



Pós-Graduação em Ciência da Computação

## "Patrulha Multi-Agente com Aprendizagem por Reforço"

Por

*Hugo Pimentel de Santana*

Dissertação de Mestrado



Universidade Federal de Pernambuco  
posgraduacao@cin.ufpe.br  
[www.cin.ufpe.br/~posgraduacao](http://www.cin.ufpe.br/~posgraduacao)

RECIFE, 02/2005



UNIVERSIDADE FEDERAL DE PERNAMBUCO  
CENTRO DE INFORMÁTICA  
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HUGO PIMENTEL DE SANTANA

## "Patrulha Multi-Agente com Aprendizagem por Reforço"

*ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM  
CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA  
UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO  
PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA  
DA COMPUTAÇÃO.*

ORIENTADOR: PROF. DR. GEBER LISBOA RAMALHO (CIN-UFPE)  
CO-ORIENTADOR: PROF. DR. VINCENT CORRUBLE (LABORATOIRE D'INFORMATIQUE DE  
PARIS VI)

RECIFE, FEVEREIRO/2005

## RESUMO

A tarefa de patrulha pode ser encontrada em diferentes domínios, desde administração de redes de computadores a simulações de jogos de guerra. Esta é uma tarefa multi-agente complexa, que requer que os agentes participantes coordenem as suas tomadas de decisão de modo a obter um bom desempenho para o grupo como um todo.

Neste trabalho, é mostrado de que maneira a tarefa da patrulha pode ser modelada como um problema de aprendizagem por reforço (AR), permitindo uma adaptação contínua e automática das estratégias dos agentes ao ambiente. Nós demonstramos que um comportamento cooperativo eficiente pode ser obtido utilizando técnicas padrão de AR, como *Q-Learning*, para treinar os agentes individualmente.

É feita uma análise detalhada da optimalidade das soluções propostas e os resultados obtidos constituem um caso de estudo positivo no uso de técnicas de aprendizagem por reforço em sistemas multi-agentes. As reflexões e técnicas apresentadas são igualmente valiosas para outros problemas que compartilham propriedades similares.

Além disto, a abordagem proposta é totalmente distribuída, o que a torna computacionalmente eficiente. A avaliação empírica comprova a eficácia da mesma, e torna este trabalho uma primeira abordagem de sucesso na obtenção de uma estratégia adaptativa para tal tarefa.

Palavras-chave: sistemas multi-agentes, aprendizagem por reforço, coordenação, patrulhamento

# ABSTRACT

Patrolling tasks can be encountered in a variety of real-world domains, ranging from computer network administration and surveillance to computer wargame simulations. It is a complex multi-agent task, which usually requires agents to coordinate their decision-making in order to achieve optimal performance of the group as a whole.

In this work, we show how the patrolling task can be modeled as a reinforcement learning (RL) problem, allowing continuous and automatic adaptation of the agents' strategies to their environment. We demonstrate that an efficient cooperative behavior can be achieved by using RL methods, such as Q-Learning, to train individual agents.

It is given a detailed treatment on the analysis of the optimality of the solutions proposed and the results obtained constitute a positive case study on the use of standard reinforcement learning techniques in multi-agent systems. The insights and techniques presented are equally valuable for other problems with similar properties.

The proposed approach is totally distributed, what makes it computationally efficient. The empirical evaluation proves the effectiveness of our approach, which constitutes a first successful approach of an adaptive strategy for this task.

Keywords: multi-agent systems, reinforcement learning, coordination, patrolling

## AGRADECIMENTOS

Agradeço primeiramente à minha família, principalmente aos meus pais e irmãos. Em especial, agradeço a meu pai por ter me dado um computador quando tinha 11 anos de idade, e ao meu tio Eduardo por me ensinar a usá-lo. Sem querer, eles foram os responsáveis por todo o meu interesse e admiração pela ciência da computação até hoje.

À Patrícia, agradeço o carinho de todos os momentos juntos: do apoio nas horas difíceis à celebração das nossas vitórias.

Ao orientador e amigo Geber, por ter me dado a oportunidade de trabalhar com ele quando ainda estava nos meus primeiros anos de universidade, pela oportunidade de “trabalhar” nas áreas de jogos e música e, é claro, pelos mais de 4 anos de orientação!

A Vincent Corruble e Bohdana Ratitch, pela ajuda fundamental no desenvolvimento deste trabalho.

A todo o pessoal da D’Accord, por terem me ensinado que trabalhar pode ser algo divertido e estimulante, e a Márcio Dahia, pelas intermináveis discussões sobre os mais diversos assuntos.

Aos demais professores do CIn, em particular a Jacques Robin e Kátia Guimarães, por serem mais do que apenas bons professores.

Finalmente, agradeço a todos os meus amigos da universidade, em especial ao time de programação HSiG (Gatis, Sobral, Mozart, Danzi e Guilherme), e aos companheiros do saudoso MaracatuRFC (RodBarros, RodCunha, Cléa, Ernesto, André) pela amizade, por tudo que aprendemos juntos e pela companhia nas madrugadas e fins de semana na faculdade.

# ÍNDICE ANALÍTICO

|   |           |
|---|-----------|
| <b>1 Introdução.....</b>  | <b>1</b>  |
| 1.1 Estrutura desta Dissertação.....                                      | 3         |
| <b>2 A Tarefa da Patrulha Multi-Agente .....</b>                          | <b>4</b>  |
| 2.1 Descrição do Problema.....  | 4         |
| 2.2 Patrulha como um Problema de Coordenação Multi-Agente.....            | 8         |
| 2.3 Abordagens Anteriores.....  | 9         |
| 2.3.1 Abordagem Pioneira: Arquiteturas Simples .....                      | 9         |
| 2.3.2 Agentes Baseados em Utilidade .....                                 | 10        |
| 2.3.3 Agentes Negociadores .....  | 11        |
| 2.3.4 Estratégia do Ciclo Único .....                                     | 11        |
| 2.3.5 Novas abordagens .....  | 12        |
| 2.4 Conclusões.....   | 12        |
| <b>3 Aprendizagem por Reforço .....</b>                                   | <b>14</b> |
| 3.1 Visão Geral do Framework .....  | 14        |
| 3.2 Processo de Decisão de markov (MDP) .....                             | 14        |
| 3.2.1 Componentes de um MDP .....   | 15        |
| 3.2.2 Q-Learning .....  | 16        |
| 3.2.3 Processo de Decisão Semi-Markoviano (SMDP).....                     | 18        |
| 3.3 Coordenação em Sistemas Multi-Agentes e Aprendizagem por Reforço .... | 19        |
| 3.3.1 Aprendizes Independentes e de Ação Conjunta.....                    | 19        |
| 3.3.2 Inteligência Coletiva: Framework COIN.....                          | 20        |
| 3.3.3 Aprendizagem por Reforço Coordenada (ARC).....                      | 21        |
| 3.4 Conclusões.....   | 22        |
| <b>4 A Patrulha como um Problema de Aprendizagem por Reforço .....</b>    | <b>23</b> |
| 4.1 Uma Instância Simplificada .....                                      | 23        |
| 4.2 Considerando Distâncias Reais Entre os Nós.....                       | 28        |
| 4.3 O Caso Multi-Agente .....   | 29        |
| 4.4 Os Agentes Aprendizes .....   | 31        |
| 4.5 Conclusões.....   | 33        |
| <b>5 Implementações e Resultados .....</b>                                | <b>35</b> |

|          |   |           |
|----------|---|-----------|
| 5.1      | Implementações Realizadas.....                                      | 35        |
| 5.1.1    | <i>RL Engine</i> .....  | 35        |
| 5.1.2    | <i>Agentes Patrulhadores</i> .....                                  | 37        |
| 5.2      | Avaliação Empírica .....  | 37        |
| 5.2.1    | <i>Metodologia utilizada</i> .....                                  | 37        |
| 5.2.2    | <i>Avaliação da Representação de Estados</i> .....                  | 40        |
| 5.2.2.1  | Resultado Experimental.....   | 40        |
| 5.2.2.2  | Discussão .....   | 43        |
| 5.2.3    | <i>Avaliação das Funções de Recompensa</i> .....                    | 45        |
| 5.2.3.1  | Resultado Experimental.....   | 45        |
| 5.2.3.2  | Discussão .....   | 48        |
| 5.2.4    | <i>Comparação com Abordagens Anteriores</i> .....                   | 50        |
| 5.2.4.1  | Resultado Experimental.....   | 50        |
| 5.2.4.2  | Discussão .....   | 52        |
| 5.3      | Conclusões.....   | 54        |
| <b>6</b> | <b>Experimentos Adicionais.....</b>                                 | <b>56</b> |
| 6.1      | Generalização do Conhecimento Aprendido .....                       | 56        |
| 6.2      | Especialização como Comportamento Emergente .....                   | 58        |
| 6.3      | Visão Simbólica do Conhecimento Aprendido .....                     | 62        |
| <b>7</b> | <b>Conclusão .....</b>  | <b>66</b> |
| 7.1      | Principais Contribuições.....                                       | 66        |
| 7.2      | Trabalhos Futuros .....   | 67        |
| 7.2.1    | <i>Aprendizagem por Reforço Coordenada na Patrulha</i> .....        | 67        |
| 7.2.2    | <i>Desenvolvimento de um novo simulador</i> .....                   | 67        |
| 7.2.3    | <i>União da abordagem CLinG com os agentes deste trabalho</i> ..... | 68        |
|          | <b>Referências.....</b>   | <b>69</b> |

# ÍNDICE DE FIGURAS

|  |    |
|--|----|
| Figura 1. Mapas A e B no simulador. Os pontos brancos (nós) representam locais específicos do mapa, e as conexões em azul (arestas) representam possíveis caminhos.....  | 4  |
| Figura 2. Ilustração da estratégia de ciclo único para dois agentes. Os agentes seguem o ciclo determinado mantendo um <i>gap</i> de distância entre eles. ....  | 12 |
| Figura 3. Visão geral de um problema de aprendizagem por reforço .....   | 14 |
| Figura 4. Pseudo-Código do algoritmo <i>Q-Learning</i> . $\#S$ e $\#A$ representam o número de elementos dos conjuntos $S$ e $A$ , respectivamente.....  | 17 |
| Figura 5. Exemplo de Grafo com Pesos. O agente (em A) deve escolher o caminho (A,B,C) ou (A,C,B) .....   | 29 |
| Figura 6. Ilustração de agentes <i>Black-Box</i> e <i>Gray-Box</i> na patrulha. Os agentes gray-box comunicam as intenções de ações e incluem as intenções dos outros em suas representações de estados. ....                          | 33 |
| Figura 7. Diagrama de Classes (UML) do RLEngine .....  | 36 |
| Figura 8. Etapas da metodologia. Fases de aprendizagem e avaliação são intercaladas até haver convergência, e o resultado passa por uma etapa final de avaliação. ....   | 39 |
| Figura 9. Cenários de simulação utilizados. (a) e (b) são grafos muito e pouco conectados, respectivamente. (c), (d), (e) e (f) representam topologias de círculo, corredor, ilhas e grid, respectivamente.....                        | 39 |
| Figura 10. Avaliação da média da ociosidade média (eixo Y), e seus respectivos desvios padrão, de arquiteturas com diferentes representações de estado. Quanto menor o valor da métrica para a arquitetura, melhor a performance. .... | 42 |
| Figura 11. Exemplo da coordenação que emerge no caso BBLA: o grafo é particionado em áreas disjuntas, com agentes especializados em áreas específicas do grafo (destacadas) .....  | 44 |
| Figura 12. Avaliação da média da ociosidade média das diferentes funções de recompensa aplicadas às arquiteturas BBLA e GBLA.....  | 47 |
| Figura 13. Avaliação da ociosidade quadrática média das diferentes funções de recompensa aplicadas às arquiteturas BBLA e GBLA.....  | 47 |
| Figura 14. Avaliação da ociosidade média durante cada sub-etapa de treinamento. A ociosidade é bastante alta no início e diminui com o aprendizado dos agentes. ....   | 48 |
| Figura 15. Comparação da média da ociosidade média das abordagens com uma população de 5 agentes. ....   | 51 |



|  |    |
|--|----|
| Figura 16. Comparação da média da ociosidade média das abordagens com uma população de 15 agentes. Os primeiros resultados da CR são 73.71 e 107.01, respectivamente.....              | 51 |
| Figura 17. Média da ociosidade média normalizada com seus respectivos desvios-padrão. Os resultados são uma média para todos os mapas.....   | 52 |
| Figura 18. Comparação da performance dos agentes GGBLA em todos os cenários, com o cenário de treinamento indicado entre parênteses. ....  | 57 |
| Figura 19. Comparação de performance do agente GGBLA treinado no mapa Ilhas com o agente GBLA (treinado especificamente para cada mapa) e o agente HPCC.....                           | 58 |
| Figura 20. Frequência de visitas a cada nó feitas por cada agente, por arquitetura. ....   | 60 |
| Figura 21. Desvio padrão do valor da frequência de visitas por nó (eixo Y), normalizado, calculado para cada agente (eixo X) das arquiteturas BBLA, GBLA, GGBLA, Aleatória e HPCC..... | 61 |
| Figura 22. Exemplo de regra aprendida pelo agente GGBLA. ....  | 63 |
| Figura 23. Comparação de agentes GGBLA que utilizam regras com diferentes níveis de detalhe.....   | 64 |

## ÍNDICE DE TABELAS

|   |    |
|---|----|
| Tabela 1 - Resumo das principais características das arquiteturas propostas por Aydano Machado .....                | 9  |
| Tabela 2 – Comparação da aplicabilidade das arquiteturas considerando restrições de comunicação e visibilidade..... | 54 |

# ÍNDICE DE EQUAÇÕES

|   |    |
|---|----|
| Equação 1. Ociosidade do nó $k$ , no instante $t$ .....   | 5  |
| Equação 2. Ociosidade instantânea do grafo .....  | 5  |
| Equação 3. Ociosidade média instantânea do grafo .....  | 5  |
| Equação 4. Pior ociosidade instantânea do grafo .....   | 5  |
| Equação 5. Média da ociosidade média .....  | 6  |
| Equação 6. Média da pior ociosidade .....   | 6  |
| Equação 7. Máxima ociosidade média .....  | 6  |
| Equação 8. Máxima pior ociosidade .....   | 6  |
| Equação 9. Ociosidade quadrática média .....  | 7  |
| Equação 10. Média da Ociosidade Média Normalizada .....   | 7  |
| Equação 11. Função valor dos estados .....  | 15 |
| Equação 12. Função valor das ações .....  | 16 |
| Equação 13. Regra de atualização da função $Q$ pelo algoritmo $Q$ -Learning. ....   | 16 |
| Equação 14. Função de decaimento do parâmetro $\alpha$ .....  | 18 |
| Equação 15. Regra de atualização da função $Q$ em SMDPs .....   | 18 |
| Equação 16. Média da ociosidade média do ciclo $I$ até $T$ .....  | 24 |
| Equação 17. Equação de recorrência da ociosidade média instantânea do grafo .....   | 25 |
| Equação 18. Função valor do agente caso $R$ seja a ociosidade média instantânea .....                                       | 25 |
| Equação 19. Equação da ociosidade média instantânea do grafo em forma fechada .....   | 26 |
| Equação 20. Equação da Média da Ociosidade Média em termos das funções $Pos$ e $\phi$ .....                                 | 26 |
| Equação 21. Subexpressão da ociosidade média dependente da política do agente .....   | 26 |
| Equação 22. Função valor do agente caso $R$ seja dado por $\Phi(Pos(t), t)$ .....   | 27 |
| Equação 23. Sub-expressão da ociosidade média no caso multi-agente, cuja minimização<br>resulta em uma política ótima ..... | 30 |
| Equação 24. Função de recompensa instantânea WTG de um agente $j$ no tempo $t$ . ....                                       | 46 |

# 1 INTRODUÇÃO

Sistemas Multi-Agentes (SMAs) são sistemas formados por um grupo de agentes autônomos que interagem em um determinado ambiente [22]. As tecnologias, métodos e teorias dos SMAs têm se mostrado extremamente relevantes e contribuído fortemente em diversos domínios de aplicações, tais como recuperação de informação, interface homem-máquina, comércio eletrônico, robótica, jogos de computadores, educação e treinamento, computação ubíqua, simulação social, entre outros [53] .

No contexto dos problemas cuja solução pode utilizar uma abordagem multi-agente, destacamos neste trabalho a **tarefa da patrulha** [31][32][33]. Literalmente, patrulhar é o “ato de percorrer uma área, em intervalos regulares, de maneira a protegê-la ou supervisá-la” [1]. A execução desta tarefa eficientemente é necessária em diversas aplicações práticas como, por exemplo, para detecção de falhas ou situações específicas em uma rede de computadores [6], para identificação de novas páginas WEB a serem indexadas por engines de busca [16][17], para identificação de objetos ou pessoas que precisam ser resgatados por robôs [38], ou no controle de entidades virtuais em jogos de computadores que precisam executar tal tarefa [21][12].

Além disto, uma solução eficiente para esta tarefa necessita de uma extrema coordenação entre os agentes. Sendo assim, tal tarefa se apresenta como um estudo de caso interessante para a pesquisa em SMAs, e os resultados obtidos na investigação de soluções para este problema podem se mostrar válidos para outros problemas com propriedades similares.

Apesar da sua alta aplicabilidade, apenas recentemente este problema foi abordado seriamente [36][42][43]. Neste sentido, pesquisadores do Centro de Informática da UFPE e do *Laboratoire d'informatique de Paris 6* têm realizado um trabalho pioneiro na investigação de tal problema e de suas possíveis soluções. Inicialmente [31][32][33], foi feito um estudo sistemático do problema e das suas aplicações, e diversas estratégias para sua resolução foram propostas. Nestes mesmos trabalhos, foram definidos critérios de avaliação do desempenho de soluções, e se construiu um simulador para realizar tais avaliações. Em trabalhos posteriores [2][3][4][14], soluções mais sofisticadas foram propostas e avaliadas, e instâncias mais complexas do problema foram tratadas. Mais recentemente, foi estudado o problema de patrulhar um terreno real com robôs [44]. Além destes trabalhos, no momento da redação

deste documento está em andamento um trabalho visando à utilização de mecanismos de negociação entre agentes para definição de estratégias de patrulhamento [36].

Embora muitas destas abordagens tenham demonstrado bons resultados empíricos, foi observado que, para todas as arquiteturas propostas, havia sempre uma instância do problema (uma topologia particular do ambiente, por exemplo) na qual a estratégia não apresentava um bom desempenho [4]. Uma das razões para este fato é que, neste domínio, é muito difícil projetar a priori uma estratégia coletiva com boa performance e geral o suficiente, dado que tal tarefa requer uma extrema coordenação entre as ações dos agentes.

Neste sentido, técnicas adaptativas como aprendizagem de máquina [37] poderiam ser bastante úteis, provendo uma maneira de se **obter coordenação automaticamente**. A aprendizagem por reforço [47], em particular, já utilizada com sucesso em outros domínios tais como futebol de robôs [38], jogos de computadores [25], entre outros, se apresenta como uma boa candidata para servir a este propósito. De fato, foi motivada pelo potencial desta técnica que esta dissertação foi desenvolvida.

O objetivo deste trabalho é investigar a criação de agentes adaptativos que aprendem a patrulhar utilizando técnicas de aprendizagem por reforço (AR). A aplicação de tais técnicas, neste caso, não é direta. Primeiramente, o uso da maioria dos algoritmos de AR envolve a modelagem do problema como um Processo de Decisão de Markov (MDP) [47], no entanto, a formulação original deste formalismo só se aplica ao caso em que há apenas um único agente aprendiz, e as extensões para seu uso em SMAs ainda são bastante incipientes. Entre os outros desafios que tiveram de ser enfrentados neste trabalho, cabe citar:

- i) Como definir um modelo de recompensas instantâneas dentro deste formalismo que leve a uma estratégia coletiva satisfatória a longo termo?
- ii) Como treinar efetivamente tais agentes?
- iii) Como aprender um conhecimento generalizável para instâncias diferentes do problema?
- iv) Como analisar tais resultados, não apenas quantitativamente, mas também do ponto de vista dos diversos comportamentos emergentes que podem surgir como resultado da aprendizagem coletiva?

Tais desafios, e possíveis caminhos para resolvê-los, serão discutidos e analisados neste trabalho.

## ***1.1 ESTRUTURA DESTA DISSERTAÇÃO***

O restante desta dissertação está estruturado da seguinte maneira. O capítulo 2 descreve a tarefa da patrulha, e traz um resumo dos esforços feitos anteriormente para resolver este problema.

O capítulo 3 descreve o *framework* da aprendizagem por reforço, bem como os trabalhos já feitos no sentido de permitir sua utilização em Sistemas Multi-Agentes.

O capítulo 4 propõe uma modelagem do problema da patrulha como um problema de aprendizagem por reforço, discutindo possíveis maneiras de construir tal modelo.

O capítulo 5 traz a avaliação experimental das alternativas de criação dos agentes aprendizes, e compara os resultados obtidos com os trabalhos anteriores.

O capítulo 6 analisa os resultados obtidos não mais de uma maneira quantitativa, mas do ponto de vista dos comportamentos emergentes do sistema e da análise dos resultados obtidos.

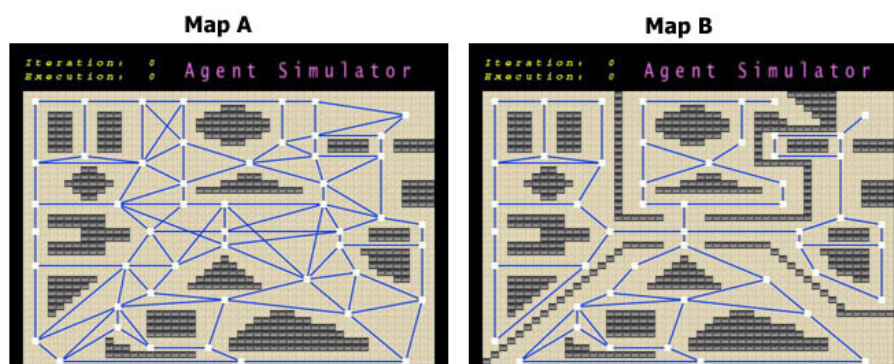
Por fim, o capítulo 7 traz as conclusões deste trabalho e aponta possíveis trabalhos futuros.

## 2 A TAREFA DA PATRULHA MULTI-AGENTE

Este capítulo se divide em três seções principais. Na seção 2.1, será dada uma definição mais formal da tarefa da patrulha, e serão definidos os principais conceitos relativos a esta tarefa, bem como critérios para avaliação de estratégias de patrulhamento. Na seção 2.2, situaremos a patrulha no contexto dos problemas de coordenação em SMAs. Na seção 2.3, será feito um resumo das abordagens propostas anteriormente a este trabalho. Finalmente, as principais conclusões deste capítulo são apresentadas na seção 2.4.

### 2.1 DESCRIÇÃO DO PROBLEMA

A tarefa da patrulha é definida como o ato de percorrer continuamente um determinado ambiente, visitando-se locais específicos do mesmo, de maneira a detectar, o mais rapidamente possível, determinadas situações que possam vir a ocorrer. De modo a obter uma representação mais abstrata, mesmo assim útil e precisa deste ambiente, os trabalhos anteriores [31][33] definiram a noção de patrulhamento em grafos, onde o ambiente a ser patrulhado é representado por um grafo cujos nós representam locais específicos (i.e. pontos de boa visibilidade, estações de trabalho em uma rede, etc.) e as arestas representam um caminho entre 2 nós. Tal representação permite uma fácil generalização do problema de terrenos e mapas para redes de computadores, por exemplo. A Figura 1 mostra um exemplo de tal representação, no simulador desenvolvido nestes mesmos trabalhos anteriores.



