
Pós-Graduação em Ciência da Computação

# "Patrulhamento Multiagente em Grafos com Pesos"

**Por**

***Alessandro de Luna Almeida***

**Dissertação de Mestrado**

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, 10/2003


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

ALESSANDRO DE LUNA ALMEIDA

# "Patrulhamento Multiagente em Grafos com Pesos "

*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

RECIFE, OUTUBRO/2003

# RESUMO

Um grupo de agentes pode ser usado para realizar tarefas de patrulhamento em uma variedade de domínios, desde administração de redes de computadores a jogos de computadores. Apesar de seu grau elevado de aplicabilidade, arquiteturas multiagente para patrulhamento ainda não foram profundamente estudadas. Por isso, o grupo de Inteligência Artificial do CIn-UFPE tem desenvolvido um trabalho pioneiro sobre o tema, o qual já rendeu uma dissertação de mestrado e algumas publicações. Essa nova dissertação reflete sobre o problema da patrulha de uma forma mais geral, enriquece a representação do ambiente a ser patrulhado ao utilizar grafos com pesos e propõe arquiteturas mais avançadas para resolver o problema. A fim de realizar esse estudo, além de propor novas arquiteturas de sistemas multiagentes, foram repensados os critérios de avaliação que deveriam ser utilizados, criados novos cenários de experimentação e modificado o simulador de patrulhamento já desenvolvido.

Palavras-chave: sistemas multiagentes, coordenação de agentes, patrulhamento, grafos

# ABSTRACT

A group of agents can be used to perform patrolling tasks in a variety of domains ranging from computer network administration to computer games. Despite its wide range of potential applications, multi-agent architectures for patrolling have not been studied in depth yet. The Artificial Intelligence research group at CIn-UFPE has been developing a pioneer work on the subject, which has produced a master dissertation and some publications. This new work reflects on the patrolling problem in a more general way, enriches the environment representation by using weighted-graphs and proposes more advanced architectures to solve the problem. In order to accomplish this study, besides proposing new architectures of multi-agent systems, we have reconsidered the evaluation criteria to be used, created new experimental scenarios, and made some modifications to the already developed simulator.

Keywords: multiagent systems, coordination, patrolling, graphs

# AGRADECIMENTOS

*"Make it a habit to tell people thank you. To express your appreciation, sincerely and without the expectation of anything in return. Truly appreciate those around you, and you'll soon find many others around you. Truly appreciate life, and you'll find that you have more of it".*

*(Ralph Marston)*

Agradeço em primeiro lugar a minha mãe, a qual, apesar de ausente fisicamente, esteve sempre presente com seus ensinamentos, certamente indispensáveis para essa dissertação tornar-se realidade.

A meu pai, meus irmãos e demais familiares, pelo apoio e incentivos dados durante toda minha vida, em particular durante a minha formação acadêmica e profissional.

Ao meu eterno orientador e amigo Geber, o qual desde o início do segundo ano de curso deposita confiança em mim na realização de trabalhos conjuntos.

A todos meus amigos, especialmente Maroka, pela compreensão e apoio dado durante as diversas fases do meu mestrado.

A Mauro, o qual me ajudou na geração de equações e, juntamente com Gustavo, Fernando, Ivan, Pedro e Sérgio, teve papel essencial na execução dos experimentos.

Ao CIn-UFPE e ao CNPQ pelos recursos físicos, humanos e financeiros, que foram contribuições indispensáveis ao desenvolvimento desse trabalho.

Enfim, a todos que direta ou indiretamente contribuíram para a realização desse projeto, os meus sinceros agradecimentos.

# ÍNDICE ANALÍTICO

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABELAS

# ÍNDICE DE EQUAÇÕES

# 1 INTRODUÇÃO

*"Digno de admiração é aquele que, tendo tropeçado ao dar o primeiro passo, levanta-se e segue em frente".*

*(Louis XIV)*

*"He that will not apply new remedies must expect new evils; for time is the greatest innovator".*

*(Sir Francis Bacon)*

Há muitas situações em que se precisa proteger, resgatar, buscar, detectar, supervisionar ou rastrear algo ou alguém. Em geral, essas situações envolvem algum tipo de patrulha. Entretanto, como podemos definir o conceito de patrulha?

Patrulhar significa literalmente "o ato de andar ou navegar em uma área, a intervalos regulares, a fim de protegê-la ou supervisioná-la" [1]. Podemos destacar dois tipos básicos de patrulha: detecção e supervisão. No problema da detecção, o objetivo é detectar algum alvo (objeto ou pessoa) que esteja localizado no ambiente. Já no problema da supervisão, o objetivo da patrulha é fazer a vigilância de uma determinada área onde possam ocorrer alguns eventos como, por exemplo, crimes numa cidade.

Nessa dissertação, consideramos o problema da patrulha como uma tarefa multiagente realizada por um grupo de agentes inteligentes.

O grupo de pesquisa de Agentes Inteligentes e Jogos do CIn-UFPE [21] vem aprofundando pesquisas na área de sistemas multiagente já há alguns anos [28][31]. Dentre os problemas de pesquisa investigados, tem-se abordado vários aspectos do problema da patrulha, o qual começou a ser investigado durante a concepção do jogo Canyon [32]. O Canyon é ambientado em um planeta hostil, que é constituído de desfiladeiros que formam um grande labirinto rochoso. Combates de espaçonaves e missões de resgates de reféns ocorrem nesse cenário. Dessa forma, é importante que o componente de Inteligência Artificial do jogo seja capaz de coordenar um grupo de unidades para patrulhar o terreno a fim de detectar a presença de inimigos e conseguir encontrar recursos e reféns.

Além do nosso jogo particular, realizar patrulha de forma eficiente pode ser útil em diversos tipos de jogos. Casos típicos são jogos de estratégia (tal como StarCraft [53], Civilization III [10]) e RPG[1] (tal como Diablo [16] e Ultima Online [59]) onde a patrulha serve para detectar personagens móveis e novos prédios inimigos, proteger cidades e recursos,

[1] Role Playing Games

etc. A patrulha multiagente também pode ser aplicada em jogos de combate (tais como Panzer Commander [41] e StarWars Rogue Squadron [54]), particularmente em FPS[2] (tais como Unreal Tournament [60] e Counter-Strike [14]), para a detecção de inimigos e descoberta de itens. Da mesma forma, jogos de *arcade* (eg. PacMan) e simuladores podem utilizar algoritmos de patrulha. Mesmo jogos de esporte coletivos (FIFA Soccer [19] e NBA Inside Drive 2000 [39], por exemplo) talvez possam tomar proveito de algum tipo de patrulha para posicionamento de personagens.

Além de simuladores e jogos de computadores, a realização de forma eficiente da tarefa de patrulhamento pode ser útil para vários outros domínios de aplicação onde vigilância, inspeção ou controle distribuídos são necessários. Por exemplo, agentes de patrulha podem:

- ser utilizados por administradores de rede na supervisão de falhas ou de uma situação específica numa intranet [4];
- detectar modificações ou novas páginas na *web* a serem indexadas por engenhos de busca [12];
- identificar e resgatar objetos ou pessoas em situações perigosas [48];
- encontrar possíveis ofertas especiais nos mercados eletrônicos distribuídos na Internet;
- atuar como agentes de saúde de maneira a percorrer uma área para identificar e tratar possíveis focos de doenças como, por exemplo, a dengue;

Apesar de sua relevância, a tarefa de patrulhamento não tem sido estudada em profundidade. Alguns estudos relacionados com o tema foram encontrados como o mapeamento de redes [37], o problema do bar *El Farol* [6], sistemas de robótica baseados em comportamento (*behavior-based)* [5][7], *steering behaviors* [47], TSP [22] e o *Minimum Latency Problem* [9]. No entanto, estes trabalhos não são voltados para resolver o problema da patrulha.

Existem também pesquisas específicas na tarefa de patrulhamento [23], [43] e [44], mas estas não avaliam as estratégias de coordenação possíveis, arquiteturas de agentes ou organizações de sociedades de agentes. Existem poucos trabalhos que levam em consideração a eficácia do patrulhamento. Os que tratam do problema de supervisionar uma área para checar se ocorre algum evento [58] são muito incipientes. Já os que tratam especificamente do

[2] First-person shooters

problema de detecção de alvos [27] apesar de serem mais completos, como poderá ser visto mais adiante, não serão o foco desse trabalho.

## 1.1 OBJETIVOS

Em trabalhos anteriores [3][31], nós apresentamos uma discussão inicial e pioneira no problema da patrulha, assim como uma avaliação empírica de possíveis soluções. A fim de realizar esse estudo, propusemos algumas arquiteturas de sistemas multiagentes, critérios de avaliação e cenários de experimentação.

Este trabalho de mestrado é uma continuação do que já foi feito em nossos trabalhos anteriores. Como primeiro objetivo, refletimos sobre o problema da patrulha de uma forma mais sistemática. Como fruto dessa reflexão, foi proposta uma tipologia através da qual os diferentes problemas de patrulha podem ser classificados de acordo com a sua natureza como um problema de detecção ou de supervisão. Foram também pensadas as possíveis peculiaridades do ambiente a ser patrulhado e as características e restrições da sociedade multiagente de patrulha, o que deu origem a duas outras tipologias: do ambiente e da solução.

Além disso, tivemos como objetivo propor arquiteturas mais avançadas para resolver o problema da supervisão de forma mais eficiente, as quais utilizam técnicas heurísticas mais avançadas de tomada de decisão e *path-finding*.

Nosso último objetivo consistiu no enriquecimento da representação do ambiente a ser patrulhado. Para isso, consideramos grafos com pesos. A fim de testar as novas arquiteturas, foi necessário criar novos ambientes de experimentação, já que apenas grafos sem pesos haviam sido utilizados. Esses novos ambientes constituem um *benchmark* original.

## 1.2 ESTRUTURA DA DISSERTAÇÃO

O restante da dissertação está estruturado da seguinte maneira. O próximo capítulo descreve as tipologias propostas, mostrando como podemos classificar os diversos problemas e aplicações de patrulha.

O capítulo 3 apresenta uma visão geral do estado da arte da pesquisa em patrulha. Nele também descreveremos o caso particular de patrulha já abordado em nossos trabalhos anteriores, bem como as métricas e arquiteturas utilizadas.

As arquiteturas propostas nesse trabalho são detalhadas no capítulo 4.

O capítulo 5 discute a metodologia adotada, que consiste nos grafos utilizados nas simulações, nas mudanças realizadas no simulador e detalhes do método experimental.

Já no capítulo 6, são expostos os resultados e as discussões dos experimentos realizados.

Por último, o capítulo 7 apresenta nossas conclusões e sugestões para trabalhos futuros.

# 2 A TAREFA DE PATRULHAMENTO

*"If those in charge of our society - politicians, corporate executives, and owners of press and television - can dominate our ideas, they will be secure in their power. They will not need soldiers patrolling the streets. We will control ourselves".*

*(Howard Zinn)*

Várias aplicações podem ser associadas a diferentes problemas de patrulha, cada um com suas peculiaridades e objetivos. Por exemplo, patrulhar um ambiente com o objetivo de detectar um inimigo móvel é diferente de vigiar constantemente áreas de uma cidade. Enquanto uma boa estratégia de detecção de inimigos não necessariamente acarretará visitas constantes a todos os locais do ambiente, uma boa estratégia de vigilância intrinsecamente fará essas visitas.

Torna-se necessário, então, classificar os diversos problemas de patrulhamento através de uma tipologia. Uma classificação inicial dividiria os problemas de patrulha em duas grandes classes de acordo com a natureza da tarefa: detecção e supervisão. De forma ortogonal, podemos classificar as aplicações de patrulha em relação às características do ambiente a ser patrulhado. Por último, pode-se também refletir sobre uma tipologia da solução, ou seja, que características ou restrições apresentam os agentes que são utilizados para resolver o problema. Tais tipologias são assuntos abordados nesse capítulo e constituem a primeira contribuição dessa dissertação.

## 2.1 DETECÇÃO

### 2.1.1 OBJETIVO

O problema da detecção parte do pressuposto que no ambiente existe algum alvo que precisa ser detectado. A tarefa de patrulhamento consiste em, dados ***N*** agentes, detectar o alvo no menor tempo possível. Na literatura [42], esses problemas se chamam *pursuit-evasion*, no qual os agentes patrulhadores são chamados de *pursuers* e os alvos, de *evaders*.

Exemplos típicos de detecção são encontrar uma criança perdida num centro comercial; resgatar pessoas em situação de perigo [48]; localizar objetos tais como terroristas, guardas, reféns ou explosivos [40]; impedir a disseminação de um dado *worm* em redes de computadores [52]; ou detectar um invasor de uma área proibida [27].

### 2.1.2 VARIAÇÕES

Quanto à capacidade de locomoção, o alvo pode ser fixo ou móvel. No primeiro caso, o alvo sempre está localizado em um determinado local do ambiente, onde ficará durante todo o processo de detecção. Caso o alvo seja móvel, ele poderá se locomover pelo ambiente, enquanto a detecção estiver sendo feita.

Em determinadas situações é possível fazer um modelo do alvo a ser detectado. No caso de uma criança perdida num centro comercial, talvez o pai dela saiba que ela gosta muito de brinquedos. Logo, pode ser feito um modelo em que a probabilidade alta da criança estar localizada em uma loja de brinquedos ou em um fliperama seja utilizada para tentar detectá-la mais rapidamente.

Além disso, quando o alvo é móvel, o mesmo pode percorrer o ambiente de forma completamente previsível ou exibir um comportamento imprevisível. A imprevisibilidade pode ocorrer devido a um movimento aleatório ou devido a um comportamento propositalmente inteligente de forma a dificultar a detecção. Situações em que um comportamento perspicaz poderá ser observado nos alvos são invasões de ambientes proibidos, pois o alvo invasor tentará fazer o possível para despistar os detectores e conseguir fugir.

Na presença de alvos móveis inteligentes, pode ser que a detecção nunca tenha sucesso, pois a quantidade de $N$ agentes talvez não seja suficiente para detectá-lo. Sendo assim, em primeiro lugar, pode ser interessante determinar a número mínimo de agentes necessário para se detectar o alvo. Em geral, para resolver esse problema, assume-se um modelo de pior caso para o alvo, no qual ele tem a capacidade de se locomover pelo ambiente usando uma velocidade ilimitada. Infelizmente, determinar essa quantidade mínima é, em geral, um problema NP-completo [27][34].

Finalmente, o problema da detecção pode variar também de acordo com a quantidade de alvos.

### 2.1.3 MÉTRICAS

A princípio a métrica mais importante que poderia ser usada para avaliar o desempenho de N agentes detectores seria o tempo gasto para detectar o(s) alvo(s). Entretanto, dependendo da situação, outras métricas podem ser utilizadas.

Quando o alvo é fixo e não existe um modelo ajudando a determinação de sua posição no ambiente, no pior caso, os agentes terão que visitar todas as possíveis localizações do

ambiente a fim de detectá-lo. Nesse caso, agentes que tentam minimizar o tempo de exploração do ambiente podem apresentar um bom desempenho.

Já nas situações em que o alvo tenta enganar os agentes detectores a fim de não ser encontrado, se os agentes detectores apresentarem um comportamento previsível, o alvo "inimigo" poderia lançar mão de um comportamento de movimentação através do qual consiga nunca ser encontrado. Nessas situações, uma boa estratégia seria adotar uma política de movimentação não-previsível, em que a forma como os agentes detectores se movimentam não apresenta padrões simples e repetitivos de movimentos, tal como andar em ciclos.

Métricas relativas ao custo da solução também podem ser adotadas. Por exemplo, a memória e o tempo de processamento utilizados pelo mecanismo de raciocínio dos agentes detectores.

## 2.2 SUPERVISÃO

### 2.2.1 OBJETIVO

No problema da supervisão, dado um ambiente, a tarefa de patrulhamento consiste em visitar continuamente determinados pontos estratégicos onde possam ocorrer alguns eventos. A priori não se tem certeza de quando e onde vão ocorrer esses eventos.

Esses eventos podem ser, por exemplo, surgimento de ofertas especiais nos mercados eletrônicos; ocorrências de crimes numa cidade; falhas em redes de computadores [4]; surgimento de focos de doenças numa cidade; ou modificações ou surgimento de páginas web [12].

### 2.2.2 VARIAÇÕES

Podem existir dois tipos básicos de eventos: eventos temporários ou eventos perduráveis. Eventos temporários são aqueles que só podem ser observados em um determinado intervalo de tempo, tais como ofertas em mercados eletrônicos ou crimes numa cidade. Caso não estejam localizados no local de ocorrência do evento no momento em que ele aconteceu, os agentes a princípio não têm como tomar conhecimento de sua existência. Por outro lado, eventos perduráveis podem ser observados a qualquer momento a partir do instante em que ele ocorreu. Exemplos típicos são falhas em redes de computadores, existência de focos de doenças, incêndios e modificações de páginas web.

**Comentário:** Permanente? , perene? , duradouro? , perdurável? , persistente?

Mais uma vez, às vezes o conhecimento do domínio pode nos fornecer um modelo dos eventos que ocorrem. Nesse caso, podem-se adotar estratégias mais precisas que facilitem a

supervisão do ambiente. Por exemplo, no caso de modificação de páginas web, informações estatísticas a respeito de eventos passados (determinadas páginas que foram modificadas) podem ser utilizadas para dar uma prioridade maior a determinados locais (servidores web) a serem patrulhados [11]. Da mesma forma, determinadas regiões de uma cidade podem ter uma maior probabilidade de ocorrência de crimes ou de surgimento de focos de doenças que outras regiões.

### 2.2.3 MÉTRICAS

Uma boa estratégia de supervisão a princípio minimizaria o intervalo de tempo entre visitas a um mesmo lugar para todos os lugares do ambiente. Essa afirmação nos leva ao conceito de ociosidade.

Ociosidade (*idleness*) instantânea de um ponto ***P*** do ambiente no tempo ***t***, $\boldsymbol{Idl_t(P)}$, é o tempo que ele passou sem ser visitado, ou seja, o tempo decorrido entre o instante da última visita de algum agente a ***P*** e o instante ***t***. A ociosidade instantânea média do ambiente ***S*** no tempo ***t***, $\boldsymbol{AvgIdl_t(S)}$, é a integral em todo o ambiente da ociosidade instantânea dos pontos. Ela é dada pela Equação 1:

$$AvgIdl_t(S) = \int_S Idl_t \, \mathrm{d}S$$

**Equação 1 - Definição matemática da ociosidade instantânea média do ambiente**

Se tomarmos a integral da ociosidade instantânea média do ambiente em um determinado período de tempo $\boldsymbol{\Pi}$, temos a ociosidade média do ambiente para esse período, $\boldsymbol{AvgIdl_\Pi(S)}$. Ela pode ser definida pela Equação 2:

$$AvgIdl_\Pi(S) = \int_\Pi AvgIdl_t(S) \, \mathrm{d}t$$

**Equação 2 - Definição matemática da ociosidade média do ambiente**

Em determinados casos, pode ser importante que a ociosidade instantânea dos pontos do ambiente não alcance um valor muito alto. Nessas situações é conveniente medir a ociosidade máxima e utilizá-la como critério de avaliação. Definimos a ociosidade instantânea máxima do ambiente ***S*** no tempo ***t***, $\boldsymbol{MaxIdl_t(S)}$, pela Equação 3:

$$MaxIdl_t(S) = max_{P \in S} \, Idl_t(P)$$

**Equação 3 - Definição matemática da ociosidade instantânea máxima do ambiente**

Similarmente, definimos a ociosidade máxima do ambiente ***S*** em um determinado período de tempo $\boldsymbol{\Pi}$, $\boldsymbol{MaxIdl_\Pi(S)}$, pela Equação 4:

$$MaxIdl_{\Pi}(S) = max_{t \in \Pi} MaxIdl_t(S)$$

**Equação 4 - Definição matemática da ociosidade máxima do ambiente**

Assim como definimos na Seção 2.1.3, métricas relativas ao custo da solução também podem ser adotadas.

Nas seções 2.1 e 2.2, vimos como os diferentes problemas de patrulha podem ser agrupados numa tipologia de acordo com o objetivo da patrulha. Analisamos algumas possíveis variações para as duas principais classes de problemas, bem como algumas métricas que podem ser utilizadas na avaliação de soluções. A Figura 1 mostra um resumo dessa tipologia exposta:

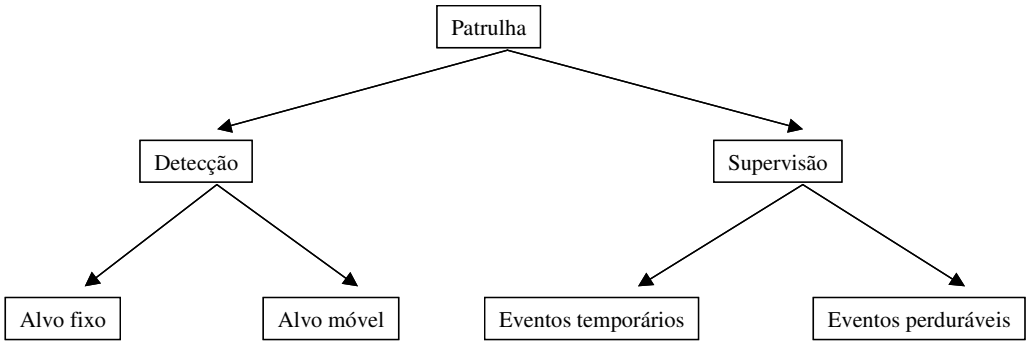


**Figura 1 - Taxonomia de problemas de patrulha**

Como veremos mais adiante, essa dissertação abordará apenas o problema da supervisão de eventos perduráveis.

## 2.3 Características do Ambiente a Ser Patrulhado

Nessa seção iremos apresentar algumas características do ambiente a ser patrulhado e o que pode ser representado dele no sistema.

### 2.3.1 Granularidade do ambiente

O ambiente a ser patrulhado pode ser discreto ou contínuo. Se existe um número limitado de percepções e ações possíveis no[1] ambiente, dizemos que ele é discreto. Caso contrário, classificamos o ambiente como sendo contínuo. Podemos citar como exemplos de problemas de patrulha que podem ser modelados sem grandes perdas em ambientes discretos: supervisão de falhas em redes de computadores, supervisão de modificações de páginas web, detecção de *worms* em redes de computadores. Em todos esses casos, existe uma quantidade

limitada de locais nos quais os agentes podem estar localizados. Além disso, as ações dos agentes se restringem a visitar um desses locais.

Em termos de área, o caso mais complexo é o de patrulhar um terreno contínuo. A localização dos objetos no ambiente é definida por coordenadas reais. Portanto, é impossível aplicar algoritmos de busca padrão de forma direta, uma vez que a quantidade de estados e ações é muito grande [49]. Exemplos típicos são patrulhas em ambientes como cidades, regiões de guerra, edificações, etc.

Nesses casos, muitas abordagens provenientes da robótica propõem maneiras de reduzir o espaço de busca contínuo em uma quantidade discreta de escolhas a se considerar. Uma das formas de obtermos essa redução é através da mudança de representação do terreno.

Uma das técnicas usadas para mudar a representação do terreno é a divisão do terreno original em linhas e colunas, criando diversas células denominadas *tiles* [55]. A movimentação dos objetos do mundo se dá entre *tiles* adjacentes. A Figura 2 (A) mostra o terreno original e a Figura 2 (B) mostra uma possível subdivisão em *tiles*.

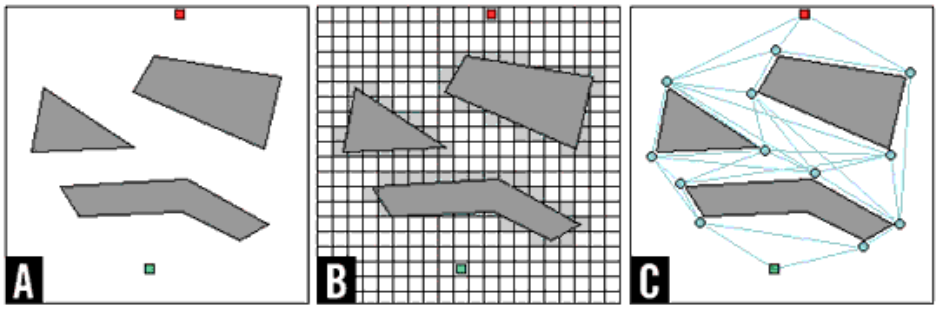


**Figura 2 - Mudança de Representação do Terreno**

Outra técnica bastante utilizada é a esqueletonização [49][55], a qual consiste em trocar o espaço de busca inicial por um grafo (*skeleton*) representando os possíveis caminhos, conforme pode ser visto na Figura 2 (C).

