *Market-driven*, que é baseado em um sistema de leilão, onde a ação executada é escolhida de forma mais rápida. Esse foi o motivo porque os testes com o time LRI-CG no espaço de estados contínuos, que naturalmente já é mais custoso sob o ponto de vista do tempo, não atingiu bons resultados em um tempo viável. Novamente destaca-se que para esses modelos não se conhece os espaços de estados a priori. Outra vantagem da coordenação *Market-driven* é que as funções de avaliação são criadas com algum conhecimento sobre o estudo do caso, podendo ser modeladas para cada ação

específica. No caso da coordenação por grafos se tenta obter uma ação ótima através de uma heurística, detalhada no Capítulo 5.

A estratégia para o futebol é fundamental, como já mencionado. Por esse motivo, foram testadas várias configurações de papéis e ações para conseguir uma estratégia de jogo boa o suficiente para superar os adversários. A mesma estratégia foi usada para os dois times, ou seja, os mesmos papéis e as mesmas ações. Analisando os resultados obtidos, observou-se que com essa mesma estratégia o time LRI-MD conseguiu fazer o dobro de gols que time LRI-CG. Constata-se que a estratégia de jogo e o modelo de coordenação têm uma grande importância para o sucesso no jogo de futebol robótico.

É importante mencionar que a estratégia do time benchmark MattiHetero é mais elaborada que a desenvolvida para os modelos RL-NFHP-MA-MD e RL-NFHP-MA-CG.

Não foi encontrado na literatura nenhum modelo de coordenação com *Reinforciment Learning* com espaço de estados contínuo, todos discretizam previamente o campo em grid. Os resultados obtidos em ambiente contínuo podem ser considerados satisfatórios já que foram similares aos obtidos em grid.

O campo com espaço de estados discreto (grid) tem uma grande vantagem frente ao campo com espaço contínuo para ambos os modelos propostos, porque os estados em grid são conhecidos previamente. Isso na prática leva também a um menor custo computacional (tempo de processamento).

No teste da jogada de defesa o time se mostrou bastante eficiente com os dois modelos de coordenação, mas foi mais eficiente para o time LRI-CG. Uma possível explicação para o baixo desempenho do time LRI-MD, na jogada de defesa, é uma inadequada especificação das funções de avaliação dos papéis de defesa do time. O modelo LRI-MD tem uma dificuldade adicional que é a escolha dos pesos para as funções, já que alterando os pesos, como é mostrado no teste 2, muda-se o comportamento do time. Portanto, uma melhor especificação da função de avaliação pode melhorar o desempenho do time LRI-MD em jogadas de defesa. Este ponto merece um estudo mais aprofundado, e por isso será tratado em trabalhos futuros.

# 7
# Conclusões

Este trabalho apresentou dois novos modelos de coordenação para Sistemas Multi-Agentes integrados a modelos neuro-fuzzy hierárquicos. O primeiro modelo, denominado RL-NFHP-MA-MD, é baseado em leilões. O segundo modelo, RL-NFHP-MA-CG, é baseado em coordenação por grafos.

O principal objetivo desta tese foi criar modelos de coordenação para sistemas multiagentes que apresentassem um bom desempenho em ambientes complexos. Isso permite que agentes trabalhem em conjunto de forma harmônica e ordenada, melhorando sua comunicação, explorando melhor uma determinada tarefa e acelerando sua conclusão.

A escolha do algoritmo RL-NFHP-MA como base do novo modelo, realizada após a etapa de levantamento bibliográfico, foi motivada especialmente pela importância de se estender a autonomia e aprendizado de agentes através do quesito inteligência, e pela sua capacidade de superar limitações presentes em algoritmos de aprendizado por reforço tradicionais.

Nesta tese se decidiu alocar papéis aos agentes ao invés de alocar tarefas; essa estratégia permite tirar proveito das condições atuais do ambiente, otimizando o desempenho dos agentes em ambientes dinâmicos.

Os modelos apresentados integraram a coordenação *Market Driven* e coordenação por grafos ao modelo RL-NFHP-MA. A ideia central desses modelos é ter uma estrutura hierárquica onde no primeiro nível de hierarquia o modelo RL-NFHP-MA aprende o papel do agente. Após a escolha do papel, no segundo nível de hierarquia as coordenações MD e CG escolhem, para cada modelo, a ação ao ser executada pelo agente.

Foram desenvolvidos dois caso de estudo, o jogo presa-predador e o futebol robótico. O futebol robótico foi escolhido para demonstrar o desempenho do modelo em ambientes complexos.