**Figura 1. Mapas A e B no simulador. Os pontos brancos (nós) representam locais específicos do mapa, e as conexões em azul (arestas) representam possíveis caminhos.**

Outro conceito desenvolvido foi o de *ociosidade*: o intervalo de tempo decorrido entre duas visitas consecutivas a um mesmo nó. Considerando  $Pos_j(t)$  como sendo o nó em que o agente  $j$  se encontra no instante  $t$ , e  $N$  o número de nós do grafo, a Equação 1 descreve a ociosidade compartilhada<sup>1</sup>,  $\phi$ , de um nó  $k$  no instante  $t$ :

$$\Phi(k, t) = t - m, \text{ onde:} \quad (\text{Equação 1})$$

$$m = \max \{s \mid s \leq t \wedge \exists j, Pos_j(s) = k\}$$

### **Equação 1. Ociosidade do nó $k$ , no instante $t$**

Desta maneira, formula-se um objetivo, ainda intuitivo, porém mais preciso para esta tarefa, como sendo a minimização das ociosidades dos nós do grafo em questão. Finalmente, foram definidos critérios precisos e quantitativos de avaliação de estratégias de patrulhamento. A seguir mostramos definições importantes que podem ser utilizadas na avaliação de uma estratégia de patrulhamento, a saber:

- *Ociosidade instantânea ( $I(t)$ )* do grafo – somatório das ociosidades dos nós do grafo em um determinado instante, definida pela Equação 2;

$$I(t) = \sum_{k=1}^N \phi(k, t) \quad (\text{Equação 2})$$

### **Equação 2. Ociosidade instantânea do grafo**

- *Ociosidade média instantânea ( $Med(t)$ )* do grafo – média das ociosidades dos nós do grafo em um determinado instante, definida pela Equação 3;

$$Med(t) = \frac{I(t)}{N} \quad (\text{Equação 3})$$

### **Equação 3. Ociosidade média instantânea do grafo**

- *Pior ociosidade instantânea ( $Max(t)$ )* do grafo – ociosidade máxima do grafo em um determinado instante, definida pela Equação 4;

$$Max(t) = \max_{k \in [1, N]} \phi(k, t) \quad (\text{Equação 4})$$

### **Equação 4. Pior ociosidade instantânea do grafo**

---

<sup>1</sup> Chamamos de *ociosidade compartilhada* a ociosidade calculada como o intervalo de tempo entre visitas consecutivas a um mesmo nó feitas por qualquer agente. A *ociosidade individual* é definida como o intervalo de tempo entre visitas consecutivas a um mesmo nó por um mesmo agente.



- *Média da Ociosidade média* ( $MedMed(t_i, t_f)$ ) e *Média da Pior ociosidade* ( $MedMax(t_i, t_f)$ ) – média dos valores da ociosidade média instantânea e da pior ociosidade instantânea, durante um determinado intervalo de tempo de simulação, definidas na Equação 5 e Equação 6, respectivamente;

$$MedMed(t_i, t_f) = \frac{\sum_{t=t_i}^{t_f} Med(t)}{t_f - t_i} \quad (\text{Equação 5})$$

#### Equação 5. Média da ociosidade média

$$MedMax(t_i, t_f) = \frac{\sum_{t=t_i}^{t_f} Max(t)}{t_f - t_i} \quad (\text{Equação 6})$$

#### Equação 6. Média da pior ociosidade

- *Máxima Ociosidade média* ( $MaxMed(t_i, t_f)$ ) e *Máxima Pior ociosidade* ( $MaxMax(t_i, t_f)$ ) – maior dos valores da ociosidade média instantânea e da pior ociosidade instantânea, durante um determinado intervalo de tempo de simulação, definidas na Equação 7 e Equação 8, respectivamente;

$$MaxMed(t_i, t_f) = \max_{t \in [t_i, t_f]} Med(t) \quad (\text{Equação 7})$$

#### Equação 7. Máxima ociosidade média

$$MaxMax(t_i, t_f) = \max_{t \in [t_i, t_f]} Max(t) \quad (\text{Equação 8})$$

#### Equação 8. Máxima pior ociosidade

- *Ociosidade quadrática média* ( $QuaMed(t_i, t_f)$ ) – Raiz quadrada da média dos quadrados das ociosidades instantâneas. Tal métrica enfatiza as performances da média da ociosidade média com alto desvio padrão, e é definida na Equação 9. Em outras palavras, tal métrica é uma medida de compromisso entre os critérios  $MedMed$  e  $MaxMed$ : duas estratégias com igual valor para o critério da média da ociosidade média podem ter valores diferentes para o critério  $QuaMed$ , com a estratégia com pior valor para a métrica  $MaxMed$  conseqüentemente apresentando também o pior valor para  $QuaMed$ .

$$QuaMed(t_i, t_f) = \sqrt{\frac{\sum_{t=t_i}^{t_f} Med(t)^2}{t_f - t_i}} \quad (\text{Equação 9})$$

### Equação 9. Ociosidade quadrática média

- *Média da Ociosidade Média Normalizada* ( $NorMed(t_i, t_f)$ ) – Média da ociosidade média normalizada. Esta normalização é feita para permitir a comparação entre simulações feitas com diferentes tamanhos de populações de agentes em grafos com diferentes números de nós, e é definida na Equação 10. Em outras palavras, ao se dividir o valor de  $MedMed$  pelo número de nós do mapa, obtém-se uma medida de desempenho independente do número de nós do mapa no qual a métrica foi recolhida, facilitando comparações entre simulações rodadas em cenários de simulação diferentes. A mesma idéia se aplica para simulações com tamanhos de população de agentes diferentes.

$$NorMed(t_i, t_f) = \frac{MedMed(t_i, t_f) \times (\#agentes)}{(\#nós)} \quad (\text{Equação 10})$$

### Equação 10. Média da Ociosidade Média Normalizada

As equações desenvolvidas nesta seção serão citadas, quando conveniente, no restante deste trabalho. Uma vez que definimos mais formalmente o problema, é importante frisar que muitas variações podem ser propostas para o mesmo. No entanto, quando mencionarmos a tarefa da patrulha no restante deste trabalho, estaremos tratando de uma instância simplificada do problema, que considera que:

- A população de agentes tem tamanho fixo e é homogênea;
- O grafo é estático (arestas não desaparecem ou são criadas, simulando portas abrindo e fechando, por exemplo);
- Todos os nós possuem a mesma prioridade de visitas.

Tal escolha foi feita por basicamente 2 razões principais. A primeira delas é manter o mesmo *setup* dos experimentos realizados nos trabalhos anteriores, permitindo uma análise comparativa. A segunda delas é que, por sua simplicidade, esta instância é mais apropriada para uma primeira análise do nosso problema de aprendizagem. Uma vez que tal instância tenha sido suficientemente entendida, pretende-se partir então para instâncias mais complexas do mesmo problema.

## 2.2 PATRULHA COMO UM PROBLEMA DE COORDENAÇÃO MULTI-AGENTE

O estudo do problema da coordenação em sistemas multi-agentes, por não ser puramente teórico, em geral é feito utilizando-se uma aplicação específica [45]. Entretanto, é importante entender como uma determinada aplicação se relaciona com outras, e até que ponto as soluções propostas para um determinado problema generalizam para outros problemas similares.

A tarefa da patrulha, em particular, pertence a uma classe de problemas denominados problemas de **coordenação espacial**. Nestes problemas, a tarefa de coordenação consiste em alocar os agentes a áreas específicas de um ambiente, de maneira sincronizada. *Sempé* [45] lista classes de tarefas que compartilham esta propriedade, a saber:

- *Foraging* – Consistem em recuperar objetos dispersos em um ambiente. Um exemplo de tarefa pertencente a esta categoria é o problema do *token collection*, onde um grupo de agentes devem recuperar moedas (*tokens*) dispersas em um ambiente, e a qualidade da solução é dada pelo valor das moedas recolhidas, e o tempo levado para o recolhimento [52];
- *Deslocamento em formação* – Um grupo de robôs formam uma figura (uma linha ou um triângulo, por exemplo) e mantêm esta forma enquanto realizam movimentos [9];
- *Cobertura* – Consiste em garantir que um determinado terreno é completamente explorado por um robô, ou um grupo de robôs [55];
- *Patrulha* – Consiste em percorrer um espaço recorrentemente, o que implica na repetição de visitas a uma mesma área periodicamente no tempo [33];

A escolha da tarefa da patrulha como objeto de estudo para o problema da coordenação em sistemas multi-agentes é motivada por vários fatores. Em primeiro lugar, a tarefa da patrulha se apresenta como uma atividade fundamental na realização de outras atividades mais complexas: por exemplo, em domínios como futebol de robôs ou simulações de jogos de guerra/estratégia exigem a realização de tarefas de patrulha [43]. Um outro ponto positivo é o fato desta tarefa possuir critérios de avaliação quantitativos e bem definidos, permitindo comparar abordagens e estimar a optimalidade de soluções. Finalmente, a tarefa da patrulha possui um nível de complexidade ajustável, ou seja, pode-se analisar o problema em instâncias simples, com agentes homogêneos e em terrenos estáticos, ou em instâncias bastante complexas, onde há diversos agentes que utilizam arquiteturas diferentes, o terreno é

dinâmico (há portas abrindo e se fechando, por exemplo), as ociosidades crescem com velocidades diferentes em determinadas áreas (regiões apresentam uma prioridade de visita maior que outras) entre outras variações.

## 2.3 ABORDAGENS ANTERIORES

Nesta seção, detalhamos os esforços já realizados na tentativa de resolver esta tarefa de maneira eficiente.

### 2.3.1 ABORDAGEM PIONEIRA: ARQUITETURAS SIMPLES

O estudo pioneiro da tarefa da patrulha, citado na seção 2.1, foi feito por Machado *et al* [31][32][33]. Neste trabalho, foram propostas diversas arquiteturas, baseadas em características como o tipo de processo de decisão utilizado (reativo ou cognitivo)<sup>2</sup>, as formas de comunicação permitidas, e as informações utilizadas na escolha do próximo nó a ser visitado, conforme mostra a Tabela 1.

| Nome                           | Tipo Básico | Comunicação | Escolha do próximo nó            |
|--------------------------------|-------------|-------------|----------------------------------|
| <i>Random Reactive</i>         | Reativo     | Nenhuma     | Aleatória                        |
| <i>Conscientious Reactive</i>  |             |             | Heurística baseada na ociosidade |
| <i>Reactive with Flags</i>     |             | Flags       |                                  |
| <i>Conscientious Cognitive</i> | Cognitivo   | Nenhuma     | Heurística baseada na ociosidade |
| <i>Blackboard Cognitive</i>    |             | Blackboard  |                                  |
| <i>Random Coordinated</i>      |             | Mensagens   | Aleatória                        |
| <i>Cognitive Coordinated</i>   |             |             | Heurística baseada na ociosidade |

**Tabela 1 - Resumo das principais características das arquiteturas propostas por Aydano Machado**

Analisando cada uma das arquiteturas mencionadas, percebe-se que as 3 primeiras arquiteturas da Tabela 1 (*Random Reactive*, *Conscientious Reactive* e *Reactive with Flags*) são reativas, ou seja, todas elas só consideram em sua tomada de decisão o próximo nó a ser visitado (nó vizinho), sem considerar um possível caminho a ser percorrido pelo agente. Desta

<sup>2</sup> O que diferencia as arquiteturas reativas e cognitivas é a presença ou não de um nó objetivo: a arquitetura reativa só toma uma decisão local, escolhendo qual vizinho imediato irá visitar, enquanto que as arquiteturas cognitivas escolhem um nó qualquer do grafo como seu objetivo e percorrem um caminho até ele.

maneira, o *Random Reactive* simplesmente escolhe um nó vizinho randomicamente, enquanto que o *Conscientious Reactive* e o *Reactive with Flags* escolhem o nó vizinho com maior ociosidade. A diferença entre os últimos é que o primeiro considera a ociosidade individual, enquanto que o segundo, por usar uma comunicação por *flags*<sup>3</sup>, utiliza a ociosidade compartilhada (ver seção 2.1).

Nas abordagens cujo processo de decisão é cognitivo, os agentes primeiro escolhem um nó objetivo (que pode ser qualquer nó do grafo), para depois percorrerem um caminho até ele, utilizando o algoritmo de menor caminho de Dijkstra [35]. Os agentes *Conscientious Cognitive* e *Blackboard Cognitive* não possuem coordenador, e cada agente individualmente escolhe como objetivo o nó do grafo com maior ociosidade individual e compartilhada, respectivamente. Os agentes *Random Coordinated* e *Cognitive Coordinated* possuem um elemento central de coordenação que toma as decisões por cada agente e as comunica através de troca de mensagens, evitando que dois agentes escolham um mesmo nó objetivo. Como o nome indica, o *Random Coordinated* escolhe o nó objetivo aleatoriamente (entre os nós que não são objetivos de nenhum outro agente), e o coordenador dos agentes *Cognitive Coordinated* escolhe o nó a ser visitado como sendo o de maior ociosidade compartilhada que não é objetivo de nenhum outro agente.

Os agentes desenvolvidos neste trabalho, embora bastante simples, serviram de base para os trabalhos posteriores nesta tarefa, em particular para os agentes baseados em utilidade, descritos na seção 2.3.2.

### **2.3.2 AGENTES BASEADOS EM UTILIDADE**

As heurísticas utilizadas pelos agentes pioneiros eram bastante simples: apenas a ociosidade do nó objetivo era utilizada, e o algoritmo de escolha do menor caminho para atingir o objetivo considerava que as arestas tinham o mesmo comprimento, e não levava em conta as ociosidades dos nós que fazem parte do próprio caminho. Sendo assim, o trabalho de Almeida *et al.* [4] aprimora os agentes pioneiros com heurísticas mais sofisticadas.

Tal melhoria foi feita basicamente de duas maneiras: i) as arquiteturas cuja escolha do nó objetivo era feita baseada na maior ociosidade agora passaram a utilizar uma função de utilidade que considera não apenas a ociosidade deste nó, mas a distância que o agente precisa percorrer para chegar no mesmo; ii) as arquiteturas que antes utilizavam um algoritmo de

---

<sup>3</sup> Na comunicação por *flags*, o agente pode deixar uma informação em um determinado nó que ele visita (*flag*), que pode ser vista por outros agentes.

escolha do menor caminho para atingir o nó objetivo, e cujo custo do caminho era dado pelo número de arestas (as arestas possuíam tamanho unitário), agora consideram uma função de utilidade que procura um caminho cujos nós por onde o agente passará estão mais ociosos, em compromisso com a minimização do comprimento total do caminho (que reflete no tempo para atingir o objetivo).

Dentre os agentes criados neste trabalho, destaca-se o *Heuristic Pathfinder Cognitive Coordinated*, uma evolução do *Cognitive Coordinated*, tendo obtido a melhor performance dentre os agentes propostos por Almeida *et al.*

### 2.3.3 AGENTES NEGOCIADORES

Neste trabalho [36], considera-se que cada agente é responsável por patrulhar um subconjunto dos nós do grafo, sendo estes subconjuntos disjuntos entre si. Do ponto de vista de cada agente, é desejável que, no conjunto de nós que ele possui, cada nó esteja o mais próximo um do outro quanto possível, diminuindo o tempo necessário para que o agente percorra seu conjunto de nós, minimizando, assim, a ociosidade dos nós sob sua responsabilidade e maximizando a função de utilidade do agente.

Desta maneira, nesta abordagem os agentes negociam seus recursos (nós), fazendo com que o conjunto de nós que cada agente precisa percorrer maximize sua utilidade. Inicialmente, os subconjuntos dos nós do grafo de cada agente são escolhidos aleatoriamente, e os agentes fazem uma série de negociações, implementadas como leilões de mercado, até que o sistema convirja para uma solução onde cada agente está satisfeito com o conjunto de nós que ele é responsável por patrulhar. As arquiteturas com melhores resultados são chamadas de *Two-Shot-Bidder Agent* e *Mediated Trade Bidder Agent*, que implementam esta abordagem sem e com o uso de um mediador, respectivamente.

### 2.3.4 ESTRATÉGIA DO CICLO ÚNICO

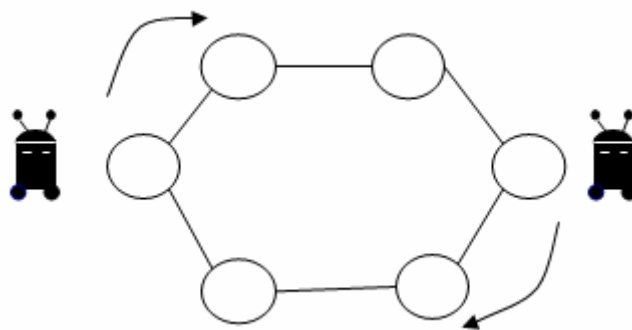
Neste trabalho [14][15], considera-se que os agentes são capazes de descobrir um ciclo<sup>4</sup> no grafo a ser patrulhado, e a tarefa de patrulhamento consiste em seguir este ciclo repetidamente. Tal abordagem é chamada de estratégia do ciclo único. Mostra-se que, no caso de um único agente, a estratégia ótima em termos do critério da Pior ociosidade máxima (ver seção 2.1) é a estratégia de ciclo único cujo ciclo é calculado como a solução ótima para o

---

<sup>4</sup> Definimos um ciclo como sendo um caminho fechado que começa e termina em um mesmo nó, passando por todos os nós do grafo, passando possivelmente mais de uma vez por um mesmo nó.

Problema do Caixeiro Viajante (*Travelling Salesman Problem* - TSP) [20][26], problema clássico já bastante estudado na área de pesquisa operacional.

Uma maneira de estender tal abordagem para o caso multi-agente é manter todos os agentes seguindo este mesmo ciclo, separados por uma distância similar e constante entre eles, conforme indicado na Figura 2.



**Figura 2. Ilustração da estratégia de ciclo único para dois agentes. Os agentes seguem o ciclo determinado mantendo um *gap* de distância entre eles.**

### 2.3.5 NOVAS ABORDAGENS

No momento do desenvolvimento deste trabalho, dois esforços relevantes foram feitos no sentido de desenvolver novas soluções para o problema da patrulha. *Sempé* [44][45] estudou o problema da patrulha em terrenos do mundo real, utilizando robôs. Para isto, foram utilizadas técnicas de navegação de robôs baseadas na teoria dos campos potenciais [7]. O algoritmo desenvolvido, chamado de estratégia CLinG, se baseia na **propagação** das informações globais sobre o estado do mundo (ociosidade dos nós, posição dos outros agentes, etc), de maneira que os agentes possam agir baseado apenas na informação do terreno (local), mas que ao mesmo tempo, esta mesma informação, resume o estado de todo o mapa (global).

Mais recentemente, em paralelo a este trabalho, um aluno do *Laboratoire D'Informatique de Paris VI* [19] está trabalhando na extensão dos agentes desenvolvidos nesta dissertação, combinando-os com a abordagem ClinG. Este trabalho é descrito em mais detalhes na seção 7.2.3.

## 2.4 CONCLUSÕES

As abordagens mostradas nesta seção apresentam alguns problemas do ponto de vista de aplicabilidade, dos quais destacamos dois principais. Primeiramente, observou-se que o desempenho relativo de uma determinada abordagem era extremamente dependente das

características do ambiente em que ela estava atuando, como, por exemplo, o tamanho da população, a topologia do grafo utilizado, etc. Sendo assim, uma abordagem que tinha um desempenho muito bom em um determinado mapa apresentava problemas em mapas com características diferentes [4][15].

Uma outra crítica que pode ser feita, também relacionada com o primeiro aspecto levantado, porém mais genérica, é a falta de adaptabilidade de determinadas abordagens. Por exemplo, a estratégia do ciclo único impõe uma coordenação fixa *a priori*, podendo assim apresentar problemas em ambientes dinâmicos, onde caminhos que eram possíveis anteriormente podem passar a conter obstáculos, ou em que os agentes precisem eventualmente desviar de seus cursos para recarregar baterias, por exemplo [44].

Todos os aspectos aqui levantados serviram como indícios de que técnicas de aprendizagem de máquina poderiam ser utilizadas com sucesso na busca por uma solução desta tarefa, não apenas do ponto de vista da obtenção automática de coordenação, mas também na obtenção de uma solução adaptativa por natureza, e que não sofresse dos problemas aqui citados. No próximo capítulo, revisaremos a Aprendizagem por Reforço, técnica escolhida neste trabalho para lidar com as questões levantadas nesta seção.



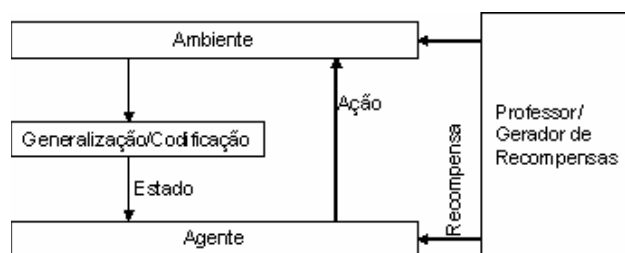
## 3 APRENDIZAGEM POR REFORÇO

Neste capítulo, mostraremos resumidamente os principais conceitos da Aprendizagem por Reforço, bem como os esforços que têm sido feitos para a sua utilização em Sistemas Multi-Agentes.

### 3.1 VISÃO GERAL DO FRAMEWORK

A aprendizagem por reforço não é caracterizada como uma técnica específica, e sim como um problema de aprendizagem. Tal problema de aprendizagem é definido como o de “aprender o que fazer (como mapear situações em ações) de maneira a maximizar um sinal numérico de recompensa” [47]. Em termos gerais, tem-se um agente que toma decisões baseado em uma determinada representação do ambiente (estado), e recebe um *feedback* sobre o resultado das suas ações (recompensa).

Diferentemente de outras formas de aprendizagem, na aprendizagem por reforço não se diz para o agente quais ações ele deve escolher, e a aprendizagem ocorre utilizando-se a própria interação com o ambiente. A Figura 3 ilustra a modelagem de um problema de aprendizagem por reforço em termos do agente, do ambiente e de um professor ou algum gerador automático de recompensas.



**Figura 3. Visão geral de um problema de aprendizagem por reforço**

Este *framework* é, em geral, definido em termos do formalismo dos processos de decisão de Markov (MDP), descrito na seção 3.2.

### 3.2 PROCESSO DE DECISÃO DE MARKOV (MDP)

Nesta seção, mostraremos os principais conceitos dos processos de decisão de Markov, bem como da sua versão mais geral, os processos de decisão Semi-Markovianos (SMDPs). Além disto, mostraremos *Q-Learning*, um dos principais algoritmos para solução de MDPs e SMDPs.

### 3.2.1 COMPONENTES DE UM MDP

Considere um agente que toma decisões seqüencialmente em um ambiente. A cada passo, este agente escolhe uma **ação** de um conjunto finito  $A$ , baseado em um sinal de **estado** do ambiente. Este estado pertence a um conjunto finito  $S$ , e incorpora as percepções atuais e passadas (eventualmente) de tal maneira que toda a informação relevante para a tomada da decisão está contida nele [37].

A dinâmica deste processo é descrita por dois componentes: a **distribuição de probabilidade da transição entre estados**,  $P$ , e a **função da recompensa instantânea esperada**,  $R$ . Estas duas funções são definidas no domínio das triplas  $\langle s, a, s' \rangle$ , onde  $s$  é o estado atual,  $a$  é a ação executada pelo agente e  $s'$  é o próximo estado no qual o agente irá estar após executar esta ação. Em outras palavras, a distribuição  $P$  é uma função que diz a probabilidade de se atingir o estado  $s'$  dado que o agente estava no estado  $s$  e executou a ação  $a$ , enquanto que a função  $R$  diz a recompensa esperada nesta mesma situação. Estas funções devem satisfazer a **propriedade de Markov**: o valor da função aplicada a uma determinada tripla  $\langle s, a, s' \rangle$  depende unicamente dos valores desta tripla, e não da seqüência de estados ou de ações anteriores do agente.