Existem várias maneiras de gerar o grafo a ser utilizado para representar o ambiente. São algumas delas: diagramas de Voronoi, grafos de visibilidade e *C-cells* [55]. Outra forma de gerar o grafo é manualmente.

Com tal abstração do ambiente criada pelo grafo, os diferentes casos de patrulha (discreto ou contínuo) se tornam semelhantes, pois estes utilizarão a mesma representação para o ambiente. De fato, uma grande vantagem da utilização de tal representação abstrata é que as diferentes soluções propostas podem ser aplicadas a vários tipos de problemas e ambientes, desde patrulha de redes de computadores a patrulhas em terrenos contínuos. Além

disso, a tarefa é sensivelmente simplificada nessa nova representação, já que o espaço de busca é reduzido.

### 2.3.2 Importância dada a Sub-regiões

Todo o ambiente pode ter a mesma prioridade ou podem ser dadas prioridades diferentes dependendo da sub-região a ser patrulhada. Prioridades são dadas a determinadas regiões principalmente quando se tem um modelo mais refinado do problema, através do qual se sabe que regiões são mais importantes de serem visitadas. Exemplos típicos já citados anteriormente são: detecção de crianças perdidas em centros comerciais, supervisão de diferentes regiões de uma cidade em busca de focos de doenças ou de ocorrência de crimes, supervisão de servidores web em busca de modificações de páginas, etc.

### 2.3.3 Dinamicidade da Estrutura

Num ambiente estático apenas as posições dos agentes no ambiente podem mudar ao longo do tempo. Todos os outros elementos pertencentes ao ambiente não sofrem nenhuma alteração. Já em ambientes dinâmicos, os pontos a serem patrulhados podem surgir ou sumir ao longo do tempo, pode haver obstáculos móveis ou as prioridades das regiões também podem mudar ao longo do tempo. Numa representação do ambiente usando grafos, isso se reflete na mudança dos nós e das arestas ao longo do tempo.

### 2.3.4 Peso das Arestas

Quando se utilizam grafos para representar o ambiente, as arestas podem ter tamanhos diferentes associados (pesos) correspondendo, por exemplo, à distância real entre os nós ou o tempo gasto para se locomover entre dois nós adjacentes. Caso não haja peso associado às arestas (todas as arestas com peso unitário), um agente com uma velocidade de uma unidade de distância por unidade de tempo conseguiria sair de um nó e chegar em outro nó adjacente em exatamente uma unidade de tempo, já que o peso de todas as arestas seria igual a uma unidade de distância. Já se considerarmos pesos, uma aresta conectando dois nós separados por mais que uma unidade de distância vai requerer do agente mais de uma unidade de tempo para ser percorrida.

### 2.3.5 Representação do Conhecimento Do Ambiente

Durante a execução da tarefa de patrulha, os agentes precisarão de diversas informações as quais representam seu modelo do mundo para realizar um patrulhamento da

forma mais eficiente possível. Tal conhecimento pode ser armazenado em estruturas de dados internas aos agentes ou no próprio ambiente.

Caso o ambiente seja representado através de um grafo, o que pode ou não ser armazenado a respeito dos nós do grafo? Por exemplo, podem-se registrar os últimos n agentes que visitaram cada nó do grafo e quando ocorreram as últimas n visitas. Caso o problema seja a supervisão, diversas informações a respeito da ociosidade podem ser armazenadas, por exemplo, como a ociosidade variou em cada nó do grafo nos últimos ***t*** segundos. Os agentes armazenam essas informações dos diversos nós? Se sim, quais delas?

Resumimos o que foi descrito nessa seção através da Tabela 1. Convém notar que, segundo definido em [49], o ambiente do problema da patrulha sempre será não-episódico e não-determinístico.

| Propriedade | Valores Possíveis |
|---|---|
| Granularidade do ambiente | Contínuo, grafo com pesos, grafos sem pesos |
| Prioridade para as regiões | Inexistente, fixa, variável |
| Dinamicidade da estrutura | Estrutura dinâmica, estrutura estática |

**Tabela 1 - Características do ambiente**

## 2.4 AGENTES PATRULHADORES

Nessa seção, iremos apresentar algumas restrições ou características dos agentes os quais resolverão o problema da patrulha, permitindo um outro nível de classificação de aplicações em que a patrulha aparece.

### 2.4.1 CAMPO DE VISÃO

Uma característica bastante importante de qualquer agente inteligente é a sua percepção do ambiente. Em se tratando do problema da patrulha, consideramos que a percepção de cada agente consiste num campo de visão de raio ***R***. Um valor de ***R=0***, o agente só percebe o lugar onde está situado. Já quando ***R=∞***, a percepção é global, ou seja, o agente consegue enxergar todo o ambiente (ambiente acessível). Valores intermediários de ***R*** permitem ao agente visualizar apenas determinadas regiões do ambiente.

Em geral, ambientes acessíveis são mais simples de serem tratados. Entretanto, nem sempre a capacidade de percepção dos agentes dará a eles acesso ao estado completo do ambiente.

### 2.4.2 Tipos de Comunicação

Agentes se comunicam para melhor atingir seus objetivos e/ou objetivos da sociedade da qual fazem parte [24]. No caso do problema de patrulha, várias informações podem ser compartilhadas como, por exemplo, os diversos locais que cada agente visitou e quando ocorreram tais visitas. Logo, de acordo com as possibilidades de comunicação, um dado agente pode ou não saber o que os outros agentes estão fazendo.

No problema da supervisão em especial, uma das informações que podem ser trocadas entre os agentes é a ociosidade do ambiente. Nesse contexto, denominamos a ociosidade individual, quando um agente considera apenas suas próprias visitas, ou seja, não se tem conhecimento do que os outros agentes estão fazendo. Já a ociosidade é compartilhada, quando o agente leva em conta o movimento de todos os outros agentes.

Além disso, os agentes podem desejar saber o que os outros agentes pretendem fazer. Nesse caso, existiria troca de informações a fim de compartilhar as intenções dos agentes.

É importante ressaltar que as informações trocadas nem sempre são confiáveis. Alguns agentes podem passar informações erradas aos outros agentes. Tal fato pode acontecer por razão de mau funcionamento (erro de hardware) ou por serem traidores (por exemplo, soldados em guerra).

Embora a comunicação talvez possa permitir alcançar os objetivos mais facilmente, não necessariamente ela será utilizada. O fato de não haver comunicação pode ser por limitação imposta pelo ambiente, pela falta de recursos dados aos agentes ou por uma decisão de estratégia deliberada. Em uma guerra, mesmo o ambiente permitindo comunicação e os agentes possuindo recursos, utilizar comunicação pode não ser uma boa estratégia, pois o inimigo pode detectar a troca de mensagens a fim de se beneficiar. Situações semelhantes podem ser observadas em detecção de invasores de prédios ou perseguições policiais. Nesses casos, a ausência de comunicação não é determinada por restrições impostas pelo ambiente nem por falta de recursos, mas sim por uma escolha da estratégia de coordenação.

Quando existe capacidade de comunicação ela pode ser: via *flags*, *blackboard* ou por mensagens. Todas essas três formas de comunicação já haviam sido estudadas em nossos trabalhos anteriores.

Na comunicação via *flags* ou bandeiras, os agentes colocam bandeiras no ambiente contendo informações que julgam importantes para os demais agentes. Uma vez que os outros agentes passam pelo local onde a bandeira está localizada, ele pode adquirir a informação nela

contida. Essa é uma forma de comunicação indireta, pois não há uma especificação explícita dos destinatários das informações contidas nas bandeiras [17][18].

O conceito fundamental de comunicação baseada em *blackboard* é o uso estruturas de dados compartilhadas (por exemplo, em um centro de comando e de controle) que são acessíveis a todos os agentes. O acesso pode ser tanto para escrever mensagens na estrutura de dados compartilhada ou para ler mensagens dela. Isso também é uma forma de comunicação indireta na qual qualquer agente interessado em determinados tipos de mensagens pode ler as seções relevantes do blackboard [18].

Na comunicação via mensagens, existe a possibilidade de um canal direto entre os agentes, os quais podem se comunicar através da troca de mensagens. Nesse caso, tem que se levar em conta a circunstância em que dois agentes podem se comunicar. Uma possibilidade é permitir a comunicação apenas quando eles se encontram em algum ponto do ambiente. Outra possibilidade é permiti-la a qualquer momento em que os agentes desejarem.

Outra questão que precisa ser levada em consideração é a rede de conhecimento dos agentes. Ou seja, dado um agente, com que outros agentes ele pode se comunicar.

Por último, é necessário estabelecer um protocolo de comunicação a fim de determinar o tipo de mensagem trocada entre os agentes e o formato dessas mensagens. Para tal, pode-se utilizar linguagens de comunicação de agentes, tais como KQML [26] e FIPA-ACL [20], ambas baseadas na teoria dos atos de fala [50].

### 2.4.3 Organização da Sociedade

Diferentes arquiteturas multiagentes podem ser propostas para resolver o problema da patrulha. Cada arquitetura é caracterizada pelo mecanismo de raciocínio e tomada de decisão utilizados na escolha das ações e da movimentação do agente. Numa dada sociedade, todos os agentes podem ser do mesmo tipo/arquitetura ou não.

Vários podem ser os motivos que levem a uma sociedade heterogênea no problema da patrulha. A implementação de agentes mais complexos pode requerer um custo financeiro, de tempo de processamento ou de memória utilizada elevado e proibitivo. Dessa forma, na ausência de tais recursos, apenas alguns agentes da sociedade seriam implementados de forma complexa e os demais de maneira mais simples. Por outro lado, talvez sociedades heterogêneas possam apresentar um melhor desempenho. Caso isso fosse comprovado, mesmo dispondo de recursos, seria mais vantajoso lançar mão de uma sociedade heterogênea.

Um caso particular de sociedade heterogênea é quando existe um coordenador central. Nesses casos, a comunicação pode ser permitida apenas com o coordenador central, o qual

constitui a rede de conhecimento de cada agente. O papel do coordenador seria realizar a divisão de tarefas entre os diversos agentes de forma a minimizar conflitos. Em outras palavras, o coordenador indicaria para cada agente que locais do ambiente ele deve visitar a fim de evitar que agentes diferentes visitem os mesmos lugares. O coordenador pode estar localizado no ambiente ou fora dele. Caso esteja localizado no ambiente, ele também pode assumir o papel de patrulhador. Caso esteja localizado fora do ambiente, ele seria uma entidade presente em algo como uma central de comando ou um satélite.

Uma outra possibilidade seria a existência de vários coordenadores, cada um tomando conta de uma determinada região do ambiente.

### 2.4.4 Dinamicidade da Sociedade

A sociedade de agentes patrulheiros pode ser aberta ou fechada. Numa sociedade aberta (ou dinâmica), a quantidade de agentes pode variar ao longo do tempo. Nesse caso, alguns agentes podem deixar de funcionar ou "morrer" durante o processo de patrulhamento e outros podem surgir durante a execução da tarefa. Já numa sociedade fechada (ou estática), não há nem criação nem destruição de agentes patrulheiros.

A Tabela 2 resume as principais características dos agentes descritas nessa seção.

| Propriedade | Valores Possíveis |
|---|---|
| Campo de visão | Raio ***R*** |
| Tipo de Comunicação | Inexistente, via *flags*, *blackboard* ou mensagens |
| Confiabilidade da comunicação | Confiável, não-confiável |
| Organização da sociedade | Homogênea, heterogênea |
| Existência de coordenador | Sim, não |
| Dinamicidade da sociedade | Sociedades abertas, sociedades fechadas |

**Tabela 2 - Características dos agentes**

## 2.5 Conclusão

O estudo do problema da patrulha é relevante, haja vista que não são poucas as áreas em que o problema pode ser encontrado.

Como o problema da patrulha pode ter inúmeras definições e várias características diferentes, mostramos, nesse capítulo, como podemos agrupar as diversas naturezas de problemas em classes de uma tipologia. Além disso, apresentamos uma tipologia do ambiente a ser patrulhado e da solução empregada. A vantagem de descrever uma tipologia é que

problemas de uma mesma classe podem ser resolvidos seguindo a mesma estratégia de patrulhamento.

No próximo capítulo, veremos que subconjunto dessas tipologias já foi tratado na literatura, fazendo uma crítica às abordagens tomadas.

# 3 ESTADO DA ARTE

Como mencionado na introdução, não obstante sua importância, a tarefa de patrulhamento em geral não havia sido estudada profundamente conforme se observa na literatura. Dessa forma, não foi encontrado um estudo que tratasse a patrulha de maneira adequada, apresentando uma análise profunda e sistemática. O primeiro estudo sistemático realizado no tema consiste numa dissertação de mestrado realizada por Aydano Machado no CIn-UFPE [31].

Em primeiro lugar, esse capítulo mostra alguns trabalhos que podem ser relacionados, de certa forma, com o problema em questão. Em seguida, será abordado o trabalho de Aydano Machado. Finalmente, será feita uma crítica ao trabalho desenvolvido, a qual serviu de motivação na realização dessa nova dissertação.

## 3.1 TRABALHOS CORRELATOS

A literatura tem trabalhos interessantes a respeito de problemas relacionados à patrulha. O problema do mapeamento de rede [37] é um deles e consiste em, dado um grafo representando uma rede de computadores, visitar todos os seus nós.

No problema do *bar El Farol* [6], uma população de agentes tem que decidir independentemente se vai a um bar numa certa noite. Todos os agentes desejam ir ao bar a não ser que ele esteja lotado (isto é, quando mais que 60% dos agentes vão). A única informação disponível é a lotação das últimas semanas. Qualquer modelo do problema que é compartilhado pelos agentes é insatisfatório. Se a maioria dos agentes prediz que o bar não estará muito lotado então todos eles irão e ele ficará lotado e, vice-versa, se todos os agentes predizem que o bar estará lotado, nenhum deles irá e ele ficará vazio. Essa situação também é observada no problema do patrulhamento, pois se todos os agentes decidirem ir a um determinado local, os demais ficarão não-patrulhados.

Sistemas de robótica baseados em comportamento (*behavior-based)* [5], [7] e *steering behaviors* [47] provêm aos agentes a habilidade de navegação num mundo de maneira que pareça real (*life-like*). A combinação de comportamentos simples pode ser usada para atingir objetivos de mais alto nível, tais como seguir um líder, sair de um lugar a outro evitando obstáculos, se unir a um grupo de agentes. Entretanto, o problema da patrulha não é abordado nesses trabalhos.

Problemas da área da Teoria dos Grafos tal como o problema do caixeiro viajante ou TSP [22] e o *Minimum Latency Problem* [9] também possuem como motivação a movimentação em um ambiente (nesse caso, o ambiente é um grafo). No TSP, dado um número finito de "cidades" e o custo de viagem entre cada par delas, deseja-se encontrar o modo mais barato de visitar todas elas e voltar para a cidade inicial. Entretanto, o objetivo é essencialmente diferente da patrulha.

No problema da latência mínima, dados um conjunto finito de pontos e o tempo de viagem entre cada par deles, desejamos encontrar um caminho que passe por eles e que minimiza a soma dos tempos de chegadas nos pontos. Imaginemos, por exemplo, que um agente se move por uma rede buscando uma informação (alvo fixo) que é igualmente provável de estar localizada em qualquer um dos pontos da rede. Então, o problema de achar uma rota que minimize o tempo esperado de busca é exatamente o problema da latência mínima. Dessa forma, o problema da detecção de alvos fixos quando não existe um modelo do alvo e o problema da latência mínima são equivalentes.

Os trabalhos da área de jogos dedicados especificamente a tarefas de patrulha [23], [43], [44] não apresentam uma avaliação sistemática das possíveis estratégias de coordenação, modelos de agentes, organizações de sociedades de agentes, restrições de comunicação, e assim por diante. Em geral, o problema abordado consiste apenas em se mover para uma série de posições especificadas em uma determinada ordem, como pode ser visto na Figura 3. A motivação nesse caso é que ter unidades se movendo na tela ao invés de paradas esperando alguma ordem faz com que o mundo se pareça muito mais ativo e dinâmico. Além disso, existe muito menos chance de as unidades serem capturadas e mais chance de elas detectarem a presença de intrusos.

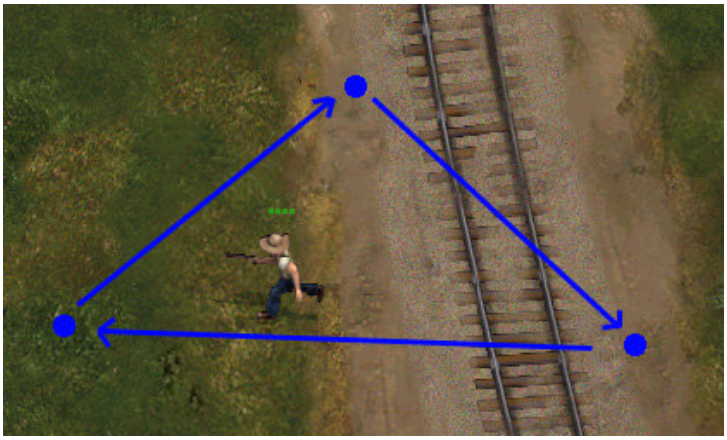

**Figura 3 - Exemplo de Patrulhamento**

O trabalho de Pottinger, em especial, também lida com movimentação de grupos de agentes, o qual envolve questões como detecção de colisão entre agentes e previsão de onde as unidades estarão localizadas no futuro. Entretanto, esses trabalhos são focados em movimento, não exatamente em um comportamento emergente que poderia melhorar a eficácia do patrulhamento.

Existem poucos trabalhos que levam em consideração a eficácia do patrulhamento. O que trata do problema da supervisão [58] é muito incipiente. Os ambientes de patrulhamento testados nesse trabalho são demasiadamente pequenos e limitados (Figura 4). Foram testados apenas 3 cenários: 2 agentes num grafo completo de 6 nós, 3 agentes no mesmo grafo completo de 6 nós e 3 agentes num grafo completo de 9 nós. Foram utilizadas duas heurísticas para a escolha do próximo nó a ser visitado. Na primeira, os agentes escolhem como o próximo nó a ser visitado aquele que está localizado mais perto do agente. A outra estratégia utilizada foi escolher o ponto mais longe dos outros robôs. Entretanto, essas duas heurísticas tiveram péssimos resultados para o critério ociosidade definido na seção 2.2.3.

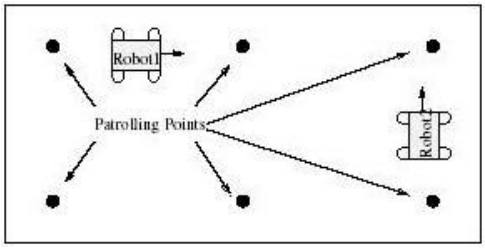


**Figura 4 - Patrulhamento no trabalho de Tangachit et al.**

Já os que tratam especificamente do problema de detecção de alvos [27] são mais completos, inclusive estabelecendo limites na quantidade de agentes necessária para detectar o alvo desejado, conforme visto na seção 2.1.2. Contudo, como nosso foco é o problema da supervisão, não entraremos em maiores detalhes nesses trabalhos.

De fato, o primeiro estudo sistemático realizado no tema da supervisão consiste no trabalho de Aydano Machado [31]. A próxima seção trata dos pontos principais desse estudo, em especial, o caso particular de patrulha abordado, as métricas utilizadas e as arquiteturas implementadas e testadas.

## 3.2 TRABALHO DE AYDANO MACHADO

A primeira preocupação do trabalho de Aydano Machado foi definir o problema que seria estudado. Segundo sua definição, dado um grafo, a tarefa de patrulha consiste em visitar continuamente todos os nós do grafo de maneira a minimizar o intervalo de tempo entre duas

visitas a um mesmo nó. Em outras palavras, a natureza do problema abordado foi a supervisão.

Em relação às características do ambiente a ser patrulhado, como dito acima, foi feita uma mudança de representação do terreno, transformando-o em um grafo. Além disso, a mesma importância foi dada a todas as regiões do ambiente, ou seja, aos nós do grafo. Foi determinado também que o grafo seria estático, sem mudanças nos nós nem nas arestas ao longo do tempo. Finalmente, por questões de simplicidade, apenas grafos sem pesos foram tratados.

### 3.2.1 Critérios de Avaliação Utilizados

Após definir o problema, o próximo passo tomado foi decidir como seria feita a avaliação das diferentes arquiteturas de agentes patrulhadores. Para tal, foram utilizados três critérios de avaliação, já descritos anteriormente nas Seções 2.1.3 e 2.2.3: *ociosidade média*, *ociosidade máxima* e *tempo de exploração*.

Esses critérios foram medidos através de simulações. A unidade de tempo utilizada em uma simulação foi denominada *ciclo*. Dessa forma, ao invés de representar o tempo de forma contínua, cada simulação compreendeu um período de tempo dividido discretamente em ciclos. Considerando a discretização do tempo e do ambiente, podemos simplificar os critérios de avaliação como se segue.

Ociosidade (*idleness*) instantânea de um nó ***i*** do grafo no ciclo ***t***, $\boldsymbol{Idl_t(i)}$, é o tempo em que ele passou sem ser visitado, ou seja, a quantidade de ciclos decorrida entre o ciclo da última visita de algum agente ao nó ***i*** e o ciclo ***t***. A ociosidade instantânea média do grafo ***G*** no tempo ***t***, $\boldsymbol{AvgIdl_t(G)}$, é a média [36] da ociosidade instantânea dos nós, ou seja a soma das ociosidades instantâneas dos nós dividida pela quantidade de nós. Matematicamente, sendo ***n*** a quantidade de nós do grafo, temos:

$$AvgIdl_t(G) = \frac{\sum_{i=1}^{n} Idl_t(i)}{n}$$

**Equação 5 - Definição matemática da ociosidade instantânea média do grafo**

Finalmente, a ociosidade média do grafo, ou simplesmente *ociosidade média*, ***AvgIdl(G),*** é a média das ociosidades instantâneas do grafo medidas ao longo de uma simulação. Matematicamente, sendo ***c*** a quantidade de ciclos da simulação, temos:

$$AvgIdl\,(G) = \frac{\sum_{t=1}^{c} AvgIdl_t\,(G)}{c}$$

**Equação 6 - Definição matemática da ociosidade média do grafo**

Seja ***V*** o conjunto de vértices do grafo, definimos a ociosidade instantânea máxima do grafo ***S*** no tempo ***t***, ***MaxIdl$_t$(G)***, pela Equação 7:

$$MaxIdl_t\,(G) = max_{i \in V}\, Idl_t\,(i)$$

**Equação 7 - Definição matemática da ociosidade instantânea máxima do grafo**

Similarmente, sendo ***c*** a quantidade de ciclos da simulação, definimos a ociosidade máxima do grafo, ou simplesmente *ociosidade máxima*, ***MaxIdl(G)***, por:

$$MaxIdl\,(G) = max_{t \in \{1,\ldots,c\}}\, MaxIdl_t\,(G)$$

**Equação 8 - Definição matemática da ociosidade máxima do grafo**

O *tempo de exploração* consiste na quantidade de ciclos necessária para os agentes visitarem pelo menos uma vez todos os vértices do grafo. Ele pode ser dado matematicamente pela Equação 9:

$$t = tempoExploracão \Leftrightarrow (\forall i \in V \bullet (Idl_t\,(i) < t)) \wedge (\neg \exists t' \bullet (t' < t) \wedge (\forall i \in V \bullet (Idl_t\,(i) < t')))$$

**Equação 9 - Definição matemática do tempo de exploração**

Explicando um pouco a equação acima: um nó *i* já foi visitado antes de um ciclo *t* quando a ociosidade do nó *i* no ciclo *t* é menor que *t*. Supondo que o tempo de exploração seja o ciclo *t*, desejamos que todos os nós já tenham sido visitados antes desse ciclo *t*, ou seja, que todos os nós tenham ociosidade menor que *t* nesse ciclo. Entretanto, não basta isso. Temos que garantir que a primeira vez que isso acontece seja no ciclo *t*. Ou seja, não pode existir nenhum ciclo *t'* menor que *t* em que todos os nós tenham ociosidade menor que *t'*.