No primeiro caso de estudo os dois modelos apresentados neste trabalho foram superiores em desempenho ao modelo original RL-NFHP-MA em todos os testes, mostrando a eficácia dos modelos. A fase de treinamento, para os dois

modelos, foi mais custosa que no modelo original, devido ao fato de conter duas etapas.

No segundo caso de estudo (futebol robótico) foi desenvolvido um time de futebol que usa como modelo de coordenação os dois modelos apresentados nesta tese. O time desenvolvido foi testado contra times usados como benchmark em diferentes trabalhos da literatura. Os dois modelos de coordenação tiveram um bom desempenho, ganhando dos times usados como benchmark em quase todos os testes. Os resultados mostraram que os modelos apresentados neste trabalho são capazes de superar técnicas convencionais utilizadas no futebol robótico.

Entre os dois modelos apresentados o que apresentou os melhores resultados, em todos os testes, além de ter um custo computacional menor, é RL-NFHP-MA-MD, mostrando-se um modelo flexível e robusto.

As principais contribuições deste trabalho são, portanto:

- Introdução na família RL-NFH do conceito de coordenação multiagente para ambientes dinâmicos e complexos, com a proposição e teste de dois novos modelos de coordenação voltado para ambientes multiagentes;
- Desenvolvimento de aplicações SMA com agentes heterogêneos. Isso pode ser observado nos casos de estudo realizados, onde foram abordados problemas em diferentes áreas de natureza competitiva e cooperativa simultaneamente;
- A proposição e teste de dois novos modelos de coordenação voltado para ambientes multiagentes, dinâmicos e complexos, que pode vir a ser usado em diversas aplicações e problemas distintos. Como contribuição secundária destaca-se um levantamento bibliográfico extenso na área de coordenação de Sistemas Multiagentes, tema recente e promissor.

Em relação a trabalhos e perspectivas futuras, que podem ser explorados a partir do desenvolvimento desta tese, podem ser consideradas:

- Otimizar os pesos das funções de avaliação do modelo RL-NFHP-MA-MD com um algoritmo que possa trabalhar integrado ao treinamento;
- Otimizar o modelo para trabalhar em espaço contínuo, melhorando os resultados e tentando diminuir os problemas de custo computacional;
- Aplicar os modelos desenvolvidos em outras aplicações multiagente como controle de navegação, sistemas de negociação e trading, gerenciamento de recursos, etc;
- Estender o modelo para explorar as propriedades de escalabilidade e contingência dos SMA, através da introdução e exclusão automática de novos agentes sempre que houver aumento/redução de demanda ou falha nos agentes do sistema.

# 8

# Referências bibliográficas


ABUL, O.; POLAT, F. and ALHAJJ, R. "Multiagent reinforcement learning using function approximation". **IEEE TSMC**, Vol. 30, No. 4, 2000.

BAKKER, B.; STEINGROVER, M.; SCHOUTEN, R.; NIJHUIS, E. and KESTER, L. "Multiagent Reinforcement Learning for Urban Traffic Control using Coordination Graphs". In: Proceedings of the Nineteenth European Conference on Machine Learning, pp. 656-671, September 2008

BALAJI P. G. and SRINIVASAN D. "Distributed geometric fuzzy multi-agent urban traffic signal control". **IEEE Trans. Intell. Transport. Syst**., vol. 11, no.3, pp.714-727, 2010.

BANERJEE and PENG, J. "Adaptive policy gradient in multiagent learning". In: Proc. 2nd Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-03), Melbourne, Australia, p.686-692, 14-18 Jul. 2003.

BEAUMONT, P. and BRAHIM, C. "Multiagent Coordination Techniques for Complex Environments". The Case of a Fleet of Combat Ships. **IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews**, pp. 373-385, vol. 37, no. 3, May 2007.

BERNDT, J. O. and HERZOG, O. "Efficient Multiagent Coordination in Dynamic Environments". IEEE / WIC / ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT) Lyon, France. **IEEE Computer Society**, pp. 188-195, 2011.

BOWLING, M. and VELOSO, M. "Multiagent learning using a variable learning rate". **Artificial Intelligence**, vol. 136, no. 2, p. 215-250, 2002.

BRENNER, W.; ZARNEKOW, R. and WITTIG, H. "Intelligent Software Agents". **Springer**, 1998.

BROOKS, R.A. "Intelligence without representation". **Artificial Intelligence**, 47, p. 139-159, 1991.

BROWN, G.W. "Iterative solutions of games by fictitious play". In: Activitiy Analysis of Production and Allocation, T.C. Koopmans, Ed. New York: Wiley, ch. XXIV, p. 374-376, 1951.

