Patrulhamento Multiagente

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Tiago Sak e aprovada pela Banca Examinadora.

Campinas, 23 de novembro de 2008.

Prof. Dr. Jacques Wainer (Orientador)

Prof. Dr. Siome Klein Goldenstein
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8a 162/2005

Sak, Tiago
Sa29p Patrulhamento multiagente / Tiago Sak -- Campinas, [S.P. : s.n.], 2008.

Orientadores: Jacques Wainer ; Siome Klein Goldenstein
Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Inteligencia artificial. 2. Sistemas multiagentes. 3. Simulação (Computadores). I. Wainer, Jacques. II. Goldenstein, Siome Klein. . III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

(cqc/imecc)

Título em inglês: Multiagent patrolling

Palavras-chave em inglês (Keywords): 1. Artificial intelligence. 2. Multiagent systems.
3. Simulation (Computers).

Área de concentração: Sistemas de Informação

Titulação: Mestre em Ciência da Computação

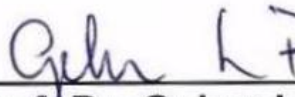
Banca examinadora: Prof. Dr. Jacques Wainer (IC-Unicamp)
Prof. Dr. Geber Lisboa Ramalho (CIn-UFPE)
Prof. Dr. Flávio Keidi Miyazawa (IC-Unicamp)

Data da defesa: 03/10/2008

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 03 de outubro de 2008, pela Banca examinadora composta pelos Professores Doutores:



Prof. Dr. Geber Lisboa Ramalho
CIN. / Universidade Federal Pernambuco.

Prof. Dr. Flávio Keidi Miyazawa
IC – UNICAMP

Prof. Dr. Jacques Wainer
IC – UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Patrulhamento Multiagente

Tiago Sak[1]

Outubro de 2008

Banca Examinadora:

- Prof. Dr. Jacques Wainer (Orientador)
- Prof. Dr. Geber Lisboa Ramalho
  Centro de Informática - UFPE
- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação - UNICAMP
- Prof. Dr. Cid Carvalho de Souza (Suplente)
  Instituto de Computação - UNICAMP
- Prof. Dr. Jaime Simão Sichman (Suplente)
  Departamento de Engenharia da Computação e Sistemas Digitais - USP

[1]Suporte financeiro de: Bolsa do CNPq 2006–2008.

Resumo

O patrulhamento está associado ao ato de percorrer um ambiente com o objetivo de protegê-lo ou simplesmente supervisioná-lo. Em geral, uma execução eficiente desta atividade demanda a alocação de um grupo de patrulhadores, de forma que, o problema do patrulhamento pode ser considerado inerentemente multiagente.

Os trabalhos anteriores relacionados com o patrulhamento multiagente, utilizaram de critérios de avaliação que buscavam prioritariamente minimizar o tempo necessário para percorrer o ambiente como um todo, sem acrescentar quaisquer restrições que evitassem o uso de soluções completamente estáticas. No entanto, a utilização de soluções que não incluem nenhum tipo de variação possibilitam que eventuais atacantes tornem-se aptos a inferir o tempo do qual dispõem para sua invasão. De forma que, embora, muito eficientes, as estratégias apresentadas permitem um planejamento prévio por parte de atacantes, e portanto, em muitos cenários as soluções propostas não serão capazes de impedir a ação desses intrusos. Buscando estender a aplicabilidade do patrulhamento multiagente, este trabalho propõe uma nova abordagem ao problema, considerando critérios de avaliação baseados em diferentes modelos de atacantes, partindo de invasores que agem de maneira randômica, mas considerando também intrusos que fundamentam suas ações em ferramentas estatísticas de predição. Tendo por base essa nova perspectiva são propostas ao longo do trabalho um conjunto de novas soluções, que buscam orquestrar a ação dos patrulhadores de forma a garantir que o patrulhamento seja bem sucedido. Propõe-se também uma metodologia de comparação e avaliação das soluções apresentadas, incluindo a elaboração de um gerador de cenários, o que possibilitou a simulação das soluções, de acordo com cada critério de avaliação, em um amplo conjunto de ambientes.

Por fim, o trabalho apresenta mais uma extensão ao problema do patrulhamento ao considerar o impacto proveniente do corrompimento de um dos agentes patrulhadores. Buscando amenizar o efeito decorrente deste corrompimento são propostas soluções específicas para esse caso. Novamente as soluções são comparadas e avaliadas de acordo com cada critério de avaliação.

Abstract

Patrolling refers to the act of walking around an area, with some regularity, in order to protect or supervise it. A group of agents is usually required to perform this task efficiently. Previous works in this field, using a metric that minimizes the period between visits to the same position, proposed static solutions that repeats a cycle over and over. But an efficient patrolling scheme requires unpredictability, so that the intruder cannot infer when the next visitation to a position will happen. This work presents various strategies to partition the sites among the agents, and to compute the visiting sequence. We evaluate these strategies using three metrics which approximates the probability of averting three types of intrusion - a random intruder, an intruder that waits until the guard leaves the site to initiate the attack, and an intruder that uses statistics to forecast how long the next visit to the site will be. We present the best strategies for each of these metrics, based on several simulations. Additionally we propose another extension to the patrolling problem by considering the possibility that one patroller have been corrupted. Specific solutions are proposed, analyzed and compared using the evaluation criteria.

Agradecimentos

Agradeço a todos os professores com que mantive contato durante estes dois anos de mestrado. Em especial agradeço aos professores Jacques Wainer e Siome Klein Goldenstein com quem tive o grande privilégio de conviver e aprender ao longo de todo este período. Agradeço ao Instituto de Computação e também ao CNPQ pelo apoio financeiro oferecido durante o decorrer deste mestrado.

Agradeço a todos os alunos do Instituto com quem pude partilhar de momentos muito agradáveis de descontração, em especial agradeço a Rodrigo Minetto, Juliana de Santi, Fernando Kronbauer e Jefersson Alex que de forma especial estiveram presentes durante todo o tempo que estive aqui. A Gabriela Prado e Fernanda Adams Schunig, que embora distantes mantiveram-se especialmente presentes nos momentos de dificuldade, meu profundo agradecimento. Agradeço também aos novos amigos com quem pude partilhar momentos muito especiais em Campinas, Alexandre e Felipe Silva, Paulo Sérgio Maciel. Pelos conselhos, carinho e acolhimento agradeço a Marisa Silva e toda sua família. Por fim, demonstro aqui meu profundo apreço e agradecimento à Juliana dos Santos por aqueles, embora breves, tão significativos momentos que pudemos compartilhar.

Presto aqui também minha homenagem e sinceros agradecimentos ao professor Elton Dias Junior e a Eduado Batista Cruz que foram de fundamental importância no processo de despertar em mim o interesse pela pesquisa e o apreciar pela ciência.

Embora palavra alguma possa absorver significado tal, a ponto de revelar todo o carinho, amor e admiração que sinto por vocês reafirmo o meu agradecimento por todo vosso investimento em minha vida. A vocês Julio, Sueli, André, Ana Paula, Jonatas e Melissa Sak um agradecer que vai além daquilo que pode ser escrito.

Por fim, àquele em quem a expressão da vida pode retornar à sua essência, àquele que mais do que criar é capaz de amar ao ponto de se dar, à Deus mais do que simplesmente o meu agradecimento.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação e Contexto

A tarefa de patrulhar determinada área consiste em encontrar uma maneira eficiente de monitorar os diversos locais que constituem aquele ambiente, visando garantir em geral sua segurança. Esta atividade pode ser considerada inerentemente multiagente já que na maioria dos casos o processo poderá ser realizado de maneira distribuída, por um grupo de agentes. Embora o patrulhamento, na maioria dos casos, esteja associado diretamente a uma atividade específica de segurança, os estudos nesta área encontram aplicação em muitos outros domínios, tornando-se importantes em qualquer problema onde fique caracterizada a necessidade de definir uma ordem segundo a qual sejam visitados um conjunto pré determinado de pontos, sistematicamente. De forma que, as pesquisas em patrulhamento multiagente poderão ser empregadas, por exemplo, em sistemas de redes ou então em alguns jogos de computador.

Adicionalmente, é importante criar soluções que sejam coerentes com as características inerentes a atacantes reais garantindo assim a aplicabilidade das soluções em cenários que também sejam reais. Por exemplo, em muitos casos, manter secreta a ordem de visita dos locais é muito importante, evitando-se assim que intrusos usem essa informação para planejar meios de impedir que sejam vistos pelos patrulhadores. A literatura atual relacionada com o problema do patrulhamento multiagente considera, no entanto, prioritariamente apenas uma dentre as várias características relacionadas com uma boa solução para o patrulhamento, dedicando-se em minimizar o tempo necessário para visitar-se todas as regiões de um ambiente, propondo algumas metodologias sem quaisquer variações na ordem com a qual os pontos são visitados. Embora esses métodos obtenham maneiras muito eficientes de realizar as visitas, a ordem com a qual cada nodo será visitado pode ser facilmente deduzido por qualquer intruso externo. Conhecendo a ordem das visitas e portanto, o intervalo de tempo entre elas, um intruso pode elaborar e realizar uma in-

trusão bem sucedida. Por exemplo, se o intruso sabe que um guarda de segurança visita um cofre com fechadura de combinação a cada 50 minutos, com regularidade, o atacante poderá abrir o cofre com sucesso, mesmo que o processo de descobrir a combinação correta leve mais de 50 minutos, isso porque, o intruso pode interromper sua tentativa de ataque um pouco antes da visita do guarda e se esconder, e como sua tentativa não deixa nenhum rastro na fechadura que poderia ser reconhecido por um patrulhador, o intruso terá tempo suficiente para tentar todas as combinações até que obtenha sucesso. Evidentemente, em casos onde a premissa de que uma tentativa de invasão não deixa rastros for falsa o ataque não poderá ser dividido. Suponha para o mesmo cenário de patrulhamento anterior, que um atacante decida obter acesso ao conteúdo do cofre por meio de um arrombamento, que naturalmente deixará marcas reconhecíveis no cofre. Se o tempo necessário para o arrombamento for de 40 minutos, e o atacante puder esperar para começar seu ataque assim que o guarda deixar o cofre, nesse caso um período constante de 50 minutos entre as visitas não irá deter uma intrusão. Portanto, dependendo do tipo de intrusão, a impredictabilidade do período entre visitas, ou ao menos a variabilidade poderá tornar-se tão importante quanto a eficiência em diminuir o intervalo de espera entre as visitas de patrulhadores.

Existem também muitos casos nos quais embora os intrusos possam observar o ambiente que pretendem invadir sem que sejam considerados suspeitos, suponha por exemplo o campus de uma grande universidade um atacante poderia permanecer coletando dados a respeito do patrulhamento sem levantar qualquer suspeita já que dificilmente poderia ser distinguido de um aluno qualquer. Para esses casos os ataques poderão ser planejados consistentemente com o modelo de patrulhamento utilizado evitando facilmente que patrulhadores que utilizam-se de estratégias sem variação sejam bem sucedidos.

Segundo a literatura o patrulhamento pode ser utilizado tendo em vista atender dois objetivos: detecção ou supervisão. No primeiro caso pressupõe-se a existência de um invasor e a tarefa do patrulhamento consiste em encontrar o alvo no menor tempo possível. Já no segundo caso, quando o objetivo é supervisionar um ambiente, o patrulhamento irá monitorar continuamente pontos importantes onde eventualmente poderão ocorrer eventos. É importante destacar que no caso da supervisão o patrulhador não dispõem de quaisquer informações a respeito de quando e onde o eventos acontecerão. Este trabalho fundamenta-se no objetivo de supervisionar um ambiente, portanto, busca localizar eventuais intrusos sem dispor de informações a respeito da posição ou momento no qual ocorrerá o ataque. Embora o objetivo específico de encontrar um alvo no menor tempo possível seja característico do problema de detecção neste trabalho essa condição é mantida já que o tempo de invasão do qual o patrulhador dispõe para tentar encontrar o intruso é geralmente curto.

As estratégias de patrulhamento propostas em geral demandam poucas decisões, e

portanto, consomem muito pouco tempo de processamento, de forma que, as escolhas do patrulhamento poderão ser realizadas em tempo real. Neste trabalho consideramos apenas ambientes estáticos, conhecidos a priori, sem distinção de regiões de prioridade. O número de agentes patrulhadores uma vez definido permanece fixo durante toda a execução e não existe qualquer comunicação entre os patrulhadores.

Outras características relativas ao problema tratado serão expostas ao longo do trabalho, principalmente no capítulo 2.

1.2 Introdução ao Problema do Patrulhamento

O patrulhamento multiagente pode ser dividido em quatro atividades: Encontrar uma *representação de dados* adequada; Escolher *critérios de avaliação* consistentes com o problema; Desenvolver *arquiteturas* que modelem a forma como o patrulhamento será realizado, levando em consideração para isso os *critérios de avaliação* que foram definidos; e, finalmente, faz-se necessário a criação de *cenários experimentais* e a elaboração de uma metodologia de comparação entre as diversas *arquiteturas* propostas.

Uma *representação de dados* adequada está relacionada com uma estrutura computacional que seja capaz de armazenar as principais características de um ambiente, a ser patrulhado. Além disso a escolha dessa estrutura deverá levar em consideração fatores adicionais como a eficiência computacional na manipulação dos dados, dentre outros.

A escolha dos *critérios de avaliação* refere-se a uma das decisões mais importantes em todo o processo de patrulhamento, isso porque, baseados nesses critérios todas as arquiteturas serão desenvolvidas e posteriormente avaliadas. Portanto, critérios de avaliação apropriados deverão levar em consideração todas as características inerentes ao problema do patrulhamento, do ponto de vista de variados cenários e perspectivas.

Uma *arquitetura* constitui o cerne do patrulhamento e está diretamente relacionada com o processo de tomada de decisões. De forma simplificada, são dois os principais passos na elaboração de uma arquitetura: Primeiro defini-se como os patrulhadores irão escolher a ordem de visita aos locais sob sua patrulha, este passo refere-se à determinação do *algoritmo de seqüenciamento*. Além disso, cabe a uma arquitetura determinar a maneira pela qual será distribuída entre todos os agentes a responsabilidade de patrulhar cada um dos locais do ambiente, por exemplo, determinando que o conjunto total de locais será dividido igualmente entre os agentes patrulhadores. Esse passo na elaboração da arquitetura será conhecido como a escolha da *extensão multiagente*. É muito importante destacar que a *qualidade* de uma arquitetura é medida de acordo com o critério de avaliação, de forma que, um engano na escolha dos critérios de avaliação compromete também o desenvolvimento e principalmente, a comparação precisa entre as diversas arquiteturas. Nesse estudo todos os patrulhadores agirão de maneira homogênea, isso porque, definida

uma arquitetura todos os agentes farão uso dela. Embora, todos os agentes façam uso de uma mesma arquitetura, todas as decisões decorrentes dessa escolha (e.g. escolhas aleatórias que façam parte do algoritmo de seqüenciamento) serão tomadas de maneira completamente independente, de forma que, cada patrulhador desconhece por completo o comportamento, posição e objetivos dos demais agentes.

Criadas as arquiteturas e definidos os critérios de avaliação faz-se necessário a geração de *cenários experimentais* que permitirão a comparação, através de uma avaliação experimental, das diversas arquiteturas, levando em consideração todos os critérios de avaliação definidos. Esse conjunto de cenários deverá ser criado em quantidade suficiente a ponto de garantir que um grande número de topologias sejam examinada, além disso, torna-se imprescindível que esses cenários embora possivelmente gerados de maneira artificial sejam muito similares com ambientes reais.

Parte do trabalho apresentado nessa dissertação foi submetido e aceito na forma de artigo, ao Simpósio Brasileiro em Inteligência Artificial (SBIA) de 2008.