### 3.2.2 Arquiteturas de Sistemas Multiagentes Propostas

Nessa seção, iremos apresentar quais restrições ou características (já descritas na Seção 2.4) das arquiteturas investigadas foram consideradas. As arquiteturas são mostradas na Tabela 3:

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

**Tabela 3 - Resumo das principais características das arquiteturas propostas por Aydano Machado**

#### 3.2.2.1 Campo de Visão

Foram considerados agentes com campo de visão local (reativos) e agentes com campo de visão global (cognitivos). O campo de visão de agentes reativos tem a profundidade de um nó, ou seja, agentes reativos só percebem os nós adjacentes a ele. Como, por definição, agentes reativos não podem planejar um caminho para nós distantes, não se faz necessário que eles possuam um campo de visão maior. Agentes cognitivos, por sua vez, podem escolher qualquer nó do grafo como nó objetivo e usam técnicas de *path-finding* (Algoritmo de Floyd-Warshall [33], no caso) para alcançar esse nó objetivo. Na criação da estrutura de dados responsável pela obtenção de melhor caminho, o algoritmo de Floyd-Warshall é executado para determinar os menores caminhos entre todos os pares de nós do grafo. Todos esses caminhos são pré-computados e já ficam armazenados nessa estrutura de dados. Conseqüentemente, quando um agente solicita o menor caminho entre o nó em que ele se localiza e o nó objetivo, a resposta é dada em tempo constante, diferentemente se fosse usado um algoritmo de busca tradicional, como o A* [15][46].

#### 3.2.2.2 Capacidade de Comunicação

Foram considerados os três tipos de comunicação: através de *flags*, via *blackboard* e direta entre os agentes por mensagens. Entretanto, quando usada, a comunicação via mensagens foi permitida apenas com o coordenador central. Supôs-se que a comunicação entre os agentes era feita de forma confiável, sem a existência de traidores.

#### 3.2.2.3 Organização da Sociedade

Foram apenas consideradas arquiteturas homogêneas, onde todos os agentes são iguais. As exceções são as arquiteturas que possuem a figura de um coordenador, sendo esse o único tipo de agente diferente dos demais.

#### 3.2.2.4 Dinamicidade da Sociedade

Durante uma mesma simulação, a quantidade de agentes não variava ao longo do tempo. Ou seja, não foram abordadas sociedades abertas.

#### 3.2.2.5 Tomada de Decisão

A tomada de decisão consiste em determinar como o próximo nó será escolhido. Foram considerados dois critérios de escolha: escolher um nó de forma aleatória ou de maneira heurística baseado nas ociosidades dos nós. No segundo caso, um agente escolhe o nó com maior ociosidade.

Obviamente, dois aspectos já citados influenciam o processo de tomada de decisão. O primeiro é o campo de visão dos agentes, que pode ser local ou global como discutido anteriormente. O segundo é o fato que, de acordo com as possibilidades de comunicação, um dado agente pode ou não saber o que os outros agentes estão fazendo. Dessa forma, denominamos a ociosidade de um nó individual, quando um agente considera apenas suas próprias visitas; compartilhada, quando o agente leva em conta o movimento de todos os outros agentes; ou compartilhada e coordenada, quando ele utiliza um coordenador. Considerando essas variações do processo de tomada de decisão, os agentes farão a escolha do próximo nó de acordo com as estratégias mostradas na Tabela 4:

| Nome da Arquitetura | Escolha detalhada do próximo nó |
|---|---|
| *Random Reactive* | Aleatória local |
| *Conscientious Reactive* | Ociosidade individual local |
| *Reactive with Flags* | Ociosidade compartilhada local |
| *Conscientious Cognitive* | Ociosidade individual global |
| *Blackboard Cognitive* | Ociosidade compartilhada global |
| *Random Coordinated* | Aleatória compartilhada e coordenada global |
| *Cognitive Coordinated* | Ociosidade compartilhada e coordenada global |

**Tabela 4 - Estratégia de tomada de decisão detalhada usada pelas arquiteturas propostas por Aydano Machado, considerando o campo de visão e a comunicação entre os agentes**

É importante salientar que a metodologia utilizada por Aydano Machado também considerou outro parâmetro de coordenação que não está mostrado na Tabela 3: a capacidade de monitoramento. Enquanto um agente cognitivo estiver seguindo um determinado caminho para o seu nó objetivo, é bastante útil que seja constantemente observado se outro agente visitou esse nó objetivo. Isso é útil para que o agente possa escolher outro nó objetivo, já que algum outro agente já realizou o objetivo que ele estava buscando realizar. Apenas é possível adicionar tal característica quando existe a possibilidade do agente saber se seu nó objetivo foi visitado por outro agente durante sua caminhada, ou seja, quando existe alguma forma de comunicação entre os agentes.

## 3.3 CRÍTICA

Concebida de forma sistemática e metodológica, a dissertação de Aydano Machado teve a originalidade como uma forte característica, pois abordou um problema ainda não havia sido atacado, o da patrulha (supervisão) multiagente. Entretanto, como todo trabalho inicial em uma área, existem muitas direções que ainda podem ser exploradas.

Em primeiro lugar, a representação do ambiente utilizada é muito simples. Grafos com pesos podem ser usados para representar de forma mais adequada o ambiente, pois refletem melhor a realidade e têm uma maior capacidade de representação. Por exemplo, os pesos das arestas podem representar o comprimento das ruas de uma cidade, a distância entre dois pontos de um ambiente (região em guerra, edifício, ...), o tempo gasto para se locomover entre dois nós de uma rede de computadores, etc. Dessa forma, esse trabalho representará o ambiente através de grafos com pesos.

Além disso, os agentes usavam técnicas muito simples de *path-finding* e tomada de decisão. Por exemplo, ao invés de buscar menores caminhos, o mecanismo de *path-finding* poderia levar em conta a ociosidade instantânea dos nós localizados entre a localização corrente e o objetivo. Da mesma forma, no processo de tomada de decisão, poderia ser considerada uma heurística mais avançada que considerasse não apenas a ociosidade de um candidato a nó objetivo, mas também a distância entre o agente e esse nó candidato.

Por último, conforme visto nas seções 2.1.3 e 2.2.3, o tempo de exploração é um critério mais adequado ao problema da detecção, não devendo ser usado para medir o desempenho de arquiteturas destinadas ao problema da supervisão. Sendo assim, não será mais utilizado o tempo de exploração para avaliar o desempenho dos agentes.

No próximo capítulo, detalharemos a importância de utilização de grafos com pesos. Além disso, veremos que técnicas avançadas de *path-finding* e tomada de decisão foram propostas nesse trabalho.

# 4 DISTÂNCIA E OCIOSIDADE

*"A arte de ser, alternadamente, audacioso e prudente é a arte de ter êxito".*

*(Napoleão)*

O capítulo 2 mostrou a primeira contribuição desse trabalho: uma tipologia de problemas de patrulha. Nesse capítulo, trataremos dos avanços que fizemos em relação ao estado da arte por considerar distância e ociosidade na confecção dos agentes.

Primeiramente, mostraremos a importância em se utilizar grafos com pesos. Em seguida, abordaremos um mecanismo de *path-finding* mais refinado que utiliza a ociosidade dos nós intermediários no cálculo do menor caminho. Posteriormente, veremos como o processo de tomada de decisão pode ser melhorado ao levarmos em consideração a distância do agente aos nós candidatos. Por último, resumiremos as características dos novos agentes que utilizam tomada de decisão e *path-finding* heurísticos. É importante ressaltar que mais uma vez apenas o problema da supervisão foi estudado, deixando o problema da detecção como trabalho futuro.

## 4.1 CONSIDERANDO GRAFOS COM PESOS

Em [31], o ambiente era representado de forma muito simplificada através de grafos sem pesos. Contudo, grafos com pesos podem ser usados para representar de forma mais adequada o ambiente, pois refletem melhor a realidade e têm uma maior capacidade de representação.

Um grafo com peso $G = (V, E)$ é um em que cada aresta $(i, j) \in E$ tem associada a ela um peso $w_{ij}$. Para muitas aplicações, é útil se as arestas do grafo puderem ser rotuladas com pesos. Por exemplo, pode-se ter um grafo representando as possíveis rotas entre cidades, onde o peso é a distância ao longo de uma dada rota. Da mesma forma, os pesos das arestas podem representar o comprimento das ruas de uma cidade, a distância entre dois pontos de um ambiente (região em guerra, edifício, ...).

Em outras aplicações, o peso pode representar o tempo levado para percorrer uma aresta (e.g., tempo para ir de uma cidade a outra ou para se locomover entre dois nós de uma rede de computadores); o custo de atravessar a aresta (e.g., custo de enviar uma mensagem, combustível gasto numa estrada, preço de uma passagem) ou qualquer outro valor associado

(por exemplo, o conforto experimentado num determinado trecho de uma viagem, o perigo de percorrer uma determinada região, ...).

Dessa forma, esse trabalho representará o ambiente através de grafos com pesos. Gostaríamos de deixar claro ao leitor que, apesar de nem sempre o peso representar a distância entre os nós, a palavra distância será empregada diversas vezes para se referir aos pesos do grafo.

## 4.2 REFINAMENTO DO PATH-FINDING

Conforme visto na seção 3.2.2.1, os agentes cognitivos podem escolher qualquer nó do grafo como nó objetivo e utilizar técnicas de *path-finding* para atingir o seu objetivo. No trabalho anterior, um agente cognitivo simples alcançava o nó utilizando o menor caminho. Entretanto, a ociosidade tenderia a ser menor caso ele escolhesse um caminho um pouco maior, mas que passasse por nós mais ociosos. Em outras palavras, ao invés de buscar menores caminhos, nós também levamos em consideração a ociosidade instantânea dos nós situados entre a localização atual do agente e do seu objetivo.

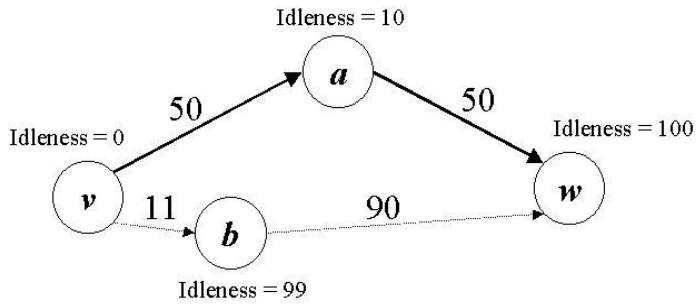


**Figura 5 - Grafo utilizado nos exemplos**

Imaginemos um agente localizado em um determinado nó ***v*** que escolheu o nó objetivo ***w*** (ver Figura 5). A escolha do menor caminho através da distância o faz escolher um caminho (arestas de traço mais espesso, ***v-a-w***) de ***v*** a ***w*** com tamanho 100 unidades de distância, mas que passa apenas pelo nó ***a***, cuja ociosidade é igual a 10. Entretanto, existe um outro caminho (arestas com tracejado pontilhado, ***v-b-w***) um pouco maior de tamanho 101 unidades de distância, mas que passa pelo nó ***b***, cuja ociosidade é igual a 99. Poderíamos pensar: o segundo caminho está muito mais ocioso e tem uma distância apenas um pouco maior, porque não ir de *v* para *w* por ele? Essa é a idéia por trás do que chamamos de *Pathfinder Agents*.

Esses agentes transformam o peso das arestas ***(x, y)*** do grafo de tal forma que o peso é uma média ponderada entre a ociosidade do nó ***y*** e a distância entre ***x*** e ***y***. Essa ponderação é obtida através da taxa de ociosidade, ou seja, a taxa de importância percentual (entre 0% e 100%) dada à ociosidade em detrimento da distância. Uma tentativa de fórmula seria a seguinte:

$$pesoAresta(x,y) = (taxaOciosidade \cdot ociosidade(y)) + ((1 - taxaOciosidade) \cdot distancia(x,y))$$

**Equação 10 - Tentativa de avaliação heurística das arestas**

Uma vez transformado o grafo, os *Pathfinder Agents* podem determinar o menor caminho para o nó objetivo utilizando qualquer algoritmo de busca de menor caminho e considerando os novos pesos das arestas.

Contudo, existem alguns problemas nessa tentativa. Em primeiro lugar, conforme dito anteriormente, quanto mais ocioso estiver um nó ***y***, maior deveria ser a probabilidade de utilizar uma aresta ***(x, y)***, para que no cálculo do menor caminho, o agente passasse por ela e, conseqüentemente, visitasse o nó ***y*** (observe-se, por exemplo, a aresta ***(v,b)*** da Figura 5). A probabilidade de escolha de uma aresta ***(x, y)*** por um algoritmo de menor caminho é maior quando o peso dela é pequeno (algoritmos de menor caminho tendem a escolher menores arestas). Ou seja, quanto maior a ociosidade de um nó ***y***, menor deveria ser o peso atribuído à aresta ***(x, y)***. Entretanto, de acordo com a Equação 10, *ociosidade* e *pesoAresta* são grandezas diretamente proporcionais e não inversamente proporcionais como desejaríamos.

Além disso, ociosidade e distância não são grandezas que possuem a mesma unidade de medida, não podendo assim ser comparadas trivialmente. Em outras palavras, não podemos comparar a princípio uma ociosidade de 100 ciclos com uma distância de 100 unidades de distância. Para tal, é necessário fazer uma normalização das grandezas ociosidade e distância. Com as grandezas normalizadas chegamos a uma outra fórmula:

$$valorAresta = (taxaOciosidade \cdot ociosidadeNormalizada) + ((1 - taxaOciosidade) \cdot distanciaNormalizada)$$

**Equação 11 - Avaliação heurística das arestas**

Duas maneiras foram analisadas para fazer essa normalização. Numa primeira forma de normalização, o agente ordena crescentemente os nós do grafo segundo a ociosidade e ordena decrescentemente as arestas segundo a distância entre os nós incidentes a ela. Atribui-se à maior ociosidade um valor 1 (um), à segunda maior ociosidade um valor 2 e assim por diante, até a menor ociosidade um valor igual a ***V***, onde ***V*** é a quantidade de vértices do grafo. Da mesma forma, atribui-se à menor distância um valor 1, à segunda menor distância um

valor 2 e assim por diante até a maior distância um valor igual a ***E***, onde ***E*** é a quantidade de arestas do grafo.

Consideremos um agente localizado no nó ***v*** do grafo da Figura 5 cujo objetivo é chegar no nó ***w*** e que dá a mesma importância à ociosidade e à distância (50% de taxa de ociosidade). Ordenando os valores das ociosidades obtemos os seguintes valores:

| Nó | Ociosidade | OciosidadeNormalizada |
|---|---|---|
| v | 0 | 4 |
| a | 10 | 3 |
| b | 99 | 2 |
| w | 100 | 1 |

**Tabela 5 - Exemplo da primeira forma de normalização da ociosidade no *path-finding***

Ao normalizarmos as distâncias obtemos os seguintes valores:

| Aresta | Distância | DistânciaNormalizada |
|---|---|---|
| (b, w) | 90 | 4 |
| (v, a) | 50 | 2 |
| (a, w) | 50 | 2 |
| (v, b) | 11 | 1 |

**Tabela 6 - Exemplo da primeira forma de normalização da distância no *path-finding***

Considerando os valores normalizados da Tabela 5 e da Tabela 6 e utilizando a fórmula descrita pela Equação 11, obtemos os seguintes valores para os pesos das arestas do grafo:

| Arestas | OciosidadeNormalizada | DistânciaNormalizada | ValorAresta |
|---|---|---|---|
| (v,a) | 3 | 2 | (0,5*3)+(0,5*2) = 2,5 |
| (a,w) | 1 | 2 | (0,5*1)+(0,5*2) = 1,5 |
| (v,b) | 2 | 1 | (0,5*2)+(0,5*1) = 1,5 |
| (b,w) | 1 | 4 | (0,5*1)+(0,5*4) = 2,5 |

**Tabela 7 - Exemplo da primeira forma de normalização dos pesos das arestas**

Uma vez transformados os pesos das arestas, obtemos o grafo da Figura 6:

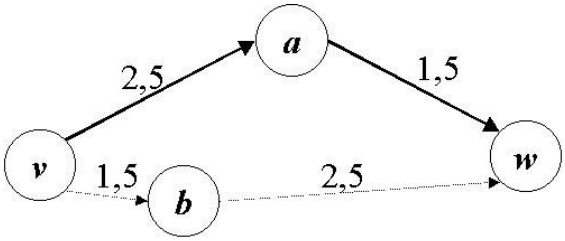


**Figura 6 - Grafo modificado pela primeira forma de normalização**

Conforme podemos observar, o agente terá uma certa dificuldade em escolher entre os dois caminhos, haja vista que os seus tamanhos são iguais (*4* unidades de distância). Porém será que eles são realmente equivalentes? Obviamente não. O caminho formado pelas arestas ***(v, b)*** e ***(b,w)*** é apenas um pouco maior que o formado pelas arestas ***(v, a)*** e ***(a, w)***, mas passa pelo nó ***b***, o qual tem ociosidade muito maior que a do nó ***a***. Logo, ele deveria ter sido escolhido. Isso nos leva à segunda forma de normalização.

Para tal, usamos a idéia de proporção e conversão entre escalas distintas. O cálculo é bastante similar à conversão entre graus Celsius e Fahrenheit. Vejamos inicialmente como pode ser feita a normalização da ociosidade.

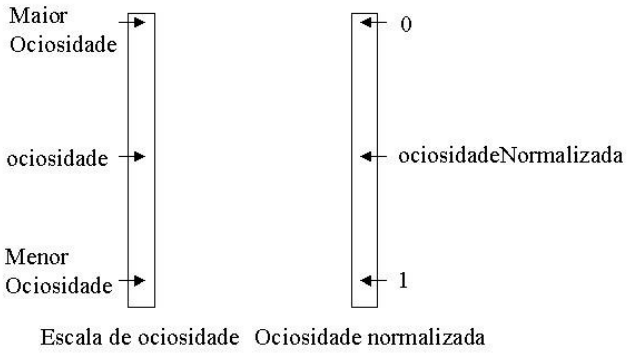


**Figura 7 - Escalas de ociosidade e de ociosidade normalizada**

Consideramos que o valor da maior ociosidade tem um valor zero quando for normalizado ao passo que a menor ociosidade recebe um valor igual a um (ver Figura 7). Valores intermediários são calculados através de proporções. Logo, temos duas escalas: uma de ociosidade que varia da maior ociosidade à menor ociosidade e outra da ociosidade normalizada variando de zero a um. Queremos determinar a que valor na escala de ociosidade normalizada corresponde um determinado valor na escala de ociosidade. Esse valor pode ser determinado pela seguinte fórmula:

$$\frac{(0 - ociosidadeNormalizada)}{(0-1)} = \frac{(maiorOciosidade - ociosidade)}{(maiorOciosidade - menorOciosidade)}$$

Simplificando obtemos:

$$ociosidadeNormalizada = \frac{(maiorOciosidade - ociosidade)}{(maiorOciosidade - menorOciosidade)}$$

**Equação 12 - Normalização da ociosidade no *path-finding***

Seguindo a mesma linha de raciocínio, pode-se chegar a uma fórmula parecida para a normalização da distância. A diferença é que atribuímos à menor distância um valor zero quando for normalizada ao passo que a maior distância recebe um valor igual a um. Mais uma vez, valores intermediários são calculados através de proporções conforme mostrado na Equação 13:

$$distanciaNormalizada = \frac{(distancia - menorDistancia)}{(maiorDistancia - menorDistancia)}$$

**Equação 13 - Normalização da distância no *path-finding***

Consideremos o mesmo agente localizado no nó ***v*** do grafo da Figura 5, cujo objetivo é chegar no nó ***w*** e que dá a mesma importância à ociosidade e à distância (50% de taxa de ociosidade). Normalizando os valores das ociosidades segundo a fórmula da Equação 12, obtemos a seguinte versão modificada da Tabela 5:

| Nó | Ociosidade | OciosidadeNormalizada |
|---|---|---|
| v | 0 | $\frac{(100-0)}{(100-0)} = 1$ |
| a | 10 | $\frac{(100-10)}{(100-0)} = 0{,}9$ |
| b | 99 | $\frac{(100-99)}{(100-0)} = 0{,}01$ |
| w | 100 | $\frac{(100-100)}{(100-0)} = 0$ |

**Tabela 8 - Exemplo da segunda forma de normalização da ociosidade no *path-finding***

Ao normalizarmos a distância segundo a Equação 13, obtemos:

| Aresta | Distância | DistânciaNormalizada |
|---|---|---|
| (b, w) | 90 | $\frac{(90-11)}{(90-11)}=1$ |
| (v, a) | 50 | $\frac{(50-11)}{(90-11)}=0{,}49$ |
| (a, w) | 50 | $\frac{(50-11)}{(90-11)}=0{,}49$ |
| (v, b) | 11 | $\frac{(11-11)}{(90-11)}=0$ |

**Tabela 9 - Exemplo da segunda forma de normalização da distância no *path-finding***

Finalmente, considerando os valores normalizados da Tabela 8 e da Tabela 9 e utilizando a fórmula descrita pela Equação 11, obtemos os seguintes valores para os novos pesos das arestas do grafo:

| Aresta | OciosidadeNormalizada | DistânciaNormalizada | ValorAresta |
|---|---|---|---|
| (v,a) | 0,9 | 0,49 | (0,5*0,9)+(0,5*0,49) = 0,695 |
| (a,w) | 0 | 0,49 | (0,5*0)+(0,5*0,49) = 0,245 |
| (v,b) | 0,01 | 0 | (0,5*0,01)+(0,5*0) = 0,005 |
| (b,w) | 0 | 1 | (0,5*0)+(0,5*1) = 0,5 |

**Tabela 10 - Exemplo da segunda forma de normalização dos pesos das arestas**

Uma vez transformados os pesos das arestas, obtemos o grafo da Figura 8:

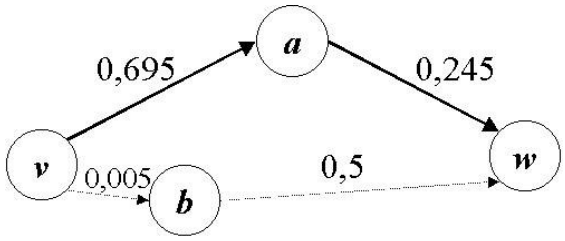


**Figura 8 - Grafo modificado pela segunda forma de normalização**

Utilizando a segunda forma de normalização, o agente não terá mais dificuldade em escolher entre os dois caminhos, pois os seus tamanhos são diferentes. O caminho formado pelas arestas com tracejado pontilhado tem tamanho *0,505*, ao passo que o caminho formado

pelas arestas com traço forte possui tamanho *0,94*. Logo, o caminho escolhido pelo agente será o que utiliza as arestas ***(v, b)*** e ***(b,w)***.

Visto que a segunda forma de normalização consegue uma diferenciação melhor nos diversos caminhos possíveis que podem ser tomados pelo agente, ela foi a normalização utilizada na implementação dos *Pathfinder Agents*.

Uma vez calculados os pesos das arestas através dessa maneira de normalização, o algoritmo utilizado para determinar o caminho é o algoritmo de Dijkstra [13]. Foi provado que a complexidade desse algoritmo é $O((|E| + |V|) * log\ |V|)$, em que $|E|$ é a quantidade de arestas do grafo e $|V|$ a quantidade de nós do grafo.

Poderia ser utilizado um algoritmo de busca heurística tal como A* para determinar o caminho entre o nó atual em que o agente está localizado e o nó escolhido como objetivo. Entretanto, não é trivial encontrar uma boa função heurística admissível que compense o uso do A* em detrimento do Dijkstra, pois os pesos das arestas também levam em consideração a ociosidade.

## 4.3 Aperfeiçoamento da Tomada de Decisão

Além de desenvolver uma nova técnica de *path-finding*, essa dissertação também propõe um aprimoramento no processo de tomada de decisão. Até então, a escolha do próximo nó podia ser apenas aleatória ou baseada na ociosidade, onde um agente escolhe o nó com maior ociosidade. No entanto, se o nó com a maior ociosidade estiver muito longe do agente, seria melhor escolher um nó mais próximo com ociosidade perto da máxima. Ou seja, ao invés de escolher nós com ociosidade máxima, nós também levamos em consideração a distância do agente ao nó candidato.

Imaginemos que um agente localizado em um determinado nó ***v*** quer escolher o próximo nó objetivo a ser visitado (ver novamente a Figura 5). Existem quatro possíveis nós candidatos: ***v, a, b*** e ***w***. Uma tomada de decisão baseada na maior ociosidade o faz escolher o nó ***w*** com ociosidade instantânea igual a 100, mas que está bastante distante (localizado no mínimo a 100 unidades de distância do nó ***v***). Entretanto, existe um outro nó, o nó ***b***, com ociosidade instantânea igual a 99, mas que está localizado a apenas 11 unidades de distância do nó ***v***. Poderíamos pensar: o segundo nó está muito mais perto e tem uma ociosidade apenas um pouco menor, porque não visitá-lo logo? É exatamente essa a idéia subjacente nos

*Heuristic Agents* [3], os quais utilizam essa nova maneira de se fazer a escolha do próximo nó: heuristicamente baseado na ociosidade e na distância.