BUSONIU, L.; BABUSKA, R. and DE SCHUTTER, B. "A Comprehensive Survey of Multiagent Reinforcement Learning". In: Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, Volume: 38, Issue: 2, Mar. 2008.

CALVO, R. e ROMERO, R.A.F. “Arquitetura Híbrida inteligente para navegação autônoma de robôs”. Dissertação de Mestrado em Ciências da Computação e Matemática Computacional - Universidade de São Paulo, 2007.

CHALKIADAKIS, G. “Multiagent reinforcement learning: Stochastic games with multiple learning players”. Dept. of Comput. Sci., Univ. Toronto. Toronto, ON, Canada, Tech. Rep. [Online]. Available: http://www.cs.toronto.edu/~gehalk/DepthReport/DepthReport.ps, Mar. 2003.

CHOI, S.P.M. and YEUNG, D.-Y. “Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control”. In: Proc. Adv. Neural Inf. Process. Syst. (NIPS-95), Denver, CO, vol. 8, p. 945-951, Nov. 1995.

CLAUS and BOUTILIER, C. “The dynamics of reinforcement learning in cooperative multiagent systems”. In: Proc. 15th Nat. Conf. Artif. Intell. 10th Conf. Innov. Appl. Artif. Intell.igence, Madison, WI, p.746-752, Jul. 1998.

CLOUSE, J. “Learning from an automated training agent”. Presented at the Workshop Agents that Learn from Other Agents, 12th Int. Conf. Mach. Learn. (ICML-95), Tahoe City, CA, Jul. 1995.

CONITZER, V. and SANDHOLM, T. “AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents”. In: Proc. 20th Int. Conf. Mach. Learn. (ICML-03), Washington, DC, p. 83-90, Ago. 2003.

CRITES, R.H. and BARTO, A.G. “Elevator group control using multiple reinforcement learning agents”. **Mach. Learning**, vol.33, no. 2-3, p. 235-262, 1998.

DAVIDSON E.M. and MCARTHUR S.D.J. “Applying multi-agent system technology in practice: Automated management and analysis of SCADA and digital fault recorder data”. **IEEE Trans. Power Syst**., vol. 21, no.2, pp. 559-567, May 2006.

DEBENHAM, J.K. "An Intelligent Multiagent System for the Management of Strategic Business Processes". In: Proceedings Seventh IFIP/IEEE International Symposium on Integrated Network Management (IM 2001), 14-18 May 2001.

DECKER, K.S. and LESSER, V.R. “Designing a family of coordination algorithms”. In: Proceedings of the First International Conference on Multi-Agent Systems, Menlo Park, California, p. 73-80, AAAI Press, Jun. 1995.

DURFEE, E.H. “Blissful ignorance: Knowing just enough to coordinate well”. In: Proceedings of the First International Conference on MultiAgent Systems (ICMAS95), Menlo Park, California, p. 406-413, AAAI Press, Jun. 1995.

EGUCHI; HIRASAWA, T.; HU, K. and J. OTA, N. “A study of evolutionary multiagent models based on symbiosis. **IEEE Transactions on Systems, Man, and Cybernetics, Part B**, pp. 179-193, 2006.

FENSTER, M.; KRAUS, S. and ROSENSCHEIN, J.S. “Coordination without communication: Experimental validation of focal point techniques”. In: Proceedings of the First International Conference on MultiAgent Systems (ICMAS95), Menlo Park, California, p. 102-108, AAAI Press, Jun. 1995.

FERNANDEZ, F. and PARKER, L.E. “Learning in large cooperative multirobot domains”. **Int. J. Robot. Autom**., vol. 16, no. 4, p. 217-226, 2001.

FIGUEIREDO, K.; VELLASCO; M.M.B.R. and PACHECO, M.A.C. “Novos Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço para Agentes Inteligentes”. Tese de Doutorado, Departamento de Engenharia Elétrica Pontifícia Universidade Católica do Rio de Janeiro, Fev, 2003.

FININ, T.; MCKAY, D.; FRITZSON, R. and McENTIRE, R. “Kqml: An information and knowledge exchange protocol”. In: Knowledge Building and Knowledge Sharing, Ohmsha and IOS Press, 1994.

FITCH, R.; HENGST, B.; SUC, D.; CALBERT, G. and SCHOLZ, J.B. “Structural abstraction experiments in reinforcement learning”. In: Proc. 18th Aust. Joint Conf. Artif. Intell. (AI-05), Lecture Notes in Computer Science, vol. 3809, Sydney, Australia, p. 164-175, Dez. 2005.