Em resumo, um **Processo de Decisão de Markov finito** é definido por uma tupla  $\langle S, A, P, R \rangle$  dos elementos definidos anteriormente. Neste formalismo, o agente age de acordo com alguma **política de ações**  $\pi(s, a)$ , que representa a probabilidade de escolha da ação  $a \in A$  quando o agente estiver no estado  $s \in S$ . O objetivo de um agente em um MDP é maximizar um critério de performance a longo prazo, chamado de **retorno**, que é geralmente definido como a soma das recompensas futuras descontadas. Mais formalmente, o agente tenta aprender, dentre o espaço de todas as políticas de ações possíveis, uma **política de ações ótima**  $\pi^*$ , que maximiza o retorno esperado, também chamado de **função de valor dos estados**,  $V^\pi(s)$ , mostrado na Equação 11.

$$V^\pi(s) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \mid s_t = s \right\},$$

(Equação 11)

onde  $\gamma$  é um **fator de desconto**,  $0 \leq \gamma < 1$ ,  $r_t$  é a recompensa instantânea recebida no passo  $t$ , e  $E_\pi$  representa o valor esperado da expressão entre chaves caso o agente siga a política de ações  $\pi$ .

#### Equação 11. Função valor dos estados

De maneira similar, define-se a **função de valor das ações**,  $Q^\pi(s, a)$ , mostrada na Equação 12 como o retorno esperado a partir do estado  $s$ , executando a ação  $a$ , e a partir daí seguindo a política de ações  $\pi$ .

$$Q^\pi(s, a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \mid s_t = s, a_t = a \right\} \quad (\text{Equação 12})$$

### Equação 12. Função valor das ações

Uma das vantagens de se trabalhar com tal função é que, uma vez aprendida uma função de valor das ações ótima  $Q^*(s, a)$ , a política de ações ótima pode ser construída de maneira gulosa: para cada estado  $s$ , a ação da política ótima neste estado (melhor ação) é aquela que maximiza  $Q^*(s, a)$ . A seção 3.2.2 mostra um algoritmo para aprendizagem da função  $Q$  ótima, *Q-Learning*.

#### 3.2.2 Q-LEARNING

*Q-Learning* [37][47][49] é um algoritmo tradicional de aprendizagem por reforço para resolução de MDPs. Uma das principais vantagens deste algoritmo, além da sua simplicidade, é o fato dele não precisar de um modelo do ambiente (*model-free*). Sendo assim, é possível utilizá-lo em um ambiente onde não se conhece *a priori* a distribuição de probabilidade da transição de estados  $P$  nem a função de recompensas esperadas instantâneas  $R$ .

Como o próprio nome indica, a idéia é computar a função valor das ações,  $Q^*(s, a)$ , ótima. É provado que a aprendizagem de tal função pode ser feita iterativamente, computando o valor de cada par estado-ação, *on-line*, utilizando a regra de atualização mostrada na Equação 13. Um pseudo-código da implementação deste algoritmo é mostrado na Figura 4 [37].

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r(s, a) + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a) \right], \quad (\text{Equação 13})$$

onde  $s$  e  $a$  são o estado e a ação atual, respectivamente,  $s'$  é o próximo estado e  $\alpha$  é a taxa de aprendizagem.

### Equação 13. Regra de atualização da função $Q$ pelo algoritmo *Q-Learning*.

**Entrada:**

*Matriz de valores reais  $Q$  ( $\#S$  linhas e  $\#A$  colunas),  
inicializada com valores zero.*

**Saída:**

*Matriz  $Q$  contendo a estimativa da função  $Q^*$ .*

**Algoritmo**

1. Observe o estado atual  $s$ ;
2. Para sempre, faça:
  - a. Execute uma ação  $a \in A$ ;
  - b. Observe a recompensa  $r$ ;
  - c. Observe o novo estado  $s'$ ;
  - d. Atualize  $Q[s][a]$  utilizando a regra de atualização da Equação 13;
  - e.  $s \leftarrow s'$

**Figura 4. Pseudo-Código do algoritmo  $Q$ -Learning.**  $\#S$  e  $\#A$  representam o número de elementos dos conjuntos  $S$  e  $A$ , respectivamente.

Uma questão em aberto na utilização deste algoritmo se refere a linha 2.a. Embora a única exigência do algoritmo com relação a escolha da ação é que cada ação seja escolhida um número suficiente de vezes (idealmente, infinito), caso este agente esteja preocupado com sua performance, ele precisará balancear a necessidade de **explorar** novas ações, ou de **usufruir** das ações que ele já sabe que darão um bom retorno em termos de recompensas. Este problema, conhecido como o dilema da exploração versus usufruto (*exploration versus exploitation*) [49], é bastante conhecido na literatura de aprendizagem por reforço, e diversas soluções já foram propostas. A mais simples delas é o método  $\epsilon$ -greedy [49], em que uma ação randômica é escolhida com probabilidade  $\epsilon$ , e o agente escolhe a melhor ação com probabilidade  $(1 - \epsilon)$ .

A taxa de aprendizagem  $\alpha$  da Equação 13 é utilizada pelo agente para que o novo valor estimado para a função  $Q$  seja uma combinação linear entre a última estimativa e a nova, ponderada por esta taxa. Uma taxa de aprendizagem igual a 0 (zero) significa que o

agente desconsiderará as novas estimativas de  $Q$ , e uma taxa de aprendizagem de 1 (um) significa que o valor antigo de  $Q$  será desconsiderado. Valores intermediários são utilizados para combinar as duas estimativas.

Para garantir convergência dos valores de  $Q$ , em um ambiente não-determinístico, é necessário decair a taxa de aprendizagem com o tempo. É provado [49] que, se tal decaimento for feito de acordo com a Equação 14, o algoritmo convergirá para estimativas ótimas da função  $Q$  também em tempo polinomial.

$$\alpha = \frac{1}{1 + \frac{\#visitas(s,a)}{alphaDecayRate}},$$

(Equação 14)

onde  $s$  e  $a$  é um par estado-ação,  $\alpha$  é a taxa de aprendizagem,  $\#visitas$  é uma função que diz quantas vezes o agente escolheu a ação  $a$  no estado  $s$ , e  $alphaDecayRate$  é uma taxa que varia a velocidade de decaimento do parâmetro  $\alpha$ .

#### Equação 14. Função de decaimento do parâmetro $\alpha$ .

### 3.2.3 PROCESSO DE DECISÃO SEMI-MARKOVIANO (SMDP)

MDPs convencionais não são capazes de representar abstrações temporais ou ações com extensão temporal: a ação tomada no tempo  $t$  afeta apenas o estado e a recompensa instantânea do tempo  $t+1$ . Esta restrição dificulta seu uso em aplicações cujas ações requerem intervalos de tempo diferentes (durações diferentes) para serem executadas, como na tarefa da patrulha, por exemplo, onde o tempo de duração da ação de percorrer uma determinada aresta do grafo depende do tamanho da mesma.

Processos de decisão Semi-Markovianos, ou SMDPs, [48], podem ser vistos como uma versão mais geral do formalismo MDP, onde as ações podem ter duração variável. Desta maneira, os fundamentos dos MDPs descritos na seção 3.2.1 continuam válidos nos SMDPs, mudando-se basicamente a maneira como a informação de tempo é interpretada, bem como a regra de atualização da função  $Q$ , utilizada por algoritmos como  $Q$ -Learning. A versão mais genérica desta regra de atualização é mostrada na Equação 15.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r(s,a) + \gamma^\tau \cdot \max_{a'} Q(s',a') - Q(s,a) \right],$$

(Equação 15)

onde  $s$  e  $a$  são o estado e a ação atual, respectivamente,  $s'$  é o próximo estado e  $\tau$  é o tempo de duração da ação  $a$ .

#### Equação 15. Regra de atualização da função $Q$ em SMDPs

Observa-se que, no caso em que todas as ações tem mesma duração, unitária, a definição da regra de atualização da função  $Q$  em SMDPs e MDPs é a mesma. Ressalta-se também que algoritmos para resolução de MDPs, tais como  $Q$ -Learning, são aplicáveis à resolução de SMDPs também em tempo tratável.

### 3.3 COORDENAÇÃO EM SISTEMAS MULTI-AGENTES E APRENDIZAGEM POR REFORÇO

A teoria tradicional de aprendizagem por reforço considera o caso em que há um único agente aprendiz. Em Sistemas Multi-Agentes, em particular em cenários cooperativos, é necessário garantir que haja coordenação entre os comportamentos aprendidos por cada agente. Uma maneira intuitiva de adaptar esta teoria para o caso multi-agente seria considerar todo o sistema de uma maneira centralizada, como um único agente [39]. No entanto, tal solução é claramente intratável, pois seria preciso considerar um número de ações que cresce exponencialmente com o aumento do número de agentes.

Em contraste a esta abordagem, existem diversas propostas de soluções distribuídas, no entanto, torna-se mais difícil a garantia de um comportamento global satisfatório. De fato, o problema de encontrar uma solução globalmente ótima para um grupo de agentes cooperativos com informação parcial já foi demonstrada como intratável em teoria [10], no entanto, muitas soluções propostas tem obtido bons resultados práticos [13][23][24][25][38][39][51]. Revisaremos nas próximas seções as abordagens distribuídas mais relevantes, no sentido de serem mais próximas dos objetivos deste trabalho.

#### 3.3.1 APRENDIZES INDEPENDENTES E DE AÇÃO CONJUNTA

As abordagens distribuídas existentes para sistemas multi-agentes cooperativos podem ser classificadas segundo o modo de interação entre os agentes [18]: de um lado, têm-se os aprendizes independentes (*independent learners* - IL), que são agentes que ignoram a existência dos outros agentes e escolhem suas ações desconsiderando o desempenho coletivo do grupo. Do outro lado, têm-se os aprendizes de ação conjunta (*joint action learners* - JAL), que sabem da existência de todos os outros agentes que atuam junto com ele.

Uma outra classificação, proposta por Riedmiller *et al* [39], divide as arquiteturas que interagem nos modos IL e JAL em 3 tipos distintos de arquiteturas. Adotaremos a mesma nomenclatura neste trabalho, a saber:

- *Agentes Black-Box* – correspondem aos agentes IL. Cada agente age de acordo com seu próprio processo de decisão de Markov, e só toma conhecimento da

existência dos outros agentes pela própria dinâmica do ambiente (os outros agentes influenciam nas suas funções  $P$  e  $R$ ). Não há comunicação explícita entre os agentes;

- *Agentes White-Box* – correspondem aos agentes JAL. Cada agente age coletivamente, aprendendo a escolher as melhores ações conjuntamente com os outros agentes.
- *Agentes Gray-Box* – Estes agentes também agem independentemente, de maneira similar ao *black-box*, no entanto, eles podem comunicar explicitamente, transmitindo informações sobre suas ações ou sobre suas **intenções de ações**.

Uma vez que até o momento da redação deste trabalho não existem algoritmos que tornem viável a utilização de agentes *White-Box* devido ao alto custo computacional dos mesmos, neste trabalho utilizaremos apenas agentes *Black-box* e *Gray-Box*.

### 3.3.2 INTELIGÊNCIA COLETIVA: FRAMEWORK COIN

Uma inteligência coletiva, ou Collective INtelligence (COIN) [51], é definida como um sistema multi-agente onde existe uma utilidade global (*world utility*) bem definida, e onde existe pouco ou nenhum controle centralizado. A teoria COIN busca atacar o seguinte problema de projeto: dado que os agentes que compõem este sistema conseguem maximizar suas próprias funções de utilidade (utilizando AR, por exemplo), que tipo de utilidade individual deve ser definida para cada agente de modo que, ao perseguir a maximização destas utilidades individuais, eles maximizem a utilidade global? Em outras palavras, de que maneira é possível, sabendo-se que os agentes são bons aprendizes individuais, garantir a qualidade do comportamento coletivo.

Para atingir tal objetivo, este *framework* busca satisfazer dois requisitos básicos. Em primeiro lugar, cada função de utilidade local deve estar “alinhada com a utilidade global”. Isto quer dizer que, se um agente está tentando maximizar sua utilidade local, necessariamente isto implica que a utilidade global será maximizada (não há conflito de interesses entre as utilidades dos agentes). Em segundo lugar, as funções de utilidade individuais devem ser possíveis de serem aprendidas (boa *learnability*), ou seja, o agente deve ser capaz de identificar os efeitos de suas ações e aprender a escolher aquelas que maximizam a sua utilidade. Um modelo de utilidade que satisfaz tais propriedades é chamado de *wonderful life utility* (WLU).

Embora tal modelo tenha sido utilizado com sucesso em problemas clássicos de coordenação multi-agente como roteamento de pacotes na internet [52] e no problema do Bar el farol [8], a sua utilização em tarefas que são resolvidas por uma **seqüência de ações** (ao invés de uma única ação de cada agente), como a patrulha, não é direta. O problema com estas tarefas é que não há uma correspondência direta entre a noção de utilidade e a noção de recompensa instantânea dos algoritmos de aprendizagem por reforço. As tentativas de aplicar este modelo de utilidade em problemas que são resolvidos por uma seqüência de ações ainda são bastante incipientes.

Uma aplicação desta abordagem foi feita no problema de *token collection* [27][54], descrito na seção 2.2. Nestes trabalhos, o modelo de recompensas WLU foi comparado aos modelos de recompensa egoísta (*selfish utility* - SU), onde cada agente tenta maximizar suas próprias recompensas individuais, e ao modelo *team-game* (TG), onde cada agente recebe como recompensa individual a utilidade global de todo o grupo.

As principais conclusões destes trabalhos foram que: i) a recompensa *team-game* apresentou baixa performance no problema aplicado, devido ao grande nível de ruído no sinal de aprendizagem. Em outras palavras, os agentes não conseguem discernir se uma determinada recompensa foi fruto de uma boa ação sua ou do restante do grupo, e a aprendizagem é extremamente lenta e pode não convergir. Este problema também é conhecido como o problema da atribuição de crédito [53]; ii) a recompensa WLU foi o melhor modelo nos trabalhos de *Wolpert et al* [54], e apresentou desempenho equivalente a recompensa SU em *Hoen et al* [27], a não ser quando foram utilizadas heurísticas particulares do domínio, onde neste caso a abordagem WLU obteve melhor resultado.

### 3.3.3 APRENDIZAGEM POR REFORÇO COORDENADA (ARC)

Aprendizagem por reforço coordenada [23][24] consiste em um método para coordenar ações de agentes de maneira a maximizar uma função valor global. Daremos nesta seção uma visão geral deste método.

Consideremos um grupo de  $G$  agentes, onde o  $j$ -ésimo agente pode escolher uma ação de um conjunto  $A_j$ , e observa um espaço de estados descrito por uma variável  $X_j$ . O estado completo do sistema é descrito por um vetor  $x \in X$ , onde  $X = (X_1, X_2, \dots, X_G)$  e a ação conjunta dos agentes pode ser dada por um vetor  $a \in A$ , onde  $A = (A_1, A_2, \dots, A_G)$ .

Consideremos também que este grupo de agentes deve escolher suas ações de maneira a maximizar a utilidade total do sistema. Em geral, esta utilidade global (que chamaremos de  $Q$ ), tem de ser definida no domínio de  $X$  e  $A$ , já que ela depende das ações de todos os



agentes. No entanto, em muitos problemas práticos, é possível aproximar a utilidade total  $Q$  pela soma das sub-utilidades locais (de cada agente)  $Q_j$ , onde  $Q = \sum_j Q_j$ .

Embora computar um vetor de ações que maximiza  $Q$  por esta fórmula pareça intratável<sup>5</sup>, este método explora a estrutura local das funções  $Q_j$  para computar a ação conjunta ótima eficientemente. A idéia é explorar esta localidade utilizando uma estrutura chamada de **grafo de coordenação**. Este grafo indica as influências entre as variáveis de ação de cada  $A_j$ . Isto é, se a ação  $a_j$  influencia  $a_k$ , haverá uma aresta conectando  $A_j$  e  $A_k$  no grafo de coordenação. Usando esta estrutura, e o algoritmo de eliminação de variáveis de redes Bayesianas [41], é demonstrado [23][24] de que maneira é possível escolher um conjunto de ações que maximiza  $Q$  eficientemente. Tal solução é exponencial apenas na largura induzida do grafo de coordenação, ao invés de ser no número de agentes como na formulação original.

Embora a utilização desta técnica no problema da patrulha se apresente como uma abordagem promissora, esta nunca foi utilizada em SMDPs, e tal adaptação requereria uma série de reflexões adicionais que, devido a restrições de escopo, não puderam ser lidadas neste trabalho. A seção 7.2.1, entretanto, sugere caminhos que os trabalhos futuros possam seguir na tentativa de utilizar esta abordagem no problema da patrulha.

### 3.4 CONCLUSÕES

Neste capítulo, foram mostrados os principais conceitos associados a aprendizagem por reforço: processos de decisão de Markov e Semi-Markovianos; *Q-Learning*, seu principal algoritmo; bem como dos principais esforços que já foram feitos até o momento para a utilização deste *framework* para obtenção de coordenação em sistemas multi-agentes, em particular naqueles em que há um cenário cooperativo.

Na próxima seção, mostraremos como a tarefa da patrulha pode ser modelada como um problema de aprendizagem por reforço, colocando-a no contexto deste *framework*, e permitindo a aplicação das técnicas de resolução de AR na mesma.

---

<sup>5</sup> Seria necessário experimentar todas as combinações de ações possíveis para cada agente, o que é exponencial no número de agentes.

## 4 A PATRULHA COMO UM PROBLEMA DE APRENDIZAGEM POR REFORÇO

Neste capítulo, discutiremos os possíveis caminhos para a modelagem da tarefa da patrulha como um problema de aprendizagem por reforço. Dado que algumas características particulares da tarefa podem ter um grande impacto na dificuldade de construção do modelo MDP, discutiremos tal modelagem de maneira incremental, a partir de instâncias mais simples do problema, até as mais complexas. Nesta discussão, serão inseridos os principais conceitos da abordagem proposta neste trabalho.

### 4.1 UMA INSTÂNCIA SIMPLIFICADA

Consideraremos, primeiramente, uma tarefa da patrulha **simplificada**, onde:

- Só existe **um único agente** patrulhador;
- A abstração do terreno a ser patrulhado consiste em um **grafo sem pesos** (distância unitária entre nós, para todas as arestas).

Para esta tarefa ser considerada um MDP, conforme foi mostrado na seção 3.2.1, é necessário mostrar que existe uma tupla  $\langle S, A, P, R \rangle$ , onde  $S$  é o conjunto dos estados possíveis,  $A$  é o conjunto das ações possíveis,  $P$  é a distribuição de probabilidade de transição entre estados e  $R$  é a função de recompensas instantâneas.

Uma modelagem natural para o espaço de ações  $A$  é considerá-lo como o conjunto de ações que permitem que o agente navegue entre nós adjacentes no grafo. Em outras palavras, com uma ação, o agente pode percorrer uma aresta na abstração do terreno, o grafo de patrulhamento.

Considerando  $P$ , é importante notar que esta instância simplificada da tarefa da patrulha é totalmente determinística: como há um único agente, o próximo estado do mundo pode ser completamente determinado (probabilidade 1) pelo estado atual e ação realizada. Sendo assim, a função  $P$  pode ser facilmente obtida, com o estado  $s'$  da tripla  $\langle s, a, s' \rangle$  com probabilidade 1 e os demais estados com probabilidade 0.

A definição da função de recompensas instantâneas,  $R$ , é dependente do critério de avaliação (ver seção 2.1) que se deseja otimizar. Consideraremos neste trabalho agentes que

tentam otimizar o critério da **Média da Ociosidade Média**. A escolha de tal critério foi uma decisão de projeto, que teve as seguintes razões:

- O fato de que algumas das abordagens anteriores só foram avaliadas segundo este critério;
- Este critério é representativo para uma grande classe de problemas que estão relacionados à tarefa da patrulha;
- O uso de tal critério se mostra natural para uma abordagem baseada em aprendizagem por reforço, como será mostrado a seguir.

Queremos então definir uma função de recompensas instantâneas  $R$  tal que, se um agente aprender uma política ótima em termos desta função, ele também estará maximizando o critério de performance de longo termo, a Média da Ociosidade Média, a qual chamaremos deste ponto em diante apenas de **Ociosidade Média**. Utilizando a Equação 5 da seção 2.1, vemos que a ociosidade média de uma simulação que dura do ciclo  $I$  (um) até o ciclo  $T$  é dada pela Equação 16.

$$MedMed(1, T) = \frac{\sum_{t=1}^T Med(t)}{T} \quad (\text{Equação 16})$$

#### **Equação 16. Média da ociosidade média do ciclo $I$ até $T$**

Pela Equação 16 vemos que, se um agente deseja minimizar a ociosidade média, uma boa estratégia é tentar, a cada ciclo, minimizar o valor da ociosidade média instantânea do grafo  $Med(t)$  (ver Equação 3). Considerando que a ociosidade de cada nó é incrementada por 1 (um) a cada ciclo da simulação (se o nó não foi visitado), a ociosidade instantânea do grafo será incrementada por  $N$ , onde  $N$  é o número de nós do grafo. Da mesma maneira, como a cada ciclo a ociosidade do nó atualmente visitado recebe o valor zero, a ociosidade instantânea do grafo no ciclo é subtraída do valor da ociosidade deste nó. Seja  $Pos(t)$  o nó visitado pelo agente no ciclo  $t$ , e  $\Phi(Pos(t), t)$  a ociosidade deste nó no ciclo  $t$ , e, o agente pode calcular o valor de  $Med(t)$  facilmente, utilizando apenas percepções locais sobre em qual nó ele está e qual a ociosidade deste nó, através da equação de recorrência mostrada na Equação 17.

$$\begin{aligned} Med(1) &= OciosidadeInicial; \\ Med(t) &= Med(t-1) + \frac{N - \Phi(Pos(t), t)}{N}, \end{aligned} \quad (\text{Equação 17})$$

onde  $OciosidadeInicial$  é uma constante qualquer, e  $N$  é o número de nós do

grafo.

### **Equação 17. Equação de recorrência da ociosidade média instantânea do grafo**

Deste modo, **uma possível função de recompensa instantânea seria a própria função  $Med(t)$** , que atuaria, na verdade, como uma função de punição: quanto maior a ociosidade instantânea do grafo a cada ciclo, maior a punição do agente. A utilização de tal função leva a uma aproximação de  $MedMed(l, T)$ , embora não leve a uma solução ótima: a solução aprendida pelo agente minimizaria a soma das ociosidades instantâneas descontadas no tempo, conforme mostra a Equação 18, diferindo do numerador da Equação 16, que não possui o fator de desconto.

$$V = \sum_{t=1}^{\infty} \gamma^t \cdot Med(t) \quad (\text{Equação 18})$$

### **Equação 18. Função valor do agente caso $R$ seja a ociosidade média instantânea**

Na realidade, para se obter performance ótima nesta tarefa utilizando esta função de recompensa, é necessário considerar métodos de aprendizagem por reforço que otimizam a função de valor com relação à recompensa média esperada, tais como *R-Learning* [34], ao invés dos métodos tradicionais de horizonte infinito cuja função valor utiliza a soma das recompensas descontadas. Não utilizamos tais métodos neste trabalho, por duas razões principais. Em primeiro lugar, seu uso não seria um ponto de partida razoável, dado que ainda há muito pouca experiência no uso destes na comunidade de aprendizagem por reforço. Em segundo lugar, apesar da função  $Med(t)$  se apresentar como uma função de recompensa promissora, ela assume que o agente possui um modelo completo de todo o ambiente: ele sabe de que maneira as ociosidades de todos os nós do grafo estão sendo alteradas a cada ciclo. Embora na nossa instância simplificada em que há um único agente isto funcione, pois o ambiente é modificado exclusivamente pelo próprio agente, em instâncias mais complexas, como no caso multi-agente, por exemplo, teria de haver comunicação entre os agentes ou com um ponto central de informação para construir o valor de tal função de recompensa para cada agente. Sendo assim, mostramos agora como uma função de recompensa mais genérica pode ser obtida.

Expandindo a Equação 17, e considerando que a constante *OciosidadeInicial* é zero<sup>6</sup>, pode-se obter a mesma equação na sua forma fechada, conforme mostrado na Equação 19.