A escolha heurística atribui um valor aos nós candidatos a objetivo e escolhe aquele com o maior valor. Esse valor é uma média ponderada entre a ociosidade do nó e a distância do nó candidato ao nó em que o agente está localizado. Essa ponderação é obtida através da taxa de ociosidade, ou seja, a taxa de importância percentual (entre 0% e 100%) dada à ociosidade em detrimento da distância. Uma tentativa de fórmula seria a seguinte:

$$valorNó = (taxaOciosidade \cdot ociosidade) + ((1 - taxaOciosidade) \cdot distancia)$$

**Equação 14 - Tentativa de avaliação heurística dos nós**

Contudo, tal como ocorreu no refinamento do *path-finding*, existem alguns problemas nessa tentativa. Em primeiro lugar, o valor atribuído ao nó deveria ser inversamente proporcional à distância, ou seja, quanto mais longe estiver um nó do agente, menor deveria ser seu valor. Dessa forma, menor seria a probabilidade de escolher nós distantes. No entanto, de acordo com a Equação 14 *distância* e *valorNó* são grandezas diretamente proporcionais. Adicionalmente, pode-se constatar novamente o problema de ociosidade e distância serem grandezas de unidades de medida diferentes.

Por conseguinte, é preciso fazer uma normalização das grandezas ociosidade e distância. Com as grandezas normalizadas chegamos à seguinte fórmula:

$$valorNó = (taxaOciosidade \cdot ociosidadeNormalizada) + ((1 - taxaOciosidade) \cdot distanciaNormalizada)$$

**Equação 15 - Avaliação heurística dos nós**

Utilizar algo semelhante à primeira forma de normalização explanada anteriormente na Seção 4.1 apresentaria os mesmos problemas.

Sendo assim, usamos outra vez a idéia de proporção e conversão entre escalas distintas. A fim de normalizar a ociosidade, consideramos que o valor da menor ociosidade tem um valor zero quando for normalizado ao passo que a maior ociosidade recebe um valor igual a um. Valores intermediários são calculados através de proporções. Logo, temos duas escalas: uma de ociosidade que varia da menor ociosidade à maior ociosidade e outra da ociosidade normalizada variando de zero a um. Queremos determinar a que valor na escala de

[3] O nome *Heuristic* talvez seja infeliz, pois a escolha do próximo nó baseada apenas na ociosidade também é heurística. Entretanto, o nome foi mantido por já ter sido utilizado anteriormente em publicações.

ociosidade normalizada corresponde um determinado valor na escala de ociosidade. Esse valor pode ser determinado pela seguinte fórmula:

$$ociosidadeNormalizada = \frac{(ociosidade - menorOciosidade)}{(maiorOciosidade - menorOciosidade)}$$

**Equação 16 - Normalização da ociosidade na tomada de decisão**

Similarmente, para a distância obtemos uma fórmula parecida, com a diferença que consideramos que o valor da maior distância tem um valor zero quando for normalizado ao passo que a menor distância recebe um valor igual a um.

$$distanciaNormalizada = \frac{(maiorDistancia - distancia)}{(maiorDistancia - menorDistancia)}$$

**Equação 17 - Normalização da distância na tomada de decisão**

Consideremos um agente localizado no nó ***v*** do grafo da Figura 5, o qual deve escolher um dos nós como objetivo e que dá a mesma importância à ociosidade e à distância (50% de taxa de ociosidade). Normalizando os valores das ociosidades através da Equação 16 obtemos os seguintes valores:

| Nó | Ociosidade | OciosidadeNormalizada |
|---|---|---|
| v | 0 | $\frac{(0-0)}{(100-0)} = 0$ |
| a | 10 | $\frac{(10-0)}{(100-0)} = 0{,}1$ |
| b | 99 | $\frac{(99-0)}{(100-0)} = 0{,}99$ |
| w | 100 | $\frac{(100-0)}{(100-0)} = 1$ |

**Tabela 11 - Exemplo da normalização da ociosidade na tomada de decisão**

Ao normalizarmos a distância segundo a Equação 17, obtemos:

| Nó | Distância de v | DistânciaNormalizada |
|---|---|---|
| v | 0 | $\frac{(100-0)}{(100-0)} = 1$ |
| a | 50 | $\frac{(100-50)}{(100-0)} = 0{,}5$ |
| b | 11 | $\frac{(100-11)}{(100-0)} = 0{,}89$ |
| w | 100 | $\frac{(100-100)}{(100-0)} = 0$ |

**Tabela 12 - Exemplo da normalização da distância na tomada de decisão**

Finalmente, considerando os valores normalizados da Tabela 11 e da Tabela 12 e utilizando a fórmula descrita pela Equação 15, obtemos os seguintes valores para os nós do grafo:

| Nó | OciosidadeNormalizada | DistânciaNormalizada | ValorNó |
|---|---|---|---|
| v | 0 | 1 | (0,5*0)+(0,5*1) = 0,5 |
| a | 0,1 | 0,5 | (0,5*0,1)+(0,5*0,5) = 0,3 |
| b | 0,99 | 0,89 | (0,5*0,99)+(0,5*0,89) = 0,94 |
| w | 1 | 0 | (0,5*1)+(0,5*0) = 0,5 |

**Tabela 13 - Exemplo da normalização dos valores dos nós na tomada de decisão**

Conforme esperado, o nó ***b*** é escolhido pelo agente, pois é o que apresenta o maior valor. De fato, ele possui uma ociosidade instantânea bem próxima do máximo e está localizado a uma pequena distância do agente.

Na próxima seção, veremos que novas arquiteturas foram geradas ao combinar as novas técnicas descritas nessa seção e na seção anterior.

## 4.4 NOVOS AGENTES

Inicialmente, todas as 7 arquiteturas já mencionadas anteriormente na Seção 3.2.2 tiveram que ser reimplementadas para dar suporte a grafos com pesos nas arestas.

Em seguida, considerando essas novas variações do processo de tomada de decisão e técnicas de *path-finding*, 11 novas arquiteturas foram implementadas, conforme mostrado na Tabela 14.

| Arquitetura anterior | Novas arquiteturas | Mudanças possíveis consideradas |
|---|---|---|
| *Conscientious Reactive* | *Heuristic Conscientious Reactive* | Tomada de decisão avançada |
| *Reactive with Flags* | *Heuristic Reactive with Flags* | Tomada de decisão avançada |
| *Conscientious Cognitive* | *Heuristic Conscientious Cognitive* | Tomada de decisão avançada |
| | *Pathfinder Conscientious Cognitive* | *Path-finding* avançado |
| | *Heuristic Pathfinder Conscientious Cognitive* | *Path-finding* e tomada de decisão e avançados |
| *Conscientious Cognitive* | *Heuristic Blackboard Cognitive* | Tomada de decisão avançada |
| | *Pathfinder Blackboard Cognitive* | *Path-finding* avançado |
| | *Heuristic Pathfinder Blackboard Cognitive* | *Path-finding* e tomada de decisão e avançados |
| *Cognitive Coordinated* | *Heuristic Cognitive Coordinated* | Tomada de decisão avançada |
| | *Pathfinder Cognitive Coordinated* | *Path-finding* avançado |
| | *Heuristic Pathfinder Cognitive Coordinated* | *Path-finding* e tomada de decisão e avançados |

**Tabela 14 - Novas arquiteturas implementadas**

O *path-finding* avançado não pode ser utilizado com agentes reativos, por questões óbvias já mencionadas na seção 2.4.1. Ao contrário do *path-finding* refinado, o aperfeiçoamento da tomada de decisão pode ser aplicado a qualquer uma das arquiteturas, à exceção das arquiteturas aleatórias.

A Tabela 15 resume as principais características dos novos agentes, seguindo o mesmo formato da Tabela 3.

<table>
<tr><th>Nome</th><th>Tipo Básico</th><th>Comunicação</th><th>Escolha do próximo nó</th></tr>
<tr><td>Heuristic Conscientious Reactive</td><td rowspan="2">Reativo</td><td>Nenhuma</td><td rowspan="3">Heurística baseada na distância e na ociosidade</td></tr>
<tr><td>Heuristic Reactive with Flags</td><td>Flags</td></tr>
<tr><td>Heuristic Conscientious Cognitive</td><td>Cognitivo</td><td rowspan="3">Nenhuma</td></tr>
<tr><td>Pathfinder Conscientious Cognitive</td><td rowspan="2">Cognitivo Heurístico</td><td>Heurística baseada na distância</td></tr>
<tr><td>Heuristic Pathfinder Conscientious Cognitive</td><td rowspan="2">Heurística baseada na distância e na ociosidade</td></tr>
<tr><td>Heuristic Blackboard Cognitive</td><td>Cognitivo</td><td rowspan="3">Blackboard</td></tr>
<tr><td>Pathfinder Blackboard Cognitive</td><td rowspan="2">Cognitivo Heurístico</td><td>Heurística baseada na distância</td></tr>
<tr><td>Heuristic Pathfinder Blackboard Cognitive</td><td rowspan="2">Heurística baseada na distância e na ociosidade</td></tr>
<tr><td>Heuristic Cognitive Coordinated</td><td>Cognitivo</td><td rowspan="3">Mensagens</td></tr>
<tr><td>Pathfinder Cognitive Coordinated</td><td rowspan="2">Cognitivo Heurístico</td><td>Heurística baseada na distância</td></tr>
<tr><td>Heuristic Pathfinder Cognitive Coordinated</td><td>Heurística baseada na distância e na ociosidade</td></tr>
</table>

**Tabela 15 - Resumo das principais características das novas arquiteturas**

Em comparação à Tabela 3, a Tabela 15 mostra um outro tipo básico de agente, os agentes cognitivos heurísticos, os quais escolhem o menor caminho utilizando o mecanismo de *path-finding* descrito na seção 4.2. Do mesmo modo, um novo método de escolha do próximo nó foi inserido, a escolha heurística baseada na distância e na ociosidade, a qual já foi descrita na seção anterior. As estratégias detalhadas de tomada de decisão são mostradas na Tabela 16.

| Nome da Arquitetura | Escolha detalhada do próximo nó |
|---|---|
| *Heuristic Conscientious Reactive* | Distância + Ociosidade individual local |
| *Heuristic Reactive with Flags* | Distância + Ociosidade compartilhada local |
| *Heuristic Conscientious Cognitive* | Distância + Ociosidade individual global |
| *Pathfinder Conscientious Cognitive* | Ociosidade individual global |
| *Heuristic Pathfinder Conscientious Cognitive* | Distância + Ociosidade individual global |
| *Heuristic Blackboard Cognitive* | Distância + Ociosidade compartilhada global |
| *Pathfinder Blackboard Cognitive* | Ociosidade compartilhada global |
| *Heuristic Pathfinder Blackboard Cognitive* | Distância + Ociosidade compartilhada global |
| *Heuristic Cognitive Coordinated* | Distância + Ociosidade compartilhada e coordenada global |
| *Pathfinder Cognitive Coordinated* | Ociosidade compartilhada e coordenada global |
| *Heuristic Pathfinder Cognitive Coordinated* | Distância + Ociosidade compartilhada e coordenada global |

**Tabela 16 - Estratégia de tomada de decisão detalhada usada pelas novas arquiteturas, considerando o campo de visão e a comunicação entre os agentes**

Maiores detalhes sobre cada uma dessas arquiteturas são dados a seguir.

### 4.4.1 Heuristic Conscientious Reactive

Essa arquitetura é uma das mais simples. Consiste em agentes reativos com estado interno, o qual armazena os nós do grafo já visitados pelo agente e quando ocorreram essas visitas. Cada agente escolhe um dos nós de sua vizinhança para ser visitado (campo de visão local). O processo de tomada de decisão utiliza uma heurística combinada baseada não só na ociosidade, mas também na distância. Sempre que existe um empate, escolhe-se um dos nós empatados de forma arbitrária.

### 4.4.2 Heuristic Reactive with Flags

A diferença dessa arquitetura para a anterior está na presença de comunicação via *flags* entre os agentes. Dessa forma, todos os agentes possuem a mesma visão das ociosidades do grafo, sabendo há quanto tempo foi visitado cada nó. Tal como na arquitetura anterior, a escolha do próximo nó a ser visitado é feita dentro da vizinhança, levando em consideração a distância e os desempates são feitos de forma aleatória.

### 4.4.3 Heuristic Conscientious Cognitive

Esses agentes são bastante parecidos com os agentes descritos na Seção 4.4.1 (*Heuristic Conscientious Reactive*). Ou seja, eles possuem memória e armazenam suas ações

antecedentes. No entanto, os integrantes dessa arquitetura podem formular um objetivo e dispõem da capacidade de planejar um caminho para ele. Isso implica na possibilidade de escolher qualquer nó do grafo como o próximo objetivo. A escolha do próximo nó é feita usando a ociosidade e a distância. Os agentes utilizam o menor caminho para alcançar o objetivo (*path-finding* simples).

### 4.4.4 Pathfinder Conscientious Cognitive

Essa arquitetura é um aprimoramento da *Conscientious Cognitive*. Da mesma forma que os *Conscientious Cognitive*, eles escolhem qualquer nó do grafo como objetivo segundo a maior ociosidade individual. Contudo, enquanto que os agentes *Conscientious Cognitive* utilizam o menor caminho para alcançar o objetivo, os *Pathfinder Conscientious Cognitive* lançam mão do mecanismo avançado de *path-finding* descrito anteriormente na seção 4.2.

### 4.4.5 Heuristic Pathfinder Conscientious Cognitive

Os agentes *Heuristic Pathfinder Conscientious Cognitive* consistem numa união das características dos dois últimos agentes descritos. O processo de tomada de decisão é idêntico ao dos agentes *Heuristic Conscientious Cognitive*, ou seja, através da função heurística dada na Equação 15. Já o mecanismo de *path-finding* utilizado para alcançar o objetivo é o mesmo dos *Pathfinder Conscientious Cognitive*, pois leva em consideração a ociosidade dos nós intermediários.

### 4.4.6 Heuristic Blackboard Cognitive

A arquitetura *Heuristic Blackboard Cognitive* está para a *Heuristic Reactive with Flags* assim como os agentes *Heuristic Conscientious Cognitive* estão para os *Heuristic Conscientious Reactive*. No caso da arquitetura *Heuristic Blackboard Cognitive*, de forma semelhante à arquitetura *Heuristic Reactive with Flags* (seção 4.4.1) existe uma comunicação entre os agentes que é feita de forma indireta. Porém, ao invés de marcas deixadas no ambiente que podem apenas ser percebidas quando estão dentro do campo de visão do agente, existe uma estrutura comum (*blackboard*) onde todos os agentes indicam sua passagem por cada nó. Dessa forma, os agentes *Heuristic Blackboard Cognitive* tem uma visão compartilhada de todo o grafo e podem escolher qualquer nó como objetivo. A escolha, tal como feita pelos agentes *Heuristic Reactive with Flags*, combina ociosidade e distância. Para alcançar o objetivo, é utilizado o menor caminho.

### 4.4.7 Pathfinder Blackboard Cognitive

Essa arquitetura é um aprimoramento da *Blackboard Cognitive*. Da mesma forma que os agentes *Blackboard Cognitive*, eles escolhem qualquer nó do grafo como objetivo segundo a maior ociosidade compartilhada. Contudo, enquanto que os agentes *Blackboard Cognitive* utilizam o menor caminho para atingir o objetivo, os *Pathfinder Blackboard Cognitive* lançam mão do mecanismo avançado de *path-finding* descrito anteriormente.

### 4.4.8 Heuristic Pathfinder Blackboard Cognitive

Os agentes *Heuristic Pathfinder Blackboard Cognitive* consistem numa união das características dos dois últimos agentes descritos. O processo de tomada de decisão é idêntico ao dos agentes *Heuristic Blackboard Cognitive*, ou seja, ponderando a ociosidade compartilhada e a distância. Já o mecanismo de *path-finding* utilizado para chegar no objetivo é o mesmo dos *Pathfinder Blackboard Cognitive*, pois leva em consideração a ociosidade dos nós intermediários.

### 4.4.9 Heuristic Cognitive Coordinated

A partir dessa seção as arquiteturas deixam de ser homogêneas, pois é adotada a estratégia de coordenação central explícita, e para tal é incluído um outro tipo de agente chamado de coordenador. Esse coordenador tem conhecimento de todas as visitas que acontecem no ambiente e determina o objetivo de cada agente. O protocolo de decisão acontece da seguinte forma: cada agente pergunta ao coordenador qual será seu objetivo. O coordenador nunca dará a dois agentes diferentes o mesmo objetivo. Uma vez determinado o objetivo, a responsabilidade de como cumprir a tarefa fica por conta do agente patrulheiro. Os agentes que executam a patrulha são cognitivos, ou seja, dado o objetivo eles planejam o caminho que deve ser percorrido.

Nessa arquitetura em especial, a decisão do coordenador é feita utilizando a distância do agente aos nós candidatos e a ociosidade instantânea desses nós. Após tomar conhecimento de qual nó deve visitar, os agentes utilizam o menor caminho para chegar até ele.

### 4.4.10 Pathfinder Cognitive Coordinated

Essa arquitetura é um aprimoramento da *Cognitive Coordinated*. Da mesma forma que o coordenador dos agentes *Cognitive Coordinated*, o coordenador dos agentes *Pathfinder Cognitive Coordinated* aloca ao agente o nó do grafo com maior ociosidade que não é objetivo de nenhum outro agente. Contudo, enquanto que os agentes *Cognitive Coordinated*

utilizam o menor caminho para alcançar o objetivo, os *Pathfinder Cognitive Coordinated* lançam mão do mecanismo refinado de *path-finding* descrito na Seção 4.1.

### 4.4.11 Heuristic Pathfinder Cognitive Coordinated

A última arquitetura implementada é a mais completa de todas. Utiliza um coordenador para evitar colisão de objetivos entre os agentes. Além disso, a estratégia de tomada de decisão do coordenador é a mais complexa de todas, pois combina ociosidade e distância. Os agentes também são os mais aprimorados, visto que determinam o caminho para o objetivo considerando não apenas a distância, mas também a ociosidade instantânea dos nós intermediários.

Convém lembrar que utilizamos monitoramento (descrito na Seção 3.2.2.5) sempre que permitido. Dessa forma, os agentes cognitivos sempre mudam seus objetivos caso ele seja visitado por outro agente. O monitoramento apenas é possível quando existe comunicação entre os agentes, ou seja, nas seguintes arquiteturas: *Heuristic Blackboard Cognitive*, *Pathfinder Blackboard Cognitive*, *Heuristic Pathfinder Blackboard Cognitive*, *Heuristic Cognitive Coordinated*, *Pathfinder Cognitive Coordinated* e *Heuristic Pathfinder Cognitive Coordinated*.

## 4.5 Conclusão

Nesse capítulo, foi abordado como os mecanismos de *path-finding* e tomada de decisão propostos no estado da arte podem ser aperfeiçoados. Vimos inicialmente a importância de se utilizar grafos com pesos na representação do ambiente.

Em seguida, abordamos algumas técnicas que podem ser usadas para considerar a ociosidade dos nós intermediários no cálculo do menor caminho. Também foram apresentadas técnicas similares as quais podem ser empregadas no desenvolvimento de uma escolha heurística do próximo nó alvo que também considera a distância do agente aos possíveis nós alvos. Por último, apresentaram-se os agentes implementados por esse trabalho.

**Comentário:** Ajeitar

No próximo capítulo, veremos que método experimental foi empregado a fim de testar as diversas arquiteturas implementadas.

# 5 MÉTODO EXPERIMENTAL

*"Genius is one percent inspiration and ninety-nine percent perspiration".*

*(Thomas A. Edison)*

Uma vez definidas as novas arquiteturas, torna-se necessário determinar como se comporta o desempenho delas em relações às arquiteturas já propostas. Para tal, adotamos um método experimental, o qual será descrito nesse capítulo.

Em primeiro lugar, levantar-se-ão algumas hipóteses a serem verificadas ou refutadas pelos experimentos. Após isso, serão apresentados os grafos utilizados na avaliação dos agentes. Em seguida, descreveremos o funcionamento básico do simulador de experimentos e que mudanças foram necessárias realizar para dar suporte aos avanços realizados nessa dissertação. Por último, será descrito como foi realizado cada experimento (ou simulação).

## 5.1 HIPÓTESES

Os experimentos terão como objetivo avaliar as diferentes arquiteturas propostas e verificar se as seguintes hipóteses são válidas:

- Arquiteturas com melhor desempenho são obtidas quando se utiliza uma tomada de decisão heurística combinando ociosidade e distância;
- Da mesma forma, combinar ociosidade e distância no *path-finding* melhora o desempenho dos agentes.

## 5.2 NOVOS GRAFOS

Com as novas arquiteturas definidas a próxima preocupação metodológica foi a de definir as situações em que esses agentes iriam atuar. Conforme dito anteriormente, esse trabalho trata do problema da supervisão em grafos com pesos. Como os trabalhos antecedentes consideravam apenas grafos sem pesos, não pudemos aproveitar os mesmos ambientes utilizados anteriormente. Assim, propusemos novos grafos, os quais são descritos nessa seção.

Antes de determinar os grafos a serem utilizados nas simulações, faz-se necessário refletir sobre que parâmetros do ambiente podem influenciar o desempenho dos agentes de patrulha durante a execução da tarefa de supervisão.

Conforme observado por Aydano Machado [31], dois fatores que influenciam a ociosidade média do grafo são a quantidade de nós por agente e a conectividade do grafo. De fato, quanto mais agentes, menor tenderá a ser a ociosidade média do grafo. Da mesma forma, quanto mais conexo um grafo, mais numerosos serão os caminhos entre os diversos nós. Logo, a probabilidade de os agentes percorrerem partes distintas do ambiente será maior e eles poderão se distribuir mais uniformemente e, conseqüentemente, menor será a ociosidade.

Entretanto, a partir do momento em que são considerados pesos nas arestas, outras características podem ser observadas. Por exemplo, o desvio padrão no peso das arestas pode influenciar o comportamento dos agentes que escolhem o próximo nó usando uma heurística baseada na distância. Caso todas arestas tenham pesos cujos valores são muito próximos (pouco desvio padrão), a influência da distância na tomada de decisão será desprezível. Além de considerar o desvio padrão, grafos com pesos representando ambientes reais podem passar a ser utilizados. Até esse trabalho, nenhum grafo representando ambientes reais havia sido feito.

Convém lembrar que não utilizamos os grafos utilizados em [31], pois os mesmos eram grafos sem pesos. Além disso, ele testou os agentes em apenas dois grafos.

Sendo assim, foram criados quatro grafos para realizar os experimentos. Os dois primeiros grafos descritos a seguir representam ambientes reais com pequeno desvio padrão. Os dois outros grafos foram criados com o intuito de variar o grau de conectividade entre as diversas regiões do ambiente e possuem um desvio padrão maior.

### 5.2.1 Shopping Guararapes

O primeiro grafo considerado representa o ambiente de um dos maiores centros comerciais de Recife, o Shopping Guararapes [51]. A planta do edifício, juntamente com sua representação em forma de grafo pode ser vista na Figura 9.

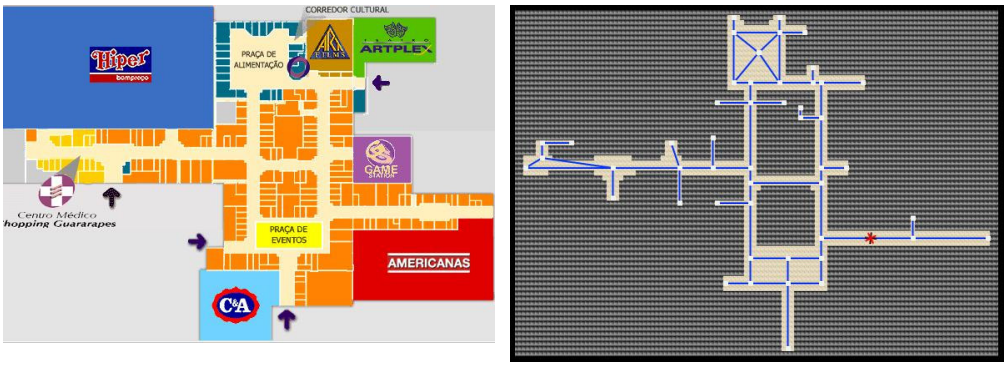


**Figura 9 - Planta do Shopping Guararapes e sua representação através de grafo**

O grafo gerado possui 49 nós e 57 arestas. Como pode ser notada na figura, a conectividade do grafo é bastante pequena. O grafo possui muito poucos ciclos e tem a aparência de uma árvore na maior parte de sua extensão. O peso das arestas corresponde a distância entre os nós. As arestas possuem um peso médio de *7,95* e o desvio padrão é igual a *4,54*.