FRANÇA, M.C. “Modelos Neuro-Fuzzy Hierárquicos com Aprendizado por Reforço para Multiagentes Inteligentes”. Tese Doutorado em Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, 2011

GENESERETH, M.R. and FIKES, R.E. “Knowledge interchange format, version 3.0 reference manual”. **Technical Report Logic921**, Computer Science Department, Stanford University, 1992.

GOOD, B.M. “Evolving multi-agent systems: Comparing existing approaches and suggesting new directions”. Masters’s thesis, University of Sussex, 2000.

GOODRICH, M.A. and QUIGLEY, M. “Satisficing Q-Learning: Efficient Learning in Problems with Dichotomous Attributes”. In: Proceedings of ICML-A-2004, Louisville, Kentucky, Dez. 2004.

GOODWINE, B. and ANTSAKLIS, P. “Multiagent coordination exploiting system symmetries”. **American Control Conference (ACC)**, Topic(s): Robotics & Control Systems, Page(s): 830-835, 2010 ,

GREENWALD, A. and HALL, K. “Correlated-Q learning”. In: Proc. 20th Int. Conf. Mach. Learn. (ICML-03), Washington, DC, p. 242-249, Ago. 2003.

GUESTRIN, C.; LAGOUDAKIS, M.G. and PARR, R. “Coordinated reinforcement learning”. In: Proc. 19th Int. Conf. Mach. Learn. (ICML-02), Sydney, Australia, p. 227-234, Jul. 2002.

HADDADI, A. “Communication and Cooperation in Agent Systems: A Pragmatic Theory”. **Springer-Verlag**, Berlin, Heidelberg, and New York, 1995.

HAO, J.Y. and LEUNG, H.F., 2012. “Maintaining Cooperation in Homogeneous Multi-agent System”. In: Proceedings 2012 IEEE International Conference on Systems, Man, and Cybernetics, Seoul, Korea, 14-17. IEEE, 301-306. October 2012.

HARATI, A.; AHMADABADI, M.N. and ARAABI, B.N. “Knowledge-Based Multiagent Credit Assignment: a Study on Task Type and Critic Information”. **Systems Journal IEEE**, vol. 1, Issue 1, p. 55-67, Set. 2007.

HLADEK, D. “Multiagent control of the robotic soccer using fuzzy logic”. In: 7th PhD Student Conference and Scientific and Technical Competition of Students of Faculty of Electrical Engineering and Informatics Technical University of Košice: Proceeding from conference and competition: Košice, 23.5.2007. Košice: TU, p. 107-108, 2007.

HOLLAND, O. “Multiagent systems: Lessons from social insects and collective robotics”. In: Adaptation,Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium, Menlo Park,CA, p. 57-62, AAAI Press, Mar. 1996.

HSU, W.-T. and SOO, V.W. “Market performance of adaptive trading agents in synchronous double auctions”. In: Proc. 4th Pacific Rim Int. Workshop Multi-Agents. Intell. Agents: Specification Model. Appl. (PRIMA-01). LNCS Series, vol. 2132, Taipei, Taiwan, R.O.C., p. 108-121, Jul. 2001.

HU, J. and WELLMAN, M.P. “Multiagent reinforcement learning: Theoretical framework and an algorithm”. In: Proc. 15th Int. Conf. Mach. Learn. (ICML-98), Madison, WI, p. 242-250, Jul. 1998.

HUBER, M.J. and DURFEE, E.H. “Deciding when to commit to action during observation based coordination”. In: Proceedings of the First International Conference on MultiAgent Systems (ICMAS95), Menlo Park, California, p.163-170, AAAI Press, Jun. 1995.

HUI, W. and Li, K.K. “An adaptive multiagent approach to protection relay coordination with distributed generators in industrial power distribution system”. **IEEE Trans. Ind. Appl**., vol. 46, no. 5, pp. 2118-2124, Oct. 2010.

HWANG, K.S.; CHEN, Y.J. and LEE, C.H. “Reinforcement learning in strategy selection for a coordinated multirobot system”. **IEEE Trans. Syst., Man, Cybern. A, Syst., Humans**, vol. 37, no. 6, pp. 1151-1157, Nov. 2007

ISHIWAKA, Y.; SATO, T. and KAKAZU, Y. “An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning”. **Robotics and Autonomous Systems**, 43(4): 245-256 (2003)

JAAKKOLA, T.; JORDAN, M. and SINGH, S.P. “On the convergence of stochastic iterative dynamic programming algorithms”. **Neural Computation**, vol. 6, no. 6, p. 1185-1201, 1994.