1.3 Principais Contribuições

Dentre as principais contribuições deste trabalho, destaca-se a mudança significativa na abordagem dada ao problema do patrulhamento. Ao definir um conjunto de atacantes, ao invés de tentar encontrar um único critério capaz de conter todas as singularidades envolvidas com o casos específicos de patrulhamento, essa nova abordagem permite avaliar e comparar quaisquer soluções propostas sob pontos de vista variados de diversos modelos de atacantes. Uma solução adequada do ponto de vista desse novo paradigma deverá portanto, ser eficaz em conter desde ataques simples até invasões muito bem elaboradas, baseadas em modelos estatísticos consolidados.

A mudança significativa dada ao problema do patrulhamento exigiu o desenvolvimento de novas arquiteturas, adaptadas aos novos domínios de avaliação. De forma que nesse trabalho são propostos dez algoritmos de seqüenciamento, e cinco extensões multiagentes.

Para permitir uma comparação precisa e consistente fundamentada nos critérios de avaliação fez-se necessário a elaboração de um novo modelo de avaliação e comparação de arquiteturas. Além disso, como o conjunto de soluções propostas é muito grande (um total de cinqüenta arquiteturas), elaborou-se um modelo inicial que visa identificar qual é a arquitetura mais indicada para cenários específicos.

Outra importante contribuição do trabalho foi a criação de um gerador de grafos, que baseando-se em diversos parâmetros provê a criação de um grande conjunto de *cenários de avaliação*, o que torna a comparação das arquiteturas mais consistente e robusta, ao considerar as mais variadas topologias de ambientes sob patrulha.

Por fim, este trabalho propõe mais uma extensão ao problema do patrulhamento, ao

considerar a possibilidade de que agentes patrulhadores sejam corrompidos. Decorrentes desse novo cenário de atuação são propostas novas soluções específicas para esse caso, que visam tornar o patrulhamento tolerante a esse tipo de falha, e portanto mais robusto.

1.4 Organização do Texto

Esta dissertação foi estruturada em seis capítulos. Além desse primeiro capítulo que contém uma breve introdução ao problema do patrulhamento, o Capítulo 2 ainda apresenta diversos conceitos introdutórios relacionados com o patrulhamento abordando, para isso, os principais trabalhos correlatos. No Capítulo 3 é apresentada a nova abordagem dada ao problema de patrulhamento multiagente, nesse capítulo expõe-se o conjunto de conceitos que compõem o corpo das principais contribuições desse trabalho, serão considerados os novos critérios de avaliação, algoritmos de seqüenciamento, extensões multiagentes, e cenários de avaliação. Expostos os principais conceitos, o quarto Capítulo considera a avaliação experimental das soluções propostas, em todos os contextos sugeridos. Para isso, são apresentados os detalhes relativos a simulação, os resultados obtidos, e uma análise desses dados. O Capítulo 5 está relacionado com a abordagem estendida dada aos critérios de avaliação, nesse capítulo são estudados os impactos causados por conta do corrompimento de um dos agentes patrulhadores, são ainda apresentadas algumas possíveis soluções que buscam minimizar o impacto proveniente da corrupção de agentes patrulhadores. Por fim, o sexto e último Capítulo concluí esta dissertação e apresenta alguns vislumbres de direções futuras que poderão ser consideradas na extensão deste trabalho.

Capítulo 2

Revisão Bibliográfica

Muitas são as pesquisas relacionadas ao uso de múltiplos agentes, agindo de forma cooperativa ou não, em problemas de busca ou patrulhamento de determinada área. A busca difere-se do patrulhamento, ao pressupor a existência de um alvo a ser localizado, seja ele um invasor ou não. Diferentemente, o patrulhamento busca diminuir o risco de uma invasão, buscando assegurar que dada a existência momentânea de um invasor na região patrulhada ele será localizado antes que seja possível ao intruso completar seu objetivo.

Os trabalhos centralizados na primeira tarefa - busca - dividem-se na literatura de acordo com diversos aspectos básicos, dentre os quais destacam-se três, de maior relevância para nosso trabalho:

a) quanto ao conhecimento inicial da região de busca, a topologia pode ser totalmente desconhecida, ou então, parcialmente/completamente informada à priori. No caso de existir, o processo de aprendizado da topologia que define a área de busca, é conhecido por *exploração* (ou *map-learning*);

b) quanto a movimentação do alvo da busca, são consideradas três formas fundamentais: o alvo movimenta-se de forma a evitar que seja encontrado (e.g., missões de busca e captura), ou então, seu deslocamento pode ser semelhante ao movimento randômico (e.g, missões de busca e resgate), por fim o alvo pode permanecer imóvel em uma posição durante determinado tempo (e.g., missões de busca de invasores);

c) quanto a informações referentes às localizações dos alvos, o caso mais comum é aquele onde não existem quaisquer dados disponíveis, enquanto em outros cenários, exitem informações, geralmente momentâneas, a respeito das localizações dos alvos. Nesses casos, o problema essencial deixa de ser descobrir a localização, e passa a ser providenciar o encontro de um agente com um alvo, seja para sua captura, resgate, identificação, ou qualquer objetivo definido previamente.

As variações desses e de outros aspectos possibilitam uma vasta literatura tratando do problema da busca. No que tange ao primeiro aspecto, a busca em regiões previamente conhecidas apresenta soluções com eficiência bem definida [23, 18]. Para casos nos quais a região é desconhecida a priori, são apresentadas duas abordagens na literatura, na primeira, existe uma fase inicial de exploração, concluído esse processo começa a busca. Já na segunda abordagem, exploração e busca ocorrem simultâneamente, e nesse caso a medida de eficiência da localização do alvo é essencialmente probabilística. Hespanha *et al.* [12] propõem um método que realiza exploração e busca em um único passo, fazendo uso de um algoritmo *guloso* que direciona os perseguidores a cada momento para a região que maximiza a chance de encontrar um alvo móvel, cuja posição e forma de movimentação são completamente desconhecidas. O tempo necessário para encontrar o alvo, no entanto, é garantidamente finito. Em [8] tem-se novamente a exploração e a busca ocorrendo simultâneamente, nesse trabalho estuda-se o problema de gerar uma trajetória próxima da ótima a ser seguida por um conjunto de agentes para o caso onde existam informações prévias referentes à uma distribuição probabilística dos alvos - regiões de alta e baixa probabilidade de se encontrar um alvo. Outro estudo, apresentado em [27] investiga o uso de agentes na busca de alvos cuja localização é periodicamente observável ficando visível durante um curto período de tempo, baseando-se nessas observações cada agente estima a próxima posição do alvo, para assim fundamentar sua decisão relativa à direção pela qual optará seguir.

Existem também muitos trabalhos considerando diretamente o problema do patrulhamento. As pesquisas nesse campo, configuram os parâmetros relacionados para o problema da busca da seguinte maneira: a área a ser patrulhada é previamente conhecida, o intruso a ser localizado (alvo da busca) permanece em um único local até completar sua tarefa intrusiva (alvos estáticos), findo o tempo de intrusão a invasão é considera bem-sucedida, não existem quaisquer informações a respeito da localização dos intrusos, além disso não é possível aos agentes prever em que momento haverá uma invasão.

Os primeiros trabalhos na área de patrulhamento dedicavam-se exclusivamente a regiões extremamente particulares, considerando polígonos com n lados. Chvatal [5] introduziu o problema da *galeria de arte*, estudando o número de agentes estacionários - que não realizam qualquer tipo de movimentação - necessários para cobrir um polígono de n lados. O'Rourke [21] estendeu o trabalho apresentando diversos problemas correlatos. Chin and Ntafos [4, 22, 20] introduziram, posteriormente, o uso de agentes móveis no patrulhamento. Ainda nessa linha, Grace e Baillieul [10] fizeram uso de regras estocásticas para guiar o movimento dos agentes, de forma a realizar a tarefa do patrulhamento. O maior interesse nesse trabalho foi definir a taxa com a qual uma cadeia de Markov converge para um estado de distribuição estável, a matriz de probabilidades de transição define a região de patrulha, destacando as probabilidades dentre as possíveis escolhas

para cada um dos vértices (dada certa localização qual a chance de se escolher cada um dos caminhos particulares possíveis).

Seguindo uma linha particular, na área de patrulhamento, estão os trabalhos de [17, 16, 3, 1, 26, 19]. Estes trabalhos apresentam o problema de patrulhamento como a busca por uma heurística que organize os agentes e seus modelos de decisão ao percorrer um grafo (que define a região de interesse) levando em consideração um critério de avaliação. A principal meta, e conseqüentemente o principal critério de avaliação apresentado, é o de reduzir o período entre duas visitas consecutivas a um mesmo vértice. Dada a grande relevância desses trabalhos no contexto da presente dissertação, os detalhes relativos a estes trabalhos serão citados nas próximas seções. A apresentação será iniciada considerando a *Representação de Dados* e os *Critérios de Avaliação*, comuns a todos os trabalhos, serão então expostas as *Metodologias* propostas como soluções, e por fim tratar-se-á dos *Cenários de Avaliação* e os detalhes relativos à *Avaliação Experimental*.

2.1 Representação de Dados

O primeiro passo na resolução do problema de patrulhamento, é a representação de um ambiente real, por meio de uma estrutura de dados abstrata que seja apropriada. Esta representação deve manter-se fiel às principais características desse ambiente, além de permitir um desempenho computacional adequado. Uma das principais técnicas utilizadas com o fim de gerar uma abstração partindo de um ambiente arbitrário é conhecido por *skeletonization* (figura 2.1), essa técnica busca extrair um conjunto de características do formato de uma região, produzindo como saída, a forma geral do ambiente representada por meio de uma simples abstração, como um *grafo*. Baseando-se nisso, a representação de dados do terreno a ser patrulhado, é um grafo, onde os vértices representam locais específicos a serem visitados, enquanto, as arestas referem-se a caminhos possíveis entre os vértices. Existem muitas implementações para aplicar a técnica de *skeletonization*, as mais relevantes no contexto de nosso trabalho são os *diagramas de Voronoi*, *C-Cells* e *grafos de visibilidade*. É preciso, no entanto, atentar para o fato de que cada um desses métodos obterá freqüentemente resultados diferentes, para um mesmo conteúdo de entrada. De forma que, a escolha do algoritmo apropriado dar-se-á de acordo com o terreno em estudo.

2.2 Critérios de Avaliação

Para possibilitar a comparação entre as diversas arquiteturas, buscando identificar aquela que obteve os melhores resultados, são apresentados alguns critérios de avaliação: a *ociosi-*

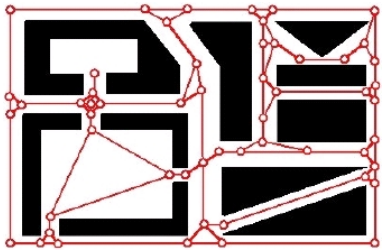


Figura 2.1: Exemplo da abstração de um ambiente em um grafo, por meio da técnica de *skeletonization*. [17]

dade, a *pior ociosidade* e o *tempo de exploração*, dentre os quais, o principal é a *ociosidade*.

Considere como *iteração* a unidade de tempo utilizada nas simulações. Dessa forma, o tempo necessário para um agente ir de um nodo até um dos nodos vizinhos será dado em *iterações*. É importante notar que a cada aresta está associado um valor, referente ao número de iterações necessárias para percorrê-la, ou seja, em termos práticos esse valor reflete quanto tempo é gasto para ir do vértice origem ao nodo objetivo[1].

A *ociosidade instantânea do nó* é o número de iterações desde a última visita de um agente qualquer a aquele vértice. Já a *ociosidade instantânea de um grafo* é a média da ociosidade instantânea para todos os nodos do grafo em uma determinada iteração. Finalmente, a *ociosidade do grafo*, ou simplesmente ociosidade, é a média da ociosidade instantânea do grafo durante todos as $\mathcal{N}_I$ iterações da simulação. Já a *pior ociosidade* de um vértice é o maior valor da *ociosidade instantânea do nó* durante toda a simulação. Finalmente, o último critério de avaliação, denominado *tempo de exploração* é o número de iterações necessárias para um agente visitar, ao menos uma vez, todos os nodos do grafo, ou seja realizar uma *cobertura completa* do grafo.

Considere um grafo $G(V, E)$, onde $n = |V|$. Seja $o(v, p)$ a ociosidade instantânea do vértice v no passo p, $O(p)$ a ociosidade instantânea de G no passo p, e O_G a ociosidade média do grafo durante toda a execução. Em uma simulação com $\mathcal{N}_I$ iterações, tem-se:

$$O(p) = \frac{\sum_{v=1}^{n} o(v, p)}{n} \tag{2.1}$$

$$O_G = \frac{\sum_{p=1}^{\mathcal{N}_I} O(p)}{\mathcal{N}_I} \tag{2.2}$$

Evidentemente que com o acréscimo de novos agentes à simulação, as arquiteturas tenderão a apresentar melhores resultados. Para normalizar o valor do critério de avaliação de acordo com o número de agentes envolvidos na simulação, faz-se:

[1]Em [17], todas as arestas possuíam um valor constante e unitário, e portanto eram equidistantes. Nos demais trabalhos foi atribuído a cada uma das aresta existentes um valor condizente com a distância euclidiana entre os vértices que a compunham.

$$\mathcal{V}_{norm} = \mathcal{V} \times \frac{\mathcal{N}_{agentes}}{\mathcal{N}_{vertices}} \tag{2.3}$$

Em todos os trabalhos sumarizados anteriormente, a ênfase principal na avaliação foi dada ao critério de *ociosidade do grafo*, buscando arquiteturas para agentes que minimizassem esse parâmetro tanto quanto possível.

2.3 Metodologias

A seguir, são apresentados o conjunto de abordagens relacionadas pelos trabalhos, como possíveis soluções ao problema do patrulhamento. Para facilitar à compreensão esta seção foi dividida em quatro subseções, dispostas na ordem cronológica de suas respectivas publicações. A primeira seção trata das arquiteturas heurísticas, introduzidas nos trabalhos [17, 16]. Posteriormente, serão considerados o uso de *Reinforcement Learning* na busca de uma solução para o problema, esta abordagem é apresentada no trabalho [26]. A arquitetura mais natural relativa aos critérios de avaliação utilizados, que utiliza-se do famoso problema do caixeiro viajante (*Traveling Salesman Problem - TSP*), foi abordada em [3, 1], essa solução será abordada na terceira subseção. A última abordagem utiliza-se de agentes negociadores, um campo remanescente de pesquisa em sistemas multiagentes, apresentada inicialmente em [1] e complementada em [19].

2.3.1 Arquiteturas Heurísticas

As primeiras abordagens para a elaboração de arquiteturas que pudessem ser aplicadas ao problema de patrulhamento, basearam-se em quatro parâmetros básicos (como disposto na tabela 2.1), os três primeiros parâmetros remetem-nos aos estudos em sistemas Multiagentes (detalhes podem ser obtidos em [7]), enquanto o último parâmetro aborda uma característica específica da tarefa de patrulhamento. O primeiro parâmetro, faz a diferenciação entre agentes reativos e cognitivos. Os agentes reativos operam baseados em sua percepção atual, enquanto os agentes cognitivos podem perseguir um objetivo. No caso dos agentes reativos, a percepção atual, está diretamente relacionada ao campo de visão do referido agente. Nos experimentos realizados, o campo de visão foi limitado a um vértice de profundidade, ou seja, o agente reativo baseia sua decisão apenas nos nodos adjacentes ao vértice no qual ele se encontra.

O segundo parâmetro considerado avalia a forma de comunicação entre diferentes agentes. O primeiro modelo de comunicação, opta por deixar marcas no ambiente (*flags*), essas marcas serão reconhecidas por todos os agentes e servirão como meio de comunicação. Já o segundo modelo, conhecido por *blackboard*, mantém uma estrutura de memória compartilhada que poderá ser acessada e alterada por todos os agentes, sempre que ocorrer

um evento (e.g.: visita a um nodo). O terceiro e último modelo, permite a comunicação direta entre diferentes agentes através do uso de *mensagens* explícitas.

Outra característica importante em sistemas Multiagentes, é a estratégia de coordenação, nela decide-se como será feita a organização da interação intra-agentes. A coordenação, de maneira simplificada, diz respeito ao comprometimento individual de cada agente em tomar decisões que irão beneficiar o grupo de agentes como um todo. Nos estudos realizados foram utilizadas dois tipos de coordenação, a *emergente* permitia que a coordenação fosse feita de maneira descentralizada, surgindo naturalmente da interação entre os diversos agentes, no segundo caso a coordenação se dava de forma *centralizada* e um coordenador central determinava qual seria o próximo passo tomado por cada um dos agentes.