### 5.2.2 Ruas do Recife

Tal como o grafo anterior, o próximo grafo criado representa um ambiente real. Ele compreende os bairros das Graças, Aflitos, Espinheiro e Santo Amaro da cidade do Recife [45]. Um mapa da região, juntamente com sua representação em forma de grafo pode ser vista na Figura 10.

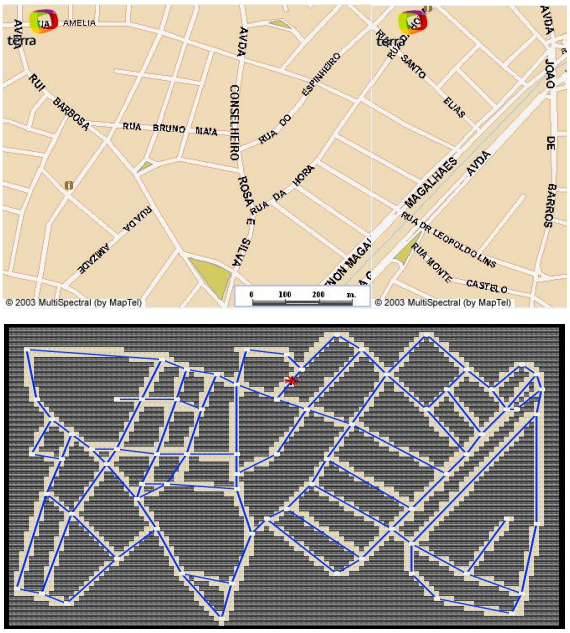


**Figura 10 - Mapa dos bairros do Recife considerados e sua representação através de grafo**

O grafo das ruas do Recife possui 76 nós e 111 arestas. A conectividade do grafo é maior que o anterior. O grafo possui diversos ciclos e tem a aparência de uma *grid* na maior parte de sua extensão. O peso das arestas corresponde ao tamanho dos trechos das ruas. As arestas possuem um peso médio de *9,08* e o desvio padrão é igual a *5,24*.

### 5.2.3 Grafos das Cinco Regiões

Como já mencionado, os dois grafos seguintes (Figura 11) não representam um terreno real. Eles possuem cinco regiões bem definidas (noroeste, nordeste, centro, sudoeste e sudeste) que são interligadas através de arestas.

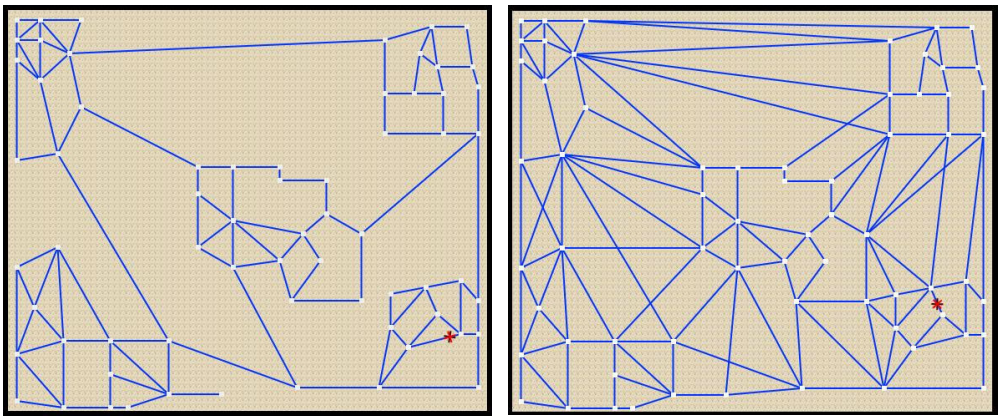

**Figura 11 - Grafos das Cinco Regiões**

O grafo da esquerda possui 66 nós e 109 arestas. Conforme pode ser observado na figura, a conectividade entre as regiões é bastante pequena, existindo poucos caminhos entre uma região e outra. As arestas possuem um peso médio de *8,66* e o desvio padrão é igual a *6,66*.

Já o grafo da direita possui 66 nós e 140 arestas. A conectividade entre as regiões é maior, existindo mais caminhos entre uma região e outra. As arestas possuem um peso médio de *11,69* e o desvio padrão é igual a *10,47*.

## 5.3 Novo Simulador

Um simulador foi desenvolvido a fim de prover um ambiente computacional no qual pudessem ser realizados experimentos para estudar e avaliar as diversas arquiteturas propostas. Outra grande motivação no desenvolvimento do simulador é a redução dos custos, pois não precisamos projetar robôs para realização de um experimento num ambiente real.

A primeira versão desse simulador foi desenvolvida por Eric Bruno Perazzo Mariz, Mozart de Siqueira Campos Araújo Filho e por mim, numa disciplina de Graduação do CIn/UFPE. A forte relação do trabalho inicial com o desenvolvimento de jogos influenciou muito a concepção do simulador e nos levou a implementá-lo utilizando C++ [56] e OpenGL [8][57], que constituem uma plataforma de desenvolvimento comumente usada na comunidade de jogos de computadores. Nessa versão inicial do simulador, apenas 4 das 18

arquiteturas atuais estavam implementadas. Os dois critérios de avaliação utilizados atualmente (ociosidade média e ociosidade máxima) estavam implementados.

Numa segunda versão, Aydano Machado refez o módulo de geração de estatísticas da simulação, implementou os agentes utilizados em seu trabalho que ainda não estavam desenvolvidos e incorporou o critério tempo de exploração.

A versão atual é de médio porte, contendo cerca de 100 arquivos com definições e implementações de classes.

Existe a opção de visualizar graficamente a simulação, como pode ser visto na Figura 12. Na visualização dos experimentos, os agentes aparecem em colorido e os nós em tonalidade de cinza (um nó com ociosidade zero aparece em branco e à medida que a ociosidade dele aumenta, ele vai escurecendo tendendo a ficar preto).

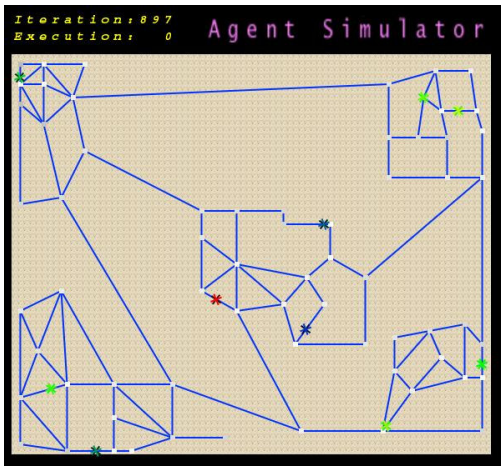


**Figura 12 - Snapshot do simulador executando com 10 agentes no mapa das cinco regiões**

O mundo é descrito em um formato proprietário, permitindo que o pesquisador indique todas as características do ambiente, tais como o tamanho do mapa, os obstáculos, o grafo de visibilidade, as posições iniciais dos agentes, a quantidade de agentes, o tipo de arquitetura a ser usado, o número de ciclos a serem rodados, a quantidade de execuções e se a simulação deve ser visualizada graficamente ou não.

A classe principal, *Simulator*, é responsável pelo gerenciamento da simulação e contém o loop principal do sistema. A execução básica do sistema é dada abaixo:

```
Leia o mapa e inicialize o mundo (chamando o método read do MapReader)
Para cada iteração {
    Desenhe na tela a cena, se for para visualizá-la
    Incremente a ociosidade dos nós do grafo
    Atualize os agentes (chamando o método reason deles)
    Gere as estatísticas da iteração
}
Compute as estatísticas finais da execução
```

A classe principal possui uma estrutura de dados (classe *PatrollingDataStructure*) que contém o grafo de visibilidade (classe *Graph*) e a estrutura de armazenamento dos menores caminhos (classe *PathDataStructure*). Dessa forma, através da classe *PatrollingDataStructure* podemos não só obter as informações de ociosidade dos nós do grafo a fim de gerar estatísticas, mas também fazer as consultas de menor caminho caso os agentes necessitem. Os algoritmos utilizados para buscar o menor caminho está implementado na classe *PathDataStructure*, sendo eles os algoritmos de *Floyd-Warshall* (*All-Pairs Shortest Path*) e o *Dijkstra*, conforme dito anteriormente.

A classe *Simulator* também possui o mundo no qual a simulação é feita. O mundo (definido na classe *World*) é composto pelo conjunto de agentes e pelo mapa (descrito na classe *SimulatorMap* através de uma matriz bidimensional de *tiles*)

Qualquer agente pode ser facilmente acoplado ao sistema. Para tal, basta herdar da classe básica que define os agentes (*ContinuousAgent*) e implementar o seu método de raciocínio (*reason*).

Para esse trabalho, algumas modificações tiveram que ser feitas ao simulador. Inicialmente, foram implementadas as 11 arquiteturas dispostas na Tabela 15. Para tal, foram adicionados na classe *PatrollingDataStructure* métodos para auxiliar os agentes na tomada de decisão heurística baseada na ociosidade e na distância. Conforme dito acima, a classe *PathDataStructure* agora provê o suporte necessário para o *path-finding* aprimorado.

Adicionalmente, o simulador teve que ser adaptado para dar suporte a pesos nas arestas. A classe *Graph* possui atualmente um grafo com pesos ao invés de sua versão anterior sem pesos. Uma vez que as arestas nesse trabalho podem possuir tamanho maior que uma unidade de distância, requerendo do agente mais de um ciclo para percorrê-la completamente, os métodos de navegação de todos os agentes tiveram que ser reimplementados para permitir que os agentes se localizem sobre uma aresta (até então os agentes só podiam se localizar sobre um nó do grafo).

Por último, alguns erros de vazamento de memória existentes em todas as versões anteriores foram resolvidos. Vários objetos que estavam sendo alocados não eram desalocados após sua utilização. Além disso, os métodos destrutores de classes possuidoras de

subclasses não estavam implementados como métodos virtuais. Dessa forma, os destrutores das subclasses não eram chamados. Exemplos desse problema foram encontrados nas classes dos agentes as quais herdam da classe *SimulatorObject*. Conseqüentemente, apesar de os métodos destrutores dos agentes possuírem código para desalocar várias estruturas de dados utilizadas por eles no processo de raciocínio, o simulador não chamava esses métodos.

## 5.4 EXPERIMENTOS REALIZADOS

Os cenários de experimentação foram os grafos descritos na Seção 5.2.

Com a preocupação de variar a quantidade de nós por agente, para cada grafo, escolhemos 5 quantidades distintas de agentes a serem simulados, as quais estão mostradas na Tabela 17.

| Grafo | Populações |
|---|---|
| Shopping Guararapes | 1, 5, 10, 20 e 30 |
| Ruas do Recife | 1, 5, 10, 20 e 40 |
| Cinco regiões (ambos os grafos) | 1, 5, 10, 20 e 35 |

**Tabela 17 - Populações utilizadas nas simulações**

Em cada grafo, foram testadas 122 arquiteturas: as 7 contidas na Tabela 3 e 115 variações das 11 arquiteturas da Tabela 15, conforme descrito a seguir.

Para os agentes do tipo *Heuristic* (*Heuristic Conscientious Reactive*, *Heuristic Reactive with Flags*, *Heuristic Conscientious Cognitive*, *Heuristic Blackboard Cognitive* e *Heuristic Cognitive Coordinated*), consideramos 5 valores (0.0, 0.2, 0.4, 0.6, 0.8) para a taxa de ociosidade como descrita na Equação 15. Um valor de 1.0 para a taxa de ociosidade corresponde à tomada de decisão com importância de 100% para a ociosidade, ou seja, na qual o agente escolhe o nó com a maior ociosidade.

Para os agentes do tipo *Pathfinder* (*Pathfinder Conscientious Cognitive*, *Pathfinder Blackboard Cognitive* e *Pathfinder Cognitive Coordinated*), também avaliamos 5 valores (0.2, 0.4, 0.6, 0.8, 1.0) para a taxa de ociosidade como descrita na Equação 11. Um valor de 0.0 para a taxa de ociosidade corresponde a um *path-finding* com importância de 100% para a distância, ou seja, que busca o menor caminho, sem considerar a ociosidade instantânea dos nós intermediários.

Finalmente, analisamos 25 variações de cada um dos agentes do tipo *Heuristic Pathfinder* (*Heuristic Pathfinder Conscientious Cognitive*, *Heuristic Pathfinder Blackboard Cognitive* e *Heuristic Pathfinder Cognitive Coordinated*). Essas variações correspondem à

combinação dos 5 valores para a taxa de ociosidade relativa a escolha heurística dos nós com os 5 valores para a taxa de ociosidade relativa a escolha heurística das arestas. Dessa forma, são gerados 25 pares: (0.0, 0.2), (0.0, 0.4), (0.0, 0.6), (0.0, 0.8), (0.0, 1.0), (0.2, 0.2), (0.2, 0.4), (0.2, 0.6), (0.2, 0.8), (0.2, 1.0), (0.4, 0.2), (0.4, 0.4), (0.4, 0.6), (0.4, 0.8), (0.4, 1.0), (0.6, 0.2), (0.6, 0.4), (0.6, 0.6), (0.6, 0.8), (0.6, 1.0), (0.8, 0.2), (0.8, 0.4), (0.8, 0.6), (0.8, 0.8), (0.8, 1.0), onde o primeiro valor corresponde à importância dada à ociosidade na tomada de decisão e o segundo valor referente à importância dada à ociosidade no *path-finding*.

Para cada tripla (grafo, população, arquitetura), por exemplo, (*ruas de recife*, *20 agentes*, *Heuristic Pathfinder Cognitive Coordinated (0.8, 1.0)*), fizemos cinqüenta execuções. Em cada uma dessas execuções, os agentes começam em pontos (nós) diferentes (i.e. posição inicial dos agentes). Para cada par (grafo, população) estas posições iniciais foram geradas de maneira aleatória e então armazenadas. Dessa forma, as mesmas posições iniciais são utilizadas na inicialização das diferentes arquiteturas. Isso permite uma melhor comparação entre as arquiteturas, pois os experimentos de cada arquitetura começaram da mesma forma.

Conforme mencionado na Seção 3.3, dois critérios foram utilizados para avaliar o desempenho dos agentes: ociosidade média e ociosidade máxima. Esses critérios já foram definidos matematicamente na Seção 3.2.1.

Até esse trabalho, cada simulação era executada por uma quantidade arbitrária de ciclos (ou iterações). Entretanto, ao considerar o problema de escolher uma quantidade apropriada de ciclos para uma simulação, uma abordagem razoável é visitar cada nó $k$ vezes. No pior caso, um nó passará $OM$ iterações sem ser visitado, onde $OM$ é a ociosidade máxima da simulação. Portanto, um limite superior para que os agentes visitem cada nó no mínimo $k$ vezes é escolher $k * OM$ iterações. No nosso caso, utilizamos um valor de $k$ igual a 15.

A fim de escolher um valor aproximado para $OM$, foram executados testes preliminares com todas as 122 arquiteturas patrulhando cada um dos quatro grafos e considerando todas as populações descritas na Tabela 17. Cada um desses testes durou 30000 ciclos. Para cada grafo, foi escolhido um valor para $OM$ que fosse superior à maioria dos valores obtidos nesses experimentos iniciais. Esses valores são mostrados juntamente com a quantidade total de ciclos na Tabela 18.

| Grafo | *OM* | Quantidade total de ciclos |
|---|---|---|
| Shopping Guararapes | 1000 | 15000 |
| Ruas do Recife | 1300 | 19500 |
| Cinco regiões pouco conectadas | 1200 | 18000 |
| Cinco regiões bem conectadas | 1000 | 15000 |

**Tabela 18 - Cálculo da quantidade total de ciclos das simulações**

No começo da simulação, consideramos que todas as ociosidades instantâneas dos nós são zero, como se eles tivessem sido visitados no momento inicial. Entretanto essa é uma consideração irreal, pois os nós não são todos visitados no momento inicial, apenas aqueles em que os agentes estão localizados inicialmente. Conseqüentemente, no início da simulação, a ociosidade instantânea do grafo tende a ser pequena. Para que esses valores pequenos da ociosidade não influenciem significativamente no valor final da ociosidade média, torna-se necessário realizar as simulações por uma quantidade muito grande de iterações, o que inviabiliza a execução dos experimentos.

Dessa forma, os experimentos podem ser divididos em duas fases: *fase transitória* e *fase não-transitória*. Na primeira, o valor da ociosidade vai aumentando (inicialmente zero) até alcançar valores não influenciados pelo estado inicial imposto. Na segunda, na qual não ocorre mais a influência do estado inicial, os valores são decorrentes do comportamento das arquiteturas. Na Figura 13, mostramos a evolução da ociosidade média na simulação de 5 agentes do tipo *Heuristic Conscientious Reactive Agent 0.6* patrulhando o grafo das ruas do Recife (ociosidade no eixo y e ciclos no eixo x). A fase não-transitória se inicia a partir do ciclo 850, o tempo de exploração do grafo.

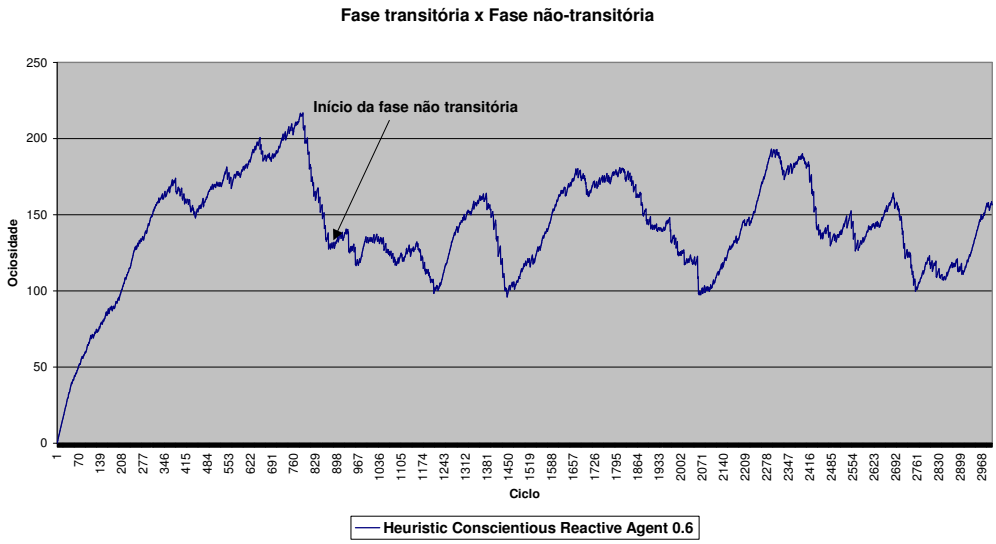


**Figura 13 - Ilustração da transição entre as fases transitória e não-transitória**

A ociosidade média do grafo é apenas medida durante a fase não-transitória. Torna-se necessário então determinar quando termina a fase transitória e se inicia a fase não-transitória. Consideramos que a transição entre uma fase e outra ocorre no momento em que todos os nós do grafo são visitados ao menos uma vez, ou seja, quando o grafo é explorado. A quantidade de ciclos necessária para isso acontecer é exatamente o tempo de exploração do grafo definido na Seção 3.2.1.

Contudo, arquiteturas aleatórias podem não conseguir explorar todo o grafo e, por conseqüência, não possuem um tempo de exploração. Arquiteturas do tipo *Heuristic* com taxa de ociosidade próxima a zero também apresentam o mesmo problema, pois os agentes tendem a ficar presos em suas posições iniciais, não visitando os demais nós do grafo.

Além disso, o tempo de exploração varia de arquitetura para arquitetura. Logo, considerar o início da fase não-transitória no tempo de exploração de cada arquitetura comprometeria a comparação entre os experimentos, visto que teríamos que comparar arquiteturas em que a ociosidade foi medida em intervalos de tempo diferentes (diferentes durações para a fase não-transitória).

Para resolver esse problema, foi escolhido um valor aproximado para o tempo de exploração (*TE*) que garantisse, para a maioria dos casos, que a transição já teria ocorrido. A fim de escolher esse valor, foram executados outros testes preliminares com todas as 122 arquiteturas patrulhando cada um dos quatro grafos e considerando todas as populações

descritas na Tabela 17. Cada um desses testes durou a quantidade de ciclos indicada na Tabela 18. Para cada grafo, foi escolhido um valor para *TE* que fosse superior à maioria dos valores obtidos nesses experimentos iniciais. Esses valores são mostrados na Tabela 19.

| Grafo | *TE* |
|---|---|
| Shopping Guararapes | 800 |
| Ruas do Recife | 1200 |
| Cinco regiões pouco conectadas | 800 |
| Cinco regiões bem conectadas | 700 |

**Tabela 19 - Cálculo do tempo de transição entre as duas fases das simulações**

## 5.5 CONCLUSÃO

Nesse capítulo foi mostrado o método experimental empregado para avaliar as diferentes arquiteturas propostas. Inicialmente descrevemos os grafos utilizados nas simulações. Em seguida, foi feita uma explicação breve do funcionamento do simulador e que mudanças foram necessárias fazer a ele a fim de dar suporte aos novos agentes. Por último, descrevemos que experimentos foram realmente executados.

No próximo capítulo, apresentaremos os resultados das simulações e faremos uma discussão a respeito deles.

# 6 RESULTADOS EXPERIMENTAIS

*"A paciência é amarga, mas seu fruto é doce".*

*(Jean Jacques Rousseau)*

Este capítulo delineia os experimentos realizados com o simulador que foi apresentado anteriormente. Em primeiro lugar, serão mostrados os resultados e, por último, uma discussão sobre os mesmos.

## 6.1 RESULTADOS

Nos gráficos que se seguem, cada tipo de linha representa uma arquitetura distinta, conforme descrito na legenda. As linhas tracejadas representam agentes com estratégia de escolha do próximo nó heurística baseada na ociosidade e na distância ou com *path-finding* que considera a ociosidade dos nós intermediários. As linhas contínuas indicam agentes com tomada de decisão e *path-finding* simples. Essa convenção de linhas será utilizada por toda a exposição de resultados gráficos.

Nas legendas dos gráficos, adotamos as seguintes abreviações: Bla – Blackboard, Cog – Cognitive, Con – Conscientious, Coo – Coordinated, Fla – Flags, Heu – Heuristic, Path – Pathfinder, Ran – Random e Rea – Reactive. O número real à direita das abreviações dos agentes *Heuristic* e *Pathfinder* corresponde à taxa de ociosidade.

As taxas de ociosidade escolhidas para serem mostradas nos gráficos foram aquelas que apresentaram um melhor desempenho entre as taxas testadas. Os resultados serão mostrados em quatro subseções, uma para cada grafo.

### 6.1.1 SHOPPING

A Figura 14 e a Figura 15 nos mostram o impacto que levar em conta a distância na tomada de decisão causa na ociosidade média do grafo.