JIANXIN, W. and YIGUANG, H. “Multi-agent coordination of networked mobile agents with hierarchical dynamic graph” WCICA.2008.4594217. In: Proceeding of: Intelligent Control and Automation, 2008. WCICA 2008.

JOUFFE, L. “Fuzzy Inference System Learning by Reinforcement Methods”. **IEEE Transactions on Systems, Man and Cybernetics, Part C**, vol. 28, n. 3, p. 338-355, 1998.

KAELBLING, L.P.; LITTMAN, M.L. and MOORE, W.A. “Reinforcement Learning: A Survey”. **Journal of Artificial Intelligence Research**, 4, p. 237-285, Mai. 1996.

KAPETANAKIS, S. and KUDENKO, D. “Reinforcement learning of coordination in cooperative multi-agent systems”. In: Proc. 18th Nat. Conf. Artif. Intell. 14th Conf. Innov. Appl. Artif. Intell. (AAAI/IAAI-02), Menlo Park, CA, p. 326-331, Jul. 28 – Aug. 1, 2002.

KARRAY F.; BASIR, O. and SONG, I.. “A Framework for Coordinated Control of Multi-Agent Systems and its Applications”. **IEEE Transactions on Systems Man and Cybernetics, Part A**, vol. 8, no. 3, 2008

KITANO, H. Editor. “RoboCup-97: Robot Soccer World Cup I”. **Springer Verlag**, Berlin, 1998.

KITANO, H.; MINORU, A.; YASOU, K.; ITSUKI, N. and EIICHI, O. “RoboCup: The Robot World Cup Initiative”. In: Proc. of IJCAI-95 Workshop on Entertainment and AI/Alife, p.19-24, 1995.

KOK, J.R.; J.’T HOEN, P.; BAKKER, P.B. and VLASSIS, N. “Utile coordination: Learning interdependencies among cooperative agents”. In: Proc. IEEE Symp. Comput. Intell. Games, Colchester, U.K., p. 29-36, Abr. 2005.

KONONEN, V. “Asymmetric multiagent reinforcement learning”. In: Proc. IEEE/WIC Int. Conf. Intell. Agent Technol. (IAT-03), Halifax, NS, Canada, p. 336-342, Out. 2003.

KOSE, H.; MERICLI, C.; KAPLAN, K. and AKIN, H.L. “All Bids for One and One Does for All: Market-Driven Multi-Agent Collaboration in Robot Soccer Domain”. In: Proceedings of Computer and Information Sciences-ISCIS 2003, 18th International Symposium, pp. 529-536, LNCS 2869, Antalya, Turkey, November 2004.

LAUER, M. and RIEDMILLER, M. “An algorithm for distributed reinforcement learning in cooperative multi-agent systems”. In: Proc. 17th Int. Conf. Mach. Learn. (ICML-00), Stanford Univ., Stanford, CA, p. 535-542, Jun. 29-Jul. 2 2000.

LEE, J.W. and OO, J.“A multi-agent Q-learning framework for optimizing stock trading systems”. In: Proc. 13th Int. Conf. Database Expert Syst. Appl. (DEXA-02). Lecture Notes in Computer Science, vol. 2453, Aixen-Provence, France, p. 153-162, Set. 2002.

LEE, J.W.; PARK, J.; LEE, J. and HONG, E. “A Multiagent Approach to Q-Learning for Daily Stock Trading”. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, vol. 37, No. 6, Nov. 2007.

LEVY, R. and ROSENSCHEIN, J.S. “A game theoretic approach to the pursuit problem”. In: Working Papers of the 11th International Workshop on Distributed Artificial Intelligence, p. 195-213, Fev. 1992.

LIANG and ZHUANG, W. "DTCoop: delay tolerant cooperative communications in DTN/WLAN integrated networks". In: Proc. IEEE VTC'10-Fall, pp. 1-5, Sept. 2010

LITTMAN, M.L. “Markov games as a framework for multi-agent reinforcement learning”. In: Proc. 11th Int. Conf. Mach. Learn. (ICML-94), New Brunswick, NJ, p. 157-163, Jul. 1994.

LITTMAN, M.L. “Value-function reinforcement learning in Markov games”. **J. Cogn. Syst. Res**., vol. 2, no. 1, p. 55-66, 2001.

LUKE, S.; HOHN, C.; FARRIS, J.; JACKSON, G. and HENDLER, J. “Co-evolving soccer softbot team coordination with genetic programming”. In: Proceedings of the First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997.