---

<sup>6</sup> Tal suposição não interfere no resultado final, apenas simplifica a visualização dos cálculos.

$$Med(t) = \frac{(t \times N) - \sum_{i=1}^t \Phi(Pos(i), i)}{N} \quad (\text{Equação 19})$$

### Equação 19. Equação da ociosidade média instantânea do grafo em forma fechada

Substituindo a Equação 19 na Equação 16, podemos obter a equação da ociosidade média em termos das funções  $Pos$  e  $\phi$ :

$$MedMed(1, T) = \frac{\sum_{t=1}^T Med(t)}{T} \rightarrow \quad (\text{Definição de } MedMed(1, T))$$

$$MedMed(1, T) = \frac{\sum_{t=1}^T \left( \frac{(t \times N) - \sum_{i=1}^t \Phi(Pos(i), i)}{N} \right)}{T} \rightarrow \quad (\text{Expandindo } Med(t))$$

$$MedMed(1, T) = \frac{\frac{N \times (T + T^2)}{2} - \sum_{t=1}^T \left( \sum_{i=1}^t \Phi(Pos(i), i) \right)}{T \times N} \rightarrow \quad (\text{Retirando } (t \times n) \text{ e } N \text{ do somatório})$$

$$MedMed(1, T) = \frac{\frac{N \times (T + T^2)}{2} - \sum_{i=1}^T ((T - i) \times \Phi(Pos(i), i))}{T \times N} \quad (\text{Equação 20})$$

### Equação 20. Equação da Média da Ociosidade Média em termos das funções $Pos$ e $\phi$

Analisando a Equação 20, observa-se que o único termo que depende da política de ações do agente é dado por:

$$\sum_{i=1}^T ((T - i) \times \Phi(Pos(i), i)) \quad (\text{Equação 21})$$

### Equação 21. Subexpressão da ociosidade média dependente da política do agente

Sendo assim, se o agente consegue aprender uma política de ações que maximiza a Equação 21, tal política seria ótima segundo o critério da ociosidade média. Consideremos então que a **função de recompensas instantâneas  $R$  seja dada por  $\Phi(Pos(t), t)$** , ou seja, a recompensa instantânea que o agente recebe ao executar uma ação é a ociosidade do nó que ele visita. Neste caso, a política aprendida por um algoritmo como *Q-Learning* é aquela que maximiza a soma destas recompensas descontadas, como mostra a Equação 22.

$$\sum_{i=1}^{\infty} (\gamma^i \times \Phi(Pos(i), i)) \quad (\text{Equação 22})$$

### Equação 22. Função valor do agente caso $R$ seja dado por $\Phi(Pos(t), t)$

Analisando as diferenças entre a Equação 21 e a Equação 22, percebe-se que ambas representam uma soma da função de recompensas descontada pelo tempo, sendo a Equação 21 descontada por uma progressão aritmética, enquanto que a Equação 22 é descontada por uma progressão geométrica. Utilizando um valor apropriado para o parâmetro  $\gamma$ , é possível aproximar a Equação 21, atingindo uma política que é ótima em muitos casos.

Além de aproximar a solução ótima, tal função de recompensa apresenta outras vantagens. Primeiro, ela é completamente local, dependendo apenas do valor da ociosidade do nó sendo visitado pelo agente. Sendo assim, o agente não precisa assumir nada sobre o resto do ambiente, o que não era verdade na função de recompensa discutida anteriormente. À primeira vista, pode parecer contraditório tentar minimizar o critério da ociosidade média utilizando recompensas que visam maximizar a ociosidade dos nós visitados. No entanto, intuitivamente, isto significa que, se a recompensa do agente é a ociosidade do nó que ele visita, ele tenderá a visitar os nós com maiores ociosidades e, por consequência, minimizará a ociosidade média global, conforme foi mostrado nos resultados teóricos desta seção.

Uma vez que definimos  $A$ ,  $P$  e  $R$ , nos resta definir o espaço de estados  $S$  para obtermos um modelo MDP completo. É necessário que a representação de estados seja escolhida de tal maneira que o modelo resultante satisfaça a **propriedade de Markov** (ver seção 3.2.1), de maneira que a escolha do próximo nó a ser visitado dependa apenas do estado atual e seja independente da sequência de posições (nós) visitados no passado pelo agente. Uma observação importante é que as posições passadas do agente só são importantes na tomada de decisão pelo fato de que elas influenciam nos valores das ociosidades dos nós do grafo como um todo. Sendo assim, o estado do ambiente pode ser totalmente capturado, satisfazendo a propriedade de Markov, pela posição atual do agente acrescido da informação da ociosidade de cada nó do mapa.

No entanto, como o conjunto dos possíveis valores das ociosidades não é finito (não há limite para o valor da ociosidade de um determinado nó, ou seja, este pode ser potencialmente infinito), tal espaço de estados não é finito. Uma possível solução para tal problema é agrupar os valores das ociosidades em um conjunto finito de faixas de valores, mas esta opção ainda leva a um grande número de estados possíveis, o que pode inviabilizar a sua utilização: se  $k$  é o número de faixas de valores, e  $N$  é o número de nós do grafo, existem

$N \times k^N$  estados possíveis, crescendo exponencialmente no número de nós. Outra possível solução é assumir que apenas as relações de ordem entre os valores das ociosidades são relevantes (maior que, por exemplo). Desta maneira, um estado atual que represente em qual nó o agente está, e qual a ordenação dos nós de acordo com os valores das suas ociosidades teria  $N \times N!$  estados possíveis, o que também é bastante custoso. Observa-se então que há um compromisso entre a complexidade da representação de estados e a qualidade da solução baseada na mesma. Esta questão se torna ainda mais problemática nas instâncias mais complexas do problema, e esta discussão será retomada nas próximas seções.

## 4.2 CONSIDERANDO DISTÂNCIAS REAIS ENTRE OS NÓS

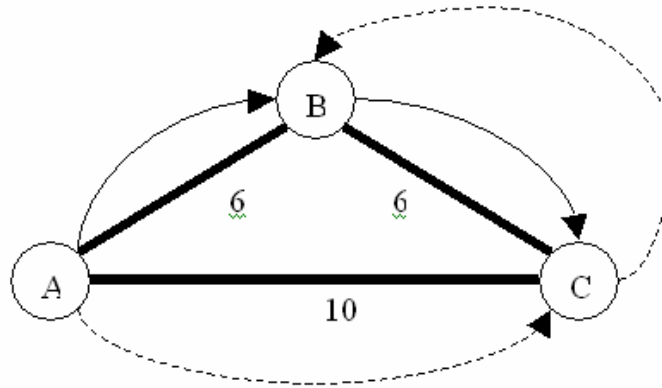
Na formulação simplificada da tarefa da patrulha mostrada na seção 4.1, nós consideramos que todas as arestas do grafo em questão possuíam comprimento igual e unitário. Tal suposição é inválida em muitos problemas reais, sendo assim, é necessário estender nosso modelo para considerar a distância real entre os nós.

Intuitivamente, o problema de utilizar diretamente a abordagem previamente proposta neste novo caso, no qual as arestas do grafo podem ter tamanhos arbitrários, é que se o agente está tentando maximizar as recompensas que ele recebe num dado período de tempo, ele deveria evitar ações que requerem um tempo maior para serem executadas (arestas longas, por exemplo), já que durante este período de tempo ele não recebe nenhuma recompensa (conseqüentemente, a ociosidade do grafo cresce).

O seguinte exemplo ilustra este problema: considere o grafo mostrado na Figura 5, onde A, B e C são nós e os valores em cada aresta representam o tempo (em ciclos de simulação) que o agente levará para percorrer tal aresta. Suponha que o agente está no nó A e precisa escolher se é melhor seguir o caminho (A, B, C) (linha cheia) ou o caminho (A, C, B) (pontilhado), considerando a utilidade de cada caminho como sendo o retorno estimado pelo algoritmo *Q-Learning*, por exemplo. Utilizando a última função de recompensas definida na seção 4.1, e considerando que os valores das ociosidades nos nós A, B e C eram 0 (zero) inicialmente, e que  $\gamma = 0.8$ , teríamos que:  $Recompensa(A,B,C) = 0 + \gamma \cdot 6 + \gamma^2 \cdot 12 = 12.5$ , e  $Recompensa(A,C,B) = 0 + \gamma \cdot 10 + \gamma^2 \cdot 16 = 18.2$ . o agente iria, então, escolher o caminho (A,C,B). No entanto, se nós utilizarmos a Equação 5 para calcularmos a ociosidade média, observa-se que o caminho (A,B,C) é, na realidade, o melhor, e deveria ter sido escolhido. De fato, este caminho requer apenas 12 (doze) ciclos para ser executado, enquanto que o caminho

(A,C,B) requer 16, mas o agente é incapaz de levar isto em conta, pois esta informação não faz parte do seu modelo MDP.

Felizmente, o formalismo SMDP (ver seção 3.2.3) já lida diretamente com esta questão. Utilizando um modelo SMDP, o fator  $\gamma$  passa a ser descontado para todos os ciclos da simulação, e não apenas nos ciclos correspondentes ao final da realização de uma ação. Neste caso, utilizando o mesmo exemplo, a utilidade do caminho (A,B,C), calculada utilizando a Equação 15, passa a ser:  $Recompensa(A,B,C) = 0 + \gamma^6 \cdot 6 + \gamma^{12} \cdot 12 = 2.4$ , enquanto que a utilidade do caminho (A,C,B) passa a ser:  $Recompensa(A,C,B) = 0 + \gamma^{10} \cdot 10 + \gamma^{16} \cdot 16 = 1.52$ . Sendo assim, o agente consideraria o caminho (A,B,C), resultando em uma política de ações correta.



**Figura 5. Exemplo de Grafo com Pesos. O agente (em A) deve escolher o caminho (A,B,C) ou (A,C,B)**

Se a função  $\Phi(Pos(t), t)$  for redefinida como sendo a ociosidade do nó visitado no tempo  $t$ , caso o agente esteja visitando algum nó neste instante, e zero caso contrário (agente está percorrendo uma aresta), todas as equações da seção 4.1 permanecem inalteradas, e o modelo desenvolvido previamente naturalmente generaliza para instâncias onde os grafos possuem pesos distintos.

### 4.3 O CASO MULTI-AGENTE

Nossa abordagem para estender o modelo apresentado nas seções anteriores para o caso multi-agente é baseada no conceito de aprendizes independentes (*independent learners*) [39], descrito na seção 3.3.1: nós tentamos resolver um problema de otimização global (coletivo), pela resolução de problemas locais (individuais). Com isto, nós nos livramos dos problemas da alta complexidade das abordagens centralizadas (ver seção 3.3), em detrimento de não termos garantias formais de convergência, além de termos a necessidade de prover



meios efetivos para garantir que haja coordenação entre os agentes. No restante desta seção, discutimos como isto pode ser feito, analisando o impacto desta mudança em cada um dos componentes  $\langle S, A, R, P \rangle$  do nosso MDP.

O conjunto de ações  $A$ , permanece o mesmo definido no caso de um único agente: uma ação para qualquer agente é percorrer uma aresta do grafo, indo de um nó a outro adjacente.

Consideremos agora a função de recompensas  $R$ . Pelos mesmos argumentos que mostraram que a política ótima com respeito ao critério da ociosidade média é aquela que maximiza a Equação 21, podemos generalizar este resultado para o caso multi-agente: se  $G$  é o número de agentes e  $Pos_j(t)$  é a função que diz a posição (nó) de um agente específico  $j$ ,  $1 \leq j \leq G$ , no tempo  $t$ , nós queremos agora maximizar a expressão mostrada na Equação 23.

$$\sum_{j=1}^G \left( \sum_{i=1}^T ((T-i) \times \Phi(Pos_j(i), i)) \right) \quad (\text{Equação 23})$$

**Equação 23. Sub-expressão da ociosidade média no caso multi-agente, cuja minimização resulta em uma política ótima**

Consideremos que cada agente do grupo usa a mesma função de recompensa definida no caso de um único agente:  $\Phi(Pos_j(t), t)$ , isto é, a ociosidade do nó visitado. Conseqüentemente, cada um deles tentará, independentemente, maximizar um dos somatórios internos da Equação 23, de maneira similar à que foi mostrada na seção 4.1. Entretanto, se relembrarmos a Equação 1, vemos que o valor de  $\Phi$ , para qualquer nó, pode ser afetado potencialmente pela política de ações de qualquer outro agente  $j$ . Sendo assim, esta abordagem é egoísta (nenhum agente deixará de tentar maximizar suas próprias recompensas em prol do grupo) e, mesmo que nós tenhamos uma representação de estados para cada agente que satisfaça a propriedade de Markov, ela não garantirá a optimalidade global da solução.

Relembrando a seção 3.3.2, este resultado é equivalente ao conceito de utilidade egoísta (*selfish utility*). Desta maneira, existem abordagens [27] para adaptar tal modelo de recompensas com o conceito de *wonderful life utility* (WLU), dando penalidades (recompensas negativas) quando agentes competem pela ociosidade (recompensa) de um mesmo nó.

Embora a utilização de uma abordagem egoísta ou WLU, utilizando o modelo apresentado nesta seção, não possua garantias de optimalidade global, ela apresenta algumas vantagens importantes. Primeiro, ela é perfeitamente adequada para uma implementação distribuída: o único feedback que cada agente recebe é a ociosidade do nó sendo visitado

atualmente. Além disto, esta recompensa é completamente local, e pode ser implementada por um esquema bastante simples de comunicação por *flags*, onde para cada nó que um agente visita, ele posiciona uma *flag* contendo o número do ciclo de simulação em que ele esteve lá, e esta *flag* pode ser vista por outros agentes.

Tendo sido definida uma extensão viável para  $R$ , voltemos a analisar agora o espaço de estados  $S$ , como o caso de um único agente já era intratável, o mesmo pode ser dito do caso multi-agente. Com isto em mente, nossa solução é representar apenas informação parcial na descrição de estado de cada agente. Em particular, nós representamos algumas características da sua posição atual e dos seus arredores (nós vizinhos), analisando empiricamente o efeito da adição de cada nova característica ao vetor de descritor do estado, como será detalhado na seção 5.2.2. Embora esta simplificação tenha consequências sérias (perda da propriedade de Markov), ela possui algumas vantagens. Em primeiro lugar, esta formulação é mais barata em termos de complexidade computacional. Em segundo, ela é mais realista do que a que foi proposta no caso de um único agente, em que o agente necessitaria ter acesso ao estado completo de todo o ambiente, o que não é possível em muitas aplicações práticas do problema da patrulha. Em terceiro lugar, tal representação é adequada para uma implementação distribuída, já que cada agente só observa sua vizinhança.

Considerando a distribuição de probabilidade de transição entre estados  $P$  no caso multi-agente, agora cada agente lida com um ambiente não-determinístico, já que ele não sabe quais ações serão executadas pelos outros agentes. Pior que isto, como todos os agentes estão adaptando seus comportamentos simultaneamente,  $P$  se torna uma distribuição não-estacionária. De maneira a diminuir os efeitos colaterais deste não-determinismo, nós introduzimos comunicação no processo de aprendizagem, investigando diferentes arquiteturas, que são detalhadas na próxima seção.

#### **4.4 OS AGENTES APRENDIZES**

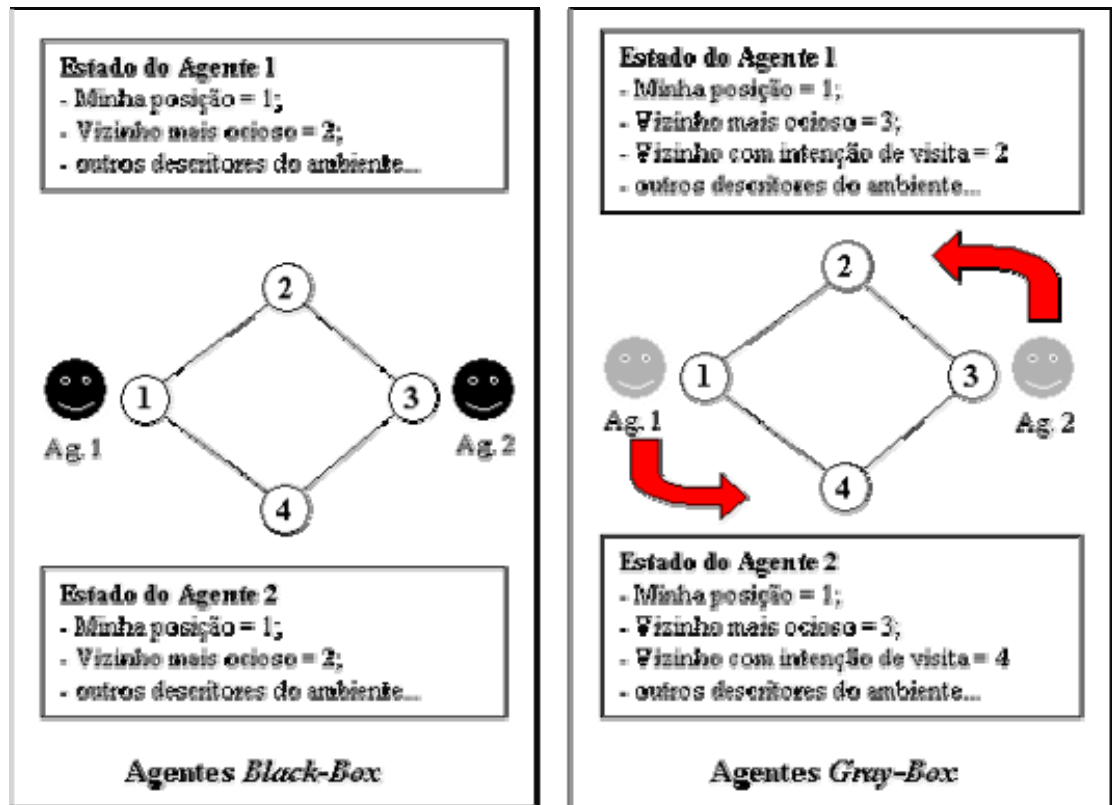
Do que foi mostrado nas seções anteriores, observa-se que, embora os resultados teóricos desenvolvidos nos sirvam de base para a modelagem proposta, a escolha da melhor implementação a ser realizada carece de experimentos envolvendo possíveis modelagens. De fato, como a escolha de alguns fatores do modelo tais como a representação de estados e a função de recompensa a utilizar não puderam ser desenvolvidos teoricamente, a finalização do mesmo carece de uma avaliação experimental de tais possibilidades. Sendo assim, diversos modelos de agentes são propostos, implementados e avaliados no próximo capítulo,

utilizando-se os conceitos chave descritos neste, porém, variando-se alguns parâmetros relacionados a características cuja melhor implementação ainda apresentava incertezas.

O primeiro parâmetro analisado foi a representação de estados. Esta representação de estados é, tradicionalmente, armazenada como um vetor de características descritoras do ambiente [47]. Uma vez que decidimos representar apenas informações locais em cada agente, devido ao alto custo computacional de uma representação de estados completa, é necessário escolher quais características são mais representativas para a tomada de decisão do agente. Os diferentes modelos são analisados na seção 5.2.2.

Um segundo parâmetro analisado nos experimentos foi a capacidade dos agentes se comunicarem. Uma vez que cada agente atua e aprende independentemente, a utilização de comunicação passa a ser imprescindível para obtenção de comportamento coordenado [53], e dois modelos de comunicação, integrados ao *framework* da aprendizagem por reforço, foram propostos e avaliados neste trabalho. Estes esquemas de comunicação utilizam o conceito de sistemas caixa preta (*black-box*) e caixa cinza (*gray-box*) [38], descritos na seção 3.3.

Na nossa abordagem, os agentes *black-box* não se comunicam diretamente, e a única comunicação que ocorre se dá através das *flags* do ambiente (o agente sabe há quanto tempo um outro agente visitou um determinado nó). Já os agentes *gray-box* são capazes de **comunicar suas intenções de ações**: se um agente decide que vai visitar um determinado nó, ele comunica esta informação aos agentes próximos a este nó (no próprio nó ou vizinhos). Estes agentes, por sua vez, incluem a informação a respeito desta intenção nas suas próprias representações de estado. Ou seja, os agentes *gray-box* utilizam para a tomada de decisão não apenas as suas percepções locais, mas também as intenções dos seus agentes e utiliza esta informação para melhorar seu aprendizado e, conseqüentemente, sua performance. A idéia do funcionamento de tais arquiteturas é ilustrada na Figura 6, e mostrada em mais detalhes na seção 5.2.2.



**Figura 6. Ilustração de agentes *Black-Box* e *Gray-Box* na patrulha. Os agentes gray-box comunicam as intenções de ações e incluem as intenções dos outros em suas representações de estados.**

Finalmente, avaliamos diferentes modelos de recompensa instantânea, baseados nos conceitos de *selfish utility*, *team-game utility* e *wonderful life-utility*, descritos na seção 3.3.2. Estas implementações são discutidas e avaliadas na seção 5.2.3.

## 4.5 CONCLUSÕES

Neste capítulo, foi analisado detalhadamente a modelagem da tarefa da patrulha como um problema de aprendizagem por reforço, apresentando-se as dificuldades em construir tal modelo desde o problema mais simples, a patrulha com um único agente em grafos unitários, até as instâncias mais complexas multi-agente e em grafos cujas arestas possuem pesos diferentes.

Vale a pena ressaltar que a modelagem proposta, embora bem fundamentada matematicamente, não possuía **nenhuma garantia** formal de que iria apresentar um bom desempenho. Não somente isto, como a hipótese de Markov, conforme nossa análise, não pôde ser completamente satisfeita, poderia ser que as estratégias aprendidas pelos agentes

utilizando tal modelo não convergissem. Este é um outro fator motivador da avaliação empírica dos modelos propostos neste trabalho, mostrada no próximo capítulo.

## 5 IMPLEMENTAÇÕES E RESULTADOS

Este capítulo está dividido em três seções principais. Na seção 5.1, as principais implementações realizadas neste trabalho são mostradas e discutidas. Na seção 5.2, temos as avaliações empíricas dos modelos, onde as diversas alternativas plausíveis de modelos de agentes que aprendem por reforço para a patrulha são avaliadas e, uma vez estabelecida uma melhor arquitetura, esta arquitetura é então comparada com as abordagens anteriores. Finalmente, discutimos os resultados mostrados neste capítulo na seção 5.3.

### 5.1 IMPLEMENTAÇÕES REALIZADAS

Para permitir a realização de toda a parte experimental deste trabalho, foi necessária a criação de uma infra-estrutura para os experimentos. Basicamente, esta infra-estrutura consistiu em implementar extensões ao simulador já existente e criar uma implementação do algoritmo *Q-Learning*. Tal implementação foi feita de maneira bastante extensível, pela criação de um componente que implementa tal algoritmo de maneira flexível. Este componente recebeu o nome de *RL Engine*, e é descrito na próxima seção.