A estratégia de raciocínio, última e principal característica, diz respeito à forma como será feita a escolha do próximo nodo a ser visitado. Essa decisão pode ser feita de forma randômica ou heurística, baseando-se no segundo caso na ociosidade dos nós. A decisão heurística prioriza sempre pela escolha do nodo com maior ociosidade instantânea.

<table>
<tr><th>Metodologia</th><th>Tipo básico</th><th>Comunicação</th><th>Estratégia de coordenação</th><th>Estratégia de raciocínio</th></tr>
<tr><td>Random Reactive</td><td rowspan="3">Reativo</td><td rowspan="2">Nenhuma</td><td rowspan="6">Emergente</td><td>Aleatória</td></tr>
<tr><td>Conscientious Reactive</td><td rowspan="4">Heurística baseada na ociosidade</td></tr>
<tr><td>Reactive with Flags</td><td>Flags</td></tr>
<tr><td>Conscientious Cognitive</td><td rowspan="6">Cognitivo</td><td>Nenhuma</td></tr>
<tr><td>Blackboard Cognitive</td><td rowspan="2">Blackboard</td></tr>
<tr><td>Blackboard Cognitive Monitored</td></tr>
<tr><td>Random Coordinator</td><td rowspan="4">Mensagens</td><td rowspan="4">Central</td><td rowspan="2">Aleatória</td></tr>
<tr><td>Random Coordinator Monitored</td></tr>
<tr><td>Idleness Coordinator</td><td rowspan="2">Heurística baseada na ociosidade</td></tr>
<tr><td>Idleness Coordinator Monitored</td></tr>
</table>

Tabela 2.1: Resumo das principais características das arquiteturas apresentadas.[16]

Segue uma sumarização ressaltando as principais diferenças entre as arquiteturas apresentadas:

- ***Random Reactive:*** A mais simples dentre todas as arquitetura, nela o agente escolhe randomicamente qualquer um dos vizinhos do nodo corrente, para ser o

próximo nodo.

- ***Conscientious Reactive*** Arquitetura na qual o agente, mantém uma memória de suas decisões anteriores, e opta sempre pelo nó adjacente ao seu, que está a mais tempo sem receber uma visita sua. No caso de empates, a escolha é aleatória.

- ***Reactive with Flags*** Semelhante à arquitetura anterior, com a diferença de que são deixadas marcas em todos os nodos na visita por qualquer agente, de forma que a decisão do nó adjacente a mais tempo sem receber visitas leva em consideração as visitas realizadas por todos os agentes.

- ***Conscientious Cognitive*** Novamente, essa arquitetura é semelhante ao *Conscientious Reactive*, no entanto, retira-se a limitação do campo de visão, e portanto, o agente opta sempre pelo nodo do grafo inteiro que possui a maior ociosidade instantânea. O percurso percorrido será o caminho mais curto (*shortest path*) até o nodo escolhido. Nessa arquitetura os agentes *concorrem* entre si, mantendo um único objetivo.

- ***Blackboard Cognitive*** Semelhante ao *Reactive with Flags*, no entanto, ao invés de deixar marcas no ambiente, as informações são deixadas em um espaço de memória compartilhado, novamente o nodo com maior ociosidade do grafo será escolhido havendo empate na escolha do nodo com maior ociosidade, um sorteio decide o nodo a ser visitado.

- ***Blackboard Cognitive Monitored*** Baseado na arquitetura anterior, ele diferencia-se por fazer uma verificação a cada passo, certificando-se de que o nodo objetivo não tenha sido visitado por algum outro agente, buscando nesse caso um novo objetivo.

Findas as arquiteturas com estratégia de coordenação emergente, serão apresentadas as arquiteturas onde as decisões relativas a cada agente serão tomadas de forma centralizada, por um agente coordenador. Esse coordenador possui o conhecimento relativo a todas as visitas no ambiente e baseado nelas determina a ação de cada agente. Como todas as arquiteturas baseadas nessa característica são cognitivos, se a escolha do próximo nodo for relativa à um vértice não adjacente ao nó corrente do agente, novamente será optado pelo uso do caminho mais curto até o nodo destino.

- ***Random Coordinator*** O coordenador decide que nodo cada agente visitará por meio de uma escolha randômica. Quando um nodo é atribuído a um determinado agente, ele deixa de pertencer a lista de nodos livres, impedindo assim que dois agentes compartilhem um mesmo nodo-objetivo.

- ***Random Coordinator Monitored*** Baseado na arquitetura anterior, ela diverge por permitir que a cada passo seja feita uma verificação que constata se o nodo-objetivo não foi visitado por outro agente (evidentemente, isso só acontecerá quando o nodo-objetivo fizer parte do caminho mais curto para algum vértice).

- ***Idleness Coordinator*** O coordenador baseia sua atribuição na ociosidade dos diversos nodos do grafo, novamente atribuindo diferentes objetivos a todos os agentes.

- ***Idleness Coordinator Monitored*** Uma integração das duas últimas arquiteturas apresentadas. Nessa arquitetura, novamente são atribuídos a cada um dos agentes os nodos com as maiores ociosidades, no entanto, a cada passo o agente verifica se o seu objetivo continua válido, ou seja, se o nodo permanece sem ter sido visitado por outro agente.

Em [1] foi apresentada uma nova heurística, chamada *Heuristic Path-finder cognitive coordinated*, semelhante a *Idleness Coordinator Monitored*. A heurística difere-se ao percorrer um caminho que busca passar por nodos de maior ociosidade, ao invés de tomar simplesmente o *caminho mais curto*, em casos onde o nodo objetivo não é um vértice adjacente ao nodo de origem.

Dentre todas as técnicas apresentadas acima, a técnica que apresentou os melhores resultados (como mostra a seção 2.5) nos cenários testados foi a última heurística apresentada (*Heuristic Path-finder cognitive coordinated*).

2.3.2 Reinforcement Learning

Para permitir a compreensão das abordagens criadas utilizando *reinforcement learning*, faz-se necessário um breve resumo da teoria de *Markov Decision Processes*, utilizada na construção das arquiteturas em estudo, uma descrição detalhada pode ser encontrada em [25].

Considere um agente que toma decisões seqüenciais em um ambiente. A cada passo, esse agente escolhe uma *ação* de um conjunto finito, baseando-se no *estado* atual do ambiente. Este estado sumariza as *sensações* presentes e passadas de forma tal que toda informação relevante é mantida. Para decidir qual ação tomar, o processo baseia-se em uma função de recompensa imediata, que recebe por parâmetros um estado, e uma ação. O objetivo primário é maximizar a soma de recompensas durante todo o período de execução.

Como essa teoria apresenta uma solução para um único agente, é necessário estender o conceito para um conjunto de agentes. A principal dificuldade nesse processo, vem do fato de que quando um grupo de agentes busca resolver uma tarefa de maneira distribuída, cada um tentando maximizar sua função de recompensa, isso não necessariamente leva a

uma solução global máxima. Nesse contexto surgiram diferentes funções de utilidades: A primeira é chamada *Wonderful Life Utility (WLU)* na qual o agente otimiza uma utilidade privada que está de acordo com uma utilidade global. Em outras palavras, a função de utilidade para cada um dos agentes garante que mais de um agente não irá trabalhar com propósitos equivalentes. Esta abordagem pode ser implementada usando penalidades quando vários agentes competem pela mesma recompensa. A segunda função, recebe o nome de *Selfish Utility (SU)*, na qual cada agente tenta maximizar sua própria função de utilidade, independentemente dos demais agentes. Existem ainda duas abordagens quanto à comunicação intra-agentes: Em sistemas *Black-Box (BB)* não existe qualquer interação entre agentes diferentes e portanto o agente não sabe a respeito das ações dos demais agentes. Já nos sistemas *Gray-Box (GB)*, agentes podem comunicar suas ações ou intenções de futuras ações. Foram implementadas no âmbito do problema corrente, quatro abordagens:

- ***Selfish Utility* com modelo de comunicação *Black-Box***: Cada agente age de maneira a otimizar sua função de recompensa, sem manter comunicação com os demais agentes.
- ***Selfish Utility* com modelo de comunicação *Gray-Box***: Novamente os agentes buscam otimizar exclusivamente sua função de recompensa, no entanto, agora eles mantém comunicação com os demais agentes, e adequão-se às funções alheias.
- ***Wonderful Life Utility* com modelo de comunicação *Black-Box***: Quando agentes diferentes compartilham de uma mesma intenção, suas funções de recompensa recebem uma punição. Novamente, não existe comunicação direta entre os agentes.
- ***Wonderful Life Utility* com modelo de comunicação *Gray-Box***: Mais uma vez, se agentes diferentes compartilham de uma mesma intenção, suas funções de recompensa recebem uma punição. Nessa arquitetura existe um canal de comunicação direto entre os agentes.

Dentre todas as técnicas apresentadas acima, a técnica que apresentou a menor ociosidade média durante a execução (como mostra a seção 2.5) nos cenários testados foi a arquitetura chamada, *Selfish Utility* com modelo de comunicação *Gray-Box*.

2.3.3 Problema do Caixeiro Viajante

Considere o problema de um único agente patrulhar uma área, buscando minimizar o tempo total da cobertura completa da região. A estratégia diretamente relacionada a

esse conceito, visa buscar um ciclo que cubra toda a área, de forma que esse ciclo seja o menor possível. Basta então fazer com que o agente cubra este ciclo continuamente. Este problema está claramente relacionado com o bem conhecido *problema do caixeiro viajante*, que visa encontrar o menor ciclo que contenha todos os vértices de um grafo. Este problema é sabidamente *NP*-difícil, o que evidentemente restringe sua aplicação prática ao problema do patrulhamento. Existem, no entanto, aproximações, a mais famosa delas foi apresentada por Christofides e dispõe de um resultado bastante razoável para fins práticos: $|S_{Chr}| \leq \frac{3}{2}|S_{TSP}|$, onde S_{TSP} é a solução ótima para o problema do caixeiro viajante. Detalhes do algoritmo são apresentadas em [24] [2].

Originalmente o problema e as respectivas soluções para o caixeiro viajante foram desenvolvidos para ambientes com um único agente, foram propostas então, duas técnicas para mapear a solução em um ambiente multiagente.

- **Estratégia cíclica**: A estratégia nesse caso refere-se a arranjar todos os agentes no mesmo ciclo, de forma que quando eles começam a movimentar-se através do caminho, todos na mesma direção, eles irão manter aproximadamente constante o espaço entre eles.

- **Estratégia baseada em particionamento**: Outra estratégia bastante intuitiva, constitui-se em particionar o território em N_A partições disjuntas, onde N_A refere-se ao número de agentes. Nesse caso cada agente patrulhará dentro de uma única região simples.

Com exceção de alguns casos particulares os resultados obtidos com o uso da *estratégia cíclica* foram melhores, do que aqueles obtidos com o uso da estratégia baseada em particionamentos, independentemente do particionamento feito. Por conta desse melhor resultado, na seção 2.5 serão apresentados apenas os resultados da utilização da estratégia cíclica.

2.3.4 Agentes Negociadores

Nessa abordagem, foram implementadas negociações, semelhante a *leilões*, entre os diversos nodos. Nesses *leilões* cada agente faz uma *oferta*, baseando-se no ganho relativo a incrementar sua lista de nodos com o nó em negociação.

Inicialmente todos os vértices do grafo são distribuídos randomicamente entre os diversos agentes. Cada agente passa então a tentar obter um conjunto de nodos próximos uns dos outros (minimizando o tempo entre duas visitas, e aumentar assim a função de utilidade do agente). Quando um agente detecta um nodo que não pode ser visitado em

[2] Outras soluções para o *TSP* serão apresentadas nos capítulos subsequentes

uma quantidade de tempo factível, ele inicia um leilão. Os outros agentes (licitantes) verificam se existe algum nodo que eles podem oferecer em troca do nó em leilão. Se for encontrado um nodo nessas condições, ele o oferece ao leiloador. Quando todos os nodos tiverem feito sua oferta, o nodo que fez o leilão, avalia a melhor oferta (o nodo-oferta mais próximo dos nós pertencentes ao seu conjunto atual), e faz a troca.

Destacam-se três arquiteturas de agentes negociadores:

- ***Two-shot-bidder Agent (TSBA)***: são ofertados somente nodos que incrementam a função de utilidade do agente. É promovido um leilão em dois lances, na tentativa de conseguir melhores ofertas.
- ***Mediated Trade Bidder Agent (MTBA)***: Nessa arquitetura, existe um agente *corretor* que informa ao leiloador qual agente pode fazer boas ofertas, e quais nodos podem ser trocados.
- ***Flexible Bidder Agent (FBA)***: As duas abordagens anteriores, trabalhavam com restrições estáticas no número de nodos que cada agente poderia trocar cada vez. No caso do FBA, semelhantemente, ao TSBA serão leiloados os piores nodos, considerando-se somente suas próprias funções de utilidade. Além disso poderão ser feitas trocas não paritárias, no sentido de que o leiloador e o licitante poderão receber em troca uma quantidade diferente de nodos do que a ofertada.

A arquitetura que apresentou melhores resultados foi a última técnica apresentada, ou seja, *Flexible Bidder Agent (FBA)*.

2.4 Cenários de Avaliação

Apesar do grande número de arquiteturas apresentadas nos trabalhos relacionados acima, elas foram avaliadas em apenas seis ambientes distintos (figura 2.2).

2.5 Avaliação Experimental

Todos os experimentos foram executados em um simulador próprio, construído exclusivamente para servir de apoio nos testes com as arquiteturas desenvolvidas para o problema em estudo. O gráfico na figura 2.3 apresenta o desempenho das técnicas que obtiveram os melhores resultados. Para cada um dos cenários apresentados na seção anterior foram executados dez experimentos por arquitetura, variando em cada um desses experimentos o nodo inicial dos agentes.

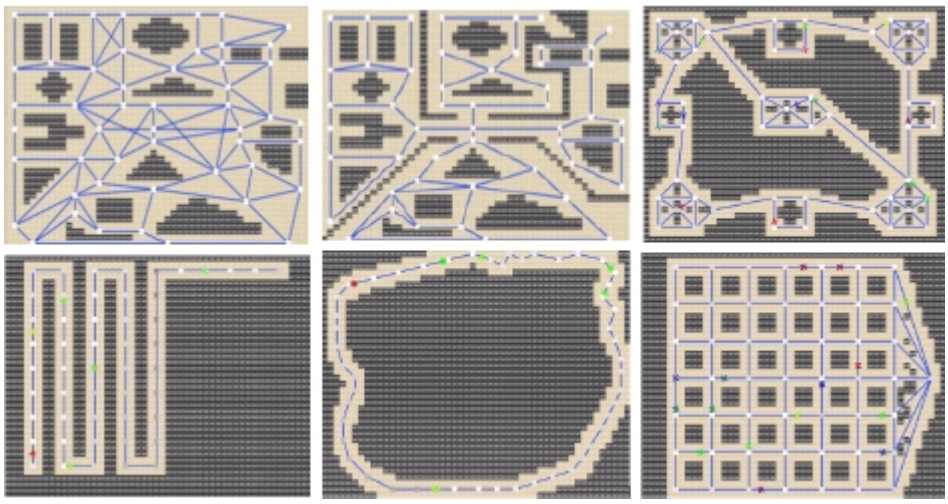


Figura 2.2: Cenários experimentais. Primeira linha: *Mapa A*, *Mapa B* e *Ilhas*; Segunda linha: *Corredor*, *Circular* e *Gride*.

Outro problema encontrado durante o processo de avaliação, foi definir o número de ciclos apropriados para a execução dos testes; baseando-se na pior ociosidade de todos os vértices, foi definido o valor de 15.000 ciclos.