LUKE, S. “Genetic programming produced competitive soccer softbot teams for RoboCup97”. In: J.R. Koza et al, editor, Genetic Programming 1998: Proceedings of the Third Annual Conference, p. 214-222. Morgan Kaufmann, 1998.

MAES, P. “Intelligent Software”. **Scientific American**, vol. 273, No. 3, 6, p. 84-8, Set. 1995.

MATARIC, M.J. “Learning to behave socially". In: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, p. 453-462, 1994.

MENG Y. and KAZEEM O. "A Hybrid ACO/PSO Control Algorithm for Distributed Multi-Agent Systems", **IEEE Swarm Intelligence Symposium**, 2007.

MIYAZAKI, K. and KOBAYASHI, S. “Rationality of Reward Sharing in Multi-agent Reinforcement Learning”. **New Generation Computing**, vol. 19, No. 2, p. 157-172, 2001.

MONSIEURS, P. “Evolving Virtual Agents using Genetic Programming”. PhD thesis, **Transnationale Universiteit Limburg**, Diepenbeek, Belgium, 2002.

MÜLLER, J.P. “The design of intelligent agents: a layered approach”. **Lecture Notes in Computer Science**, Springer-Verlag, Heidelberg, vol. 1177, 1996.

NAEINI, A.T. and PALHANG, M. “Evolving a multiagent coordination strategy using Genetic Network Programming for pursuit domain”. In: Proceeding of:

Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China

NASH, J. "The Bargaining Problem". **Econometrica, Econometric Society**, vol. 18(2), p. 155-162, April, 1950. Cybernetics

NERI, J.R.F. “A Proposal of QLearning to Control the Attack of a 2D Robot Soccer Simulation Team”. **Robotics Symposium and Latin American Robotics Symposium (SBR-LARS)**, 2012 Brazilian

NODA, I. “Soccer server: a simulator of robocup”. In: Proceedings of AI symposium’95. **Japanese Society for Artificial Intelligence**, p. 29-34, Dez. 1995.

NWANA, H. “Software Agents: An Overview”. **Knowledge Engineering Review**, vol. 11, No. 3, October/November, p. 205-244, 1996.

NWANA, H. and AZARMI, N. “Software Agents: and Soft Computing”. **Springer**, 1997.

OLFATI-SABER, R. “Flocking for multi-agent dynamic systems: Algorithms and theory”. **IEEE Trans. Autom. Control**, vol. 51, no. 3, pp. 401-420, Mar. 2006.

OO, J.; LEE, J.W. and ZHANG, B.-T. “Stock trading system using reinforcement learning with cooperative agents”. In: Proc. 19th Int. Conf. Mach. Learn. (ICML-02), Sydney, Australia, p. 451-458, Jul. 2002.

PANAIT, L. and LUKE, S. “Cooperative Multi-Agent Learning: The State of the Art”. **Autonomous Agents and Multi-Agent Systems**, 11(3): 387-434, 2005.

POWERS, R. and SHOHAM, Y. “New criteria and a new algorithm for learning in multi-agent systems”. In: Proc. Adv. Neural Inf. Process. Syst. (NIPS-04), Vancouver, BC, Canada, vol. 17, p. 1089-1096, Dez. 2004.

PRICE, B. and BOUTILIER, C. “Accelerating reinforcement learning through implicit imitation”. **J. Artif. Intell. Res**., vol. 19, p. 569-629, 2003.

QINGGUO, L.I. and PAYANDEH, S. “Multi-agent cooperative manipulation with uncertainty: a neural net-based game theoretic approach”. In: Proceedings - IEEE International Conference on Robotics and Automation, 2003

RAMANI, G. and SUBRAMANIAN, R. “Strategies of Teams in Soccerbots”. **International Journal of Advanced Robotic Systems**, Vol. 5, No. 4 (2008)

RAJU, C.; NARAHARI, Y. and RAVIKUMAR, K. “Reinforcement learning applications in dynamic pricing of retail markets”. In: Proceedings of IEEE Conference on Electronic Commerce, CEC-2003, New Port Beach, California, p.339-346, Jun. 2003.

REIS, L. P. and LAU N. “FC Portugal - High-level Coordination Methodologies in Soccer Robotics”. **International Journal of Advanced Robotic Systems: Soccer Robotics**, Edited by Pedro Lima, 2007.

RIBEIRO, C.H.C. “A Tutorial on Reinforcement Learning Techniques”. In: International Joint Conference on Neural Networks ed.: INNS Press, 1999.