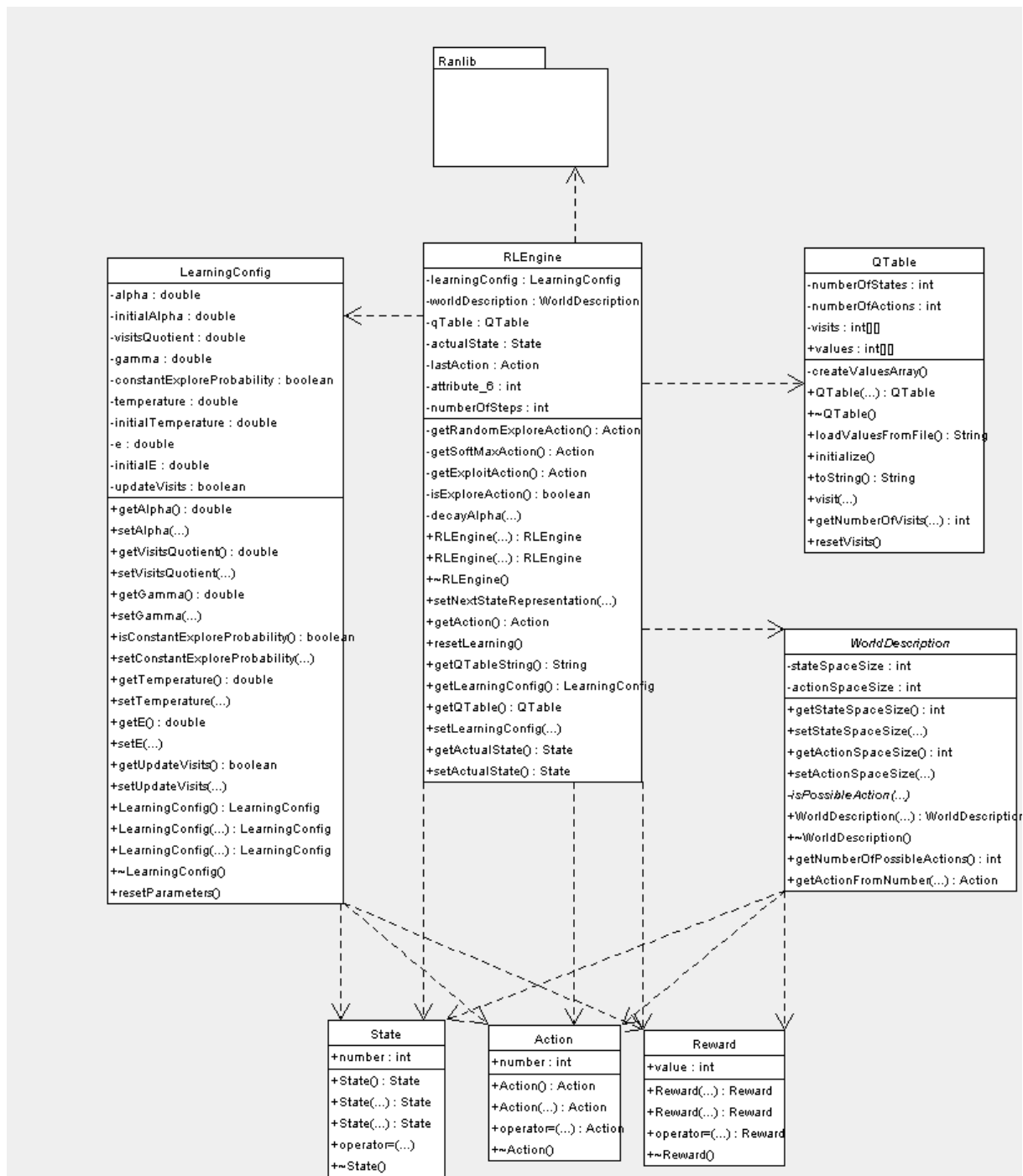
#### 5.1.1 RL ENGINE

A implementação do algoritmo *Q-Learning* se dá através de um componente chamado de *RL Engine*. A idéia consistiu na criação de uma implementação que satisfizesse as seguintes propriedades:

- Fosse independente da aplicação, o que a torna extensível;
- Fosse eficiente;
- Permitisse o uso das diversas variações do algoritmo (mecanismo de seleção de ações  *$\epsilon$ -greedy* [49] ou *Softmax* [11], ambiente determinístico ou não-determinístico [37], etc.).

A Figura 7 mostra o diagrama de classes do *RL Engine*. A sua implementação foi feita em C++ [46], por ser eficiente e permitir integração com o simulador da tarefa da patrulha já existente, que também foi desenvolvido nesta mesma linguagem. Do ponto de vista do usuário do componente, só é necessário herdar a classe *WorldDescription*, criando uma descrição do ambiente no qual o componente vai ser efetivamente utilizado, e utilizar a classe *RL Engine*, através de métodos como *getAction()*, que retorna a ação escolhida pelo algoritmo *Q-Learning* e *setNextStateRepresentation()*, que informa ao componente o estado do mundo atualmente.

As configurações da aprendizagem são carregadas através de arquivos de configuração, e as funções que necessitam de componentes aleatórios utilizam a biblioteca Ranlib [30], da linguagem C.



**Figura 7. Diagrama de Classes (UML) do RLEngine**

Este componente já foi utilizado com sucesso em outro projeto no CIn-UFPE, o Knock-em [5], o que demonstra a sua extensibilidade.

### 5.1.2 AGENTES PATRULHADORES

O simulador da patrulha [33] foi criado utilizando a linguagem C++, e o modo de visualização em OpenGL. Este simulador foi utilizado nos experimentos desenvolvidos neste trabalho, tendo sido necessário apenas a criação de novos agentes patrulhadores, utilizando aprendizagem por reforço.

A implementação dos agentes patrulhadores é bastante simples, consistindo basicamente na extensão da classe *Agente* do simulador da patrulha, e utilização do *RLEngine*, na maneira descrita na seção 5.1.1.

## 5.2 AVALIAÇÃO EMPÍRICA

Nesta seção, a avaliação experimental da nossa abordagem é detalhada sob diferentes aspectos. A seção 5.2.1 explica em linhas gerais a metodologia utilizada nos experimentos. A seção 5.2.2 e 5.2.3, avalia e compara as diferentes abordagens referentes a representação de estados e função de recompensa, respectivamente. Na seção 5.2.4, os resultados obtidos neste trabalho são comparados com as abordagens desenvolvida nos trabalhos anteriores.

### 5.2.1 METODOLOGIA UTILIZADA

Nesta seção, descrevemos em linhas gerais os procedimentos utilizados na avaliação experimental dos agentes. Os detalhes específicos a cada experimento serão detalhados apropriadamente nas próximas seções.

Quase todos os experimentos realizados consistiam de dois esforços principais: uma **etapa de treinamento**, e uma **etapa de avaliação**. Na etapa de treinamento, o agente em questão tem como objetivo adquirir e armazenar o conhecimento obtido por aprendizagem por reforço. Em outras palavras, deseja-se acumular a maior quantidade de conhecimento possível utilizando-se a arquitetura em questão. Atingir tal limite, entretanto, pode ser inviável na prática, devido ao grande número de iterações necessárias para atingi-lo, e um compromisso entre qualidade e viabilidade do treinamento teve de ser obtido. Sendo assim, em cada experimento da etapa de treinamento, os seguintes parâmetros foram estimados:

- **Número de iterações** – o número de ciclos de simulação utilizados para treinar o agente;
- **Taxa de decaimento de  $\alpha$  (taxa de aprendizagem)** – ver Equação 14 da seção 3.2.2;
- **Probabilidade de exploração  $\epsilon$**  - ver seção 3.2.2;
- **Fator de desconto  $\gamma$**  - ver seção 3.2.1.



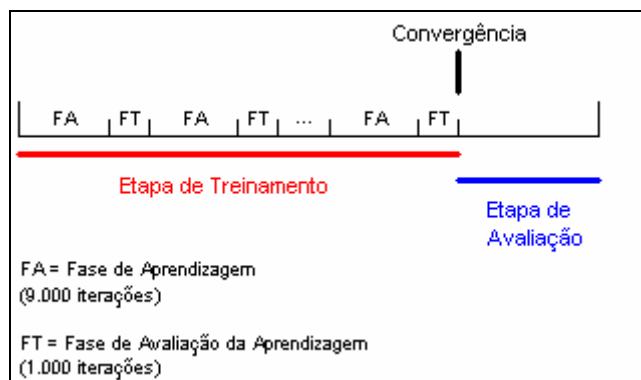
O número de iterações necessárias para a etapa de treinamento foi estimado dividindo-se a mesma em **fases de aprendizagem** (9.000 iterações<sup>7</sup>), intercaladas por **fases de avaliação da aprendizagem** (1.000 iterações<sup>7</sup>). A performance de cada fase de avaliação da aprendizagem é calculada, plotada em um gráfico, e assim, pode-se acompanhar a evolução da performance do agente durante a etapa de treinamento. Desta maneira, foram detectados manualmente limites onde havia convergência, ou nos quais o custo-benefício do ganho em performance obtido pelo aumento na duração do experimento não era mais significativo.

Para estimação dos parâmetros  $\alpha$ ,  $\gamma$  e  $\epsilon$ , foram avaliados empiricamente 3 valores para cada parâmetro, para cada uma das arquiteturas, resultando em nove experimentos por arquitetura. Estes 3 valores foram obtidos intuitivamente, levando-se em conta a experiência obtida em experimentos informais, e levando-se em conta a diferença entre as arquiteturas, em termos da quantidade de estados nas suas representações. Observou-se que o parâmetro  $\gamma$  de 0.9 obtinha sempre a melhor performance independente da arquitetura, mas que o parâmetro  $\alpha$  era bastante sensível ao número de estados possíveis da arquitetura em questão: quanto mais estados possíveis possui uma arquitetura, mais rápido pode ser o decaimento deste parâmetro, ou seja, menos visitas por estado são necessárias para garantir convergência. Os parâmetros  $\alpha$  e  $\epsilon$ , durante as fases de avaliação de aprendizagem e durante a etapa de avaliação da aprendizagem, nas quais o agente está apenas sendo avaliado e não precisa aprender, foram estimados em 0 e 0.01 (1%), respectivamente. Ou seja, o agente não aprende nenhum conhecimento e explora com probabilidade muito pequena. Nas fases de aprendizagem, o parâmetro  $\epsilon$  estimado foi de 0.1 (10% do tempo o agente escolhe uma ação exploratória), e o parâmetro  $\alpha$  inicia-se com 0.5, mas possui uma taxa de decaimento que varia de acordo com a arquitetura conforme descrito anteriormente.

Para a etapa de avaliação, o número de iterações utilizadas foi de 15.000, desprezando-se as 3.000 iterações iniciais, seguindo a metodologia utilizada nos trabalhos anteriores da patrulha, descrita em [33]. A Figura 8 mostra uma visão geral da metodologia seguida, mostrando as etapas e fases utilizadas no processo.

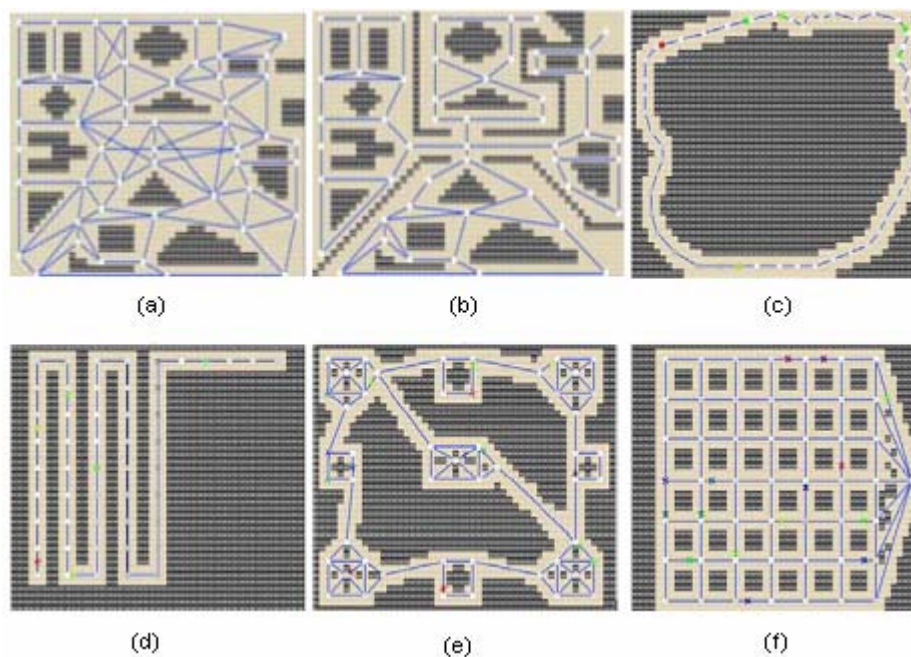
---

<sup>7</sup> Estes valores foram estimados experimentalmente, observando-se os resultados da performance dos agentes durante uma etapa completa de aprendizagem.



**Figura 8. Etapas da metodologia. Fases de aprendizagem e avaliação são intercaladas até haver convergência, e o resultado passa por uma etapa final de avaliação.**

Foram criados também neste trabalho diversos cenários de simulação, que fossem representativos das diferentes topologias que um agente pode vir a enfrentar em situações reais. A Figura 9 mostra os mapas utilizados para os experimentos, embora alguns destes tenham sido realizados em apenas um subconjunto deles. Os mapas (a) e (b) representam topologias genéricas, muito e pouco conectadas, respectivamente. O mapa (c) (*circle*) representa uma topologia de círculo. O mapa (d) (*corridor*) representa um corredor. O mapa (e) (*islands*) representa uma topologia de ilhas, onde há uma grande variância nos tamanhos das arestas. Finalmente, o mapa (f) (*grid*) representa uma topologia de *grid*.



**Figura 9. Cenários de simulação utilizados. (a) e (b) são grafos muito e pouco conectados, respectivamente. (c), (d), (e) e (f) representam topologias de círculo, corredor, ilhas e grid, respectivamente.**

Vale a pena ressaltar a complexidade de tal projeto experimental. Além dos parâmetros de aprendizagem citados, há diversos parâmetros do próprio problema, tais como o tamanho da população de agentes, ou o mapa a ser utilizado, que exigem um número exponencial de experimentos para garantir a completude, mas que se torna inviável na prática. Sendo assim, em muitos destes experimentos, alguns destes parâmetros serão fixados, tendo em vista a viabilidade dos mesmos. Os demais detalhes particulares a cada experimento são mostrados nas próximas seções.

### 5.2.2 AVALIAÇÃO DA REPRESENTAÇÃO DE ESTADOS

Um primeiro passo no desenvolvimento da nossa solução AR consiste em definir qual representação de estados utilizar. Em outras palavras, é necessário escolher um conjunto de atributos descritores do estado para utilizar na representação de cada agente. No entanto, como a complexidade de algoritmos depende bastante do número total de estados possíveis em que o agente pode vir a estar, é necessário também validar o conjunto de atributos escolhido para: i) comprovar que eles são realmente relevantes na tomada de decisão dos agentes e ii) tentar minimizar a quantidade de atributos necessária, visando melhorar a eficiência do agente, e diminuir a quantidade de iterações necessárias para realizar a etapa de treinamento.

Nesta seção, apresentamos uma análise empírica de tais representações, apontando também vantagens e desvantagens de cada uma delas. A metodologia utilizada e os resultados obtidos são descritos na próxima seção.

#### 5.2.2.1 Resultado Experimental

Primeiramente, foi selecionado um grupo de atributos que se apresentavam como uma escolha plausível para esta tarefa. Tal escolha foi feita por especialistas, de maneira intuitiva. Uma vez escolhidos estes atributos, foram feitos experimentos específicos visando medir a contribuição de cada descritor em particular para o desempenho da representação em questão, visando reduzir a quantidade total de descritores utilizados e comprovar a eficiência dos mesmos.

Conforme explicado na seção 4.4, os agentes desenvolvidos podem ser classificados em dois grupos, conforme o modelo de comunicação utilizado. Chamaremos estes grupos de *Black-Box Learner Agents* (BBLA) e de *Gray-Box Learner Agents* (GBLA). Estes agentes possuem um conjunto de atributos em comum nas suas representações de estados, mas os agentes GBLA incluem informação sobre a **intenção** das ações dos outros agentes. Os atributos em comum são:

- **NodeID (NID)**, representando em qual nó o agente está (número do nó) – tipicamente 50 valores possíveis nos mapas utilizados;
- **Últimas k Arestas**, indicando por qual aresta o agente chegou ao nó atual (qual foi sua última ação), o que representa informação sobre o passado para o agente – tipicamente  $5^k$  valores possíveis nos mapas utilizados<sup>8</sup>. Por conveniência, denotaremos por **UA** a informação sobre a última aresta visitada incluída na representação de estados;
- **Ociosidade Relativa dos Nós Vizinhos**, representando qual vizinho é o mais ocioso, qual o segundo mais ocioso, qual o menos ocioso, etc – tipicamente  $5^P$ , onde P é o número de sub-atributos utilizados. Por conveniência, denotaremos por **MO** a informação sobre qual o nó vizinho mais ocioso, por **SMO** a informação sobre o segundo vizinho mais ocioso, e por **EO**, o vizinho menos ocioso;

Para os agentes GBLA, é necessário incluir em sua representação a informação advinda da comunicação das intenções:

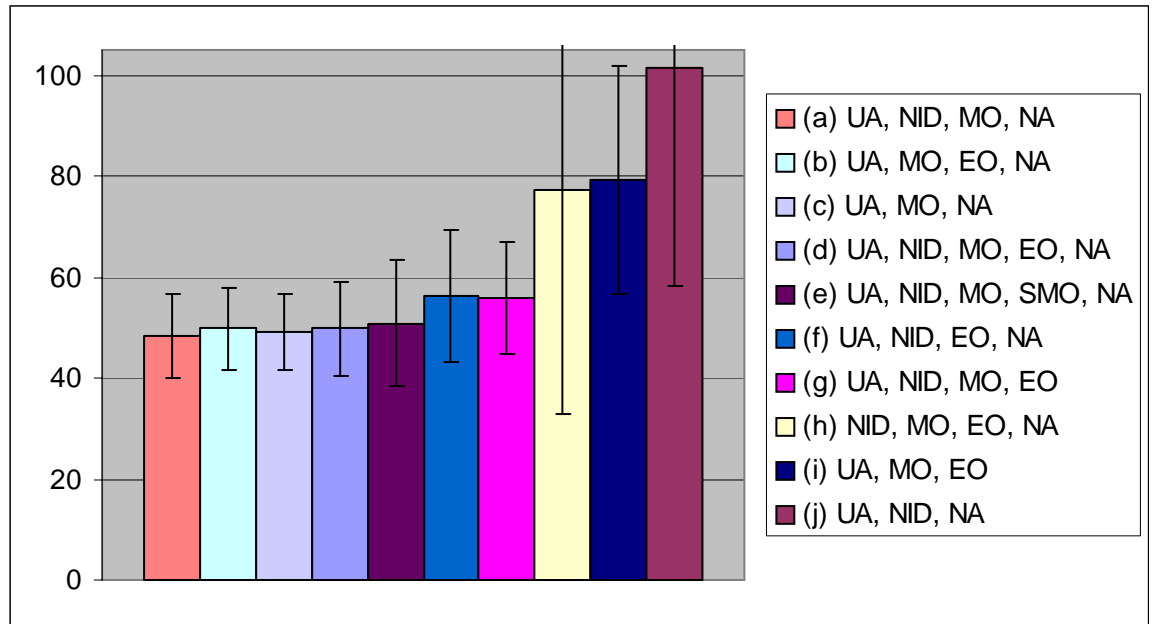
- **Nós vizinhos que outros agentes possuem intenção de visitar**, ou seja, se um agente **a** receber uma mensagem de um outro agente **b**, informando que um nó vizinho à posição atual de **a** será visitado por **b**, o agente **a** inclui esta informação na sua representação de estados. Denotaremos de **NA** o nó adjacente para o qual há pelo menos um outro agente com intenção de visita.

Com exceção do atributo NodeID, além da informação sobre as últimas arestas visitadas, a representação dos demais atributos pode ser implementada de maneira parcial: representar apenas quem é o nó mais ocioso e desprezar a ociosidade dos demais ou representar apenas um nó vizinho que está para ser visitado por outro agente.

Neste experimento, foram avaliadas as diversas representações de estado possíveis utilizando-se estas características, com o aprendizado feito utilizando-se uma função de recompensa egoísta. Os resultados mais significativos são mostrados na Figura 10. Foi utilizado o cenário de simulação (b) e os parâmetros de cada arquitetura foram obtidos seguindo o processo descrito na seção 5.2.1.

---

<sup>8</sup> A informação sobre o “último nó visitado” seria mais clara para o leitor, no entanto, representando a “última aresta” visitada, obtém-se uma informação igualmente informativa para o agente, porém tipicamente mais barata computacionalmente, como se observa pelo número de valores possíveis destas informações nos mapas utilizados, onde o número de nós vizinhos a um outro nó é no máximo 5.



**Figura 10. Avaliação da média da ociosidade média (eixo Y), e seus respectivos desvios padrão, de arquiteturas com diferentes representações de estado. Quanto menor o valor da métrica para a arquitetura, melhor a performance.**

Antes de analisar tais resultados, é importante ressaltar as diferenças fundamentais entre as arquiteturas listadas na Figura 10. As arquiteturas que utilizam a informação NA necessariamente necessitam de um mecanismo de comunicação de intenções, ou seja, são arquiteturas do tipo GBLA. As demais arquiteturas, a saber, a (g) e a (i) são arquiteturas BBLA, ou seja, não necessitam de comunicação explícita entre os agentes. As arquiteturas que não incluem a informação de NodeID, como (b), (c) e (i) possuem uma propriedade particular de serem mais genéricas que as demais pois, como não incluem informação sobre qual nó especificamente o agente está visitando, seu conhecimento pode ser generalizado para diferentes cenários de simulação. Tal peculiaridade destas arquiteturas será analisada com mais detalhes na seção 6.1.

Observando os resultados da Figura 10, vemos que as 5 primeiras arquiteturas listadas possuem desempenho estatisticamente equivalentes, com a arquitetura mais simples (c) sendo a mais estável (menor desvio padrão dos resultados), e as arquiteturas mais complexas (d e e) possuindo uma menor estabilidade. Estas arquiteturas têm algumas características em comum: todas elas são do tipo GBLA, e todas representam a informação UA e alguma informação sobre a ociosidade dos nós vizinhos (MO, EO ou SMO), o que indica a importância destes atributos.

A pior performance foi da arquitetura que não representava nenhuma informação sobre a ociosidade dos nós vizinhos ( $j$ ), o que mais uma vez indica a importância de incluir esta informação na representação de estados dos agentes. A arquitetura que obteve a segunda pior performance ( $i$ ) não representa a informação de localização do agente (NID), nem a informação advinda da comunicação (NA), sendo uma arquitetura do tipo BBLA. Comparando-se as arquiteturas ( $h$ ) e ( $d$ ), verifica-se a importância da informação sobre a última aresta (UA), já que este é o único atributo que os diferencia na representação, e as performances são bastante distintas.

Observa-se que, embora a informação sobre qual vizinho possui maior ociosidade se mostre importante, ela pode ser substituída pela informação de qual vizinho possui a menor ociosidade, sem grande perda de performance (ver a arquitetura  $f$ , em comparação com  $a$ ). Um outro fato interessante mostrado no gráfico é que, embora a informação sobre o nó atual (NID) seja relevante para as arquiteturas BBLA (comparando a performance das arquiteturas  $g$  e  $i$ ), ela é praticamente irrelevante nas arquiteturas GBLA em termos de performance (comparando a performance de  $a$  e  $c$ ). Este fato será analisado na seção 6.2.

A título de simplicidade, e levando em conta o desempenho neste experimento, nos demais experimentos realizados neste trabalho e mostrados neste documento, utilizaremos apenas as arquiteturas ( $a$ ), ( $c$ ) e ( $g$ ), e as chamaremos simplesmente de **GBLA**, **GGBLA** (GBLA generalizável, por não incluir o NID) e **BBLA**, respectivamente.

### 5.2.2.2 Discussão

O resultado mais surpreendente deste experimento foi observar que arquiteturas bastante simples, como as do tipo BBLA, por exemplo, apresentaram performance bastante satisfatória. Tal resultado foi inesperado, pois não havia comunicação entre os seus agentes, e os mesmos não satisfaziam a propriedade de Markov em suas representações de estado. No entanto, não apenas os resultados para a métrica da média da ociosidade média foram muito bons, mas a estratégia que se observa dos agentes no simulador é um surpreendente exemplo de coordenação emergente: os agentes são capazes de automaticamente particionar o grafo em questão, se especializando em regiões distintas. Tal resultado é ilustrado na Figura 11, e este comportamento emergente será analisado em mais detalhes na seção 6.2.



**Figura 11. Exemplo da coordenação que emerge no caso BBLA: o grafo é particionado em áreas disjuntas, com agentes especializados em áreas específicas do grafo (destacadas)**

Uma conclusão importante deste experimento diz respeito à descoberta da importância de cada característica a ser representada no estado, em termos de performance considerando a métrica da média da ociosidade média. Observou-se que, independente da arquitetura, os atributos sobre qual o nó vizinho mais ocioso e qual a última aresta seguida são muito importantes. Observa-se também que as arquiteturas GBLA (as que utilizam a informação NA) superam em performance as arquiteturas BBLA.