Fica evidente que a estratégia que revelou os melhores resultados na maioria dos casos foi, conforme o esperado, aquela que é baseada no caixeiro viajante. E conforme discutido anteriormente, essa estratégia mantém um comportamento completamente estático durante toda a execução, já que encontrado um ciclo, ele é percorrido continuamente. Essa alta previsibilidade, conforme abordado na seção introdutória do trabalho, pode em muitos casos invalidar a ação de patrulhamento.

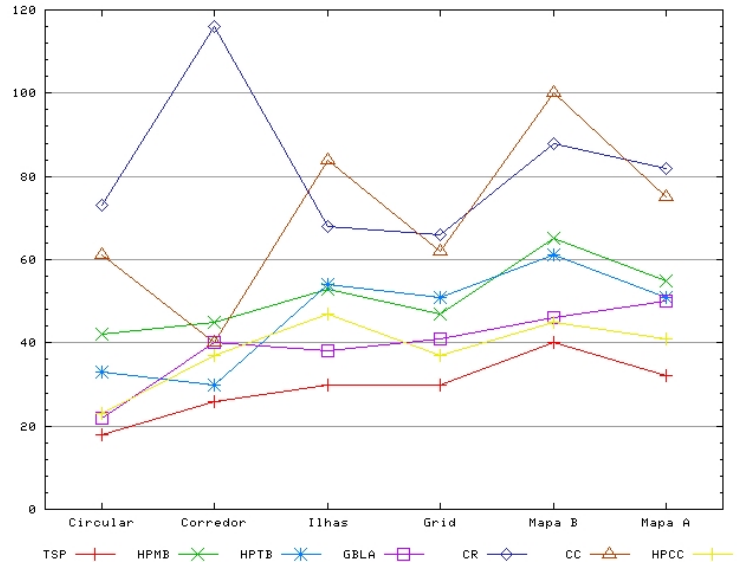


Figura 2.3: Gráfico demonstrativo dos resultados experimentais. CR: *Conscientious Reactive*, CC: *Cognitive Coordinated* e HPCC: *Heuristic Cognitive Coordinated*. (seção 2.3.1). GBLA: *Gray-Box Learner Agent* (seção 2.3.2). TSP: *Traveling Salesman Problem*. (seção 2.3.3) HPMB: *Heuristic Pathfinder Mediated Trade Bidder* e HPTB: *Heuristic Pathfinder Two-Shots bidder* (seção 2.3.4) (Este gráfico apresenta os resultados descritos em [1]).

Capítulo 3

Patrulhamento Multiagente

Este capítulo apresenta a nova abordagem dada ao problema do patrulhamento multiagente, propondo uma série de alterações e acréscimos nas tarefas que compõem sua resolução. Será mantida, no entanto, a ordem tradicional de exposição das seções relativas a este problema: o primeiro item a ser apresentado é a *representação de dados*, que aborda os motivos que fundamentam a escolha da estrutura utilizada, posteriormente são apresentadas algumas definições que decorrem da escolha dessa representação, além disso serão apresentados diversos conceitos inerentes a abordagem específica considerada nesta dissertação. Feito isso, serão introduzidos os *critérios de avaliação*, elementos que permitem a comparação entre as diversas soluções, além disso, esses critérios formalizam de maneira clara possíveis objetivos de uma patrulha, e portanto, definem quais serão as características desejáveis a serem consideradas durante o desenvolvimento de soluções. Posteriormente, baseando-se nas propriedades especificadas por meio dos critérios de avaliação serão apresentados os *algoritmos de seqüenciamento*, que buscam definir como será a ordem de visita aos locais que deverão ser patrulhados. É preciso ainda definir como será feita a distribuição de tarefas entre os diversos agentes. Detalhes dessa divisão de trabalho serão apresentados na seção que trata da *extensão multiagente*. Para finalizar serão apresentados os *cenários de avaliação*, ambientes criados artificialmente de acordo com diversos parâmetros, que possibilitam a avaliação experimental das soluções propostas de acordo com os critérios de avaliação definidos.

3.1 Representação de Dados e Principais Definições

Conforme mencionado anteriormente, o primeiro passo na resolução do problema de patrulhar determinada área refere-se à escolha de uma representação de dados apropriada. Esta estrutura deverá manter-se fiel às principais características da área especificada, além de garantir um desempenho computacional adequado. Na seção 2.1 desta dissertação

abordou-se o uso da técnica conhecida por *skeletonization*, utilizada com o fim de gerar uma abstração computacional de um ambiente real. Esta técnica produz como saída a forma geral do ambiente representada por meio desta abstração, fazendo uso, geralmente de um *grafo*. Baseando-se nisso, a estrutura de dados utilizada para representar um local a ser patrulhado, é um grafo. Segue a definição adotada ao longo de toda a dissertação. Um grafo $G(V, E)$ é composto por dois conjuntos V e E. V é um conjunto de elementos intitulados *vértices*, *nodos* ou *nós*; o segundo conjunto E contém as *arestas* de G. Uma aresta é um par não-ordenado $e = (v_i, v_j)$, onde $v_i, v_j \in V$. Costuma-se dizer que e é *incidente* aos vértices v_i e v_j, e que v_i e v_j são os *extremos* da aresta e. Se existe uma aresta ligando dois nodos quaisquer se diz que eles são vizinhos ou vértices adjacentes. Em nosso estudo serão utilizados somente *grafos simples*, o que significa que não existirão duas arestas diferentes incidentes ao mesmo par de vértices, chamadas de *arestas múltiplas* ou *paralelas*. Além disso em um grafo simples não são permitidos *laços*, que são arestas com extremos coincidentes, isto é, em um grafo simples $\forall e \in E | e = (v_i, v_j) \Rightarrow v_i \neq v_j$. Além disso optou-se pelo uso exclusivo de *arestas não orientadas*, de forma que, toda aresta poderá ser percorrida em ambos os sentido ($v_i \rightarrow v_j$ ou $v_j \rightarrow v_i$). No grafo gerado a partir do processo de abstração de um ambiente qualquer os vértices representarão locais específicos que requerem patrulhamento, já as arestas definirão os caminhos possíveis entre esses locais, e o *custo* ou *peso* associado a cada uma das arestas (ω_e ou $\omega(v_i, v_j)$) está relacionado com o tempo necessário para ir do nodo origem ao nodo destino, essa medida de tempo é proporcional à distância euclidiana entre os nodos.

Apresentam-se, adiante, os detalhes relativos a abordagem seguida na resolução do problema do patrulhamento: A unidade de tempo utilizada nas simulações foi a *iteração*, dessa forma todo valor associado a tempo será medido em número de iterações, por exemplo, o tempo necessário para um agente ir de um nodo (v_i) até um de seus nodos vizinhos (v_j) será dado em iterações. Um dos principais conceitos utilizados no decorrer do trabalho refere-se ao *tempo de espera* que é o intervalo de tempo, portanto o número de iterações, entre duas visitas, geralmente contíguas, a um determinado vértice. Este conceito é muito semelhante à definição de *ociosidade* apresentado na seção de trabalhos correlatos. O momento (ou iteração) no qual o vértice v recebeu sua i-ésima visita será representado por $\theta_v(i)$, já $\Delta_v(i, j)$ é o tempo de espera entre a i-ésima e a j-ésima visita ao nodo v (veja a equação 3.1). Durante a simulação, tendo em vista calcular estatísticas utilizadas por algoritmos de seqüenciamento será necessário obter o instante em que um vértice v recebeu sua última visita, esse valor será dado por θ_v; N_v é o número de visitas que v recebeu durante toda a execução; $\mu_v(i)$ é a média do tempo de espera de v até sua i-ésima visita e é definida pela equação 3.2. Em especial $\mu_v(N_v)$ denotada simplesmente por μ_v, é o tempo de espera médio de v durante toda a simulação; $\sigma_v(i)$ refere-se ao desvio padrão do histórico de intervalos de tempo entre as visitas ao nodo v, levando

em consideração somente as observações anteriores a i-ésima visita (veja a equação 3.3). Novamente, $\sigma_v(N_v) = \sigma_v$ diz respeito ao desvio padrão do conjunto total de observações do vértice v, durante toda a simulação.

$$\Delta_v(i_x, i_y) = \theta_v(i_y) - \theta_v(i_x) \tag{3.1}$$

$$\mu_v(i) = \frac{\sum_{j=1}^{i-1} \Delta_v(j, j+1)}{i-1} = \frac{\Delta_v(1, i)}{i-1} \tag{3.2}$$

$$\sigma_v(i) = \sqrt{\frac{1}{i-2} \sum_{j=1}^{i-1} (\Delta_v(j, j+1) - \mu_v(i))^2} = \sqrt{\frac{1}{i-2} \left(\sum_{j=1}^{i-1} \Delta_v(j, j+1)^2 - i\mu_v(i)^2 \right)} \tag{3.3}$$

3.2 Critérios de Avaliação

Os *critérios de avaliação* representam um dos papeis mais importantes no estudo do patrulhamento, por meio deles, são definidas as medidas de desempenho para toda e qualquer arquitetura proposta. Por meio destes critérios de avaliação foi possível realizar uma comparação precisa, verificando em quais situações o uso de cada uma das arquiteturas é mais adequado. Os critérios de avaliação estão intimamente ligados a modelagem de atacantes e seus respectivos ataques, isso porque a medida de eficiência do patrulhamento está diretamente relacionada a eficiência com que eventuais invasores sejam descobertos.

Destacando-se como uma das principais contribuições desse trabalho, são propostos três critérios de avaliação novos, em uma escala crescente do nível de complexidade do atacante. Cada um dos critérios avaliará as metodologias de patrulhamento propostas posteriormente, baseando-se para isso em contextos diferentes de invasão.

Nesse estudo foi considerado o caso particular no qual a intrusão refere-se a um objetivo específico, em um único local, requerendo uma quantidade de tempo determinada para ser concretizada com sucesso. Como em geral não existe qualquer informação a respeito de um possível local a ser alvo da intrusão, ou seja, todos os nodos da região sob patrulha são igualmente prováveis, sem qualquer região de patrulhamento prioritário, foi decidido que durante a simulação o ataque seria feito a um nodo escolhido de forma aleatória. Portanto, para todos os critérios de avaliação, foram medidos o desempenho de cada uma das metodologias propostas em conter ataques realizados a um único local específico, local esse definido previamente de maneira randômica.

Outro aspecto muito relevante na definição dos critérios de avaliação, diz respeito ao *intervalo de intrusão* que é o tempo do qual o invasor dispõe para realizar seu ataque. Se durante o período de ataque, um agente visitar o nodo atacado, a invasão é considerada

mal sucedida e portanto, o patrulhamento foi bem sucedido, ao contrário, se durante todo o intervalo da invasão não houver qualquer visita, o ataque terá sido realizado bem sucedido, enquanto o patrulhamento será considerado ineficiente. Buscando uma avaliação criteriosa, o intervalo de intrusão foi definido com base em duas variáveis, inerentes aos aspectos da simulação, bem como uma constante, que visa variar os intervalos a serem comparados. O primeiro parâmetro busca normalizar o intervalo de intrusão com relação a grafos de topologias e tamanhos completamente distintos, para isso, o cálculo do intervalo de intrusão é diretamente proporcional ao tempo necessário para percorrer o ciclo do TSP do grafo (o ciclo de menor tamanho que contém uma única visita a cada um dos vértices do grafo). Como são executadas simulações com um número variável de agentes, o segundo parâmetro torna o intervalo inversamente proporcional ao número de agentes na simulação ($\mathcal{N}_{agentes}$). Para permitir uma comparação precisa entre variações bem definidas do intervalo de intrusão, foi definida uma constante ($\mathcal{C}_{FRAC}$), com a alteração desse parâmetro é possível avaliar o comportamento das metodologias propostas de acordo com uma variação constante do intervalo de invasão. Por fim, a função ($\mathcal{I}_{FRAC}$) que define o valor do intervalo de invasão para um determinado grafo, no caso específico de uma simulação, é calculada da seguinte maneira:

$$\mathcal{I}_{FRAC} = \mathcal{C}_{FRAC} \times \frac{|\mathcal{TSP}_{ciclo}|}{\mathcal{N}_{agentes}} \tag{3.4}$$

Considere, por exemplo, que o ciclo do TSP exija 80 iterações para ser percorrido completamente, em uma execução com 2 agentes, e o seguinte conjunto de valores para a constante de variabilidade $\mathcal{C}_{FRAC} = \{\frac{1}{4}, \frac{1}{2}, 1\}$, calculando tem-se:

$$I_{\frac{1}{4}} = \frac{1}{4} \times \frac{80}{2} = 10; I_{\frac{1}{2}} = \frac{1}{2} \times \frac{80}{2} = 20; I_1 = 1 \times \frac{80}{2} = 40. \tag{3.5}$$

Assim, para esse grafo, nesse caso particular da simulação, cada um dos critérios de avaliação irá comparar os resultados das metodologias levando em consideração os seguintes valores para os intervalos de invasão $I = \{10, 20, 40\}$.

Conforme mencionado anteriormente, os atacantes serão apresentados de acordo com uma ordem crescente de complexidade, este grau de complexidade está diretamente relacionado à quantidade de informação da qual o intruso tem acesso no planejamento de seu ataque. Dessa forma, o primeiro intruso apresentado não utiliza ou possui quaisquer informações à priori do local a ser atacado, já o segundo e o terceiro modelos de atacantes podem observar e coletar informações a respeito do local onde pretendem realizar a invasão. A principal diferença entre esses dois modelos de atacantes consiste na janela de tempo das observações utilizada para planejar o ataque, enquanto, o *intruso escondido* faz uso apenas da informação referente a última visita por um agente, o *intruso estatístico*

utiliza-se de um histórico (geralmente longo) das visitas para planejar o momento adequado de um ataque.

As próximas seções definem de forma específica cada um dos critérios de avaliação, modelando-os conforme cada atacante específico:

3.2.1 Intruso Randômico

O ***intruso randômico*** não possui qualquer informação referente a metodologia de patrulhamento escolhida, por conta disso, a sua decisão em definir o momento no qual a invasão será iniciada é puramente aleatória. Em um ambiente real de patrulhamento, esse tipo de invasão está correlacionada a locais que impossibilitem completamente um planejamento prévio por parte dos invasores ao impedir qualquer observação externa desse ambiente. É importante destacar, no entanto, que esse tipo de modelo, poderá ser aplicado apenas em casos muito particulares do patrulhamento de ambientes reais, isso porque muito dificilmente, um local poderá ser completamente isolado a ponto de impossibilitar observação externa. Outra consideração relevante é a de que ataques sem planejamento prévio baseado na observação do local a ser invadido, estão geralmente, ligados a alvos de menor valor, com pouca ou até mesmo inexistente patrulha, já alvos com valores elevados, e portanto, objetos que necessitam prioritariamente de patrulhamento, são alvos de ataques bem elaborados, com observação e planejamento estruturado por parte dos atacantes. Por isso as próximas seções tratarão de invasores que consideram o uso de informações relativas ao ambiente a ser atacado no planejamento de suas invasões, o que torna a tarefa de patrulhamento uma tarefa mais complicado do que minimizar o tempo entre as visitas a um determinado nodo.

É importante notar que independentemente do tamanho do intervalo de invasão, a heurística que maximiza a chance de um intruso randômico ser capturado durante seu ataque é aquela na qual ocorrem um maior número de visitas ao nodo avaliado. No entanto, para que o acréscimo do número de visitas ao vértice aumente as chances de uma invasão ser contida, essas visitas deverão ser bem distribuídas durante todo o tempo da simulação. Além disso, como a escolha do nodo a ser avaliado também é aleatória, a heurística deverá se preocupar em maximizar o número de visitas para todos os vértices do grafo. Dessa forma, fica evidente a similaridade desse critério de avaliação, com os critérios apresentados nos trabalhos correlatos, onde de forma geral, buscou-se reduzir a ociosidade média dos nodos de um grafo, ou seja, primou-se pela eficiência em reduzir o tempo entre as visitas a cada um dos nodos.