RIEDMILLER, M.A.; MOORE, A.W. and SCHNEIDER, J.G. “Reinforcement learning for cooperating and communicating reactive agents in electrical power grids”. In: Balancing Reactivity and Social Deliberation in Multi-Agent Systems, M. Hannebauer, J. Wendler, and E. Pagello, Eds. NY: Springer, p.137-149, 2000.

RUMMERY, G. and NIRAJAN, M. "On-line Q-learning using connectionist sytems". **Technical Report CUED/FINFENG-TR 166**, Cambridge University, UK, 1994.

RUSSEL, S.J. and NORVIG, P. “Artificial Intelligence: A Modern Approach”. 3rd edition, Prentice Hall, 2010.

SAGHIR, H. and MEGHERBI, D.B. “An Efficient Comparative Machine Learning-based Metagenomics Binning Technique Via Optimal Feature-reduction”. **Methods International Conference on Bioinformatics and Computational Biology (BICoB)**, Honolulu, Hawai, March 4-6 2013

SAVKIN A.V. and TEIMOORI H. “Decentralized Navigation of Groups of Wheeled Mobile Robots with Limited Communication”. **IEEE Transactions on Robotics**, 26(10), pp. 1099-1104, 2010.

SCHAERF, A.; SHOHAM, Y. and TENNENHOLTZ, M. “Adaptive load balancing: A study in multiagent learning”. **Journal of Artificial Intelligence Research**, vol. 2, p. 475-500, 1995.

SELLNER, B.P. and HEGER, F. "Coordinated Multi-Agent Teams and Sliding Autonomy for Large-Scale Assembly". In: Proceedings of the IEEE - Special Issue on Multi-Robot Systems, vol. 94, No. 7, July 2006.

SHOHAM, Y.; POWERS, R. and GRENAGER, T. “Multi-agent reinforcement learning: A critical survey”. **Comput. Sci. Dept., Stanford Univ**., Stanford, CA, Tech.Rep.Available:http://multiagent.stanford.edu/papers/MALearning_ACritical Survey_2003_0516.pdf, Mai. 2003.

SIMON, H. “The Sciences of the Artificial”. **MIT Press**, 3rd edition, 1996.

SINGH, S.; KEARNS, M. and MANSOUR, Y. “Nash convergence of gradient dynamics in general-sum games”. In: Proc. 16th Conf. Uncertainty Artif. Intell. (UAI-00), San Francisco, CA, p. 541-548, Jun-Jul 2000.

SRINIVASAN, D. and WONG, D. "Multi-agent coordination for DER in MicroGrid". **Sustainable Energy Technologies**, ICSET 2008. IEEE International Conference on, pp. 77-82, 24-27 Nov. 2008.

STEPHAN, V.; DEBES, K.; GROSS, H.-M.; WINTRICH, F. and WINTRICH, H. “A reinforcement learning based neural multi-agent-system for control of a combustion process”. In: Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks. (IJCNN-00), Como, Italy, p. 6217-6222, Jul. 2000.

STONE, P. and VELOSO, M. “Multiagent systems: A survey from the machine learning perspective”. **Auton. Robots**, vol. 8, no. 3, p. 345-383, 2000.

SUEMATSU, N. and HAYASHI, A. “A multiagent reinforcement learning algorithm using extended optimal response”. In: Proc. 1st Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-02), Bologna, Italy, p. 370-377, Jul. 2002.

SUTTON, R.S. and BARTO, A.G. “Reinforcement Learning: An Introduction”. Cambridge, MA: MIT Press, 1998.

SUTTON, R.S. “Generalization in Reinforcement learning: Succesful examples using sparse coarse coding”. In: Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E., editors, Advances in Neural Information Processing Systems 8, p.1038-1044, MIT Press, 1996.

TAN, M. “Multi-agent reinforcement learning: Independent vs. cooperative agents”. In: Proc. 10th Int. Conf. Mach. Learn. (ICML-93), Amherst, OH, p. 330-337, Jun. 1993.

TESAURO, G. “Extending Q-learning to general adaptive multi-agent systems”. In: Proc. Adv. Neural Inf. Process. Syst. (NIPS-03), Vancouver, BC, Canada, vol. 16, Dez. 2003.

TESAURO, G. and KEPHART, J.P. “Pricing in agent economies using multiagent Q-learning”. **Auton. Agents Multi-Agent Syst**., vol. 5, no. 3, p. 289-304, 2002.

TILLOTSON, P.; WU, Q. and HUGHES, P. “Multi-agent learning for routing control within an Internet environment”. **Eng. Appl. Artif. Intelligence**, vol. 17, no. 2, p. 179-185, 2004.

TOUZET, C.F. “Robot awareness in cooperative mobile robot learning”. **Auton. Robots**, vol. 8, no. 1, p. 87-97, 2000.