O fato de o atributo NID ser importante em termos de performance em arquiteturas BBLA e não nas GBLA não significa, entretanto, que ele é irrelevante nas arquiteturas GBLA. O que acontece é que a arquitetura GBLA, por utilizar comunicação, possui uma maior capacidade de adaptação, sendo capaz de reorganizar o comportamento dos agentes mantendo a performance do grupo (isto será mostrado em mais detalhes na seção 6.2).

Esta propriedade dos agentes GGBLA (GBLA sem NID) lhes trazem 2 vantagens fundamentais. Em primeiro lugar, a eliminação da informação sobre em qual nó o agente está (NID) reduz bastante a complexidade da representação de estados dos mesmos, dado que o número de valores possíveis deste atributo é potencialmente muito grande, sendo igual ao número de nós do grafo (tipicamente 50, enquanto os demais possuem 5, para os mapas utilizados). Com isso, diminui-se o tempo de treinamento necessário e reduz-se a quantidade de memória necessária à implementação dos agentes. A segunda vantagem é que o treinamento de agentes que utilizam a informação sobre qual nó específico ele está passa a ser extremamente dependente do cenário de simulação (mapa) utilizado no treinamento. Ou seja, para cada novo cenário em que o agente for interagir, será necessário um novo treinamento. Sendo assim, a retirada deste atributo da representação de estados torna o conhecimento obtido pelo treinamento dos agentes mais **generalizável**. Esta propriedade será analisada com mais detalhes na seção 6.1.

Embora os agentes GBLA obtenham uma melhor performance, além das outras vantagens citadas, esta abordagem apresenta algumas desvantagens em relação a BBLA. Em primeiro lugar, a aplicabilidade da arquitetura GBLA é restrita a ambientes onde os agentes podem se comunicar explicitamente<sup>9</sup>. Outra desvantagem diz respeito à comunicação entre os agentes, uma vez que há um custo associado à transmissão das mensagens contendo as intenções dos agentes. Tal transmissão, para cada agente GBLA patrulhador, embora não precise ser feita para todos os outros agentes, precisa ser feita para todos aqueles a uma distância menor ou igual a dois nós do grafo de patrulhamento, e acarreta um custo extra para o agente.

Os resultados deste experimento são fundamentais para uma implementação efetiva dos agentes utilizando a versão tabular do algoritmo *Q-Learning*, dado que é necessário restringir bastante os atributos a serem utilizados na representação de estados do agente. No entanto, mesmo com implementações mais flexíveis, que utilizem aproximação de função [47], a seleção de atributos ainda assim é relevante dado o alto grau de complexidade da tarefa, conforme descrito no capítulo 4.

### 5.2.3 AVALIAÇÃO DAS FUNÇÕES DE RECOMPENSA

Conforme descrito na seção 3.3, o desempenho de um sistema multi-agente que utiliza aprendizagem por reforço é extremamente influenciado pelo modelo de recompensas instantâneas utilizado. O objetivo do experimento aqui descrito consiste em avaliar a performance dos modelos de recompensa aplicáveis ao nosso problema, utilizando as arquiteturas de agentes aprendizes analisadas na seção 5.2.2<sup>10</sup>.

#### 5.2.3.1 Resultado Experimental

Para realizarmos este experimento, avaliamos o desempenho das arquiteturas GBLA e BBLA, utilizando os quatro modelos de recompensa descritos a seguir, três deles baseados nos conceitos descritos na seção 3.3.2:

- **Selfish Utility (SU)** – Neste modelo, cada agente recebe como recompensa a ociosidade do nó visitado por ele atualmente, conforme descrito na seção 4.3;

---

<sup>9</sup> Usaremos o termo “comunicação explícita” para nos referir a comunicação através de troca de mensagens feita pelos agentes GBLA. Os agentes BBLA se comunicam também, porém implicitamente, através de *flags* colocadas em cada visita a um nó, e utilizados para obter a informação de ociosidade.

<sup>10</sup> Os agentes GGBLA não serão analisados, pois a questão da generalização não é relevante a este experimento. Sendo assim, comparamos as arquiteturas BBLA e GBLA.



- **Team Game Utility (TGU)** – Neste modelo, cada agente recebe como recompensa a soma das ociosidades dos nós visitados por todos os agentes atualmente. Sendo assim, a performance global do grupo é utilizada como critério para a recompensa instantânea individual de cada agente;
- **Wonderful Life Utility (WLU)** – Seguindo a sugestão de implementação deste modelo dada em *Hoën et al* [27], o agente receberá uma recompensa igual à recompensa SU, no entanto, recebendo uma punição (reforço negativo) de igual valor no caso de competir pela recompensa com outro agente (ex. dois agentes visitando o mesmo nó simultaneamente);
- **Weighted Team Game<sup>11</sup> (WTG)** – Neste modelo, cada agente receberá a recompensa global do grupo (calculada como a variação da ociosidade instantânea do grafo), ponderado pela recompensa obtida individualmente (ociosidade do nó visitado por ele mesmo em relação à ociosidade dos nós visitados pelo grupo). Em outras palavras, esta recompensa representa um compromisso entre as recompensas TG e SU: é utilizada como recompensa a performance global (TG), no entanto, ponderada pela performance local (SU). Sendo  $\Phi(Pos_j(t), t)$  a ociosidade do nó visitado pelo agente  $j$  no tempo  $t$ , e  $I(t)$  a ociosidade instantânea, a recompensa WTG em um tempo  $t$  é mostrada na Equação 24.

$$R(j, t) = \frac{\Phi(Pos_j(t), t) \times (I(t-1) - I(t))}{\sum_{i=1}^G \Phi(Pos_i(j), j)} \quad (\text{Equação 24})$$

, onde  $G$  é o número de agentes

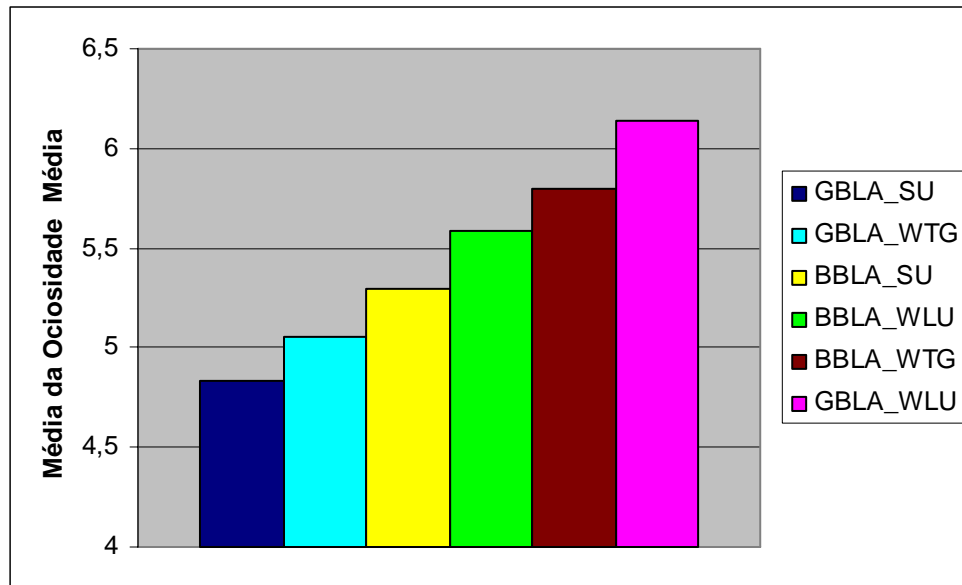
#### **Equação 24. Função de recompensa instantânea WTG de um agente $j$ no tempo $t$ .**

Foram utilizados para este experimento as combinações BBLA\_SU, BBLA\_TG, BBLA\_WLU, BBLA\_WTG, GBLA\_SU, GBLA\_TG, GBLA\_WLU e GBLA\_WTG. Os resultados dos agentes que utilizavam a recompensa TG, entretanto, não convergiram, e estão em uma escala tão diferente dos demais que não mostraremos os mesmos nos resultados a seguir. Tal resultado não é inesperado, dadas as observações da seção 3.3.2., onde mostrou-se que em experimentos similares também não apresentaram convergência.

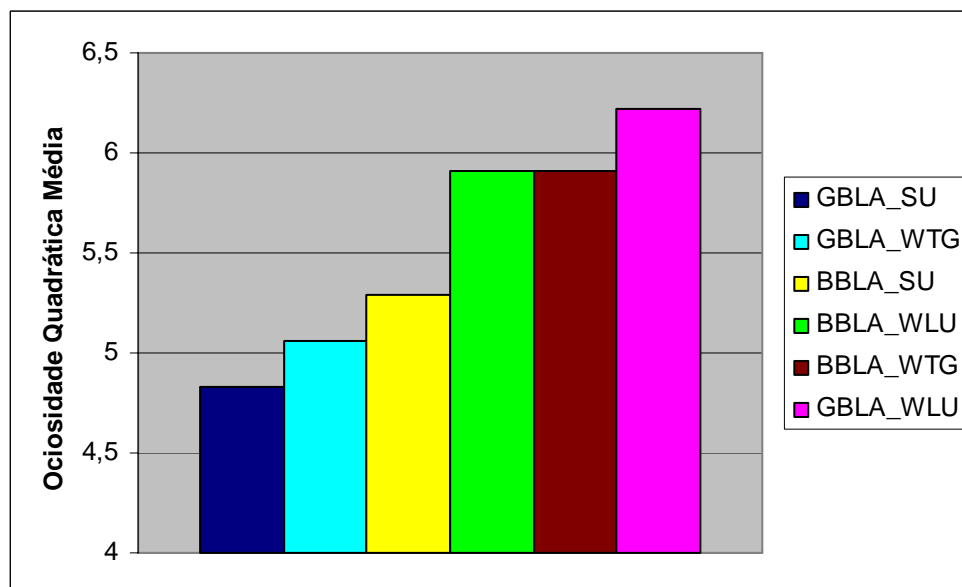
---

<sup>11</sup> Esta função de recompensa foi sugerida pela pesquisadora Bohdana Ratitch, da McGill University, Canadá, que participou no desenvolvimento deste trabalho.

A Figura 12 mostra o desempenho obtido utilizando o critério da média da ociosidade média na etapa de avaliação das arquiteturas descritas, e a Figura 13 mostra os resultados com relação a ociosidade quadrática média (ver Equação 9), que enfatiza os valores com alto desvio padrão, do mesmo experimento. Os parâmetros de cada arquitetura foram obtidos seguindo o processo descrito na seção 5.2.1. Os resultados mostrados foram obtidos no cenário de simulação (b), e resultados similares foram obtidos nos demais cenários.

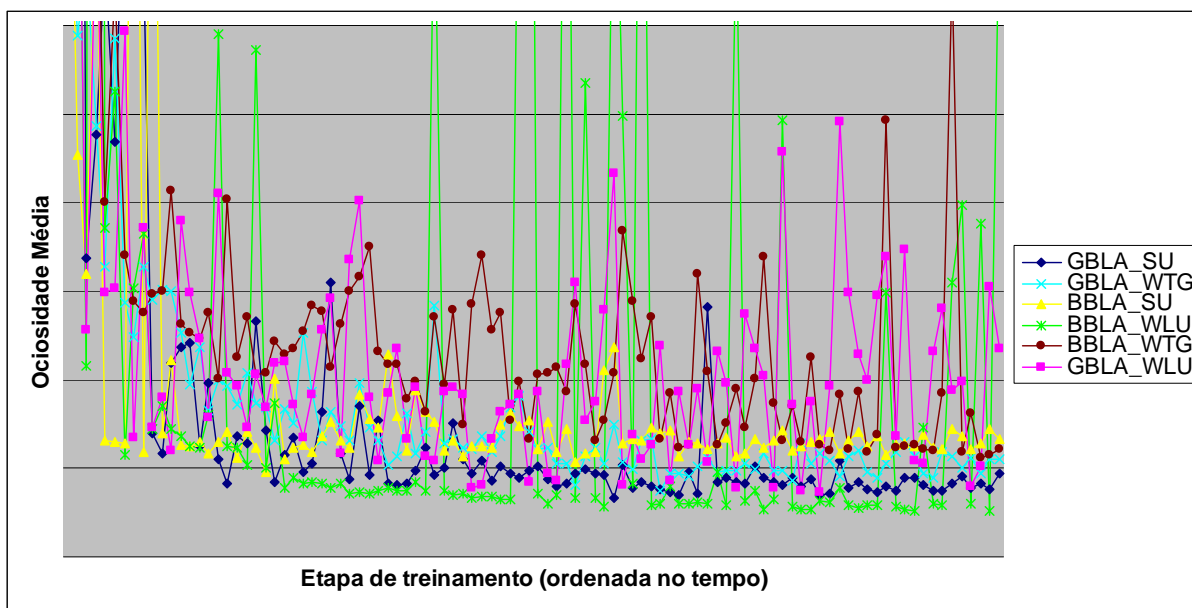


**Figura 12. Avaliação da média da ociosidade média das diferentes funções de recompensa aplicadas às arquiteturas BBLA e GBLA.**



**Figura 13. Avaliação da ociosidade quadrática média das diferentes funções de recompensa aplicadas às arquiteturas BBLA e GBLA.**

Embora os resultados utilizando-se as duas métricas sejam bem semelhantes, observa-se que as performances das abordagens BBLA\_WLU e BBLA\_WTG se igualam quando é utilizada a métrica da ociosidade quadrática média. Isto sugere que a arquitetura BBLA\_WLU seja mais instável que a sua correspondente que utiliza WTG. Esta suposição pode ser confirmada analisando a performance dos agentes durante a etapa de treinamento (ver seção 5.2.1), mostrada na Figura 14.



**Figura 14. Avaliação da ociosidade média durante cada sub-etapa de treinamento. A ociosidade é bastante alta no início e diminui com o aprendizado dos agentes.**

Observa-se que a arquitetura BBLA\_WLU consegue obter o melhor desempenho em boa parte da etapa de treinamento, melhores até do que as da arquitetura GBLA\_SU. No entanto, esta é alternada por picos de péssima performance que pioram bastante a performance geral. De fato, observa-se que as arquiteturas WLU são mais instáveis que as demais. As arquiteturas egoístas se mostraram bastante estáveis, com a GBLA obtendo performance melhor que a BBLA em todas as etapas de treinamento, como era esperado, e a arquitetura GBLA\_SU obtendo a melhor performance dentre todas as outras. Uma observação interessante é que a arquitetura GBLA não interage bem com o modelo WLU, obtendo a pior performance, pior ainda que a arquitetura BBLA\_WLU. Estes resultados são discutidos na próxima sessão.

### 5.2.3.2 Discussão

Os experimentos mostrados nesta seção apresentam um fato surpreendente: a melhor performance obtida foi de uma abordagem egoísta. Embora este resultado pareça estranho,

uma análise mais aprofundada do experimento mostra que é um resultado plausível. Também há outro relato na literatura onde arquiteturas egoístas igualam em performances arquiteturas WLU [27]. Levantamos aqui 3 pontos principais com relação a performance destes modelos na patrulha. Em primeiro lugar, ao observar a evolução do treinamento dos agentes, nota-se que a arquitetura BBLA\_WLU consegue manter em muitas iterações uma performance superior a todas as outras, no entanto, seus picos de baixa performance degradam o resultado geral. Analisando o que ocorre no simulador durante estes picos, percebe-se que estes correspondem a momentos em que algum nó do grafo deixa de ser visitado pelos agentes. Há uma explicação para este fato. Observa-se que é comum as abordagens que utilizam aprendizagem por reforço aprenderem estratégias cíclicas (ver seção 2.3.4). Caso haja algum nó que não faz parte do ciclo de nenhum agente, a ociosidade deste nó vai tender a infinito, e a performance do grupo vai cair independentemente de quão bem eles estejam patrulhando os demais nós. Devido a isto, todas as abordagens baseadas em AR possuem uma melhor performance quando utilizam uma pequena taxa de exploração (perto de 1%), fazendo-as mudar algumas vezes dos ciclos originalmente aprendidos e aprendendo novos. No entanto, este problema é mais crítico nas estratégias WLU, especialmente em nós estratégicos do mapa, que tendem a ser visitados por muitos agentes: a punição recebida pelos agentes WLU ao competir por recompensas no mapa faz com que nós que são freqüentemente fonte de conflitos não sejam visitados por nenhum agente, decaindo a performance dos mesmos.

Um outro ponto se refere à performance dos agentes WTG. Embora eles tenham obtido um bom desempenho, acreditamos que não exploramos ainda todo o potencial desta abordagem. O fato de termos obtido uma boa performance utilizando uma função de recompensa com informação global em agentes que utilizam uma representação de estados essencialmente local leva a acreditar que a performance a ser obtida utilizando uma representação de estados mais global é promissora. No entanto, tal ganho de desempenho viria acompanhado de uma certa restrição na aplicabilidade de tais agentes: haveria a necessidade de uma maior comunicação entre os agentes ou com um ponto de informação central para obter tal informação global, dificultando implementações distribuídas.

Por fim, o ótimo desempenho das abordagens egoístas é fruto de uma excelente coordenação que emerge do sistema, e os agentes são capazes de particionar o grafo (no caso das arquiteturas BBLA) ou de resolver conflitos localmente utilizando-se de comunicação (GBLA e GGBLA). Este comportamento emergente será estudado em detalhes na seção 6.2.

Um fato curioso observado é que a arquitetura BBLA obtém melhor performance que a GBLA quando ambos utilizam o modelo de recompensas WLU. Isto se deve ao fato de, ao

utilizar a recompensa WLU, aplica-se uma punição artificial visando evitar conflitos entre os agentes que querem visitar um mesmo nó. No entanto, como os agentes GBLA comunicam suas intenções de ações, eles podem naturalmente aprender se a visita a um determinado nó irá resultar em conflito ou não, e a informação dada pela recompensa WLU passa a ser supérflua, tornando mais lento o processo de aprendizagem (se torna um ruído no sinal de reforço), degradando a performance final.

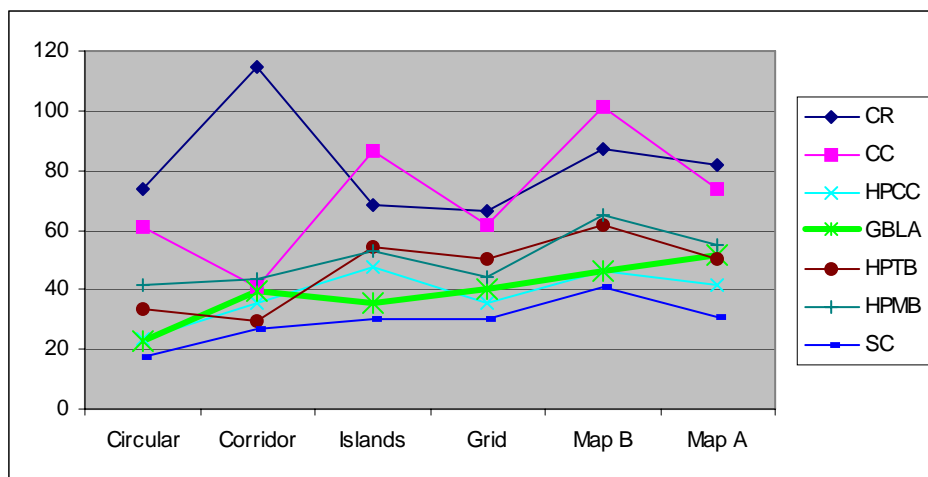
#### 5.2.4 COMPARAÇÃO COM ABORDAGENS ANTERIORES

Nesta seção, comparamos este trabalho com os trabalhos anteriores. Para isto, utilizaremos a arquitetura GBLA\_SU, que obteve melhor desempenho nos experimentos realizados até agora neste trabalho, e a compararemos com as demais abordagens dos trabalhos passados em todos os cenários de simulação da Figura 9. Os experimentos com as demais arquiteturas foram feitos pelos seus criadores, utilizando-se o mesmo ambiente de experimentação [2].

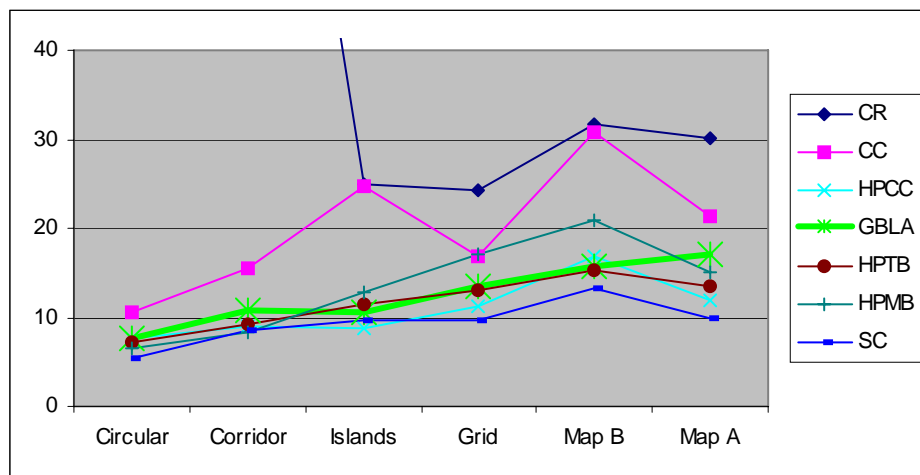
É importante frisar que o desempenho de cada arquitetura, avaliado neste experimento, não é o único parâmetro a ser levado em conta no momento de optar por uma determinada abordagem. Embora uma arquitetura possa ter obtido o melhor desempenho segundo um determinado critério de avaliação, ela pode ter restrições quanto a sua aplicabilidade. Por exemplo, esta pode não ser aplicável em cenários onde não seja permitida comunicação global entre os agentes, em cenários mais dinâmicos onde as regiões possuem prioridades diferentes, em tarefas onde não se conhece o cenário a priori, em circunstâncias onde os recursos computacionais são escassos (ex. em aplicações embarcadas) entre outras situações. Sendo assim, a comparação de desempenho realizada nesta seção fornece apenas um dos parâmetros de comparação entre as arquiteturas, e os demais fatores são discutidos a seção 5.3.

##### 5.2.4.1 Resultado Experimental

A arquitetura GBLA\_SU foi comparada com as abordagens descritas na seção 2.3, utilizando-se 6 diferentes ambientes de experimentação e populações de 5 e 15 agentes. Os resultados, utilizando-se a métrica da média da ociosidade média são mostrados na Figura 15 e na Figura 16, respectivamente, onde CR = *Conscientious Reactive*, CC = *Cognitive Coordinated* (ver seção 2.3.1), HPCC = *Heuristic Cognitive Coordinated* (ver seção 2.3.2), GBLA = *Gray-Box Learner Agent* (agente desenvolvido neste trabalho), HPTB = *Heuristic Pathfinder Two-shots Bidder*, HPMB = *Heuristic Pathfinder Mediated Trade Bidder* (ver seção 2.3.3), SC = *Single-Cycle* (ver seção 2.3.4).



**Figura 15. Comparação da média da ociosidade média das abordagens com uma população de 5 agentes.**



**Figura 16. Comparação da média da ociosidade média das abordagens com uma população de 15 agentes. Os primeiros resultados da CR são 73.71 e 107.01, respectivamente.**

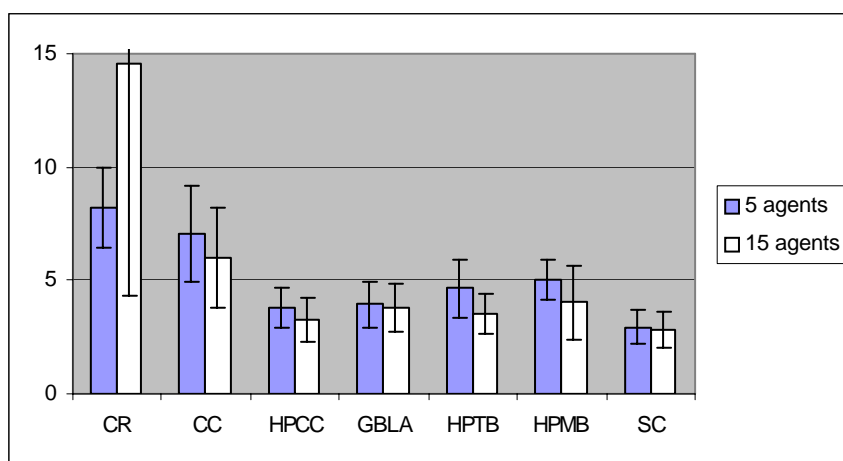
Observa-se que a abordagem de ciclo-único (SC), obteve o melhor resultado na grande maioria dos cenários, e as abordagens pioneiras (CR e CC), apresentam a pior performance em todos os cenários. Quanto as demais arquiteturas, nos experimentos com 5 agentes, os agentes HPCC e GBLA apresentaram o segundo melhor desempenho, seguidos pelo HPTB. Para 15 agentes, os agentes HPCC, HPTB e GBLA obtiveram uma performance equivalente.