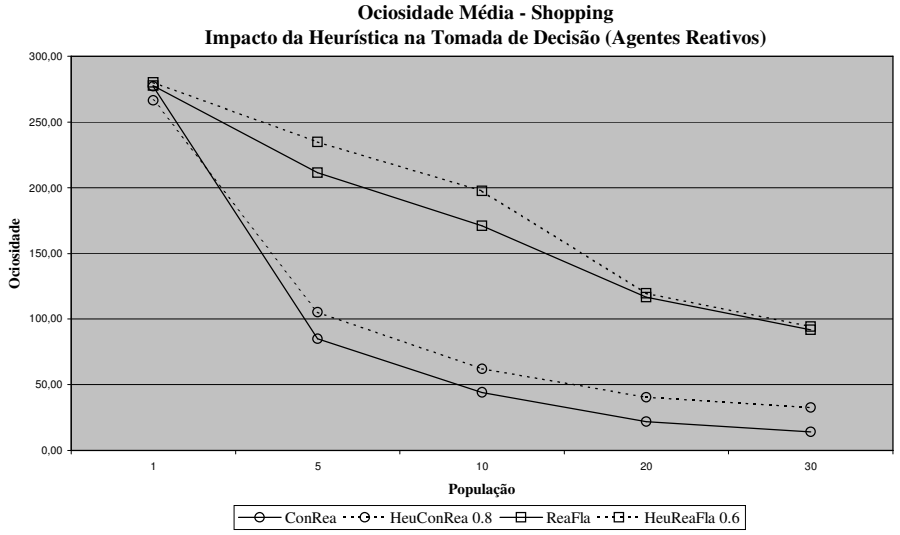


**Figura 14 - Comparação da ociosidade média dos agentes *Heuristic Reactive* com suas versões mais simples no grafo do Shopping**

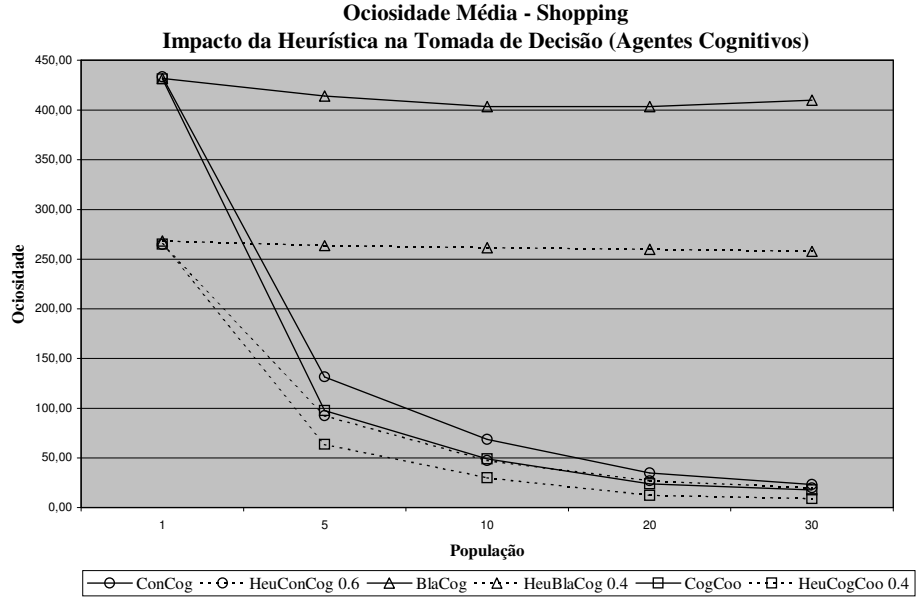


**Figura 15 - Comparação da ociosidade média dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo do Shopping**

Ao observarmos o desempenho dos agentes reativos[4], podemos observar que a ociosidade não diminui com a adição de heurística baseada na ociosidade e na distância. Isso acontece porque o grafo possui uma baixa variância nos pesos das arestas. Dessa forma, dado um nó do grafo, todas as suas arestas incidentes possuem praticamente o mesmo peso. Por conseguinte, a distância não vai ter uma influência positiva na escolha do próximo nó feita pelos agentes.

Já os agentes cognitivos conseguem um ganho significativo de performance quando utilizam uma tomada de decisão mais avançada (cerca de 30%, chegando a 50% para os agentes coordenados).

Ao compararmos a arquitetura *Conscientious Reactive* com a *Conscientious Cognitive*, observamos que as arquiteturas que tinham uma abordagem local (*Conscientious Reactive*) tiveram melhores resultados nesse tipo de grafo em relação aos agentes cognitivos equivalentes (*Conscientious Cognitive).* A melhoria foi entre 35% e 40%. Estas observações mostram que estratégias locais dão bons resultados para mapas onde o acesso a determinadas regiões seja difícil, ou seja, onde existam poucos caminhos entre as regiões do grafo. De fato, na estratégia local, os agentes tendem a permanecer por mais tempo separados nas diversas regiões, o que é uma vantagem para este tipo de mapa.

Podemos observar também que os agentes que se comunicam sem coordenação (*Reactive with Flags* e *Blackboard Cognitive*) tem pior desempenho que suas versões sem comunicação (*Conscientious Reactive* e *Conscientious Cognitive*). Existindo uma estrutura de dados compartilhada e sem coordenação, esses agentes tendem a escolher os mesmos nós para visitar, indo para os mesmos lugares ao mesmo tempo.

Por último, quando os agentes coordenados (*Cognitive Coordinated*) são comparados com os agentes correspondentes sem coordenação (*Conscientious Cognitive*), pode-se constatar que a coordenação melhora o desempenho dos agentes. De fato, os agentes coordenados foram os que apresentaram o melhor desempenho entre todos os agentes testados.

A Figura 16 nos mostra o impacto que levar em conta a ociosidade no *path-finding* causa na ociosidade média do grafo.

[4] Convém lembrar que agentes reativos escolhem como próximo nó, um dos nós vizinhos de onde ele está localizado no momento.

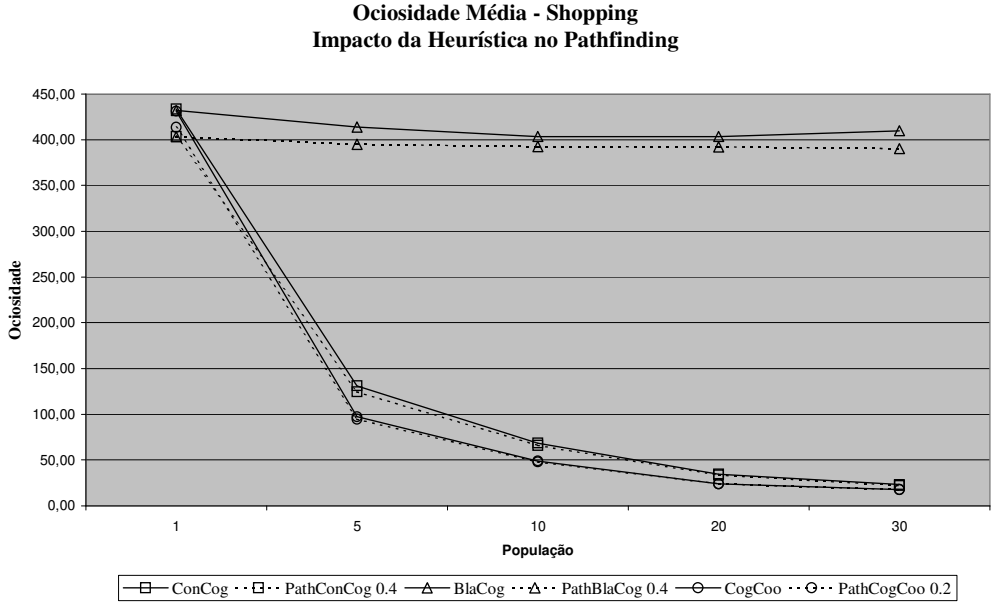


**Figura 16 - Comparação da ociosidade média dos agentes *Pathfinder* com suas versões mais simples no grafo do Shopping**

Como podemos observar na figura, os agentes *Pathfinder* também obtêm um melhor desempenho em relação aos agentes que usam *path-finding* baseado apenas no menor caminho. Entretanto, o ganho de performance é de apenas 5% em média. Isso se deve ao fato de que o grafo é pouco conexo, existindo poucas opções de caminhos entre dois nós distintos. Dessa forma, os agentes *Pathfinder* não possuem muitas opções de escolha para melhorar o desempenho.

Em relação ao critério ociosidade máxima, a Figura 17 e a Figura 18 nos evidenciam as alterações que ocorrem quando se considera a distância na tomada de decisão.

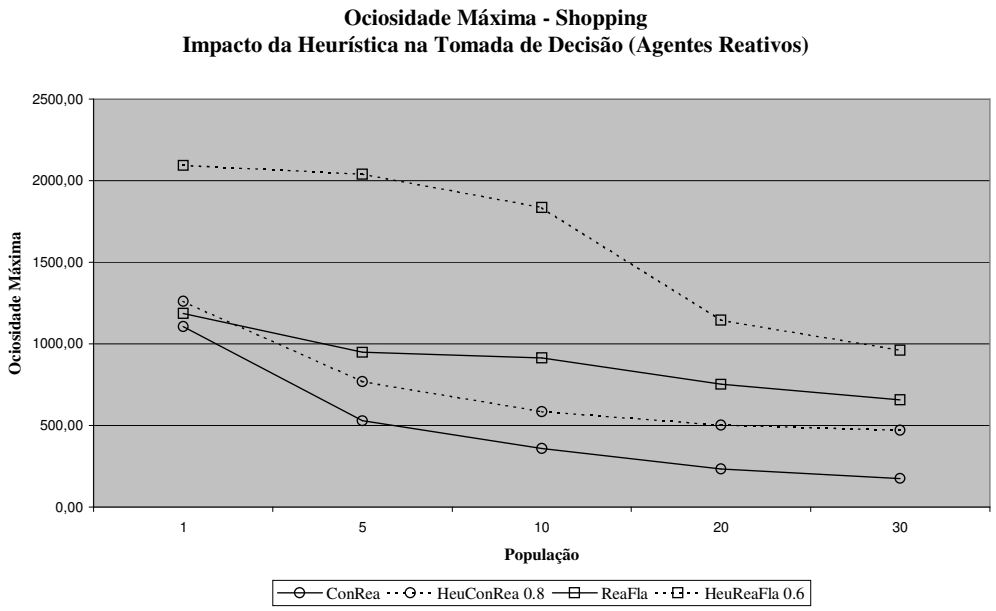


**Figura 17 - Comparação da ociosidade máxima dos agentes *Heuristic Reactive* com suas versões mais simples no grafo do Shopping**

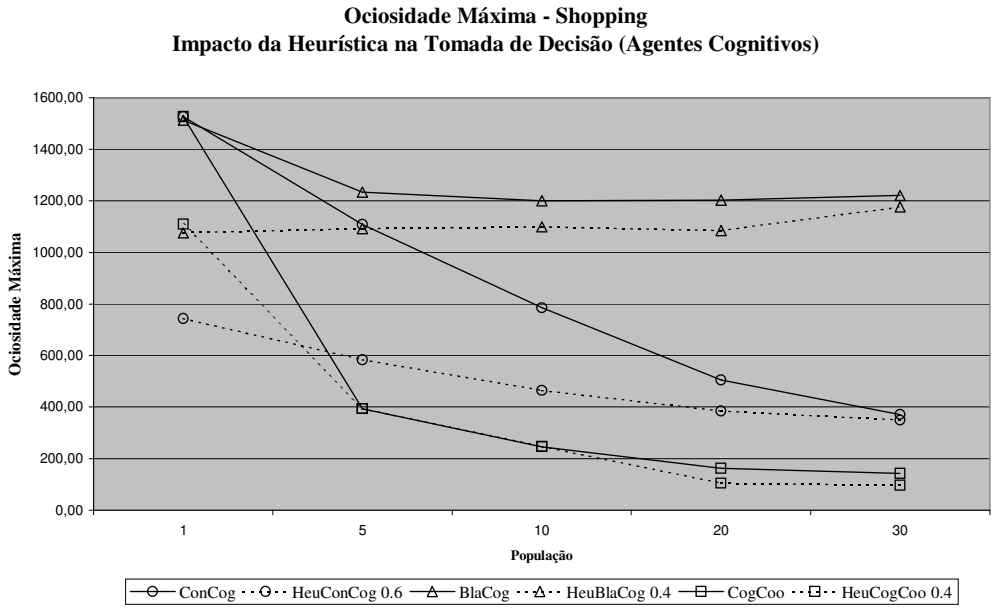


**Figura 18 - Comparação da ociosidade máxima dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo do Shopping**

Por último, a Figura 19 indica as alterações no resultado da ociosidade máxima do grafo por se levar em conta a ociosidade no *path-finding*.

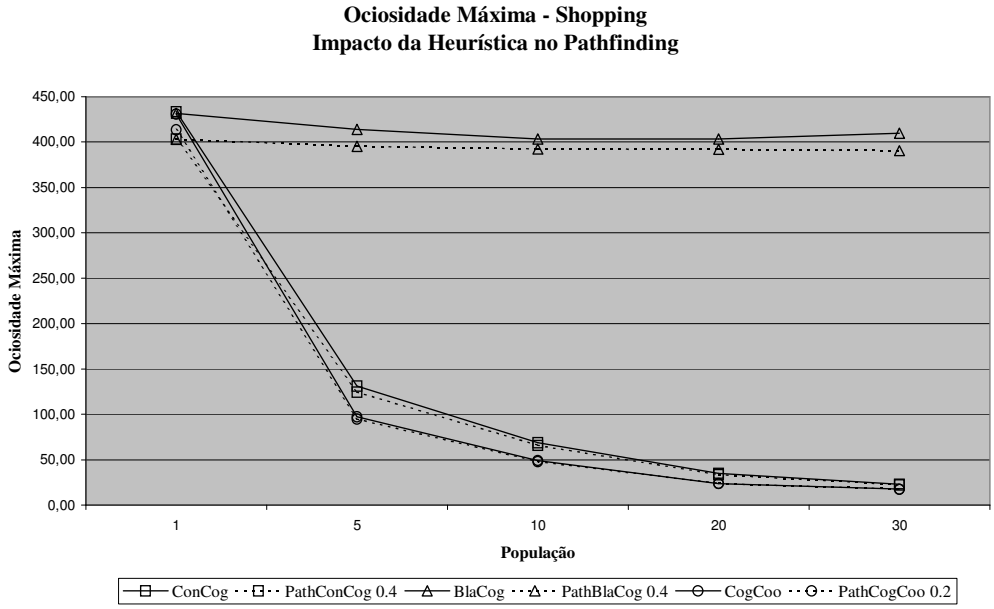


**Figura 19 - Comparação da ociosidade máxima dos agentes *Pathfinder* com suas versões mais simples no grafo do Shopping**

No que diz respeito à ociosidade máxima, constata-se o mesmo comportamento obtido no critério ociosidade média: o desempenho dos agentes reativos não é melhorado ao contrário do que ocorre com os agentes cognitivos. Similarmente, os agentes *Pathfinder* obtêm um melhor desempenho em relação aos agentes que usam *path-finding* baseado apenas no menor caminho.

### 6.1.2 Ruas de Recife

A Figura 20 e a Figura 21 nos mostram o impacto que levar em conta a distância na tomada de decisão causa na ociosidade média do grafo.

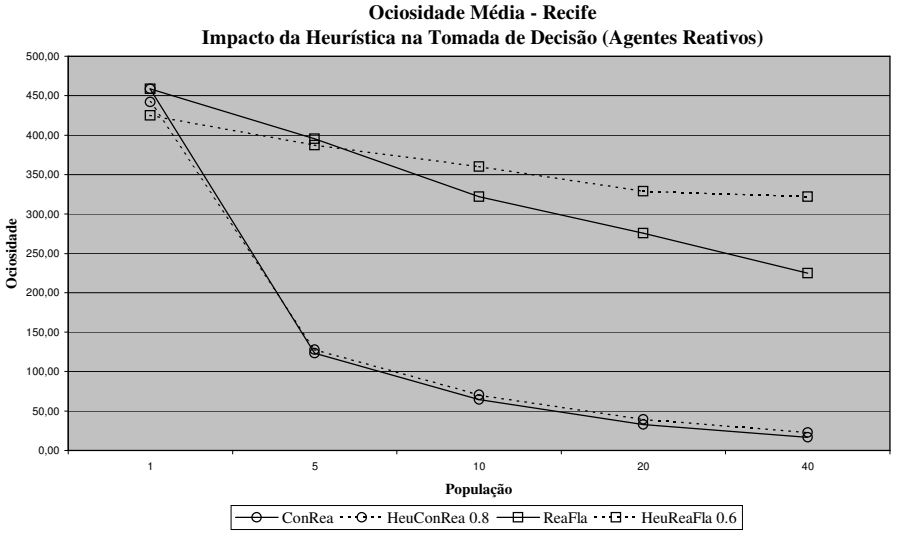


**Figura 20 - Comparação da ociosidade média dos agentes *Heuristic Reactive* com suas versões mais simples no grafo do Recife**

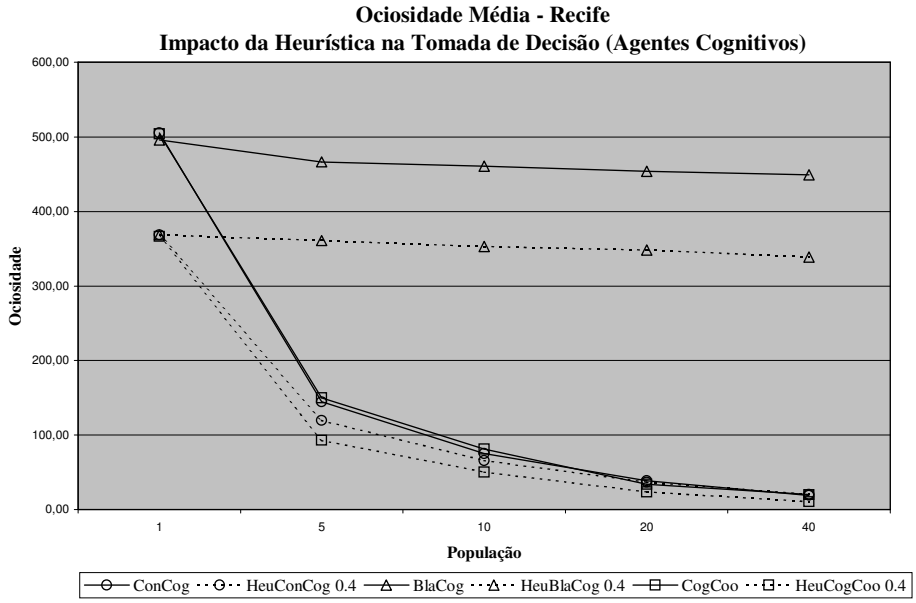


**Figura 21 - Comparação da ociosidade média dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo do Recife**

Tal como no grafo do Shopping, a ociosidade não diminui com a adição de heurística baseada na ociosidade e na distância para os agentes heurísticos. Entretanto, a diferença entre os agentes *Heuristic Reactive* e os reativos puros foi menor que no caso anterior, pois esse grafo apresenta uma variância um pouco maior nos pesos das arestas.

Assim como no primeiro grafo, os agentes cognitivos conseguem um ganho significativo de performance quando utilizam uma tomada de decisão mais avançada (cerca de 25%). Ao comparar 40 agentes do tipo *Heuristic Cognitive Coordinated* com 40 agentes *Cognitive Coordinated*, pode-se notar uma melhoria de quase 50%.

As arquiteturas com abordagem local ainda possuem melhores resultados nesse tipo de grafo em relação aos agentes cognitivos. No entanto, a melhora ficou apenas entre 15% e 20%, pois esse grafo é bem mais conectado que o grafo anterior.

Podemos observar também que os agentes que se comunicam sem coordenação (*Reactive with Flags* e *Blackboard Cognitive*) têm pior desempenho que suas versões sem comunicação (*Conscientious Reactive* e *Conscientious Cognitive*).

Conforme mencionado na introdução da Seção 6.1, as taxas de ociosidade dos gráficos são as de melhor desempenho entre as taxas testadas. Uma vez que utilizamos a arquitetura *Heuristic Conscientious Cognitive 0.6* no primeiro grafo e a arquitetura *Heuristic Conscientious Cognitive 0.4* no segundo (ver os gráficos da Figura 15 e Figura 21), não necessariamente a mesma taxa de ociosidade vai ter o melhor desempenho para todos os grafos.

Por último, quando os agentes coordenados (Cognitive Coordinated) são comparados com os agentes correspondentes sem coordenação (Conscientious Cognitive), pode-se constatar que a coordenação melhora o desempenho dos agentes.

A Figura 22 nos mostra o impacto que levar em conta a ociosidade no *path-finding* causa na ociosidade média do grafo.

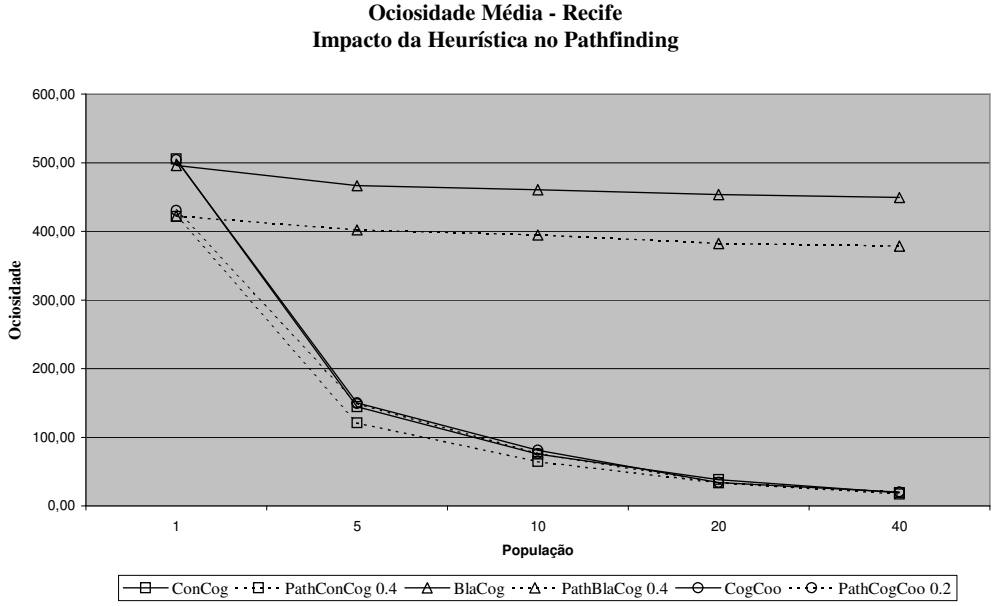


**Figura 22 - Comparação da ociosidade média dos agentes *Pathfinder* com suas versões mais simples no grafo do Recife**

Como podemos observar na figura, os agentes *Pathfinder* também obtêm um melhor desempenho em relação aos agentes que usam *path-finding* baseado apenas no menor caminho. O ganho de performance é de 10% em média, o dobro do grafo da seção anterior. A percentagem é maior para o grafo do Recife, pois ele é mais conectado que o do Shopping.

Em relação ao critério ociosidade máxima, a Figura 23 e a Figura 24 nos evidenciam as alterações que ocorrem quando se considera a distância na tomada de decisão.

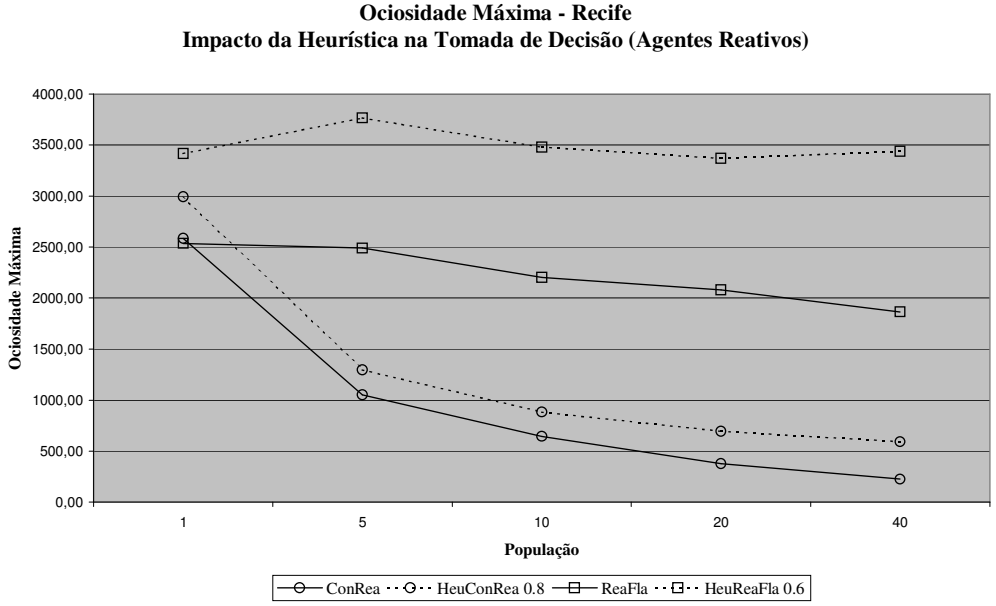


**Figura 23 - Comparação da ociosidade máxima dos agentes *Heuristic Reactive* com suas versões mais simples no grafo do Recife**

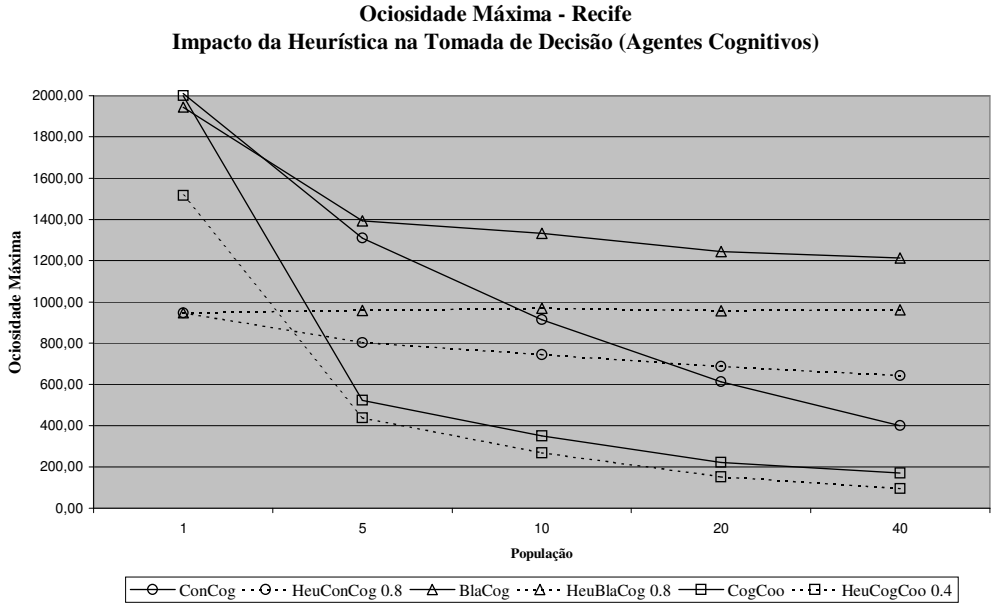


**Figura 24 - Comparação da ociosidade máxima dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo do Recife**

Por último, a Figura 25 indica as alterações no resultado da ociosidade máxima do grafo por se levar em conta a ociosidade no *path-finding*.