Fundamenta-se agora a decisão por utilizar-se de uma escolha aleatória para definir o momento no qual terá início a tentativa de invasão, ao invés de avaliar diretamente todos os passos da simulação, como foi feito nos trabalhos citados anteriormente. Além da

similaridade com o caso real de invasores com comportamento semelhante ao randômico, a principal motivação para o uso de escolhas aleatórias foi a grande aceitação e uso do método estatístico de *Monte Carlo* em simulações. A idéia fundamental do método de Monte Carlo é estimar o valor de uma função calculando-se para isso, o valor exato, apenas para um conjunto restrito de amostras aleatórias do domínio da entrada de dados. Não existe uma definição exclusiva que determine os passos seguidos no método de Monte Carlo, existem no entanto, conjuntos de classes amplamente utilizadas, essas abordagens costumam seguir um padrão particular de passos, como descrito a seguir:

1. Definir um domínio de possíveis entradas;
2. Escolher randomicamente algumas entradas do domínio definido no passo anterior, e executar uma computação determinística nessas entradas;
3. Reunir os resultados computacionais individuais em um resultado final completo.

Dois dos exemplos clássicos muito utilizados para demonstrar a aplicabilidade do método de Monte Carlo são os cálculos aproximados de π e da área de determinada região. Detalhes do método podem ser encontrados em [9].

3.2.2 Intruso Escondido (ou Observador)

O *intruso observador* irá esperar, escondido, monitorando o local alvo de seu ataque, e logo após toda visita de um agente ao vértice escolhido o intruso irá iniciar o ataque. Dessa forma, esse critério de avaliação está relacionado, evidentemente, a ambientes onde é possível observar o local alvo do ataque, antes de realmente atacá-lo. Esse atacante, no entanto, ainda faz um uso bastante limitado das informações que lhe são disponíveis. Por isso, o próximo atacante, além de observar, mantém um histórico e através de medidas estatísticas utiliza das informações relevantes da patrulha para um ataque ainda mais consistente.

No caso específico do intruso randômico, apresentado anteriormente, o ataque poderia ocorrer a qualquer momento, portanto, a visita de patrulhadores ao nodo não influenciava em nada a decisão do instante em que o ataque seria iniciado. Nesse caso, diminuir o tempo entre as visitas dos patrulhadores foi apontado como o principal objetivo a ser alcançado pelas arquiteturas de seqüenciamento, para obter sucesso no patrulhamento. Já no caso particular do intruso escondido, diminuir o tempo entre as visitas pode não ser o suficiente para que o patrulhamento seja bem sucedido. Isso porque, uma solução ótima que minimiza o tempo entre as visitas para todos os vértices será estática, mantendo portanto, um período constante entre as visitas a um mesmo nodo. Dessa forma, sempre que esse período constante for maior que o tempo necessário para a invasão, o ataque será

bem sucedido. Existem muitos casos onde a tarefa de invasão pode ser dividida em várias etapas, sem que o concluir de uma etapa seja visível a um agente que esteja fazendo o patrulhamento. Nesse caso seria extremamente razoável que o intruso dividisse o ataque e a cada etapa concluída voltasse a ficar escondido, esperando uma nova visita de um agente para voltar a atacar e passar ao próximo passo. Por exemplo, suponha um caso trivial onde um intruso deseja abrir um cofre tentando cada uma das combinações possíveis, nesse caso ele poderá dividir sua invasão ao fazer de cada uma das etapas a tentativa de um conjunto pequeno de combinações. A cada conjunto completo, o atacante volta a ficar escondido esperando a próxima visita de um agente para passar ao próximo conjunto de tentativas. De forma que, apesar do patrulhamento ser possivelmente ótimo do ponto de vista da eficiência em cobrir os locais sob patrulha, ele não conterá um ataque, desde que esse ataque possa ser dividido em curtos períodos de tempo. Para impedir que esse tipo de estratégia de invasão possa ser seguida, propõem-se o uso de arquiteturas com variabilidade. Um algoritmo de seqüenciamento com alta variabilidade torna possível a captura de intrusos escondidos, porque embora, em alguns momentos o tempo de espera no nodo alvo seja superior ao intervalo entre visitas existente no caso da alta eficiência, frequentemente durante a simulação o tempo de espera poderá ser bem inferior, ao ponto de tornar possível capturar o invasor mesmo quando o intervalo de intrusão for pequeno.

3.2.3 Intruso Estatístico

O *intruso estatístico* irá coletar estatísticas relacionadas com o período de espera entre as visitas ao nodo escolhido e só iniciará um ataque quando os dados assegurarem que a invasão será bem sucedida. Para isso o intruso estatístico mantém um histórico de visitas e por meio de uma *regressão linear simples* e seu respectivo *intervalo de confiança*, o período do qual o atacante dispõe para um eventual ataque é estimado. Considere que o último período de espera foi p_i, é feita então uma regressão linear tendo como base o histórico das visitas até o penúltimo período de espera (p_{i-1}), o valor de p_i é então utilizado para estimar por meio da regressão qual será o tempo de espera até a próxima visita (p_{i+1}). Buscando garantir a eficácia da estimativa feita fez-se uso do intervalo de confiança da regressão linear. Considera-se então o valor do limite inferior do intervalo de confiança para a estimativa feita ($\mathcal{IC}_{inf}(p_{i+1})$). Se esse valor for superior ao tempo necessário para invadir com sucesso, o intruso fará seu ataque, caso contrário, ele esperará pela próxima visita de um agente quando repetirá todo o processo na tentativa de realizar uma invasão. Evidentemente, para permitir o uso adequado da regressão linear será necessário possuir um histórico coerente de observações, por isso, antes de tentar atacar, o intruso estatístico, permanece durante determinado tempo somente coletando informações de visitas de patrulhadores ao nodo, criando assim um histórico inicial que

será utilizado pela regressão linear na predição.

Tendo em vista o uso da regressão linear, serão apresentados, brevemente, alguns detalhes relativos a esse método: A modelagem por meio de uma regressão linear representa um poderoso método para a estimativa dos valores assumidos por uma variável numérica, e representa um dos métodos amplamente utilizados em analises estatísticas. A idéia básica pode ser expressa como: estimar o valor esperado de uma variável por meio de uma combinação linear de outros atributos, cada um deles com determinado peso. No caso específico onde utiliza-se um único atributo para predizer o valor de uma variável, a regressão é dita *simples*. A regressão é chamada *linear* por que considera a relação entre as variáveis e a resposta como sendo uma função linear. Outra característica importante vem do fato de que uma *regressão linear simples* pode ser graficamente expressa, através de uma linha reta que aproxima a relação entre uma variável preditora e o valor estimado.

Para fazer a estimativa é utilizada a seguinte equação linear (trata-se precisamente da equação da reta):

$$y_i = \beta_1 + \beta_2 x_i + \epsilon_i$$

onde x_i é a variável independente, y_i é a variável dependente, a ser estimada a partir do valor de x_i, β_1 é uma constante que representa numericamente o ponto de interceptação da reta com o eixo vertical, β_2 é outra constante que define o ângulo de declive da reta e ϵ_i é o erro observado.

É importante destacar que no caso específico da utilização da regressão por um atacante para predizer o próximo período entre visitas, o conjunto de dados utilizados para calcular os valores das constantes β_1 e β_2 será dado pelo histórico existente até o momento. Considere, por exemplo, que até o momento aconteceram N_v visitas ao nodo, de forma que $\mathcal{M}_v = \{i_1, i_2, i_3, \ldots, i_{N_v}\}$, e portanto, tem-se um histórico com $N_v - 1$ itens que representam os intervalos entre as visitas recebidas por v ($\Delta_v = \{\delta_1, \delta_2, \ldots, \delta_{N_v-1}\}$). Como o objetivo do uso da regressão é a estimativa do próximo intervalo de espera, as variáveis utilizadas no cálculo das constantes da regressão serão determinadas da seguinte maneira: $X = \{\delta_1, \delta_2, \ldots, \delta_{N_v-2}\}$ e $Y = \{\delta_2, \delta_3, \ldots, \delta_{N_v-1}\}$. De forma que para cada item $x_i = \delta_i$ tem-se $y_i = \delta_{i+1}$.

Os cálculos estimados dos valores de β_1 e β_2 geralmente são feitos através do *método dos mínimos quadrados*, que faz a estimativa em função dos valores conhecidos de X e Y (n é o número de observações existentes em cada um dos conjuntos, no caso específico do patrulhamento $n = N_v - 2$), seguem as equações:

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n} x_i^2 \times \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} (x_i \times y_i) \times \sum_{i=1}^{n} x_i}{n \times \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \tag{3.6}$$

$$\hat{\beta_2} = \frac{n \times \sum_{i=1}^{n} (x_i \times y_i) - \sum_{i=1}^{n} x_i \times \sum_{i=1}^{n} y_i}{n \times \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x)^2} \tag{3.7}$$

Além da predição através da regressão linear simples, fez-se uso do *intervalo de confiança* do valor estimado. O intervalo de confiança prove uma maneira de assegurar a qualidade da predição feita. A seguinte equação define o cálculo do intervalo de confiança:

$$\hat{y} \pm t\left(1 - \frac{\alpha}{2}; n-2\right) \times \sigma_{predicao} \tag{3.8}$$

onde $\hat{y}$ é o valor estimado de y para a nova observação ($\hat{x}$), calculado por meio da equação 3.2.3. α corresponde ao nível de confiança desejado (também chamado de coeficiente de confiança), $t(\alpha, \theta)$ corresponde ao valor da tabela de distribuição *t-student* com nível de confiança α, e θ graus de liberdade. Finalmente $\sigma_{predicao}$ é um estimador não tendencioso dado por:

$$\sigma_{predicao} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \mu_y)^2}{n-2} \times \left(1 + \frac{1}{n} + \frac{(\hat{x} - \mu_x)^2}{\sum_{i=1}^{n}(x_i - \mu_x)^2}\right)} \tag{3.9}$$

O uso de um modelo de predição como uma regressão linear simples busca compreender o comportamento adotado por parte do conjunto de agentes nas visitas ao nodo alvo do ataque. Conforme mencionado anteriormente, um algoritmo de seqüenciamento eficiente está geralmente atrelado a um comportamento altamente previsível e por isso ineficiente perante um atacante que use um modelo preditor. Já a variação, apesar de possivelmente dificultar a previsão de modelos, pode não ser suficiente para tornar o modelo imprevisível. Por exemplo, modelos com variação associada a processos puramente determinísticos em geral, são exemplos de modelos variáveis que são previsíveis. Um dos algoritmos de seqüenciamento proposto (*TSP Original e acréscimo de variabilidade*), faz uso de um modelo com alta variabilidade e ainda assim na maioria dos casos suas ações podem ser facilmente previstas. Dessa forma, geralmente, processos que incluam imprevisibilidade estão relacionados com algum grau de aleatoriedade. Dessa forma, em geral as soluções que obtém bons resultados para esse tipo de atacante são aquelas que incluem componentes aleatórios.

Finda a definição dos critérios de avaliação, apresentam-se, a partir desse ponto, as soluções propostas, iniciando com os *algoritmos de seqüenciamento*.

3.3 Algoritmos de Seqüenciamento

Um algoritmo de *seqüenciamento* está diretamente relacionado à necessidade de definir o percurso a ser seguido por cada um dos agentes envolvidos no patrulhamento, por isso, em termos gerais o algoritmo de seqüenciamento determina qual será a ordem na qual os vértices serão visitados. Alguns dos algoritmos de seqüenciamento apresentados definem a

cada passo qual será o próximo nodo a ser escolhido, a maioria dos algoritmos, no entanto, definem objetivos a longo prazo.

Na seção anterior destacou-se que cada um dos critérios de avaliação propostos irão exigir comportamentos diferentes por parte dos agentes patrulhadores. *Intrusos randômicos*, por exemplo, demandam agentes que minimizem o tempo entre as visitas a todos os vértices do grafo, já os *intrusos escondidos*, requerem variabilidade além da eficiência na ação do patrulhadores. Por fim, no caso dos *intrusos estatísticos* é necessário que os agentes vigilantes façam uso de modelos impredizíveis. Tendo em vista a grande diferença nas exigências feitas por cada um dos modelos de intrusos para com os agentes patrulhadores, um algoritmo de seqüenciamento robusto deverá ser capaz de atender essas demandas diferenciadas. De forma que, a arquitetura deverá balancear eficiência com imprevisibilidade (e portanto, variabilidade). A imprevisibilidade bem como a variabilidade é geralmente alcançada incluindo-se componentes randômicos ao processo, por isso o primeiro grupo de soluções baseia-se exatamente em um conjunto de soluções que fundamentam-se prioritariamente na aleatoriedade (seção 3.3.1). De outro lado, aumentar a eficiência geralmente requer soluções exatas e determinísticas, por isso, o segundo grupo de arquiteturas propostas baseia-se em percorrer o ciclo do TSP, adicionando no entanto mudanças significativas nesse processo, para garantir a imprevisibilidade (seção 3.3.2).

É importante destacar que se houverem múltiplos agentes em uma simulação cada um deles agirá de forma completamente independente, não influenciando nas decisões dos outros agentes, isso porque, as soluções propostas nesta dissertação são fundamentalmente não centralizadas do ponto de vista de sistemas multiagentes. Os detalhes relativos ao uso de diversos agentes patrulhadores durante a simulação serão considerados na seção 3.4.

3.3.1 Arquiteturas Fundamentalmente Randômicas

Existem importantes áreas de estudos que avaliam o uso de soluções randômicas em sistemas computacionais, a vertente que trata especificamente do uso da aleatoriedade em simulações semelhantes ao nosso objeto de estudo é conhecido por *random walk*. Os estudos apresentam importantes aplicações em ciência da computação, matemática e física. A idéia intuitiva no *random walk* é solucionar um problema tomando-se para isso, passos sucessivos, cada um em uma direção randômica. Um estudo específico do uso desse método para grafos é apresentado em [2]. Existem resultados teóricos que apresentam uma medida de tempo com a qual um vértice qualquer de um grafo será alcançado, são definidos também limites inferiores e superiores para a cobertura completa de um grafo (a cobertura completa, nesse caso, esta relacionada a visita de todos os vértices de um grafo).

Tomando por base a garantia de que todo vértice será visitado, pode-se considerar o

uso de arquiteturas fundamentalmente randômicas no processo de patrulha. Propõem-se a seguir, duas arquiteturas:

- **Arquitetura randômica local:** A escolha do próximo nodo a ser visitado será realizada de forma totalmente aleatória, restringindo, no entanto, essa decisão a qualquer um dentre os nodos que sejam vizinhos ao vértice corrente do agente patrulhador. Nesse caso, o agente é dito *reativo*, pois ele age baseado-se exclusivamente em sua percepção atual, percepção essa relacionada com o campo de visão do agente que é limitado a apenas um vértice de profundidade.

- **Arquitetura randômica global:** Novamente, a escolha do próximo nodo a ser visitado é feita de maneira completamente randômica, nesse caso no entanto, não existem restrições quanto aos nodos que podem ser escolhidos, e portanto, qualquer nodo do grafo poderá ser selecionado como o próximo vértice a ser visitado. No caso de não existir uma aresta que conecte o nodo corrente ao vértice objetivo, o agente irá percorrer o *caminho mais curto* entre eles ([6] apresenta os principais algoritmos para encontrar o caminho mais curto entre dois vértices quaisquer). Diferentemente da arquitetura anterior o agente nesse caso é chamado de *cognitivo*, pois ele pode perseguir objetivos bem definidos, com uma percepção que vai além dos nodos vizinhos da posição atual. É importante notar que o uso do caminho mais curto na solução irá algumas vezes privilegiar certos nodos que com freqüência façam parte desses caminhos entre outros nodos. Além disso o uso de caminhos mais curtos está diretamente relacionado a alguns fatores como o grau dos vértices, isso porque, quanto menor o número de arestas, mais provável será que a escolha implicará no uso de um caminho não direto.

Em ambos os casos, o processo de escolha do próximo nodo a ser visitado será repetido continuamente até o término do tempo da simulação. A principal diferença entre as arquiteturas definidas é a de que a segunda perspectiva mantém um estímulo extra para que se visite todos os nodos repetidamente, isso porque, se os valores randômicos são bem distribuídos, todos os vértices serão escolhidos com uma proporção semelhante.