TUMER, K. and AGOGINO, A.K.; “Improving Air Traffic Management with A Learning Multiagent System”. **Intelligent Systems**, 24(1), IEEE, Computer Society, 2009.

TUMMOLINI, L.; CASTELFRANCHI, C.; RICCI, A.; VIROLI, M. and OMICINI, A. "Exhibitionists" and "voyeurs" do it better: A shared environment approach for flexible coordination with tacit messages”. **Lecture Notes in Artificial Intelligence**, 3374, p. 215-231, 2005.

VLASIS, N. “Aconcise introduction tomultiagent systems and distributed AI”. Fac. Sci. Univ.Amsterdam, Amsterdam, The Netherlands, Tech. Rep. [Online]. Available: http://www.science.uva.nl/vlassis/cimasdai/cimasdai.pdf, Set. 2003.

VLASIS, N. “Collaborative multiagent reinforcement learning by payoff propagation”. **Journal of Machine Learning Research**, 7: 1789-1828, 2006.

VON NEUMANN, J. “Zur theorie der gesellschaftsspiele” Math. Annalen. 100: 295-320, 1928. English translation: Tucker, A.W. and Luce, R.D. "On the Theory of Games of Strategy". **Contributions to the Theory of Games**, v. 4, p. 13-42, 1959.

WANG, X. “Planning while learning operators”. In: Proc. of the Third International Conference on AI Planning Systems, Mai. 1996.

WANG, X. and SANDHOLM, T. “Reinforcement learning to play an optimal Nash equilibrium in team Markov games”. In: Proc. Adv. Neural Inf. Process. Syst. (NIPS-02), Vancouver, BC, Canada, vol. 15, p. 1571-1578, Dez. 2002.

WATKINS, C.J.C.H. and DAYAN, P. “Q-learning”. **Mach. Learn**., vol. 8, p. 279-292, 1992.

WEINBERG, M. and ROSENSCHEIN, J.S. “Best-response multiagent learning in non-stationary environments”. In: Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-04), New York, NY, p.506–513, Ago. 2004.

WEISS, G. and SEN, S. “Adaptation and Learning in Multi-Agent Systems”. In: IJCAI "95 workshop, Montréal, Canada, August 21, 1995. Berlin: Springer, 238 p, 1996.

WEISS, G. “Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence”. Edited by Gerhard Weiss, 1999.

WELLMAN, M.P.; GREENWALD, A.R.; STONE, P. and WURMAN, P.R. “The 2001 trading agent competition”. **Electron. Markets**, vol. 13, no. 1, p. 4-12, 2003.

WIERING, M. “Multi-agent reinforcement learning for traffic light control”. In: Proc. 17th Int. Conf. Mach. Learn. (ICML-00), Stanford Univ., Stanford, CA, p. 1151-1158, Jun-Jul 2000.

WOLPERT, D.H. and TUMER, K. “An Introduction to Collective Intelligence”. **Technical Report NASA-ARC-IC-99-63**, NASA Ames Research Center. URL: http://ic.arc.nasa.gov/ic/projects/coin pubs.html. To appear in Hand-book of Agent Technology, Ed. J. M. Bradshaw, AAAI/MIT Press, 1999.

WOLPERT, D.H. and TUMER, K. “Optimal payoff functions for members of collectives”. **Advances in Complex Systems**, p. 265-279, 2001.

WOLPERT, D.H.; TUMER, K. and FRANK, J. “Using collective intelligence to route internet traffic”. **Advances in Neural Information Processing Systems**, 11, p. 952-958. MIT Press, 1999.

XIAO, D. and AH-HWEE TAN. “Self-Organizing Neural Architectures and Cooperative Learning in Multi-Agent Environment”. **IEEE Transactions on Systems, Man, and Cybernetics**, 2007.

YANG, E. and GU, D. “Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey”. **CSM-404**, Technical Reports of the Department of Computer Science, University of Essex, 2004.

ZHOU, P. and HONG, B. "On-line profit sharing algorithm". **International Journal of Electrical and Computer Engineering**, p. 424-430, 2006.

ZINKEVICH, M. “Online convex programming and generalized infinitesimal gradient ascent”. In: Proc. 20th Int. Conf. Mach. Learn. (ICML-03), Washington, DC, p. 928-936, Ago. 2003.

ZLOTKIN, G. and ROSENSCHEIN, J.S. “Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains”. In: Proceedings of the Twelfth National Conference on Artificial Intelligence, Menlo Park, California, p. 432-437, AAAI Press, Ago. 1994.