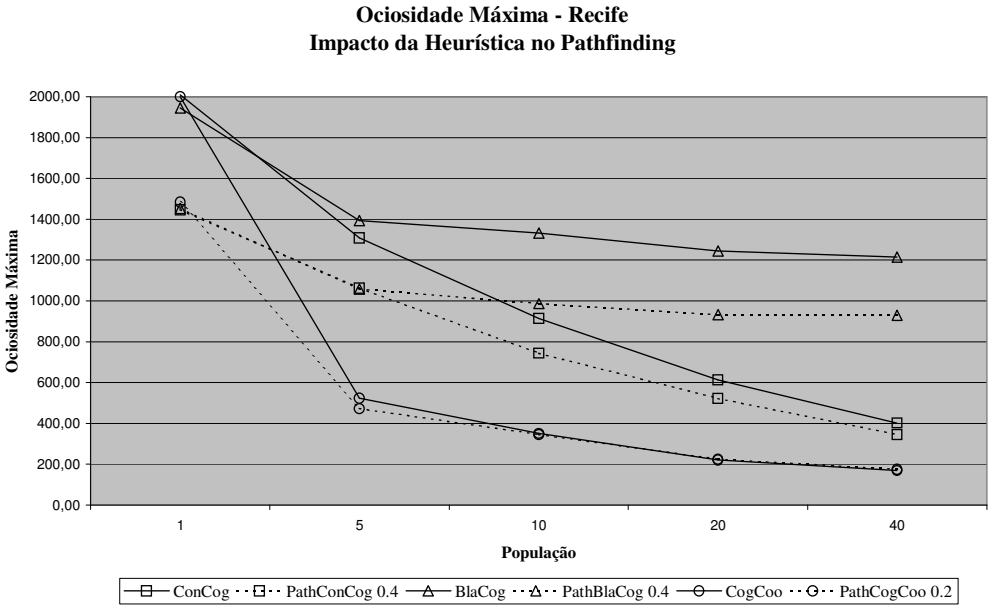


**Figura 25 - Comparação da ociosidade máxima dos agentes *Pathfinder* com suas versões mais simples no grafo do Recife**

No que diz respeito à ociosidade máxima, constata-se um comportamento parecido com o obtido no critério ociosidade média: o desempenho dos agentes reativos não é melhorado ao contrário do que ocorre com os agentes cognitivos. Similarmente, os agentes *Pathfinder* obtêm um desempenho ainda melhor em relação aos agentes que usam *path-finding* baseado apenas no menor caminho (20% em média de redução da ociosidade máxima, ao passo que apenas 10% de redução na ociosidade média).

Se a Figura 21 e a Figura 24 forem comparadas, será observado que uma taxa de ociosidade de 40% obtém melhor resultado para a arquitetura *Heuristic Conscientious Cognitive* no critério ociosidade média. Já considerando o critério ociosidade máxima, convém usar uma taxa de ociosidade de 80% para a mesma arquitetura. Situação similar acontece com a arquitetura *Heuristic Blackboard Cognitive*. Como já observado, não necessariamente a melhor taxa de ociosidade para o critério ociosidade média será a mais adequada quando se deseja otimizar a ociosidade máxima.

É interessante notar que todos os resultados obtidos com esse grafo foram piores que os observados no grafo anterior. A ociosidade média e a ociosidade máxima foram maiores no

mapa do Recife porque o peso médio das arestas é maior. Portanto, os agentes passam mais tempo no deslocamento entre os nós que no mapa do Shopping.

### 6.1.3 Regiões Pouco Conectadas

Inicialmente, a Figura 26 e a Figura 27 ilustram o efeito na ociosidade média do grafo pouco conectado da Figura 11 ao se utilizar uma escolha do próximo nó heurística baseada não apenas na ociosidade, mas também na distância.

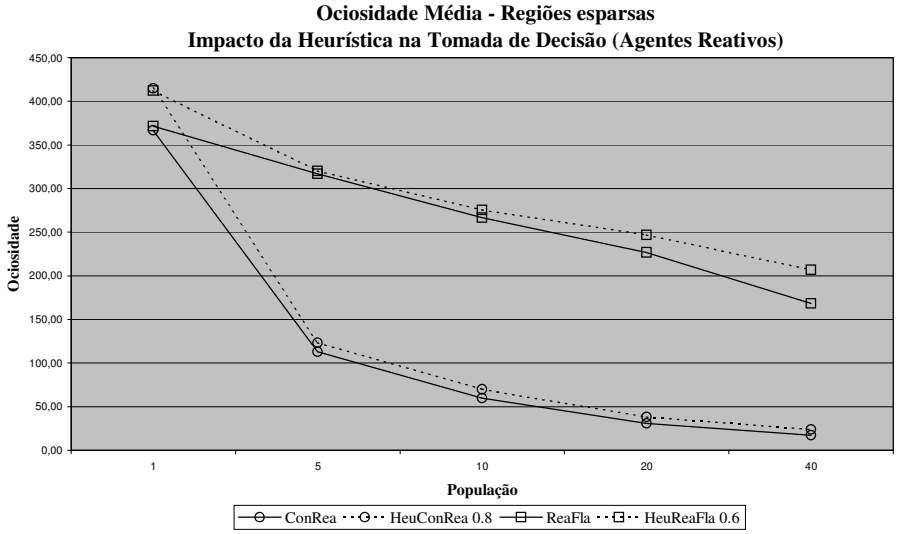


**Figura 26 - Comparação da ociosidade média dos agentes *Heuristic Reactive* com suas versões mais simples no grafo das regiões pouco conectadas**

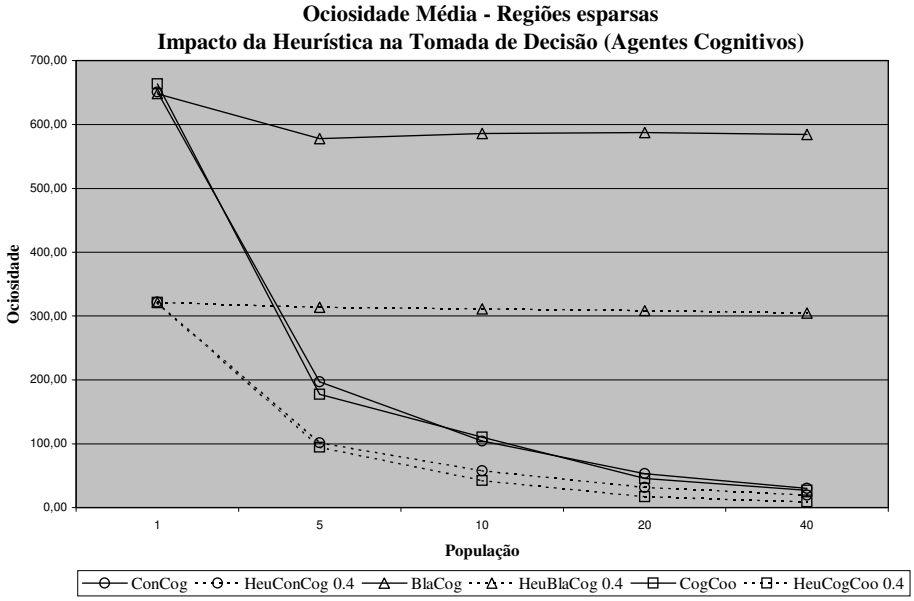


**Figura 27 - Comparação da ociosidade média dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo das regiões pouco conectadas**

Os agentes do tipo *Heuristic Reactive* continuam apresentando uma ociosidade média maior que suas versões mais simples. Todavia, a diferença é a menor entre os três grafos já estudados, haja vista que entre eles esse é o grafo que apresenta a maior variância nos pesos das arestas.

Assim como nos demais grafos, arquiteturas do tipo *Heuristic Cognitive* obtêm melhores performance que as *Cognitive*. O ganho de performance é o mais acentuado nesse grafo esparso: 50% em média, alcançando valores de 65% para agentes coordenados. A utilização da distância na escolha do próximo nó permite aos agentes ficarem localizados numa mesma região.

Considerando os três grafos já descritos, esse é o que os agentes reativos mais se destacam em relação aos agentes cognitivos simples. A média de ganho de desempenho é de 45%. De fato, o grafo das regiões pouco conectadas é o que apresenta a menor possibilidade de caminhos entre as diversas regiões (grafo mais esparso).

A Figura 28 nos mostra o impacto que levar em conta a ociosidade no *path-finding* causa na ociosidade média do grafo.

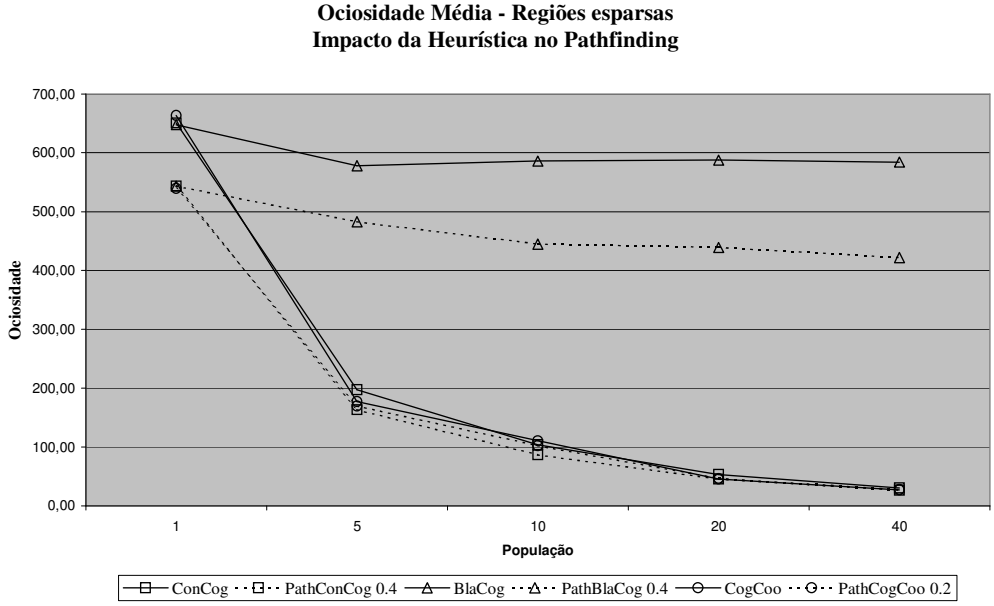


**Figura 28 - Comparação da ociosidade média dos agentes *Pathfinder* com suas versões mais simples no grafo das regiões pouco conectadas**

Como podemos observar na figura, os agentes *Pathfinder* também obtêm um melhor desempenho em relação aos agentes que usam *path-finding* baseado apenas no menor caminho. O ganho de performance é cerca de 20%, o maior de todos os grafos já vistos. A princípio poderia se estranhar esse resultado, pois as regiões são pouco conectadas, o que diminuiria a quantidade de caminhos a serem considerados pelo *Pathfinder*. Entretanto, dentro de uma mesma região, existem muitos caminhos interligando os vários nós. Por isso, esse grafo é um bom cenário de aplicação aos agentes *Pathfinder*.

Finalmente, a Figura 29, a Figura 30 e a Figura 31 trazem os resultados para o critério ociosidade máxima.

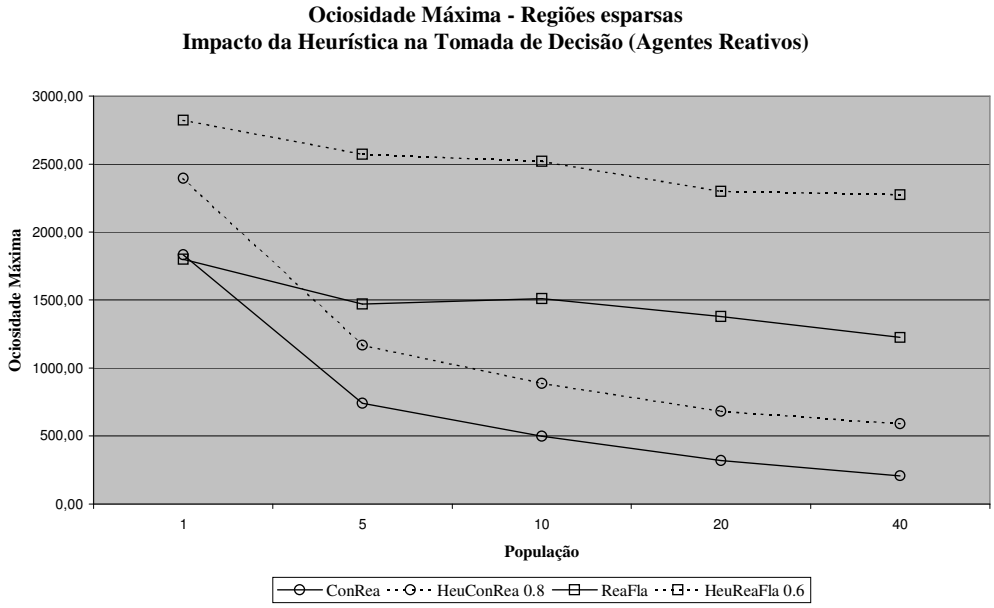


**Figura 29 - Comparação da ociosidade máxima dos agentes *Heuristic Reactive* com suas versões mais simples no grafo das regiões pouco conectadas**

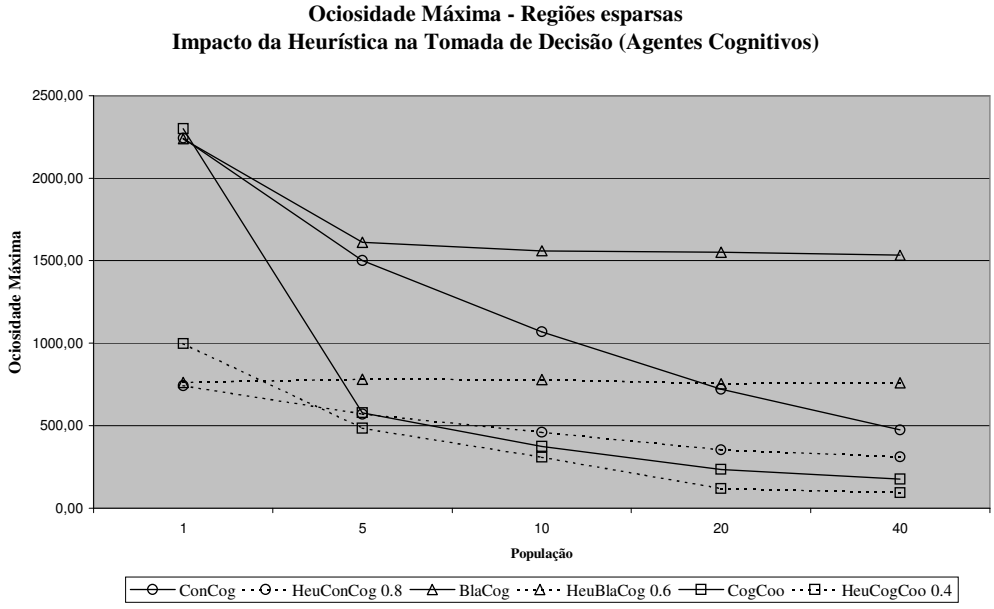


**Figura 30 - Comparação da ociosidade máxima dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo das regiões pouco conectadas**

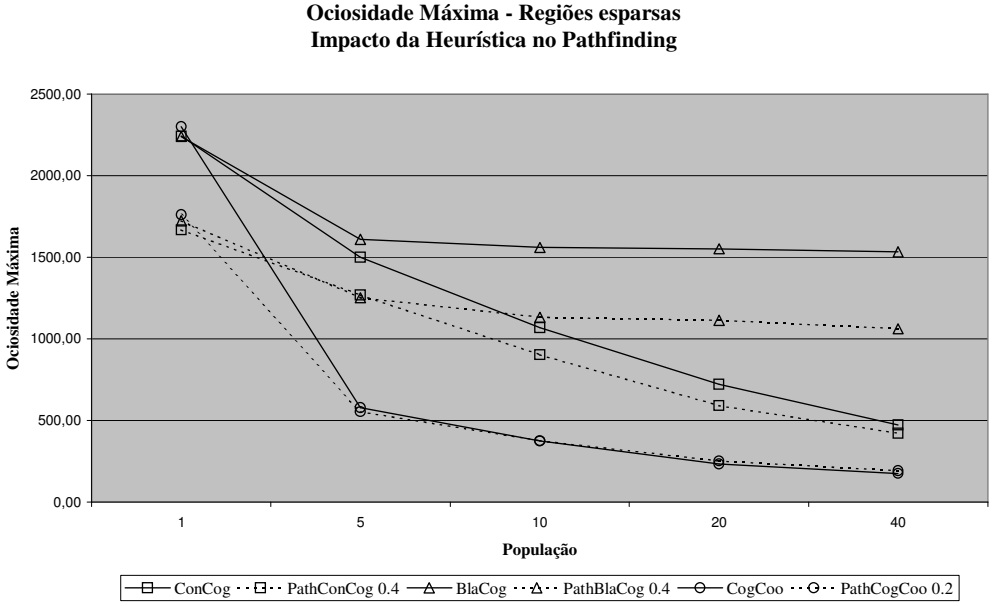


**Figura 31 - Comparação da ociosidade máxima dos agentes *Pathfinder* com suas versões mais simples no grafo das regiões pouco conectadas**

Como pode ser visto nos gráficos acima, o comportamento no critério ociosidade máxima é bastante semelhante ao que já foi discutido no critério ociosidade média.

Embora o grafo avaliado nessa seção possua um peso médio das arestas bem próximo ao grafo do Recife, os agentes mais sofisticados conseguem se distribuir mais uniformemente no grafo das regiões e, conseqüentemente, atingir ociosidades bem menores. Entretanto, o poder de distribuição ao longo do grafo não é suficiente para obter melhor desempenho em relação ao grafo do Shopping, pois a diferença entre a média do peso das arestas é maior quando comparamos o grafo das regiões pouco conectadas com o grafo do Shopping.

### 6.1.4 Regiões Bem Conectadas

A Figura 32 e a Figura 33 comparam a ociosidade média dos agentes *Heuristic* com suas versões mais simples no grafo das regiões bem conectadas (Figura 11).

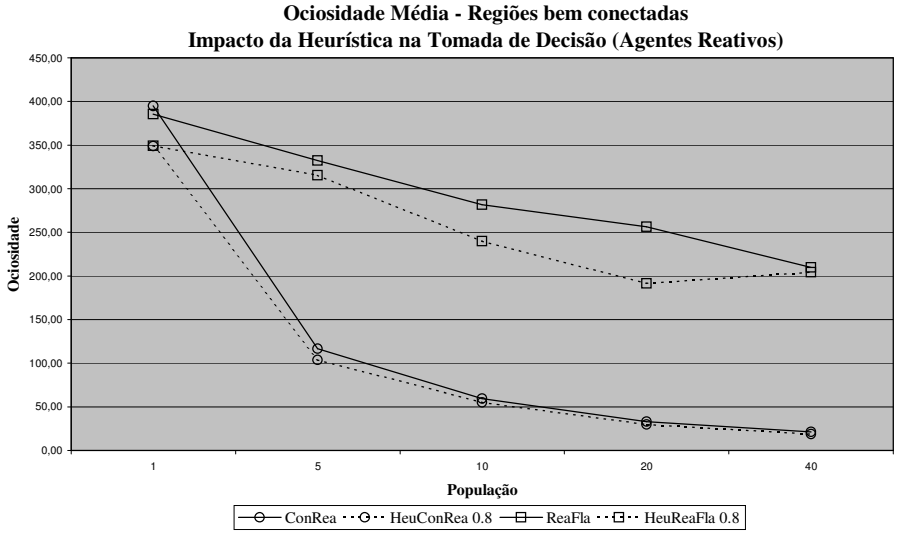


**Figura 32 - Comparação da ociosidade média dos agentes *Heuristic Reactive* com suas versões mais simples no grafo das regiões bem conectadas**

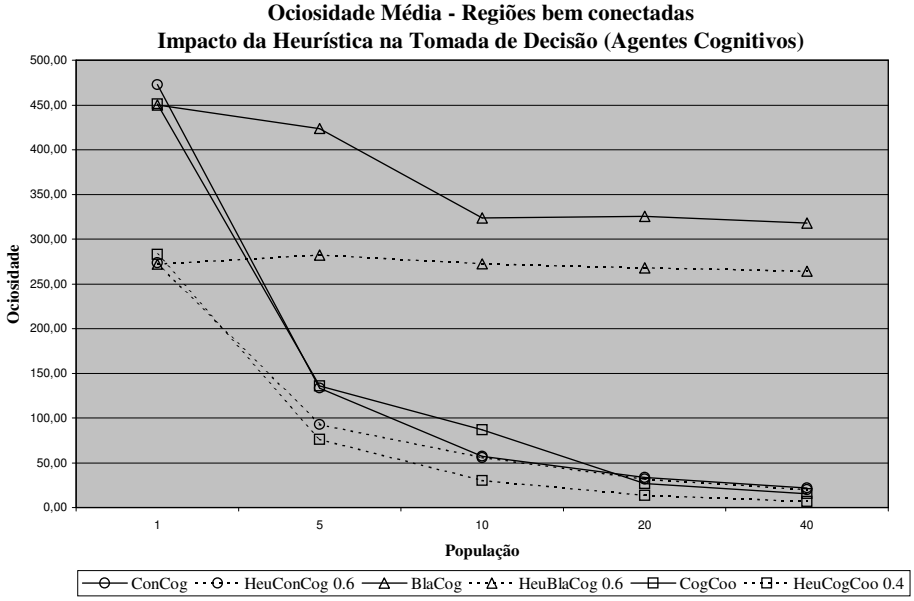


**Figura 33 - Comparação da ociosidade média dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo das regiões bem conectadas**

Diferentemente dos grafos anteriores, os agentes do tipo *Heuristic Reactive* tiveram uma ociosidade média menor que suas versões mais simples. Isso se deveu ao fato de que esse grafo é bastante conectado e com variância alta. Conseqüentemente, os agentes reativos conseguem distinguir bem os seus nós vizinhos no processo de tomada de decisão.

Assim como nos demais grafos, arquiteturas do tipo *Heuristic Cognitive* obtêm melhores performance que as *Cognitive*. O ganho de performance é o menor entre todos os grafos por se tratar de um grafo bem conectado: 20% em média, alcançando valores de 65% para agentes coordenados.

Outra diferença que pode ser notada nesse grafo é o fato de agentes reativos não possuírem mais um desempenho melhor que os agentes cognitivos simples. Como dito anteriormente, uma menor possibilidade de caminhos entre as diversas regiões do grafo favorecem os agentes reativos, o que não é o caso do grafo de regiões bem conectadas.

A Figura 34 nos mostra o impacto que levar em conta a ociosidade no *path-finding* causa na ociosidade média do grafo.