É interessante analisar a influência das topologias dos grafos e do tamanho da população na performance de cada arquitetura. Para melhor entender este ponto, a Figura 17 mostra os resultados de cada arquitetura utilizando a média da ociosidade média normalizada

(para permitir ver Equação 10), junto com seus respectivos desvios-padrão, sobre todos os mapas utilizados.

No que diz respeito à influência da topologia do grafo, as abordagens SC, GBLA e HPCC apresentam o melhor desempenho (menor ociosidade), e são menos afetados pela mudança de topologia (desvio padrão dos resultados em diferentes mapas variando entre 0.77 e 0.99). O desempenho dos agentes que utilizam mecanismos de negociação varia de acordo com o tamanho da população: HPTB possui um desvio padrão de 1.26 para 5 agentes, e a HPMB de 1.63 para 15 agentes.

As abordagens CC e GG também são as piores sobre esta perspectiva, e seus desvios padrão variam de 1.79 a 10.24. Finalmente, considerando a variação de população, SC e GBLA exibem o comportamento mais estável (mesma performance relativa independentemente do número de agentes). As demais variam suas performances (para o melhor, ou para o pior): adicionar agentes a população melhora a performance normalizada, mas retirar agentes a piora.



**Figura 17. Média da ociosidade média normalizada com seus respectivos desvios-padrão. Os resultados são uma média para todos os mapas.**

#### 5.2.4.2 Discussão

Conforme mostrado na seção passada, a abordagem SC apresentou os melhores resultados sob os diferentes aspectos mostrados. Este desempenho excelente pode ser explicado observando que esta abordagem é totalmente centralizada e utiliza um esquema rígido e explícito de coordenação: os agentes seguem um mesmo caminho e mantém a mesma distância entre eles. No entanto, como este esquema é centralizado, predefinido e fixo, este tipo de arquitetura terá problemas em certas situações, tais como em ambientes dinâmicos (ex.

robôs precisando recarregar baterias); em grafos grandes (devido a complexidade das soluções aproximadas do TSP); e em tarefas de patrulha onde as regiões tenham diferentes prioridades.

De fato, embora a abordagem GBLA tenha obtido a terceira melhor colocação em termos de desempenho (ligeiramente inferior a HPCC), ela é a única das três que é naturalmente distribuída, e utiliza apenas informação local, ao contrário do HPCC, por exemplo, que requer informação global e utiliza um elemento central de coordenação (além de requerer comunicação de todos os agentes com o ponto central). Com o intuito de poder melhor comparar tais abordagens, é necessário analisar as suas respectivas aplicabilidades. A Tabela 2 mostra as restrições de cada arquitetura no que diz respeito à **visibilidade** da informação e as necessidades de **comunicação**.

Uma outra descoberta interessante deste experimento diz respeito à fraca performance das arquiteturas pioneiras (CR e CC), o que demonstra que é necessário um bom grau de sofisticação nas arquiteturas para conseguir uma boa performance neste problema. No entanto, isto não indica que tais abordagens não sejam úteis, havendo cenários onde, por exemplo, a única arquitetura aplicável é a arquitetura CR, conforme mostrado na Tabela 2.

| Arquitetura   | Restrições     |          |        |        |              |        |
|---|----------------|----------|--------|--------|--------------|--------|
|   | Comunicação    |          |        |        | Visibilidade |        |
|   | Sem            | Só Flags | Local* | Global | Local**      | Global |
| <i>Conscious Reactive</i>                           | A              | A/N      | A/N    | A/N    | A/N          | A/N    |
| <i>Cognitive Coordinated</i>                        | N              | N        | N      | A      | N            | A      |
| <i>Heuristic Cognitive Coordinated</i>              | N              | N        | N      | A      | N            | A      |
| <i>Black-Box Reinforcement Learner</i>              | N <sup>1</sup> | A        | A/N    | A/N    | A            | A/N    |
| <i>Gray-Box Reinforcement Learner</i>               | N              | N        | A      | A/N    | A            | A/N    |
| <i>Generalizable Gray-Box Reinforcement Learner</i> | N              | N        | A      | A/N    | A            | A/N    |



|   |                |                |                |   |                |     |
|---|----------------|----------------|----------------|---|----------------|-----|
| <i>ClinG</i>                                      | N <sup>1</sup> | N <sup>1</sup> | N <sup>1</sup> | A | N <sup>2</sup> | A   |
| <i>Black-Box CLinG Reinforcement Learner</i>      | N <sup>1</sup> | N <sup>1</sup> | N <sup>1</sup> | A | N <sup>2</sup> | A   |
| <i>Gray-Box CLinG Reinforcement Learner</i>       | N              | N              | N <sup>1</sup> | A | N <sup>2</sup> | A   |
| <i>Heuristic Pathfinder Two-Shots Bidder</i>      | N              | N              | N              | A | A              | A/N |
| <i>Heuristic Pathfinder Mediated Trade Bidder</i> | N              | N              | N              | A | A              | A/N |
| <i>Single-Cycle</i>                               | N <sup>1</sup> | N <sup>1</sup> | N <sup>1</sup> | A | N              | A   |

A = Aplicável; N = Não Aplicável; A/N = Aplicável, mas subutilizado.

\* nós vizinhos, inclui *flags*; \*\* nós vizinhos

<sup>1</sup> Aplicável caso haja Visibilidade Global; <sup>2</sup> Aplicável caso haja Comunicação Global

**Tabela 2 – Comparação da aplicabilidade das arquiteturas considerando restrições de comunicação e visibilidade.**

### 5.3 CONCLUSÕES

Nesse capítulo foi mostrado o método experimental empregado para avaliar as diferentes arquiteturas propostas: diferentes representações de estados, funções de recompensa instantânea e modelos de comunicação utilizados, bem como a comparação com as abordagens anteriores.

O desempenho das arquiteturas propostas se mostrou bastante satisfatória. No entanto, algumas questões ainda estão em aberto, e necessitam de uma análise mais profunda, a saber:

- Como é a estratégia que emerge do aprendizado de cada arquitetura?
- Por que uma determinada arquitetura obteve melhor performance que outra? Qual o tipo de estratégia aprendida pelos agentes e como esta se relaciona com a performance obtida?
- Que tipo de conhecimento é aprendido pelo agente? É possível entender as regras de mapeamento de estados em ação aprendidas?
- Quão geral é este conhecimento, em termos de cenários de simulação?

No próximo capítulo buscamos as respostas a tais perguntas, fazendo uma análise aprofundada do comportamento de cada estratégia.

## 6 EXPERIMENTOS ADICIONAIS

Uma vez implementadas e avaliadas as abordagens propostas, percebeu-se que era necessário um maior entendimento das mesmas, não apenas do ponto de vista quantitativo dos resultados, mas sim de como interpretar os mesmos. Neste capítulo, faremos esta análise das abordagens em três etapas. Na seção 6.1, analisamos o poder de generalização do conhecimento obtido por um agente para outros cenários. Na seção 6.2, buscamos entender melhor o comportamento de especialização (particionamento do grafo) que emerge em algumas arquiteturas, através de medidas quantitativas que sugerem a existência de tal comportamento. Finalmente, na seção 6.3, propomos um método para obter regras legíveis (facilmente interpretáveis por um ser humano) a partir do conhecimento obtido por aprendizagem por reforço.

### 6.1 *GENERALIZAÇÃO DO CONHECIMENTO APRENDIDO*

Um problema com a arquitetura GBLA utilizada nos experimentos do capítulo anterior é que, como ela inclui em sua representação de estados a informação sobre qual nó especificamente cada agente está visitando (NodeID), o conhecimento aprendido em um determinado cenário de simulação não pode ser reutilizado em um outro cenário.

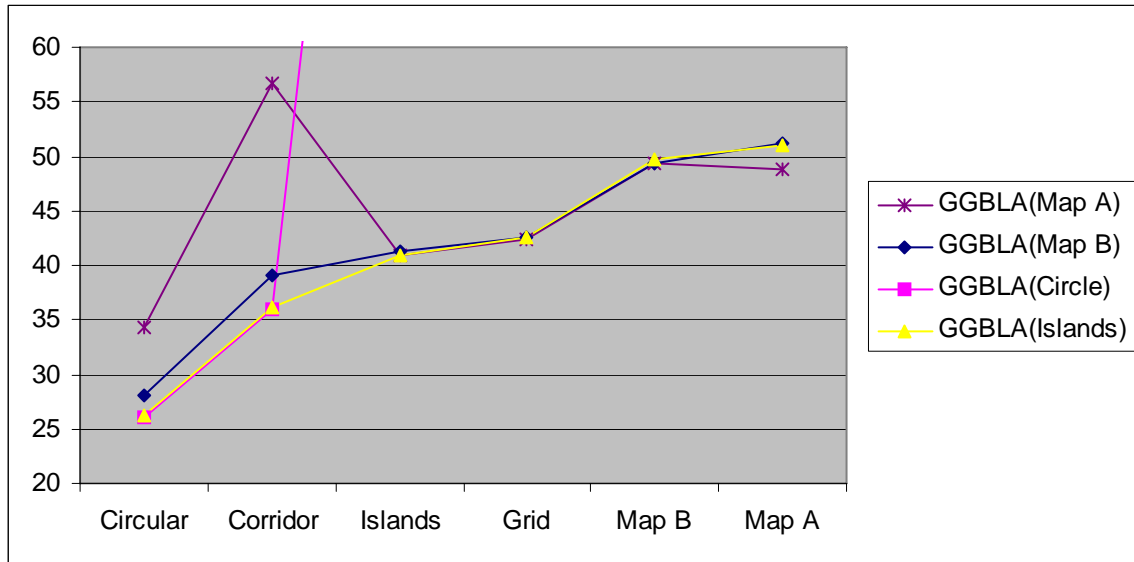
Nesta seção, investigamos os agentes GGBLA (*generalizable gray-box learner agents*). Estes agentes possuem uma representação de estados similar à dos agentes GBLA, mas não incluem a informação NodeID. Tentaremos responder a duas questões básicas nos próximos experimentos, a saber: i) que tipo de treinamento é necessário (ex. que cenário de simulação utilizar no treinamento) para se obter um conhecimento suficientemente geral e capaz de obter boa performance em outras instâncias do problema? e ii) qual a diferença de performance entre agentes treinados especificamente para atuarem em um determinado cenário e agentes que possuem um conhecimento geral ?

A Figura 18 mostra o desempenho de 4 agentes GGBLA que diferem entre si pelo treinamento inicial que receberam<sup>12</sup>: cada agente foi treinado em um mapa específico, e foi avaliado em todos os mapas. Observa-se, como era de se esperar, que os agentes obtêm a melhor performance nos seus próprios cenários de treinamento. No entanto, os agentes

---

<sup>12</sup> Não foi mostrado o agente GGBLA(*Corridor*) porque o conhecimento aprendido (e conseqüentemente o desempenho obtido) foi idêntico ao do agente GGBLA(*Circle*)

treinados no mapa *B* e no mapa *Islands* possuem claramente um desempenho superior aos demais, e os agentes treinados no mapa *Circle* e *Corridor* não conseguem generalizar o conhecimento obtido para os demais mapas, obtendo performance muito inferior.

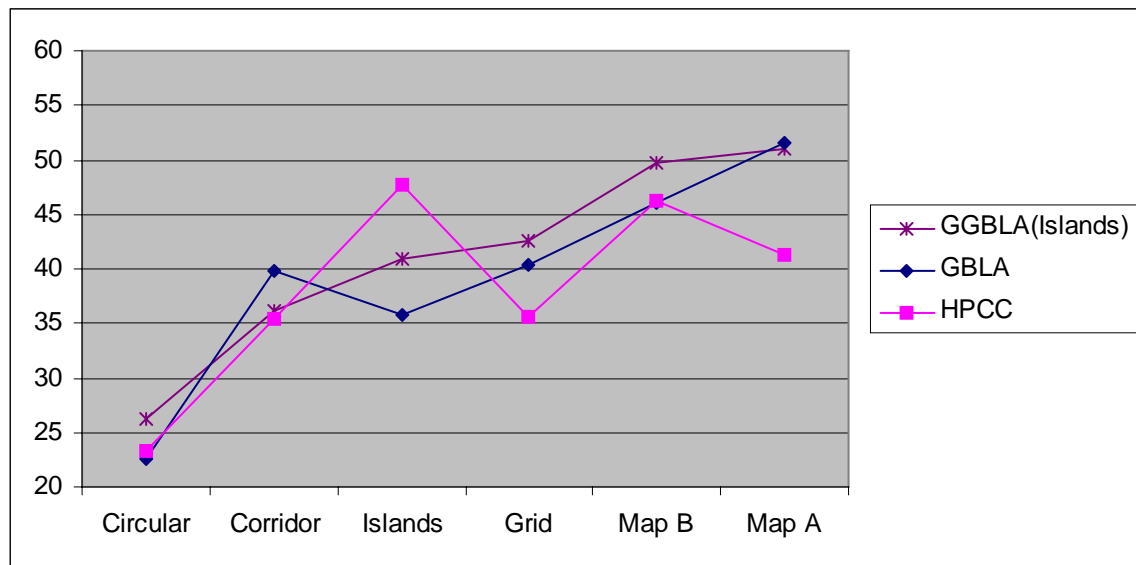


**Figura 18. Comparação da performance dos agentes GGBLA em todos os cenários, com o cenário de treinamento indicado entre parênteses.**

Tal resultado mostra que, para um treinamento inicial ser efetivo, é preciso que este tenha sido feita em uma topologia suficientemente complexa (contendo subgrafos com diferentes topologias). Sendo assim, os treinamentos realizados nos mapas de topologias mais simples como os mapas *Circle* e *Corridor* apresentaram a pior performance, enquanto que os de topologia mais complexa (*Islands* e *Map B*), o melhor resultado.

A Figura 19 mostra o “preço” que é pago por esta generalidade. Nela, temos a performance do agente GGBLA treinado no mapa *Islands* (o que obteve melhor performance), comparado a arquitetura GBLA tradicional, treinada especificamente para cada mapa avaliado, e à arquitetura HPCC<sup>13</sup>. Os resultados mostram que a perda de performance é baixa, e chega a ser zero em alguns mapas (*A* e *Corridor*), e ele chega a superar a abordagem HPCC no mapa mais complexo, o *Islands*.

<sup>13</sup> Como o objetivo deste experimento era avaliar a generalização de abordagens treinadas com AR, não incluímos as demais arquiteturas anteriores, incluindo apenas a HPCC para ilustrar como uma abordagem não adaptativa se comporta segundo este aspecto.



**Figura 19. Comparação de performance do agente GGBLA treinado no mapa Ilhas com o agente GBLA (treinado especificamente para cada mapa) e o agente HPCC**

Por fim, a principal conclusão que se pode tirar deste experimento é que, em situações onde o cenário de simulação no qual a arquitetura será utilizada é desconhecido, ou em que a arquitetura será utilizada em diferentes cenários de simulação, e não há tempo suficiente para executar um treinamento específico para cada mapa, a arquitetura GGBLA surge como uma opção extremamente viável para aplicação das idéias propostas neste trabalho. Caso não hajam estas restrições, e o custo computacional não for um problema, a arquitetura GBLA continua sendo a melhor opção por apresentar um melhor desempenho final se devidamente treinada.

## 6.2 ESPECIALIZAÇÃO COMO COMPORTAMENTO EMERGENTE

Observando certas arquiteturas de agentes no simulador, observa-se claramente o resultado da coordenação emergente do sistema: os agentes são capazes de particionar o grafo original em sub-regiões distintas, e se especializar em uma destas regiões, executando uma estratégia cíclica em seus nós.

Nesta seção, é proposta uma medida quantitativa de tal comportamento, de maneira que seja possível entender: i) por que tal comportamento emerge do sistema? ii) qual a diferença de comportamento entre as arquiteturas propostas neste trabalho?

Foram selecionadas para este experimento as três arquiteturas propostas neste trabalho (BBLA, GBLA e GGBLA), mais uma arquitetura Aleatória reativa, para servir como base na comparação. Na Figura 20, mostramos um histograma de visitas a cada nó, por cada agente,

medidos em uma simulação no Mapa B com 5 agentes. No eixo X de cada gráfico, temos os nós (NID), e no eixo Y temos o número de vezes que aquele nó foi visitado por um determinado agente. Observa-se que há claramente um particionamento dos nós para a arquitetura BBLA, há uma especialização clara do agente 4 da arquitetura GBLA, e não há sinais deste tipo de comportamento nos agentes GGBLA ou nos agentes aleatórios.

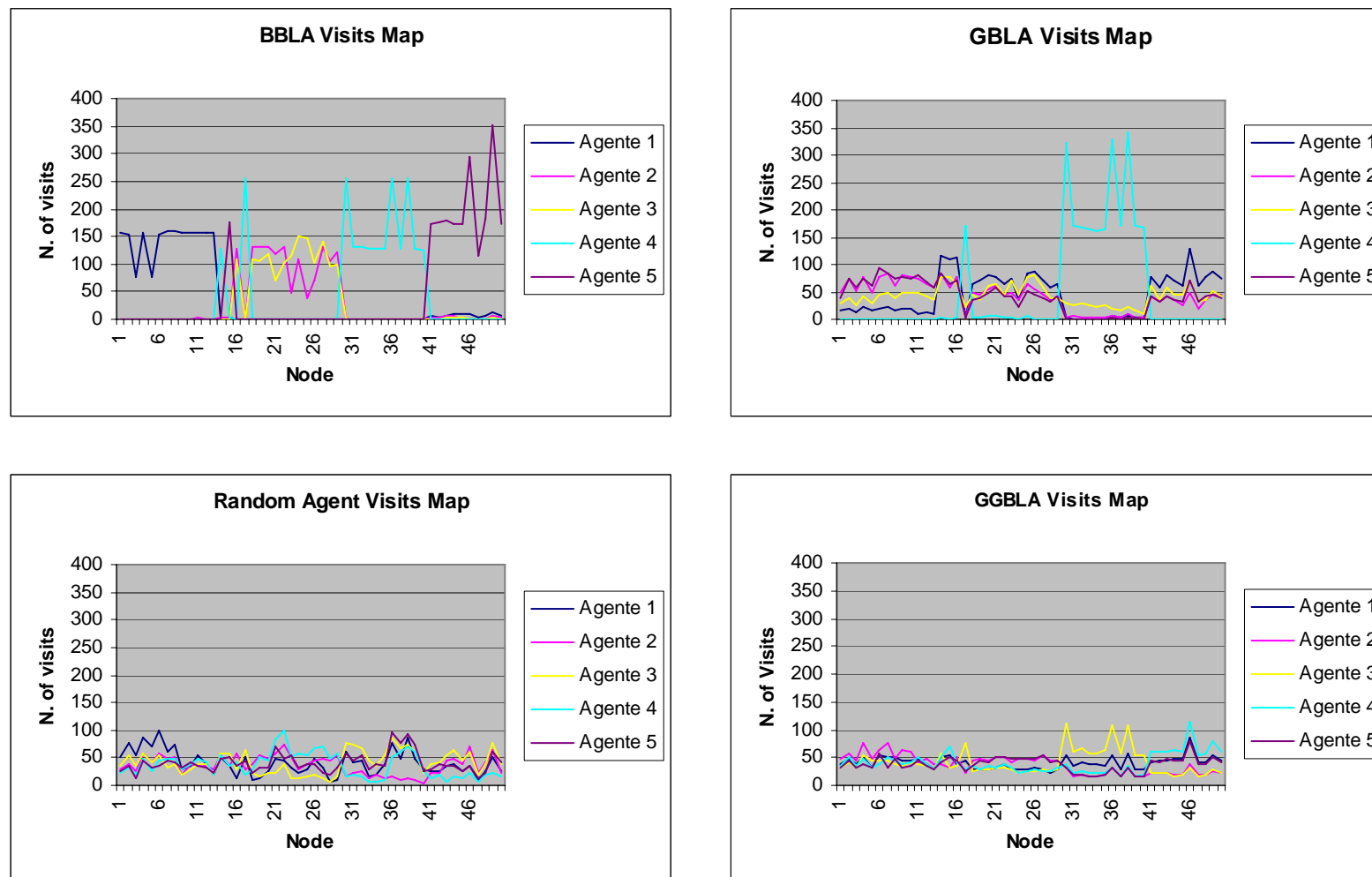
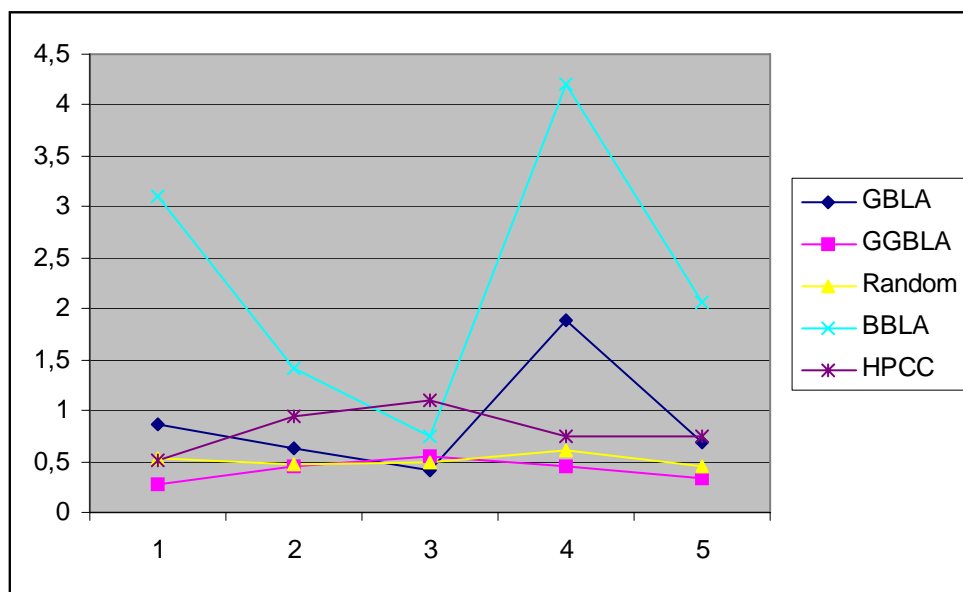


Figura 20. Frequência de visitas a cada nó feitas por cada agente, por arquitetura.

De maneira mais precisa, podemos medir quantitativamente o nível de particionamento das estratégias de cada arquitetura analisando o desvio padrão das curvas mostradas no histograma de visitas de cada agente da Figura 20: quanto maior este desvio, mais especializada é a sua estratégia. Para que este resultado seja independente do número total de visitas ou do número de nós do mapa, faz-se uma normalização dividindo o mesmo pelo valor esperado do número de visitas por nó (frequência média de visitas). Os resultados da coleta desta métrica são mostrados na Figura 21.



**Figura 21. Desvio padrão do valor da frequência de visitas por nó (eixo Y), normalizado, calculado para cada agente (eixo X) das arquiteturas BBLA, GBLA, GGBLA, Aleatória e HPCC.**

Analisando tais dados, e observando-se os agentes no simulador, chega-se a uma explicação a respeito do surgimento de tal comportamento emergente dos agentes BBLA. Sabe-se que tais agentes não se comunicam explicitamente, logo, quando eles estão atuando em uma mesma região, eles tendem a entrar em conflito e decair sua performance. Mais ainda, quando há a presença de outros agentes, cada agente passa a ter uma representação de estados não markoviana, e o ambiente passa a se comportar de maneira não-estacionária (as probabilidades de transição de estados e a função de recompensas passa a satisfazer uma distribuição não-estacionária devido a aprendizagem simultânea com os demais agentes). Neste contexto, a especialização surge para as arquiteturas BBLA como a única maneira de obter convergência na sua performance, e lidar com um problema markoviano: ao particionar



o grafo em regiões disjuntas, eles se isolam e diminuem a influência das ações dos outros agentes em suas próprias dinâmicas.