3.3.2 Arquiteturas baseadas no ciclo do TSP

De acordo com o que foi descrito anteriormente, a arquitetura baseada exclusivamente no ciclo do *caixeiro viajante* (**T**ravelling **S**alesman **P**roblem) alcança a máxima eficiência possível, em diminuir o tempo entre visitas para todos os nodos do grafo, no entanto, como os critérios de avaliação propostos demandam soluções que incluam variabilidade e imprevisibilidade, é preciso adaptar o ciclo para que ele possa incluir essas características,

por isso propõem-se três arquiteturas básicas que modelam alterações no ciclo original do TSP. Além disso, para cada uma das arquiteturas básicas propõem-se novas alterações que visam aumentar a variabilidade das soluções, para isso ao invés de utilizar-se unicamente de escolhas aleatórias no processo de incluir a variabilidade nos sistemas, algumas das decisões são tomadas de forma determinística, visando o aumento da variabilidade.

É importante destacar, que encontrar o ciclo do TSP é um problema *NP-Difícil*, isto significa que encontrar a solução para instâncias de dados grandes pode demandar muito tempo. Existem, no entanto, algumas abordagens que permitem resolver instâncias médias (aproximadamente 250 vértices) em um tempo razoável. Evidentemente, as arquiteturas baseadas no ciclo do TSP que serão apresentadas ficam atreladas a domínios de entrada de dados para os quais seja possível resolver o problema satisfatoriamente. Em particular, nos experimentos descritos nessa dissertação, para encontrar o ciclo do TSP foi usado um método que usa um relaxamento *1-tree* em um algoritmo *branch and bound* com poda, os detalhes do método são apresentados em [28, 29, 14]. Apresenta-se agora, brevemente, o funcionamento do método:

Seja G um grafo completo não-orientado com n vértices, e V o conjunto de vértices de G. Uma 1-árvore geradora mínima (*minimal spanning 1-tree*) em G refere-se a uma árvore gerador mínima em $V \setminus \{v_i\}$ combinada com as duas arestas de menor peso do vértice v_i – $V \setminus \{v_i\}$ refere-se ao conjunto de vértices V excluindo-se o nodo v_i. Pode-se então reformular o problema do TSP (simétrico) como sendo: encontrar uma 1-árvore geradora miníma com uma restrição adicional de que todo vértice terá que necessariamente possuir grau igual a dois. O algoritmo utilizado conforme o mencionado é dividido em dois passos, o primeiro é conhecido por *branching*, enquanto o segundo é chamado de *bound*. O passo relativo ao *branching* no algoritmo, leva ao particionamento do conjunto corrente de soluções factíveis em subconjuntos disjuntos. Este processo é feito encontrando nodos em uma 1-árvore com grau maior que dois e determinando que arestas incidentes a esse vértice deverão ser exigidas ou proibidas. Por fim faz-se uso do segundo passo (*bound*), tendo em vista um limite superior é possível eliminar algumas das soluções garantidamente não ótimas. As heurísticas utilizadas para calcular esse limite superior são versões simplificadas dos algoritmos de Christofides (1976) e Lin (1965) encontrados em [24]. Além de utilizar os passos de *branching and bound*, identificando-se arestas que podem ser eliminadas ou que são garantidamente parte da solução o número de arestas disponíveis é reduzido drasticamente. Isto é feito avaliando-se trocas de arestas na melhor 1-árvore, procurando por arestas não ótimas usando condições e testando a factibilidade baseando-se em algumas proibições, este passo é conhecido por poda, maiores detalhes podem ser encontrados em [14].

Passa-se agora a enunciar os detalhes das alternativas propostas que baseiam-se no ciclo do TSP:

TSP com visitas locais

Nessa arquitetura o ciclo do TSP é percorrido repetidamente, mas sempre que um vértice é visitado decide-se randomicamente com probabilidade $\mathcal{CH}_{VL}$ se uma *visita local* será feita ou não. Uma *visita local* consiste em visitar um dos nodos adjacentes ao vértice corrente, depois dessa visita o agente retorna ao nodo original e continua a percorrer o ciclo do TSP (veja a figura 3.1). O parâmetro $\mathcal{CH}_{VL}$ é definido antes do início da execução e permanece inalterado até o fim da mesma.

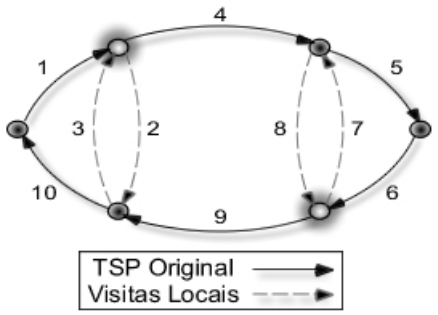


Figura 3.1: Exemplo de um ciclo da arquitetura *TSP com visitas locais*. Os nodos destacados exemplificam vértices nos quais foi decidido fazer uma *visita local*, os números em cada uma das arestas referem-se a ordem na qual cada uma dessas arestas foi percorrida.

Suponha, por exemplo, que o grafo em estudo possui seis vértices, e seu ciclo do TSP é definido pela seguinte ordenação dos nodos: $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, além disso, $\mathcal{CH}_{VL} = 35\%$. A simulação inicia-se em um vértice qualquer do grafo, por exemplo v_1, então é decidido se será feita uma visita local, para isso é escolhido aleatoriamente um número $\mathcal{N}_1$ de forma que

$$0 \leq \mathcal{N}_1 \leq 100$$

se $\mathcal{N}_1 \leq \mathcal{CH}_{VL}$ far-se-á uma visita local. Suponha $\mathcal{N}_1 = 80$, então não será realizada uma visita local, e portanto, o próximo nodo na ordem definido pelo ciclo do TSP será visitado, ou seja, o agente visitará v_2. Novamente, será escolhido um valor aleatório para $\mathcal{N}_1$, por exemplo, $\mathcal{N}_1 = 25$, e portanto optou-se por uma visita local, para escolher o vértice a ser visitado um novo valor aleatório é escolhido tal que

$$0 \leq \mathcal{N}_2 \leq \delta(v_2)$$

onde $\delta(v_2)$ é o grau do vértice v_2, nodo no qual o agente encontra-se atualmente. Busca-se então na lista de adjacência do vértice v_2 o nodo de índice $\mathcal{N}_2$, por exemplo v_6, e então faz-se a visita $v_2 \rightarrow v_6$, e depois $v_6 \rightarrow v_2$. Feita a visita local, novamente o próximo nodo

do ciclo do TSP será visitado (v_3). O algoritmo continuará dessa maneira, seguindo o ciclo do TSP, e optando de acordo com o parâmetro $\mathcal{CH}_{VL}$ se fará ou não visitas locais à vértices adjacentes.

TSP com visitas locais e acréscimo de variabilidade

Conforme mencionado na introdução desta seção, para cada uma das arquiteturas apresentadas foram modeladas alterações que visavam aumentar a variabilidade da arquitetura. Essa arquitetura portanto, é uma extensão da última arquitetura apresentada (*TSP com visitas locais*) e, novamente, fundamenta-se na alteração do ciclo do TSP com a inclusão de visitas locais. A decisão de fazer uma visita local também baseia-se em uma constante predefinida ($\mathcal{CH}_{VL}$), que determina de maneira probabilística a freqüência com que serão realizadas visitas locais. Portanto, como na arquitetura anterior, ao se percorrer o ciclo do TSP cada vez que um nodo for visitado será decidido aleatoriamente (tendo por base $\mathcal{CH}_{VL}$) se será feita ou não uma visita local. No entanto, sempre que optar-se pela realização de uma visita local, ao invés de escolher randomicamente o nodo adjacente a ser visitado, a escolha será feita de maneira determinística, optando-se sempre pela visita que trará o maior aumento do desvio-padrão do tempo entre visitas (do nodo escolhido). Para encontrar esse vértice, serão necessários os dados referentes à média e o desvio-padrão de todos os nodos adjacentes ao vértice atual, o cálculo dos dois estimadores foram feitos conforme as equações 3.2 e 3.3, respectivamente. Evidentemente, é necessário possuir um histórico de observações razoavelmente longo até que seja possível obter valores não tendenciosos da média e do desvio padrão. Por esse motivo faz-se necessário uma fase de inicialização, que deve garantir um histórico com amostras suficientes e coerentes. Evidentemente, durante a construção desse histórico inicial de visitas, não será possível utilizar-se do processo de escolha determinística dos nodos adjacentes a receberem uma visita local, por esse motivo, decidiu-se pela utilização do histórico de visitas completo resultante da simulação da arquitetura anterior (*TSP com visitas locais*) como histórico inicial, sendo esse naturalmente, um conjunto grande e coerente de amostras, já que os dados provem de uma execução verdadeira. Os passos necessários, para determinar o nodo adjacente com maior ganho de variação, são os seguintes (o nodo atual será representado por v_a e os nodos vizinhos, portanto, candidatos, serão representados por v_b):

1. Para cada um dos nodo adjacentes são calculados a média (μ_{v_b}) e o desvio-padrão (σ_{v_b}) do histórico contendo o tempo de espera entre visitas. Além disso serão obtidos o momento no qual o vértice recebeu sua última visita (θ_{v_b}) e o instante no qual a execução encontra-se atualmente será dado por θ.

2. Calcula-se então $\Delta'_{v_b} = (\theta + \omega(v_a, v_b)) - \theta_{v_b}$, onde $\omega(v_a, v_b)$ é o peso associado a aresta que liga o vértice origem ao nodo adjacente. Dessa forma, ao somar-se o instante

atual ao tempo necessário para ir do nodo corrente até o vértice em análise, simula-se uma visita ao vértice, subtrai-se então o momento em que v_b recebeu sua última visita, para assim obter-se o intervalo entre a última visita real e a visita simulada (Δ'_{v_b}).

3. Feito isso, calcula-se $\sigma'_{v_b} = |\Delta'_{v_b} - \mu_{v_b}|$, assim encontra-se o desvio do tempo de espera da visita simulada para com a média do tempo de espera do nodo.

4. Para finalizar obtem-se o ganho de variação resultante da visita ao nodo v_b por meio do seguinte cálculo $\mathcal{G}_{v_b} = \sigma'_{v_b} - \sigma_{v_b}$.

5. O nodo com o maior valor $\mathcal{G}_{v_b}$ será escolhido como vértice a ser visitado.

Naturalmente, pode acontecer que $\mathcal{G}_{v_b}$ seja inferior a zero para todo vértice v_b adjacente a v_a, nesse caso, naturalmente, a visita a nenhum dos vértices resultará em um aumento do desvio padrão, quando isso ocorrer o vértice com menor perda será o escolhido, e portanto, mantém-se a escolha do maior valor de $\mathcal{G}_{v_b}$, mesmo nesse caso específico.

De maneira semelhante ao exemplo da arquitetura anterior, suponha um grafo com seis vértices, seu ciclo do TSP novamente é definido pela seguinte ordem $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, e $\mathcal{CH}_{VL} = 35\%$. Iniciando-se a simulação no vértice v_1, será escolhido aleatoriamente um número $\mathcal{N}_1$ tal que

$$0 \leq \mathcal{N}_1 \leq 100$$

suponha $\mathcal{N}_1 = 45$, como $\mathcal{N}_1 > \mathcal{CH}_{VL}$ não será feita uma visita local. Visita-se então o próximo nodo segundo a ordem definida pelo ciclo do TSP, ou seja, v_2. Escolhe-se aleatoriamente o valor de $\mathcal{N}_1$, por exemplo $\mathcal{N}_1 = 20$, como $\mathcal{N}_1 < \mathcal{CH}_{VL}$ será realizada uma visita local. Suponha que são nodos adjacentes a v_2 os vértices $\{v_1, v_3, v_6\}$, para definir qual dentre esses nodos receberá uma visita local serão seguidos os passos listados anteriormente:

1. Suponha que $\mu_{v_b} = \{25, 30, 20\}$, para v_b iguais a v_1, v_3 e v_6, respectivamente. Já $\sigma_{v_b} = \{2, 10, 5\}$ e $\theta_{v_b} = \{45, 20, 10\}$, mantendo a mesma ordem dos nodos. Considere ainda que atualmente a simulação encontra-se no passo 50, portanto $\theta = 50$ e o tempo necessário para percorrer $v_a \rightarrow v_b$ (ou seja, $\omega(v_a, v_b)$) é igual a $\{4, 5, 5\}$, novamente, para $v_a = v_2$ e $v_b = \{v_1, v_3, v_6\}$.

2. Considerando os valores definidos no passo anterior calcula-se Δ'_{v_b}, para cada vértice adjacente, obtendo-se os seguintes valores $\{9, 35, 45\}$.

3. Com base nos passos anteriores $\sigma'_{v_b} = \{16, 5, 25\}$.

4. Finalmente o ganho de variação simulado resultante da visita aos nodos adjacentes será $\mathcal{G}_{v_b} = \{14, -5, 20\}$. Note que no caso específico do vértice v_3 o valor $\mathcal{G}_{v_3}$ é negativo, e portanto, conforme mencionado anteriormente haveria uma diminuição do desvio padrão, caso esse fosse o valor escolhido.

5. Baseando-se nos itens acima, o nodo escolhido para receber a visita local será v_6.

TSP com permutações

Nessa arquitetura propõem-se que a cada ciclo completo do TSP o agente fará uma permutação randômica simples entre dois vértices quaisquer do ciclo original. Para isso, dois nodos são escolhidos aleatoriamente e a ordem na qual eles aparecem no ciclo do TSP original é invertida. Por exemplo, considere que o ciclo do TSP seja $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, e que os dois nodos escolhidos foram os vértices v_2 e v_5, o novo ciclo com a permutação será $\{v_1, v_5, v_3, v_4, v_2, v_6\}$ (conforme a figura 3.2). Até que a simulação seja concluída, a escolha dos nodos e o processo de permutação se repetirão continuamente toda vez que o ciclo for percorrido por completo.

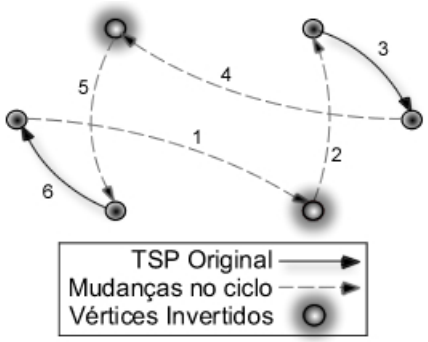


Figura 3.2: Exemplo de um ciclo da arquitetura *TSP com permutações*. Os nodos destacados tiveram sua ordem no ciclo original do TSP invertida, o número em cada uma das arestas refere-se à ordem na qual cada uma dessas arestas será percorrida nesta arquitetura.

TSP com permutações e acréscimo de variabilidade

Este algoritmo de seqüenciamento é muito semelhante ao método anterior (*TSP com permutações*), diferenciando-se ao não decidir aleatoriamente os dois vértices que farão parte da permutação. No caso dessa arquitetura esta escolha será feita de forma deter-

minística, buscando encontrar o par de vértices que acrescentará maior variabilidade ao grupo completo de nodos.