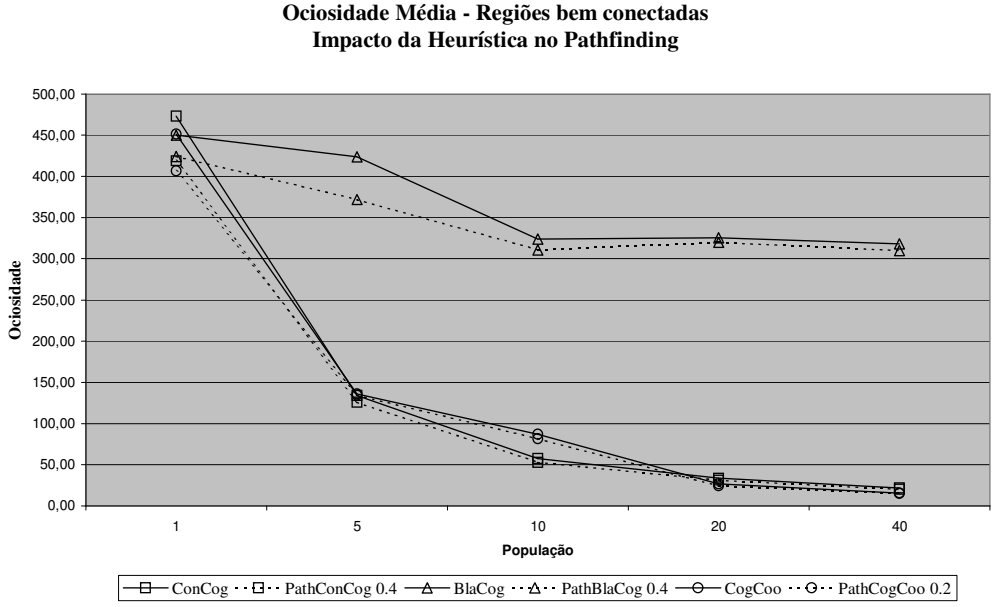


**Figura 34 - Comparação da ociosidade média dos agentes *Pathfinder* com suas versões mais simples no grafo das regiões bem conectadas**

Como podemos observar na figura, os agentes *Pathfinder* também obtêm em geral um melhor desempenho em relação aos agentes que usam *path-finding* baseado apenas no menor caminho. O ganho de performance é cerca de 5%.

Finalmente, a Figura 35, a Figura 36 e a Figura 37 trazem os resultados para o critério ociosidade máxima.

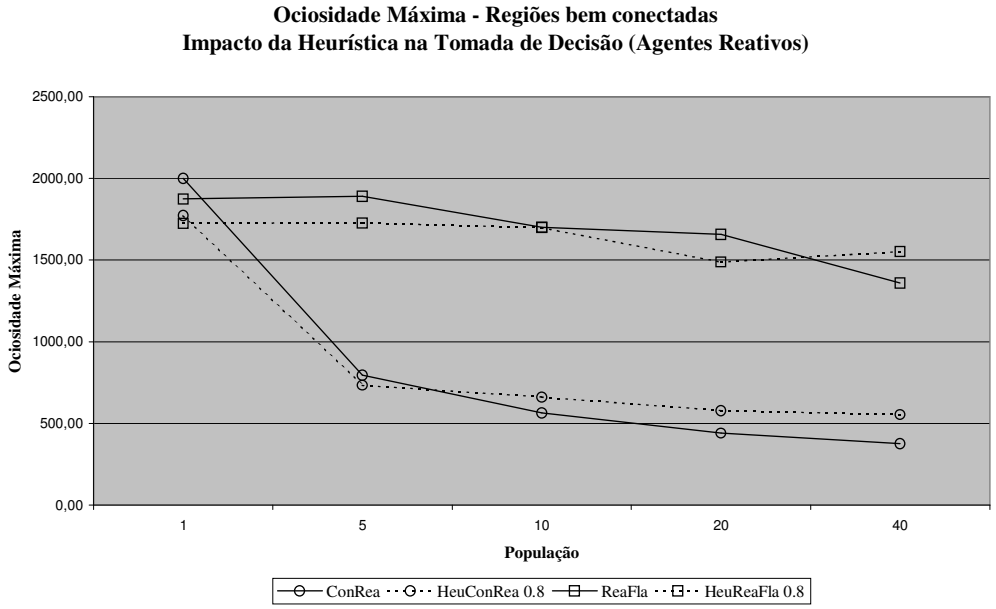


**Figura 35 - Comparação da ociosidade máxima dos agentes *Heuristic Reactive* com suas versões mais simples no grafo das regiões bem conectadas**

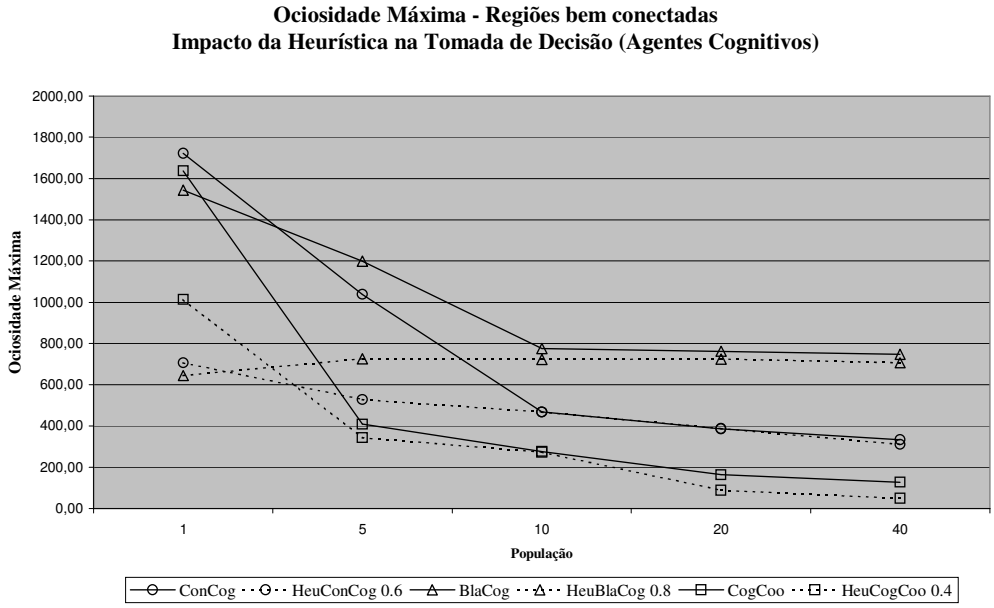


**Figura 36 - Comparação da ociosidade máxima dos agentes *Heuristic Cognitive* com suas versões mais simples no grafo das regiões bem conectadas**

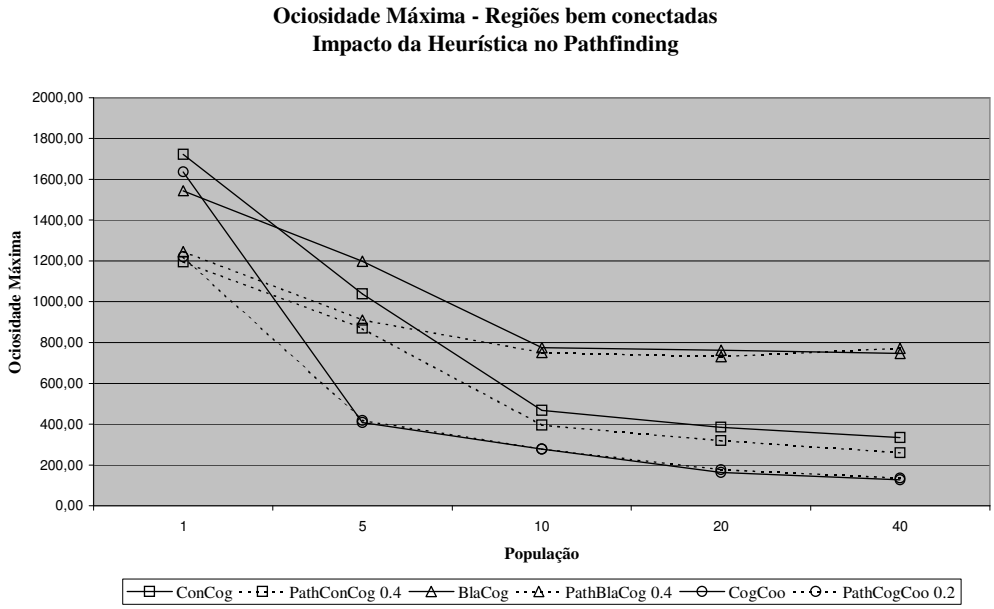


**Figura 37 - Comparação da ociosidade máxima dos agentes *Pathfinder* com suas versões mais simples no grafo das regiões bem conectadas**

Mais uma vez os resultados são bem parecidos quando comparamos os critérios ociosidade máxima e ociosidade média.

## 6.2 DISCUSSÃO

Os experimentos realizados puderam confirmar as hipóteses levantadas. À exceção dos agentes reativos, arquiteturas que combinam distância e ociosidade na tomada de decisão são sempre melhores que suas versões correspondentes que consideram apenas a ociosidade. Pudemos observar que quanto mais esparso e particionado o grafo, melhor será o desempenho dos agentes *Heuristic Cognitive*, pois eles tenderão a se dividirem entre as partições, onde ficarão localizados durante toda a simulação. Já para os agentes *Heuristic Reactive*, quanto mais conectado o grafo, melhor sua performance (eles conseguem ser melhor que os *Conscientious Reactive* para o grafo das regiões bem conectadas).

Os agentes *Pathfinder* também alcançaram melhores resultados que suas versões mais simples que consideram apenas o menor caminho. Em geral, eles são mais beneficiados (apresentam maior ganho) quando os grafos são mais conectados, existindo uma maior quantidade de escolhas de caminhos entre os nós.

Convém ressaltar que não necessariamente a melhor taxa de ociosidade para a ociosidade média será também a melhor no critério de ociosidade máxima. Da mesma forma, grafos diferentes podem requerer taxas diferentes.

Agentes reativos dão bons resultados para ambientes onde existem poucos caminhos entre as regiões do grafo. De fato, os agentes tendem a permanecer por mais tempo separados nas diversas regiões, o que é uma vantagem.

Os agentes *BlackBoard Cognitive* e *Reactive with Flags* tenderam a visitar o mesmo nó ao mesmo momento. Isto aconteceu porque existiu o compartilhamento da informação e não existiu nenhum mecanismo de coordenação que ordenasse as ações que os agentes deveriam executar. Conseqüentemente, esse grupo obteve praticamente os mesmos resultados para qualquer população, como se existisse apenas um agente no ambiente. Para solucionar esse problema, a coordenação (agentes *Cognitive Coordinated*) pode ser aplicada.

Ao combinar os agentes do tipo *Heuristic* com os *Pathfinder*, e utilizando comunicação via coordenador central, consegue-se a melhor arquitetura de todas em todos os critérios: *Heuristic Pathfinder Cognitive Coordinated*.

Comentário:

Apesar de não mostrados nos gráficos, os agentes aleatórios (Random Reactive e Random Coordinator), obtiveram um desempenho insatisfatório, tal como ocorreu em nossos

trabalhos anteriores. Entretanto, uma quantidade grande de agentes aleatórios tende a se distribuir em longo prazo pelo grafo, levando a resultados relativamente bons.

Os resultados são bastante semelhantes nos dois critérios empregados (ociosidade média e ociosidade máxima).

Em ambientes em que a comunicação não é possível, em geral é melhor utilizar agentes reativos (*Conscientious Reactive*). Entretanto, caso se permita comunicação entre os agentes, a arquitetura mais adequada entre as propostas em nossos trabalhos é a *Heuristic Pathfinder Cognitive Coordinated*.

## 6.3 CONCLUSÃO

Estes experimentos mostram algumas diretrizes preliminares para o projeto das arquiteturas de sistemas multiagente para o problema da supervisão. O principal passo é entender o ambiente de aplicação, suas restrições e características. Isto é essencial para determinar:

- As ociosidades média e máxima desejada;
- Se existe possibilidade de comunicação entre os agentes;
- As características do grafo (conectividade, número de nós, variância no peso das arestas, etc);

Conhecendo estas características e restrições do domínio de aplicação e os resultados deste trabalho fica mais fácil determinar qual é a melhor arquitetura para a situação em questão. Para tal basta ver quais as arquiteturas que satisfazem as restrições do problema em questão e escolher aquela que melhor convier.

A discussão exposta na seção 6.2 serve de guia nessa escolha.

# 7 CONCLUSÃO

*"I like the dreams of the future better than the history of the past".*

*(Thomas Jefferson)*

O estudo do problema da patrulha é de extrema importância e relevância na comunidade científica atual, haja vista as diversas situações nas quais ele pode ser encontrado. Os estudos realizados pelo CIn-UFPE na área já renderam a publicação de vários artigos [29], [30] e [2]. Esta pesquisa representou uma continuação mais aprofundada desses estudos.

Essa dissertação possuiu como primeiro objetivo propor uma tipologia inicial para o problema da patrulha, na qual classificamos as aplicações de patrulha em relação à natureza do problema e às características do ambiente a ser patrulhado. Como parte da tipologia, foram repensados os critérios de avaliação das arquiteturas.

Além disso, foram realizados progressos significativos nas arquiteturas multiagente para resolver o problema. Os avanços foram obtidos ao combinar ociosidade e distância tanto na determinação do melhor caminho quanto na escolha do nó a ser visitado. Os resultados experimentais comprovam uma melhora no desempenho das arquiteturas ao se combinar ociosidade e distância.

Adicionalmente, a representação do ambiente a ser patrulhada foi enriquecida ao se utilizarem grafos com pesos, os quais representam mais fielmente os ambientes reais. Dessa forma, quatro grafos foram propostos como *benchmark* de avaliação, nos quais experimentos foram realizados e analisados a fim de comparar as diferentes arquiteturas implementadas. Obtivemos novas conclusões sobre o impacto das características dos grafos no desempenho dos agentes. Por último, mudanças foram realizadas no simulador de patrulha para dar suporte às novas arquiteturas e à nova representação do ambiente.

É importante ressaltar que uma das maiores dificuldades desse trabalho consistiu na execução dos experimentos. Conforme visto anteriormente, foram realizados 4 x 5 x 122 x 50 = 122000 experimentos, correspondendo a 4 grafos, 5 tamanhos de populações, 122 arquiteturas e 50 posições iniciais diferentes. Já que a duração de cada experimento poderia chegar até 30 segundos, as simulações foram divididas entre sete pessoas e executadas por um período de duas semanas.

Como trabalhos futuros, pode-se aumentar ainda mais a complexidade das arquiteturas. Isso inclui características como:

- Pesquisar novas formas de combinar ociosidade e distância na tomada de decisão e no *path-finding*;
- Variar o campo de visão dos agentes, permitindo valores intermediários entre nós vizinhos e o grafo todo;
- Propor algoritmos de partição estática do grafo em regiões às quais os agentes seriam alocados;
- Utilizar informações estatísticas tais como médias das ociosidades dos nós e o histórico de visitas dos agentes a cada nó. Talvez não valha a pena para um agente escolher um nó cuja média de ociosidade seja inferior ao tempo necessário para se chegar a ele, pois quando o agente alcançar o nó, muito provavelmente ele já vai ter sido visitado por outro agente.

Além do mais, grafos dinâmicos e com prioridades diferentes entre seus nós podem ser considerados. Pode-se testar também o comportamento das diversas arquiteturas caso a quantidade de agentes varie ao longo do tempo (sociedades abertas).

Outro possível trabalho futuro seria estudar o problema da detecção de alvos estáticos e móveis. Inicialmente as arquiteturas já implementadas seriam testadas a fim de verificar o comportamento delas na detecção. Eventualmente, novas arquiteturas específicas para a detecção poderiam ser propostas e comparadas com as já desenvolvidas.

Por último, convém ressaltar que duas vertentes também estão sendo estudadas no CIn-UFPE. Dessa forma, existe um trabalho na área de negociação entre os agentes [35] e outro aplicando técnicas de aprendizagem de máquina [38], em especial *Reinforcement Learning* [25], ao problema de patrulha.

# REFERÊNCIAS

[1] Abate, F.R. *The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers*. Oxford University Press, 1996

[2] Almeida, A. L., Castro, P. M. M., Menezes, T. R. et al. Combining Idleness and Distance to Design Heuristic Agents for the Patrolling Task. To appear in: PROCEEDINGS OF SECOND BRAZILIAN WORKSHOP ON GAMES AND DIGITAL ENTERTAINMENT. Salvador, Brasil: 2003

[3] Almeida, A.: Coordenação na Tarefa de Patrulhamento em Sistemas Multiagentes. Trabalho de Graduação - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2002.

[4] Andrade, R. de C., Macedo, H. T., Ramalho, G. L. et al. Distributed Mobile Autonomous Agents in Network Management. In: PROCEEDINGS OF INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS. Las Vegas, USA: 2001. p. 7-12.

[5] Arkin, R. C. Behavior-Based Robot Navigation for Extended Domains. In: ADAPTIVE BEHAVIORS. 1992. v. 1(2), p. 201-225.

[6] Arthur, W. B. Inductive Reasoning and Bounded Rationality (The Bar el Farol Problem). In: AMERICAN ECONOMIC ASSOCIATION ANNUAL MEETINGS. 1994. p. 406-411.

[7] Balch, T., Arkin, R. C. Behavior-Based Formation Control for Multi-robot Teams. In: IEEE TRANSACTIONS ON ROBOT AND AUTOMATION. 1999. v. 20

[8] B Nehe Productions (Open GL). Disponível em: http://nehe.gamedev.net

[9] Blum, A., Chalasani, P., Coppersmith, D. et al. The Minimum Latency Problem. In: PROCEEDINGS OF THE TWENTY-SIXTH ANNUAL ACM SYMPOSIUM ON THE THEORY OF COMPUTING. Montreal, Canada: 1994. p. 163 – 171.

[10] Civilization 3. Disponível em: http://www.civilization3.com/

[11] Cho, J., Garcia-Molina, H., Page, L. Efficient crawling through URL ordering. In: COMPUTERS NETWORKS AND ISDN SYSTEMS. 1998. p. 161-172

[12] 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.

[13] Cormen, T. H., Leiserson, T., Rivest, R. et al. *Algoritmos: Tradução da Segunda Edição Americana*. Campus, 2002. p. 470-474

[14] Counter-Strike. Disponível em: http://www.counter-strike.net/

[15] DeLoura, M. *Game Programming Gems*. 2000

[16] Diablo. Disponível em: http://www.blizzard.com/diablo/

[17] Dorigo, M., Maniezzo, V. and Colorni, A. The Ant System: Optimization by a colony of cooperating agents. In: IEEE TRANSACTIONS ON SYSTEM, MAN AND CYBERNETICS. 1996. v. 26. p. 29-41

[18] Ferber, J. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999. p. 439-445.

[19] FIFA 2003 Online – Soccer Gaming. Disponível em: http://www.fifa2003.com/

[20] FIPA Communicative Act Library Specification. Disponível em: www.fipa.org/specs/fipa00037/SC00037J.html

[21] Grupo de Inteligência Computacional. Disponível em: http://www.cin.ufpe.br/~compint/

[22] Helvig, C. S., Robins, G., Zelikovsky, A. Moving Target TSP and Related Problems. In: PROCEEDINGS 6TH ANNUAL EUROPEAN SYMPOSIUM ON ALGORITHMS. 1998. p. 453-464.

[23] Howland, G. A Practical Guide to Building a Complete Game AI: Volume II. Disponível em: http://www.lupinegames.com/articles/prac_ai_2.html, 1999

[24] Jennings, N.R., Faratin, P., Lomuscio, A.R. et al. Automated Negotiation: Prospects, Methods and Challenges. In: INTERNATIONAL JOURNAL OF GROUP DECISION AND NEGOTIATION. 2001. v. 2. p. 199-215.

[25] Kaelbling, L., Michael, L. and More, A. Reinforcement Learning: A Survey. In: JOURNAL OF ARTIFICIAL INTELLIGENCE RESEARCH 4, 1996. p. 237-285

[26] Labrou, Y., Finnin, T. A semantics approach for KQML, a general purpose communication language for software agents. In: THIRD INTERNATIONAL CONFERENCE OF INFORMATION AND KNOWLEDGE MANAGEMENT. 1994. p. 447-455.

[27] LaValle, S. M., Lin, D., Guibas, L. J. et al. Finding an Unpredictable Target in a Workspace with Obstacles. In: PROCEEDINGS OF THE 1997 INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION. Albuquerque, USA: 1997. p. 737-742.

[28] Macedo, H. T. *Mobilidade, Autonomia e Distribuição em Agentes para o Gerenciamento Corporativo de Sistemas*. Dissertação (Mestrado em Ciência da Computação) - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2001.

[29] 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

[30] 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

[31] 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.

[32] Madeira, C. *FORGEV8: Um framework para o desenvolvimento de jogos de computador e aplicações multimídia*. Dissertação (Mestrado em Ciência da Computação) - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2001.

[33] Manber, Udi. *Introduction to Algorithms: A Creative Approach.* Addison-Wesley, 1989. p. 204-214.

[34] Megiddo, N., Hakimi, S., Garey, M., et al. The complexity of searching a graph. In: JOURNAL OF THE ACM. 1988. v. 35. p. 18-44.

[35] Menezes, T. R. *Negociação em Sistemas Multiagente para Patrulhamento.* Trabalho de Graduação - Centro de Informática da Universidade Federal de Pernambuco. Recife: UFPE, 2002.

[36] Meyer, Paul. *Probabilidade Aplicações à Estatística.* Rio de Janeiro: LTC Editora, 1983. p. 137-144.

[37] Minar N., Hultman K, Maes P. Cooperating Mobile Agents for Mapping Networks. In: PROCEEDINGS OF THE FIRST HUNGARIAN NATIONAL CONFERENCE ON AGENT BASED COMPUTING, 1999.

[38] Mitchell, Tom M. *Machine Learning*. McGraw-Hill Press, 1997

[39] NBA Inside Drive 2000. Disponível em: http://www.microsoft.com/sports/insidedrive2000/

[40] Ortiz, C. L., Hsu, E. Structured Negotiation. In: THE FIRST INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. Bologna, Italy: 2002.

[41] Panzer Commander. Disponível em: http://www.panzercommander.com/

[42] Parsons, T. D. Pursuit-evasion in a graph. In: THEORY AND APPLICATION OF GRAPHS. Berlin: Springer Verlag, 1976. p. 426-441

[43] Pottinger, D. C. Coordinated Unit Movement. Game Developer, Jan. 1999. p. 42-51.

[44] Pottinger, D. C. Implementing Coordinated Unit Movement. Game Developer. Feb. 1999. p. 48-58.

[45] Prefeitura do Recife. Disponível em: http://www.recife.pe.gov.br/

[46] Rabin, S. *AI Game Programming Wisdom*. 2002

[47] Reynolds, C. W. Steering Behaviors for Autonomous Characters. In: PROCEEDINGS OF GAME DEVELOPERS CONFERENCE. San Jose, California: 1999. p. 763-782. Disponível em: http://www.red3d.com/cwr/steer/

[48] RoboCup Rescue home page. Disponível em: http://www.r.cs.kobe-u.ac.jp/robocup-rescue/

[49] Russell, Stuart, Norvig, Peter. *Artificial Intelligence: A Modern Approach*. 1. ed. New Jersey: Prentice Hall, 1995. p. 111-114, 796-808.

[50] Searle, J. (1972). What is a speech act, Penguin Books Ltd, Harmoudsworth, Middlesex.

[51] Shopping Guararapes. Disponível em: http://www.shoppingguararapes.com.br

[52] Spirakis, P. G., Tampakas, B. Distributed Pursuit-Evasion: Some Aspects of Privacy and Security in Distributed Computing. In: PROCEEDINGS OF THE THIRTEENTH ANNUAL ACM SYMPOSIUM ON PRINCIPLES OF DISTRIBUTED COMPUTING. Los Angeles, California, USA: ACM, 1994. p. 403.

[53] Starcraft. Disponível em: http://www.blizzard.com/starcraft/

[54] Star Wars Rogue Squadron. Disponível em: http://www.lucasarts.com/products/rogue/

[55] Stout, B. W.: Smart Moves: Intelligent Path-Finding. Game Developer. Oct. 1996. p. 28-35.

[56] Stroustrup, Bjarne. *The C++ Programming Language*. 3. ed. Addison-Wesley, 1997.

[57] Woo, M., Neider, J., Davis, T., et al. *OpenGl (r) programming guide: the official guide to learning OpenGL, version 1.2*. 3. ed. Addison-Wesley, 1999.

[58] Tangamchit, P., Dolan, J. M., Khosla, P. K. Learning-Based Task Allocation in Decentralized Multirobot Systems. In: PROCEEDINGS OF THE 5TH INTERNATIONAL SYMPOSIUM ON DISTRIBUTED AUTONOMOUS ROBOTIC SYSTEMS. 2000. p. 381-390.

[59] Ultima Online. Disponível em: http://www.uo.com/

[60] Unreal Tournament. Disponível em: http://www.unrealtournament.com/