Nas arquiteturas GBLA, embora haja também a especialização de determinados agentes, não chega a haver particionamento: como os agentes podem se comunicar, eles podem “invadir” regiões patrulhadas por outros agentes sem entrar em conflito com eles, e sem degradar a performance do grupo. Pelo contrário, conseguem com isso melhorar ainda mais a performance. Não somente isso, observa-se que, ao retirar a informação de NodeID da representação de estados da arquitetura GBLA (obtendo a arquitetura GGBLA), os agentes são capazes de manter a performance obtida (ver seção 6.1), mesmo agindo de maneira totalmente distinta da arquitetura original (GBLA). Em tal arquitetura, é muito difícil haver especialização, pois os agentes não têm a informação geográfica sobre em quais nós eles estão. No entanto, por intermédio do mecanismo de comunicação eles conseguem não entrar em conflito e manter o desempenho. A arquitetura Aleatória foi mostrada apenas para servir como base de comparação, dado que a própria topologia do cenário de simulação utilizada poderia induzir a uma certa especialização por parte dos agentes.

### 6.3 VISÃO SIMBÓLICA DO CONHECIMENTO APRENDIDO

Embora a utilização de aprendizagem por reforço tenha se mostrado bastante eficaz neste trabalho, observa-se uma deficiência desta abordagem: o conhecimento obtido não pode ser facilmente interpretável por um especialista. Em outras palavras, o conhecimento aprendido pelos agentes fica concentrado em suas *Q-Tables*, e as regras aprendidas (mapeamento estado-ação) não podem ser analisadas facilmente. Além disto, como é difícil se observar tal conhecimento, conseqüentemente se torna difícil entender como ou porque o mesmo funciona, e também se torna difícil qualquer tarefa de manutenção.

Recentemente, algumas abordagens estão começando a tentar lidar com esta limitação, combinando formalismos lógicos com os processos de decisão de Markov [29], ou utilizando-se de representações de estado relacionais [23]. Nesta seção, propomos um método para obter regras legíveis (facilmente interpretáveis por um ser humano) a partir do conhecimento obtido por aprendizagem por reforço. Uma vez mostrado tal método, mostramos sua aplicação na tarefa da patrulha e sugerimos outras aplicações práticas para o mesmo.

Para obter regras simbólicas a partir do conhecimento obtido pela aprendizagem por reforço (tabela Q), nós seguimos o seguinte processo:

1. Transformar cada entrada da tabela Q em um exemplo de aprendizagem;

2. Utilizar um algoritmo de aprendizagem de regras como ID3 [41] utilizando os exemplos do passo 1.

Um exemplo de aprendizagem, citado no passo 1, é uma tupla formada pelos valores dos atributos descritores do estado, acrescidos de um atributo indicando a melhor ação a ser tomada naquele estado (a que possui maior valor Q). Um exemplo de aprendizagem possível, onde as siglas utilizadas para representar os atributos são as mesmas descritas na seção 5.2.2, seria a tupla: <MO=4, NA=3, PE=2, AÇÃO=4>, que significa que quando os três primeiros atributos (MO, NA e PE) apresentarem os valores indicados (4, 3 e 2, respectivamente), a ação escolhida pelo agente é a ação 4.

Como o número de estados das abordagens GBLA e BBLA era bastante grande, analisamos a aplicação deste processo nos agentes GGBLA, que possuem cerca de 125 estados possíveis para os cenários de simulação utilizados. A Figura 22 mostra um exemplo de regra aprendida pelo agente GGBLA, e obtida utilizando o algoritmo ID3 e o processo aqui descrito.

```

SE MO = 4 ENTÃO
  SE NA = 4 ENTÃO
    AÇÃO = VISITAR 1
  SENÃO
    SE PE = 4 ENTÃO
      AÇÃO = VISITAR 3
    SENÃO
      AÇÃO = VISITAR 4

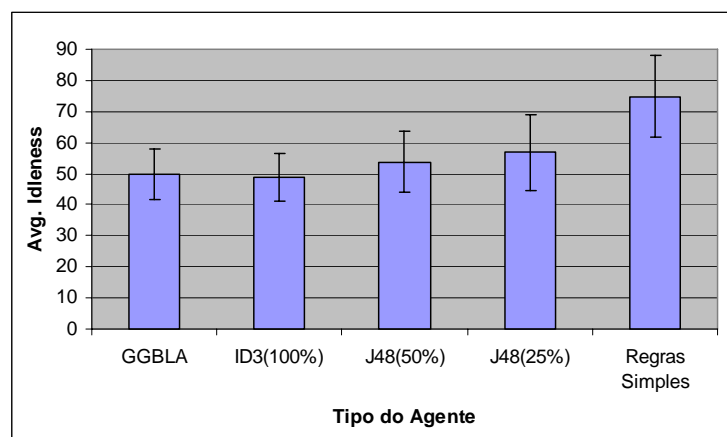
```

**Figura 22. Exemplo de regra aprendida pelo agente GGBLA.**

A regra é bastante simples de ser entendida: se o vizinho mais ocioso é o 4, o agente deve visitar este vizinho, a não ser que exista um outro agente com a intenção de visitá-lo, ou que este tenha sido o último nó em que ele esteve, onde nestes casos ele deve visitar os vizinhos 1 e 3, respectivamente. Embora todas as regras sejam simples de serem lidas, algumas regras aprendidas são bastante intuitivas (fica claro para o especialista porque ela funciona bem), e outras não, pois são fruto de resultados observados estatisticamente durante o aprendizado do agente. Uma vez observado isto, foi formulada a seguinte questão: será que estas regras intuitivas são as únicas responsáveis pelo bom comportamento do agente, e as demais poderiam ser excluídas sem haver perda de desempenho pelos agentes?

O seguinte experimento foi realizado visando mostrar que, se o objetivo do agente desenvolvido é maximizar o seu desempenho, todas as regras aprendidas, intuitivas ou não, fazem diferença sim no desempenho do agente. Utilizando o algoritmo J48 [50], extensão do algoritmo ID3, é possível parametrizar a geração das regras por um fator de confiança. Este fator de confiança indica o grau de precisão aceitável em cada regra (número de exemplos positivos cobertos, dividido pelo número de exemplos negativos). Quanto menor o grau de precisão, menos regras são geradas pelo algoritmo, levando a uma provável maior generalidade das mesmas. Em outras palavras, regulando-se este parâmetro é possível eliminar as regras não intuitivas e deixar apenas as regras mais intuitivas, e que coincidentemente cobrem maior parte dos exemplos de treinamento.

Desta maneira, foram criados 5 agentes semelhantes, mas que diferem entre si pelo nível de detalhe das suas regras. A Figura 23 mostra uma comparação entre estes 5 agentes. Em um dos extremos, temos a arquitetura GGBLA, contendo o conhecimento original, com regras intuitivas e não intuitivas, e no outro uma arquitetura que utiliza uma generalização das regras aprendidas feita por um especialista: o agente deve visitar o nó vizinho mais ocioso, a não ser que exista um outro agente com a intenção de visitá-lo, ou que este tenha sido o último nó em que ele esteve, onde, nestes casos, ele escolhe, dentre os outros vizinhos, aquele que possui a maior ociosidade. Os três outros agentes utilizados foram obtidos a partir da engenharia reversa do conhecimento do GGBLA, utilizando o processo descrito nesta seção, e utilizando diferentes níveis de precisão em suas regras.



**Figura 23. Comparação de agentes GGBLA que utilizam regras com diferentes níveis de detalhe.**

Observa-se que os agentes GGBLA e seu correspondente, cujas regras foram obtidas pelo algoritmo ID3, possuem performance estatisticamente equivalente. Entretanto, quanto mais se generaliza as regras obtidas, perde-se um pouco em performance.

Vale a pena ressaltar, entretanto, as vantagens de se utilizar tal simplificação de agentes. Em primeiro lugar, quanto menos regras/pares estado-ação possuem as abordagens (quanto mais a direita estão no gráfico da Figura 23), mais fáceis de serem entendidas por um humano que trabalhe na engenharia do conhecimento de tais agentes. De fato, um problema com abordagens numéricas de aprendizagem de máquina, como aprendizagem por reforço, é a dificuldade de validação do conhecimento obtido por um especialista. É possível realizar tal validação utilizando o método aqui proposto, e mais fácil será a mesma quão mais simples for a arquitetura.

Outra vantagem das arquiteturas mais simples obtidas diz respeito à complexidade em espaço de sua implementação. Enquanto a abordagem original GGBLA requer o armazenamento de uma tabela cujo tamanho é potencialmente muito grande, e ainda cresce com o tamanho do mapa, as abordagens mais simples decaem de maneira brusca o número de espaço requerido para a sua implementação. Isto permite sua aplicação em ambientes onde haja restrições quanto à complexidade em memória, como em aplicações para dispositivos móveis, por exemplo.

## 7 CONCLUSÃO

Este trabalho investigou diversas alternativas para o uso de aprendizagem por reforço na criação de uma solução para tarefa da patrulha mutli-agente, e representa o primeiro esforço realizado com sucesso no sentido de aplicar estratégias adaptativas neste contexto. Destacamos neste capítulo as principais contribuições deste trabalho, e apontamos trabalhos futuros.

### 7.1 PRINCIPAIS CONTRIBUIÇÕES

As soluções propostas, embora baseadas em técnicas e algoritmos tradicionais, como *Q-Learning*, trabalharam em um contexto não usual, a saber, em um ambiente não estacionário, provocado pela adaptação independente e simultânea realizada por todos os agentes. Apesar disto, a maioria das arquiteturas propostas se apresentaram bastante estáveis e o processo de aprendizagem convergiu para soluções muito boas. O esforço necessário para o projeto da estratégia de coordenação também foi muito pequeno do ponto de vista de engenharia do conhecimento, já que o comportamento coordenado emergiu quase que automaticamente como resultado do processo de aprendizagem coletiva. Além disto, a natureza distribuída da abordagem proposta a torna computacionalmente muito eficiente e, conseqüentemente, atrativa para aplicações em tempo real.

Foi dado um tratamento detalhado das questões envolvidas na modelagem da tarefa da patrulha dentro do *framework* de AR e na análise da optimalidade das soluções, em particular para o critério da média da ociosidade média. Estes resultados complementam trabalhos existentes para a análise teórica de limites inferiores para a tarefa da patrulha [15].

Consideramos que este trabalho constitui um caso de estudo positivo da aplicação das técnicas padrão de AR em sistemas multi-agentes complexos e dinâmicos. Além de incrementar o *benchmark* das soluções existentes para a patrulha atualmente, as reflexões e técnicas aqui apresentadas podem ser igualmente valiosas em outros problemas com propriedades similares, como os mostrados na seção 2.2.

Uma outra contribuição que pode ser levantada diz respeito à união de técnicas de comunicação e modelos de recompensa baseados no *framework* COIN [52], que não havia sido feita na literatura. Muitas conclusões interessantes foram obtidas e aspectos esclarecidos com respeito a sua interação. Além disto, propusemos diferentes técnicas e métricas de análise que nos permitiram **entender** melhor os resultados obtidos. Dentre estas, a métrica de

particionamento do grafo e os experimentos de generalização se apresentam como novas formas de analisar estratégias de patrulhamento, e a técnica para visão simbólica do conhecimento pode ser aplicada a outros problemas de aprendizagem por reforço onde se possa aplicar algoritmos como *Q-Learning*.

Por fim, pode-se citar como contribuição o fato deste trabalho estar servindo de base para outros trabalhos. No centro de informática da Universidade Federal de Pernambuco, há uma dissertação de mestrado em andamento cujos experimentos estão utilizando o componente *RLEngine*, aqui desenvolvido. No *Laboratoire d'Informatique de Paris VI* (LIP6), há um trabalho de pesquisa em desenvolvimento que é uma extensão dos agentes desenvolvidos neste trabalho. Explicamos melhor este último na seção 7.2.3.

## 7.2 TRABALHOS FUTUROS

Destacamos aqui os esforços atuais e futuros para a continuação deste trabalho.

### 7.2.1 APRENDIZAGEM POR REFORÇO COORDENADA NA PATRULHA

A aplicação desta técnica na tarefa patrulha envolve dois esforços principais. Primeiramente, é necessário adaptar o método de ARC para trabalhar com processos de decisão semi-markovianos. Um segundo ponto a ser tratado diz respeito à estrutura de grafo de coordenação descrita na seção 3.3.3. Este método foi desenvolvido pressupondo que o grafo de coordenação é uma estrutura estática. Em outras palavras, assume-se que as influências entre as ações dos agentes são constantes, e não variam com o tempo.

Na tarefa da patrulha, entretanto, as ações entre os agentes têm um impacto claramente local: os agentes mais afetados pela visita a um nó são os agentes mais próximos aquele nó. Como os agentes estão constantemente mudando suas posições geográficas, estas influências variam dinamicamente, e o grafo de coordenação tem suas arestas alteradas também dinamicamente.

Uma vez atacadas estas duas questões, a utilização de tal método na patrulha passa a ser direta, e se apresenta como uma abordagem bastante promissora, a ser estudada em trabalhos futuros.

### 7.2.2 DESENVOLVIMENTO DE UM NOVO SIMULADOR

Está em andamento no Centro de Informática da UFPE, um projeto de desenvolvimento de um novo simulador para a patrulha, na qual o autor deste trabalho

participa. Dentre as novas funcionalidades que estão sendo implementadas, pode-se citar, entre outras:

- Cenários mais dinâmicos: mapas com regiões cujas prioridades de visita são diferentes e mapas cuja topologia é alterada com o tempo;
- Arquitetura cliente/servidor melhorada, independente de plataforma;
- Novo visualizador de simulação, permitindo a re-exibição de uma determinada simulação, a partir de uma iteração específica;

### *7.2.3 UNIÃO DA ABORDAGEM CLING COM OS AGENTES DESTE TRABALHO*

Um trabalho em desenvolvimento consiste em unir a abordagem CLinG (ver seção 2.3.5) com os agentes desenvolvidos neste trabalho. Como foi explicado, a abordagem CLinG consiste em um mecanismo de propagação da informação, que faz com que os agentes possam utilizar em seu processo de decisão uma informação global, mesmo que ele só analise a sua vizinhança local. Os agentes que utilizaram originalmente a abordagem CLinG eram agentes baseados em utilidade, que escolhiam o nó vizinho cuja função de utilidade (obtida através de informação propagada globalmente) possuía maior valor.

Tal abordagem foi combinada com os agentes BBLA e GBLA, substituindo a informação sobre o nó vizinho com maior ociosidade (MO), pela informação de ociosidade propagada globalmente. Deste modo, os agentes podem utilizar todos os mecanismos de aprendizagem por reforço, e ainda tirar proveito de uma informação global, sem explodir seu espaço de estados.

## REFERÊNCIAS

- [1] Abate, F.R. *The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers*. Oxford University Press, 1996
- [2] Almeida, A., Santana, H., Menezes, T., Ramalho, G., Tedesco, P. & Corruble, V. Recent Advances on Multi-Agent Patrolling. Brazilian Symposium on Artificial Intelligence (SBIA 2004).
- [3] Almeida, A. L., Castro, P. M. M., Menezes, T. R. et al. Combining Idleness and Distance to Design Heuristic Agents for the Patrolling Task. Proceedings of the Second Brazilian Workshop on Games and Digital Entertainment. Salvador, Brasil: 2003
- [4] Almeida, A.: Coordenação na Tarefa de Patrulhamento em Sistemas Multiagentes. Trabalho de Graduação - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2002.
- [5] Andrade, G., Santana, H., Furtado, A., Leitão, A. & Ramalho, G. Online Adaptation of Computer Games Agents: A Reinforcement Learning Approach. Proceedings of the Brazilian Symposium on Computer Games and Digital Entertainment (SBGAMES 2004), Curitiba, Brazil.
- [6] Andrade, R. de C., Macedo, H. T., Ramalho, G. L. et al. Distributed Mobile Autonomous Agents in Network Management. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. Las Vegas, USA: 2001. p. 7-12.
- [7] Arkin, R. C. Behavior-Based Robot Navigation for Extended Domains. In: Adaptive Behaviors. 1992. v. 1(2), p. 201-225.
- [8] Arthur, W. B. Inductive Reasoning and Bounded Rationality (The Bar el Farol Problem). In proceedings of the American Economic Association Annual Meetings. 1994. p. 406-411.
- [9] Balch, T., Arkin, R. C. Behavior-Based Formation Control for Multi-robot Teams. In: IEEE Transactions on Robot and Automation. 1999. v. 20.
- [10] Bernstein, D., Zilberstein, S., Immerman, N. The Complexity of Decentralized Control of Markov Decision Processes. In Proceedings. of the 16th Conf. on Uncertainty in Artificial Intelligence, 2000.
- [11] Bridle, J. S. (1990). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimates of parameters. In Touretzky, D. S., editor,



Advances in Neural Information Processing Systems 2, pages 211-217, San Mateo, CA. Morgan Kaufmann

- [12] Civilization 3. Disponível em: <http://www.civilization3.com/>
- [13] Chalkiadakis, G. Boutilier, C. Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03), pp.709-716, Melbourne (2003).
- [14] Chevaleyre, Y.: The patrolling problem. Tech. Rep. of Univ. Paris 9. (2003) available at <http://lamsade.dauphine.fr/~chevaley/patrol>
- [15] Chevaleyre, Y. & Sempé, F. A theoretical analysis of multi-agent patrolling strategies. Poster at the International Conference on Autonomous Agents and Multi-Agents Systems (AAMAS) (2004).
- [16] Cho, J., Garcia-Molina, H., Page, L. Efficient crawling through URL ordering. In: Computers Networks and ISDN Systems. 1998. p. 161-172
- [17] Cho, J., Garcia-Molina, H. Synchronizing a database to Improve Freshness. In Proceedings of 2000 ACM International Conference on Management of Data (SIGMOD). 2000. p. 117-128.
- [18] Claus, C., Boutilier, C. The dynamics of reinforcement learning in cooperative multiagent systems. National. Conference on AI, 1998.
- [19] Cohen-Solal, J. Les Algorithmes de patrouilles dans les systemes multi-agents. Technical Report of University of Paris VI. (2004)
- [20] Cormen, T. H., Leiserson, T., Rivest, R. et al. *Algoritmos: Tradução da Segunda Edição Americana*. Campus, 2002. p. 470-474
- [21] Counter-Strike. Disponível em: <http://www.counter-strike.net/>
- [22] Ferber, J. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999.
- [23] Guestrin, C. (2003). Planning Under Uncertainty in Complex Structured Environments. Doctoral dissertation, Stanford University.
- [24] Guestrin, C., Lagoudakis, M., Parr, R. (2002) Coordinated Reinforcement Learning. In Proceedings of the International Conference on Machine Learning (ICML-02)
- [25] Guestrin, C., Koller, D., Gearhart, C. & Kanodia, N. Generalizing Plans to New Environments in Relational MDPs. In International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, August 2003.
- [26] Helvig, C. S., Robins, G., Zelikovsky, A. Moving Target TSP and Related Problems. In: Proceedings of the 6<sup>th</sup> Annual European Symposium on Algorithms. 1998. p. 453-464.

- [27] Hoen, P.J., Bohte, S.M. Collective Intelligence with Sequences of Actions: Coordinating actions in Multi-Agent Systems. European Conference in Machine Learning (ECML), 2003.
- [28] Kaelbling, L., Michael, L. and More, A. Reinforcement Learning: A Survey. In: Journal of Artificial Intelligence Research 4, 1996. p. 237-285
- [29] Kersting, K., De Raedt, L. Logical markov decision programs. In Working Notes of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data (SRL-03), pp. 63-70, 2003.
- [30] L'Ecuyer, P. and Cote, S. Implementing a Random Number Package with Splitting Facilities. ACM Transactions on Mathematical Software, 17:98-111 (1991)
- [31] Machado, A. P., Almeida, A., Ramalho, G. L. et al. Multi-Agent Movement Coordination in Patrolling. In: Workshop on Agents in Computer Games. Edmonton, Canada: 2002
- [32] Machado, A. P., Ramalho, G. L., Zucker, J. D. et al. Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures. In: Third International Workshop on Multi-Agent Based Simulation. 2002
- [33] Machado, A. P. *Patrulha Multiagente: uma análise empírica e sistemática*. Dissertação (Mestrado em Ciência da Computação) - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2002.
- [34] Mahadevan, S. (1996) Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results. Machine Learning, 22, 159-195.
- [35] Manber, Udi. *Introduction to Algorithms: A Creative Approach*. Addison-Wesley, 1989. p. 204-214.
- [36] Menezes, T. R. *Negociação em Sistemas Multiagente para Patrulhamento*. Trabalho de Graduação - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2002.
- [37] Mitchell, Tom M. *Machine Learning*. McGraw-Hill Press, 1997
- [38] Riedmiller, M. Merke, A., Meier, D., Hoffmann, A., Sinner, A., Thate, O., Ehrmann, R. (2001) Karlsruhe Brainstormers - A Reinforcement Learning Approach to Robotic Soccer. Lecture Notes in Computer Science, 2019.
- [39] Riedmiller, M., Merke, A. (2002) Using Machine Learning Techniques in Complex Multi-Agent Domains, in Perspectives on Adaptivity and Learning, LCNS, Springer
- [40] RoboCup Rescue home page. Disponível em: <http://www.r.cs.kobe-u.ac.jp/robocup-rescue/>

- 
- [41] Russell, Stuart, Norvig, Peter. *Artificial Intelligence: A Modern Approach*. 1. ed. New Jersey: Prentice Hall, 1995. p. 111-114, 796-808.
- [42] Santana, H., Ramalho, G., Corruble, V. & Ratitch, B. Multi-Agent Patrolling with Reinforcement Learning. Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, New York, NY, EUA.
- [43] Santana, H. Ramalho, G. & Corruble, V. Criando Agentes que Aprendem a Cooperar: Um Estudo de Caso na Patrulha Multi-Agente. III Workshop on Games and Digital Entertainment (WJOGOS), Salvador, Brazil.
- [44] Sempé, F., Drogoul, A. Adaptive Patrol for a Group of Robots. International Conference on Intelligent Robots and Systems, 2003.
- [45] Sempé, F. "Auto-organisation d'une collectivité de robots : application à l'activité de patrouille en présence de perturbations". Tese de Doutorado – Laboratoire d'Informatique de Paris VI. Paris: UPMC, 2004.
- [46] Stroustrup, Bjarne. *The C++ Programming Language*. 3. ed. Addison-Wesley, 1997.
- [47] Sutton, R., Barto, A. (1998) Reinforcement Learning: An Introduction. Cambridge, MA.
- [48] Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112, 181-211.
- [49] Watkins, C. (1989). Learning from Delayed Rewards. Thesis, University of Cambridge, England.
- [50] Witten, I., Frank, E. Data Mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco, 2000.
- [51] Wolpert, D., Tumer, K. An Introduction to Collective Intelligence. Technical Report from NASA-ARC-IC-99-63. Handbook of Agent Technology, AAAI Press / MIT Press, 1999.
- [52] Wolpert, D., Wheeler, K., Tumer, K. Collective Intelligence for control of distributed dynamical systems. *Europhysics letters*, 49(6), Março 2002.
- [53] Woolridge, M. *Introduction to Multi-Agent Systems*. John Wiley & Sons, 2001.
- [54] Tumer, K., Agogino, A., Wolpert, D. Learning sequences of actions in collectives of autonomous agents. In *Autonomous Agents & Multiagent Systems*, p. 378–385, part 1. ACM press, 2002.

- [55] Zlot, R., Stentz, A., Dias, M., Thayer, S. Multi-Robot Exploration Controlled by a Market Economy. In Proceedings of the IEEE International Conference on Robotics and Automation (IRCA), p. 3016-3023, Washington, DC, 2002.