1. Para cada par de vértices $v_x \neq v_y \in V$ faz-se a permutação dos nodos na ordem original do ciclo do TSP. Evidentemente, como a permutação de $\{v_1, v_2\}$ e $\{v_2, v_1\}$ é equivalente, somente uma delas será computada;

2. O ciclo do TSP permutado é então percorrido de maneira simulada, computando-se a soma da alteração do desvio padrão para cada um dos nodos visitados, essa soma será representada como $\Sigma_{\{v_x \to v_y\}}$ e será calculada da seguinte maneira:

$$\Sigma_{\{v_x \to v_y\}} = \sum_{v \in V} \mathcal{G}_v$$

   onde, $\mathcal{G}_v$ refere-se a alteração do desvio padrão para o nodo v durante a execução simulada do ciclo alterado do TSP, o cálculo desse parâmetro é feito de forma semelhante ao descrito no algoritmo *TSP com visitas locais e acréscimo de variabilidade*. Suponha que v é o nodo atual, μ_v e σ_v são a média e o desvio padrão do tempo de espera de v, respectivamente; θ_v é o instante em que v recebeu sua última visita, já θ refere-se a *iteração* atual da simulação, finalmente, $proximo(v)$ é uma função que retorna o nodo seguinte a v, obedecendo-se a ordem permutada do ciclo do TSP:

   (a) Faz-se $\theta = \theta + \omega(v, proximo(v))$ e $v = proximo(v)$;

   (b) Calcula-se então $\Delta'_v = \theta - \theta_v$, para assim obter-se o intervalo entre a última visita real e a visita simulada a v;

   (c) Feito isso, calcula-se $\sigma'_v = |\Delta'_v - \mu_v|$, assim encontra-se o desvio do tempo de espera da visita simulada para com a média do tempo de espera do nodo;

   (d) Para finalizar obtem-se o ganho de variação resultante da visita ao nodo v por meio do seguinte cálculo $\mathcal{G}_v = \sigma'_v - \sigma_v$;

   (e) $\mathcal{G}_v$ é então somado a $\Sigma_{\{v_x \to v_y\}}$;

   (f) Até que o ciclo seja completamente percorrido volta-se para (a).

3. Dentre todos os valores é escolhido o maior $\Sigma_{\{v_x \to v_y\}}$, e $\{v_x, v_y\}$ serão os nodos permutados na ordem original do ciclo do TSP.

Rank de soluções do TSP

Esta arquitetura baseia-se no fato de que embora, possivelmente, exista apenas uma solução ótima, soluções quase-ótimas para o problema do TSP podem ser suficientemente eficientes, além de prover a variabilidade necessária para a arquitetura, isso porque, as

soluções em geral são significativamente distintas, fazendo no mínimo uma permutação na ordem de dois nodos no ciclo original do TSP. Para gerar a lista de soluções quase-ótimas utilizadas como *Rank de soluções* fez-se uma alteração no algoritmo discutido na introdução desta seção que encontra soluções para o TSP. Toda vez que uma solução admissível é encontrada pelo método, ela é mantida em uma lista ordenada. De forma que o resolvedor do TSP irá retornar não somente a solução ótima (que é o primeiro elemento da lista), mas os $\mathcal{K}$ primeiros elementos da lista ordenada, assegurando, no entanto, que o custo de todas as soluções retornadas seja inferior a duas vezes o custo da solução ótima. É importante ressaltar que a alteração no resolvedor do TSP não garante que as melhores soluções quase-ótimas serão encontradas, por exemplo o segundo e o terceiro melhores ciclos contendo todos os vértices não necessariamente, farão parte da lista retornada pelo programa.

Para cada ciclo completo será escolhido aleatoriamente uma das $\mathcal{K}$ soluções contidas na lista ordenada, findo o ciclo outra solução será escolhida e percorrida, esse processo se repetirá até o fim da simulação. A escolha da solução a ser percorrida é totalmente aleatória, sem qualquer prioridade a soluções mais eficientes, isso porque a tendência em geral, é de que soluções menos eficientes gerem uma variabilidade maior, e portanto as soluções em ambos os extremos são desejáveis, gerando por um lado maior variabilidade e do outro primando pela eficiência. A figura 3.3 apresenta um exemplo do TSP original e uma solução alternativa.

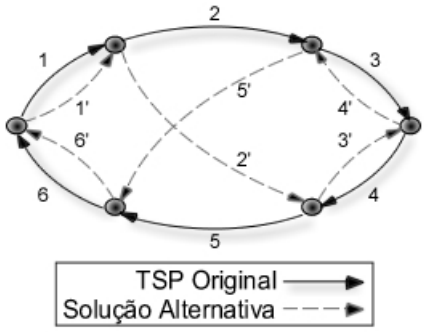


Figura 3.3: Exemplo do *Rank de soluções do TSP*. A linha contínua apresenta a solução original do TSP, e em pontilhado é apresentada uma solução alternativa.

Rank de soluções do TSP e acréscimo de variabilidade

Novamente este é um algoritmo que baseia-se, fundamentalmente, na arquitetura anterior (*Rank de soluções do TSP*), alterando a maneira como é feita a escolha da próxima

solução do TSP a ser percorrida. De forma muito semelhante ao que é feito no algoritmo de seqüenciamento *TSP com permutações e acréscimo de variabilidade*, todas as soluções do *rank* são percorridas de maneira simulada, e aquela que obtiver o maior ganho de variabilidade será a solução escolhida.

TSP Original

Para fins de comparação com as arquiteturas apresentadas, foi incluída na simulação uma arquitetura que faz uso do ciclo do TSP sem quaisquer acréscimos buscando imprevisibilidade, ou seja, nessa arquitetura o ciclo original do TSP é percorrido repetidamente até o fim do tempo de execução.

TSP Original e acréscimo de variabilidade

No caso particular desse algoritmo de seqüenciamento com acréscimo de variabilidade, naturalmente, não é feita a alteração de uma escolha randômica no método original para uma decisão que aumente a variabilidade. Essa arquitetura busca o acréscimo da variabilidade ao incluir determinadas repetições de visitas. Considerando novamente que v é o nodo inicial, e $proximo(v)$ é uma função que retorna o nodo seguinte a v, obedecendo-se a ordem original do ciclo do TSP, a arquitetura baseia-se nos seguintes passos:

1. Partindo-se de v faz-se uma visita a $proximo(v)$ ($v \rightarrow proximo(v)$);
2. O percurso inverso é feito, voltando para v, a partir de $proximo(v)$ ($proximo(v) \rightarrow v$);
3. Novamente é feita uma visita de v a $proximo(v)$ ($v \rightarrow proximo(v)$);
4. De $proximo(v)$ segue-se no percurso do TSP para $proximo(proximo(v))$ ($proximo(v) \rightarrow proximo(proximo(v))$);
5. Faz-se $v = proximo(proximo(v))$;
6. Retorna-se ao passo (1).

Seguindo os passos acima descritos, é possível gerar um ciclo com uma média de tempo entre visitas muito próxima do tempo alcançado no caso do ciclo original do TSP, garantindo assim uma alta eficiência, ao mesmo tempo em que a variação do tempo é muito alta. Considerando que o peso médio das arestas no ciclo original do TSP seja dado por $\bar{\omega}$, e que o número de vértices é n, os cálculos da média (μ) e do desvio padrão (σ) do tempo de espera podem ser expressos da seguinte maneira: cada nodo receberá uma visita depois de ter esperado apenas o tempo de ida e volta para um nodo adjacente,

(portanto, $2 \times \bar{\omega}$), e depois disso receberá outra visita somente quando o ciclo tiver sido completamente percorrido, ao fazer isso, metade das arestas serão trilhadas três vezes enquanto, a outra metade será percorrida apenas uma vez, dessa forma tem-se:

$$2\mu = (2 \times \bar{\omega}) + \left[\left(\frac{n}{2} \times 3 \times \bar{\omega}\right) + \left(\frac{n}{2} \times \bar{\omega}\right)\right] = (2 \times \bar{\omega}) + 4 \times \left(\frac{n}{2} \times \bar{\omega}\right)$$

$$\mu = \bar{\omega} + n \times \bar{\omega} \approx n \times \bar{\omega}$$

ou seja, o tempo médio de espera é aproximadamente o tempo necessário para percorrer o ciclo completo do TSP, portanto o algoritmo é extremamente eficiente. Além disso

$$\sigma^2 = \frac{1}{2} \times \sum (x_i - \mu)^2 = \frac{[(2 \times \bar{\omega}) - (\bar{\omega} + n \times \bar{\omega})]^2 + [(2 \times n \times \bar{\omega}) - (\bar{\omega} + n \times \bar{\omega}))]^2}{2}$$

$$\sigma^2 = \frac{[(n-1) \times \bar{\omega}]^2 + [(n-1) \times \bar{\omega}]^2}{2} = [(n-1) \times \bar{\omega}]^2$$

$$\sigma = (n-1) \times \bar{\omega} \approx n \times \bar{\omega}$$

dessa forma, o desvio padrão é praticamente tão alto quanto a média do tempo, e portanto, o algoritmo possui uma variabilidade altíssima, conforme o desejado.

Considerando o exemplo utilizado até agora, do grafo com 6 vértices com o seguinte ciclo do TSP $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, iniciando a execução em v_1 ($v = v_1$) tem-se:

1. Como $v = v_1 \Rightarrow proximo(v_1) = v_2$ e portanto, faz-se a visita $v_1 \rightarrow v_2$;
2. É feito o percurso inverso $v_2 \rightarrow v_1$;
3. Novamente faz-se a visita $v_1 \rightarrow v_2$;
4. Como $proximo(v_1) = v_2$ então $v = proximo(v_2) = v_3$;
5. Retorna-se ao passo (1).

Portanto a execução dar-se-á da seguinte maneira: $\{v_1, v_2, v_1, v_2, v_3, v_4, v_3, v_4, v_5, v_6, \ldots\}$.

3.4 Extensões Multiagentes

Depois de definir como será feita a escolha do próximo nodo a visitar, através dos algoritmos de seqüenciamento, é importante definir qual será o campo de ação de cada um dos patrulhadores. O campo de ação está relacionado aos locais, portanto os vértices, que deverão ser patrulhados por cada um dos agentes. Existem de forma geral, duas alternativas, na primeira todos os agentes ficam responsáveis pelo patrulhamento de todos os vértices, portanto nesse caso, o campo de ação de todos os patrulhadores será o mesmo — o grafo

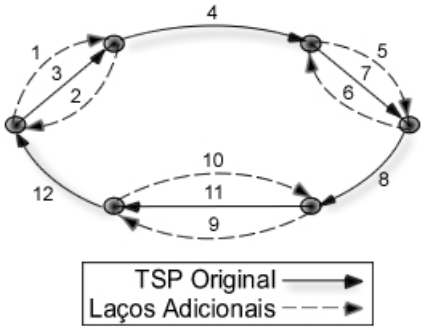


Figura 3.4: Exemplo de um ciclo da arquitetura *TSP original e acréscimo de variabilidade*. A linha contínua apresenta a solução original do TSP, enquanto em pontilhado são apresentados os laços adicionados ao ciclo original. O número em cada uma das arestas refere-se a ordem na qual cada uma das arestas será percorrida.

completo. A segunda alternativa divide os locais a serem patrulhado entre os agentes, de forma que cada agente ficará responsável por um subconjunto limitado de vértices do grafo completo. Existem, evidentemente, diversas formas de particionar o grafo, por esse motivo, são apresentados diversos métodos tradicionais de particionamento nas próximas seções.

É importante destacar que a principal diferença do ponto de vista da simulação é que no primeiro caso com um grupo maior de locais a serem visitados, o tempo de espera entre as visitas de um mesmo agente a um vértice específico tende a ser bem superior, no entanto, o número de agentes que farão visitas a aquele nodo será também maior. Apesar de seguirem premissas bastante diferentes, a relação entre as duas alternativas pode ser expressa da seguinte forma:

$$\frac{ciclo(N_v)}{N_{ag}} \approx ciclo\left(\frac{N_v}{N_{ag}}\right)$$

onde $ciclo(N)$ diz respeito ao ciclo do TSP gerado a partir dos N vértices. Evidentemente, durante a avaliação experimental buscar-se-á definir de forma objetiva e precisa a relação entre essas duas alternativas.

3.4.1 Estratégias sem Particionamento

Conforme destacado anteriormente, nesta estratégia todos os agentes irão patrulhar o mesmo conjunto de vértices. Embora os vértices aos quais todos os agentes terão acesso sejam os mesmos, os patrulhadores agirão de forma independente, tomando cada um suas decisões sem qualquer influência de outros agentes.

Como nessa estratégia os agentes compartilham um mesmo conjunto de vértices, poderia-se alternar entre diferentes agentes o uso do ciclo original do TSP com o ciclo reverso (ordem inversa na qual os nodos aparecem no ciclo original) isso é claro, nas heurísticas de seqüenciamento que se baseiam no TSP. Com essa alteração seria possível incluir mais uma fonte de variação nas arquiteturas. No entanto, essa modificação não foi implementada, de forma que, todos os resultados apresentados, que se referem a heurísticas de seqüenciamento relacionadas com o TSP utilizam a mesma ordem do ciclo do TSP para todos os agentes. Além dessa decisão, ao atribuir a patrulha de todos os vértices ao conjunto completo de agentes é importante definir como será feita a distribuição inicial de cada um dos agentes no ambiente a ser patrulhado. Cada um dos agentes irá iniciar sua patrulha em um vértice específico, não será permitido, portanto, que um agente comece a patrulha ao longo de uma aresta. O vértice escolhido para ser o ponto de origem de um patrulhador será chamado *raiz*. Novamente essa é uma escolha que poderá ser feita de forma a primar pela eficiência (distribuição equidistante dos agentes) ou pelo aumento da variabilidade (distribuição aleatória). São apresentadas a seguir, as duas alternativas propostas:

Raízes Randômicas

Neste caso os agentes serão distribuídos de maneira aleatória entre todos os vértices do grafo, a única restrição seguida será a de que, um único vértice não poderá ser atribuído a mais de um agente.

Raízes Equidistantes

A segunda alternativa opta pela distribuição dos agentes de maneira equidistante no ciclo do TSP, ou seja, de um agente até o próximo no ciclo do TSP, haverá uma distância constante, repetida para todos os agentes. Esta distância constante, naturalmente, refere-se ao tamanho do ciclo do TSP dividido pelo número de agentes. Em muitos casos, entretanto, não será possível distribuir os agentes mantendo uma distância constante entre eles, isso porque cada um dos agentes deverá ter por raiz um nodo e ao manter os agentes perfeitamente equidistantes eles seriam posicionados sobre arestas. Esta estratégia opta então por encontrar a melhor solução, que conforme explicado não necessariamente será ótima. Isto é feito, verificando-se todas as possibilidades de distribuições dos agentes e selecionando-se aquela com a menor distorção (erro) da distribuição equidistante ideal.

3.4.2 Estratégias com Particionamento

O problema de agrupar dados em sub-conjuntos disjuntos de acordo com características previamente definidas (no caso específico desse trabalho a característica segundo a qual os dados serão agrupados é a distância) é bem conhecido e apresenta-se como sendo muito importante em um grande conjunto de aplicações. Dessa forma, existem muitos métodos que objetivam definir e resolver o problema de agrupar dados, cada um desses métodos baseia-se frequentemente em diferente premissas e objetivos. Neste trabalho são exploradas três possibilidades.

Depois do processo de particionamento, cada agente será atribuído a um dos sub-conjuntos de vértices que foi gerado. A tarefa de patrulhar sua região será então realizada por cada um dos agentes, novamente, sem nenhuma influência ou conhecimento de outros patrulhadores.

A primeira alternativa fundamenta-se em uma solução para o problema específico de particionar um grafo, buscando agrupar os vértices em sub-conjuntos com uma mesma quantidade de nodos, de forma que, espera-se que mantendo esse tamanho constante nas partições a quantidade de trabalho que cada um dos agentes terá ao patrulhar cada uma das partições será semelhante ao de outros agentes em partições distintas. As outras duas alternativas apresentadas particionam o conjunto de nodos baseando-se para isso na distância euclidiana entre os vértices, sem levar em consideração as particularidades de um grafo.

Multilevel

O problema de *particionar um grafo* em k partes é definido da seguinte maneira: Dado um grafo $G = (V, E)$ com $|V| = n$; onde $|V|$ é o tamanho do conjunto V, particionar V em k sub-conjuntos, $V_1, V_2, \ldots, V_k$ tais que

$$V_i \cap V_j = \emptyset, \forall i \neq j, |V_i| = \frac{n}{k}, e \bigcup_i V_i = V$$

além disso o número de arestas de corte em E tem de ser minimizado – uma aresta de corte é aquela na qual os vértices incidentes pertencem a partições distintas.

Dentre os principais algoritmos que utilizam-se do problema de particionar um grafo, destaca-se a implementação eficiente da distribuição de tarefas em paralelo, isso porque o particionamento de um grafo em k sub-conjuntos pode ser utilizado para assinalar cada uma das tarefas aos k processadores. Como o particionamento designa um número igual de tarefas a cada um dos processadores a quantidade de trabalho fica balanceada, e com a minimização das arestas de cortes, a comunicação intra-processadores é diminuída. No problema específico do patrulhamento designar a cada um dos agentes uma quantidade

equivalente de vértices, novamente, busca manter a quantidade de trabalho entre os agentes balanceado. A minimização das arestas de corte busca contribuir para a maximização da conectividade interna das partições.

Este modelo de particionamento, portanto, dá prioridade ao tamanho das partições resultantes. Embora, no caso do patrulhamento multiagente uma partição justa seria aquela na qual a distância viajada por cada um dos agentes para cobrir todas os vértices seja igual, este requerimento foi aproximado ao determinar sub-conjuntos com o mesmo número de vértices.

K-Means

O *K-Means* [11, 13] é um método para agrupar dados de acordo com seus atributos, buscando minimizar a distância dos dados pertencentes a uma única partição. Ele é muito similar ao algoritmo *EM* (*expectation-maximization*) pois em ambos procura-se encontrar os centros de partições naturais dos dados. A premissa na qual esse método baseia-se é que os atributos dos objetos formam um espaço vetorial. O número de partições k novamente é constante, e definido antes da execução do algoritmo, e o objetivo principal pode ser definido como alcançar a menor variância dos dados de uma mesma partição, ou, a função de erro quadrático (*squared error*):

$$EQ = \sum_{i=1}^{k} \sum_{x_j \in P_i} (x_j - \mu_i)^2$$

onde k é o número de partições, as partições são dadas por P_i ($i = 1, 2, \ldots, k$), e μ_i é o centróide da i-ésima partição, ou seja, a média de todos os pontos $x_j \in P_i$.

Existem diversas variantes do algoritmo K-means, principalmente no que diz respeito a escolha das sementes (centróides iniciais). A versão mais comum do método (atribuída a Lloyd) começa escolhendo aleatoriamente a posição dos k centros de partições (centróides), cada um dos dados a serem agrupados é então assinalado à partição cuja distância euclidiana ao centro seja menor. A posição dos centróides é recalculada atribuindo-se como novo valor a média dos pontos pertencentes àquela partição. Os novos centróides são então utilizados. O algoritmo segue numa abordagem iterativa, e portanto cada um dos passos acima é repetido, até que não existam mudanças nas partições ou o número limite de iterações seja alcançado ($\mathcal{MAX}_{ITER}$).

A busca do algoritmo *K-means* no conjunto de partições possíveis fica restrita a uma parte muito pequena desse conjunto. É possível que um boa solução para o particionamento perca-se quando o algoritmo converge para um mínimo local da função de pontuação, ao invés, de continuar buscando pelo mínimo global. Uma das formas de diminuir o impacto dessa limitação é executar o algoritmo diversas vezes. É importante destacar

que o uso da inicialização randômica dos centróides leva a uma solução não-determinística, e portanto, execuções diferentes resultarão frequentemente em particionamentos distinto. Por isso nos experimentos realizados o algoritmo do K-means foi executado diversas vezes, com inicializações diferentes, sendo utilizada para a simulação somente o melhor resultado obtido. Para comparar as soluções obtidas, em prol de escolher a melhor delas, calcula-se para cada uma das soluções a soma dos ciclos do TSP de todas as suas partições, a solução que obtiver a menor soma será utilizada nos passos seguintes da simulação como representante do particionamento por meio do K-means.

Agrupamento Hierárquico Aglomerativo

Enquanto métodos baseados em particionamento iniciam o processo com um número predeterminado de partições buscando dentre as possíveis alocações de objetos em sub-conjuntos encontrar uma combinação que otimize alguma função de pontuação, os métodos hierárquicos [11, 13] agrupam dados ou dividem grandes partições. Nessa constatação podem ser identificados os dois tipos de métodos hierárquicos: o aglomerativo, no qual os pontos serão agrupados, e o método divisivo, no qual partições maiores são quebradas em sub-conjuntos menores. O método aglomerativo é o mais usado.

Os métodos hierárquicos aglomerativos são baseados de forma fundamental em medidas de distância entre partições, as principais variantes dessas medidas são a ligação simples (*single-linkage*), ligação completa (*complete-linkage*) e variância mínima. Dentre essas, as ligações simples e completa são os algoritmos mais utilizados. Fundamentalmente, a função de um medida de distância é diferenciar a forma como é caracterizada a similaridade entre pares de partições. No método de ligação simples, a distância entre duas partições é dada pela distância mínima dentre todos os pares de dados que pertencem as duas partições (um objeto do primeiro conjunto e o outro do segundo). Já na ligação completa, ao contrário, a distância será dada pela maior distância entre pares de dados dos conjuntos disjuntos. O algoritmo de ligação completa tende a gerar partições compactas com limites bem definidos. Já no algoritmo de ligação simples, em contraste, as junções frequentemente são feitas de maneira encadeada, tendendo a produzir partições mais alongadas. Considerando a correspondência do uso da ligação simples, com a idéia intuitiva de que vértices próximos deveriam ser assinalados a um mesmo agente patrulhador, optou-se pelo uso dessa métrica de distância nas simulações realizadas.

Escolhida uma das medidas de distância, o método aglomerativo funciona da seguinte maneira: dado um conjunto de partições, as duas partições mais próximas (segundo a métrica escolhida) serão agrupadas, reduzindo assim o número de partições existentes. Isto será repetido, sempre agrupando as duas partições mais próximas, até que haja somente uma partição. Outra possibilidade, frequentemente utilizada quando o número de partições desejadas é previamente estabelecido é parar o processo quando for alcançado

esse número desejado de partições. Geralmente, o método é iniciado considerando que cada um dos pontos dados é uma partição.

Outra importante característica do método hierárquico está no fato de que é possível obter uma representação gráfica adequada, que ilustra de forma clara os detalhes do método, explicitando as junções (ou divisões) visualmente, essa representação gráfica é chamada *dendograma*.

3.5 Cenários de Avaliação

Dadas as severas restrições impostas pelo número de cenários estudados até hoje, decidiu-se por criar um *gerador de grafos* que pudesse suprir a necessidade de aumentar o conjunto de topologias diferentes, garantindo assim a avaliação das soluções apresentadas em domínios mais abrangentes de estudos. A criação dos grafos feita por meio do gerador de grafos, baseia-se em diversos parâmetros, e segue um conjunto bem definido de passos apresentados adiante[1]:

1. Decide-se aleatoriamente o número de vértices do grafo a ser gerado, obedecendo

$$\mathcal{MIN}_n \leq n \leq \mathcal{MAX}_n$$

   onde n é o número de nós, $\mathcal{MIN}_n$ e $\mathcal{MAX}_n$ delimitam o número mínimo e máximo de vértices permitidos, respectivamente.

2. Define-se uma *superfície quadrada bi-dimensional* que conterá todos os n vértices, este quadrado terá tamanho

$$|\mathcal{QUAD}| \times |\mathcal{QUAD}|$$

3. Cada um dos vértices $v \in V$ terá suas coordenadas $2D$ escolhidas aleatoriamente, garantindo-se, no entanto, que vértices diferentes estejam suficientemente distantes um do outro, ou seja

$$\forall (v_i \neq v_j) \in V, dist(v_i, v_j) \geq \mathcal{MIN}_{dist}$$

   onde $dist(v_i, v_j)$ representa a distância euclidiana entre os nodos v_i e v_j; $\mathcal{MIN}_{dist}$ define a distância mínima permitida.

4. Criam-se as arestas de forma que o grafo G torne-se *completo*, isto é

$$\forall (v_i \neq v_j) \in V, \exists e \in E | e = (v_i, v_j)$$

[1]Todos as variáveis definidas em letra caligráfica maiúscula (e.g. $\mathcal{MAX}_n$) são parâmetros de configuração definidos antes da execução que são passados ao programa gerador de grafos.

5. Algumas das arestas são eliminadas, garantindo-se, no entanto, que o grafo permanecerá conexo. Este processo foi dividido em duas etapas:

   (a) Para cada vértice $v \in V$ decide-se de forma aleatória quantas serão as arestas incidentes a v que serão removidas. Este processo, é guiado pela seguinte inequação

   $$0 \leq elim_v \leq \mathcal{CH}_{elim} \times \delta_v$$

   onde $elim_v$ refere-se ao número de arestas a serem removidas, δ_v é o grau do vértice v (como G é completo, sabe-se que $\delta_v = n-1$), e $\mathcal{CH}_{elim}$ é a porcentagem máxima de arestas que poderão ser removidas de um único vértice.

   (b) Escolhe-se randomicamente quais serão as arestas efetivamente removidas.

6. Para cada uma das arestas remanescentes em E é calculada a distância euclidiana entre os vértices que a compõem, este será o peso da aresta, e definirá o número de iterações requeridas, durante a simulação, para ir de um extremo ao outro da aresta. Como em muitos ambientes reais, a distância euclidiana entre dois pontos não define perfeitamente o *custo* real de deslocar-se pelo caminho (e.g. uma subida), algumas das arestas têm seu peso original distorcido, da seguinte maneira:

   (a) Para cada aresta do grafo será decidido se esta aresta terá seu peso distorcido ou não. Esta decisão leva em consideração o parâmetro $\mathcal{CH}_d$ que define a porcentagem máxima de arestas do grafo que terão seu peso distorcido.

   (b) Se no passo anterior optou-se por alterar o valor original da aresta, define-se qual será a proporção da distorção, chamado d_e. Novamente, esse valor é limitado por uma constante $\mathcal{MAX}_d$, que quantifica o máximo de distorção permitido.

   (c) Para finalizar, o peso da aresta é alterado fazendo-se

   $$\omega'_e = \omega_e + (d_e \times \omega_e)$$

   onde ω'_e é o novo valor atribuído como peso da aresta e.

7. Para manter válida a inequação triangular[2] restrição exigida em alguns cálculos, (como o do TSP no caso do uso da heurística de Christofides). Calcula-se o valor do caminho mínimo (*shortest path*) entre todo par de vértices do grafo. O valor calculado é atribuído como novo peso a toda aresta onde o valor distorcido seja superior ao peso obtido no caminho mínimo.

[2] $\forall v_a \neq v_b \neq v_c \in V, \omega(v_a, v_c) \leq \omega(v_a, v_b) + \omega(v_b, v_c)$.

Capítulo 4

Resultados Experimentais

Neste capítulo serão apresentados os principais resultados obtidos através da simulação das diversas arquiteturas propostas. Inicialmente serão apresentados os *detalhes da simulação*, finda esta breve introdução ao ambiente experimental, seguirão duas seções, onde as soluções expostas ao longo do trabalho serão comparadas e avaliadas. Na primeira destas seções a comparação será baseada em dois estimadores que permitem mensurar a eficiência e a variabilidade de cada uma das soluções propostas. Na segunda seção a abordagem utilizada para comparar as soluções irá quantificar a eficiência dos agentes ao realizar o patrulhamento, prevenindo sempre que possível tentativas de ataque e impedindo que ataques iniciados sejam bem sucedidos. Por fim, haverá uma seção que apresentará brevemente como realizar a escolha de uma das arquiteturas propostas para a utilização no patrulhamento de um cenário específico.

4.1 Detalhes da Simulação

No capítulo anterior foram apresentados diversos parâmetros utilizados por parte dos algoritmos de seqüenciamento, extensões multiagentes e principalmente pelo gerador de grafos. Durante os testes realizados para a simulação efetiva foram avaliados diversos valores para cada um dos parâmetros especificados. Seguem agora os valores utilizados na simulação final, na qual, foram gerados os resultados, que permitirão a comparação das arquiteturas, os dados obtidos com essa simulação serão apresentadas nas seções subsequentes:

- **Número de grafos:** 1000;
- **Número mínimo de vértices de um grafo ($\mathcal{MIN}_n$):** 80;
- **Número máximo de vértices de um grafo ($\mathcal{MAX}_n$):** 100;

- **Tamanho da superfície quadrada bidimensional ($|\mathcal{QUAD}| \times |\mathcal{QUAD}|$):** 100×100;
- **Distância mínima permitida entre dois vértices ($\mathcal{MIN}_{dist}$):** 2;
- **Porcentagem máxima de arestas que serão removidas ($\mathcal{CH}_{elim}$):** 15%;
- **Porcentagem máxima de arestas que terão seu peso distorcido ($\mathcal{CH}_{d}$):** 15%;
- **Distorção máxima do peso de uma aresta ($\mathcal{MAX}_{d}$):** 20%;
- **Número de Agentes:** Cada experimento foi executado com 2, 4 e 6 agentes.

Para cada um dos cenários de avaliação criados, foram rodados $750 = 10 \times 5 \times 3 \times 5$ experimentos, que correspondem a 10 algoritmos de seqüenciamento, 5 extensões multiagentes, 3 variações no número de agentes (2, 4 e 6 agentes) e 5 intervalos de intrusão distintos. Dadas as diferenças entre os grafos, o tempo de execução para as simulações baseia-se no tempo necessário para percorrer o ciclo completo do TSP e é dado por $\mathcal{T}_{TOTAL} = 100 \times |\mathcal{TSP}_{ciclo}|$. O *tempo de espera* para cada um dos vértices não é computado antes da primeira visita, como conseqüência, o *tempo de espera* estará disponível somente da segunda visita em diante.

Os critérios de avaliação, definidos pelos intrusos randômico, escondido e estatístico, avaliam as soluções propostas baseando-se em um único vértice do grafo, nodo este escolhido de maneira aleatória. No caso do intruso randômico, são gerados 50 ataques, conforme destacado na seção 3.2.1 cada uma dessas intrusões é iniciada em um momento escolhido aleatoriamente. Já para o intruso escondido o número de ataques esta relacionado com a quantidade de visitas feitas por agentes ao vértice alvo. Finalmente, para o atacante estatístico o número de tentativas de invasão varia, já que o intruso só irá atacar quando seus dados lhe assegurarem que sua tentativa será bem sucedida.

Foram utilizados 5 constantes para o cálculo do intervalo de intrusão apresentado na seção 3.2, por meio da equação 3.4. Os valores utilizados foram

$$\mathcal{C}_{FRAC} = \left\{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2\right\}$$

Para cada instância do *K-means* visando particionar o grafo entre os agentes, o algoritmo foi executado 5 vezes, com diferentes *sementes* iniciais, e a melhor solução foi escolhida.

Para a realização dos experimentos foi desenvolvido um simulador próprio, que implementa todas as funções apresentadas no capítulo anterior: algoritmos de seqüenciamento, extensões multiagentes e o gerador de grafos, dentre outras funcionalidades que serão abordadas posteriormente. Foram utilizados pelo simulador dois programas de terceiros:

conforme mencionado anteriormente, os algoritmos de seqüenciamento que faziam uso do ciclo do TSP, utilizaram a solução obtida através do algoritmo *branch and bound* com relaxamento e poda [28, 29, 14]. Para o particionamento de grafos definido como *multilevel*, foi utilizado o software *METIS* [15].

Dentre os algoritmos de seqüenciamento, o TSP com visitas locais faz uso de uma constante que define a probabilidade de se optar por fazer uma visita local sempre que é feita uma visita a um nodo durante o percurso do ciclo do TSP ($\mathcal{CH}_{VL}$). Para todas as execuções foram rodadas cinco instâncias desse algoritmo com os seguintes valores $\mathcal{CH}_{VL} = \{1\%, 2\%, 5\%, 10\%, 15\%\}$, no entanto, apenas o melhor valor obtido em cada uma das simulações foi considerado para fins de comparação.

4.2 Demonstrativo da Média e Desvio Padrão do Intervalo entre Visitas

Serão agora apresentados dois dos principais estimadores relacionados com o tempo de espera entre visitas ao nodo alvo. O primeiro valor refere-se à média do intervalo entre as visitas ao vértice alvo para todos os grafos durante a simulação completa. Já o segundo estimador quantifica o desvio padrão do tempo de espera entre as visitas ao mesmo nodo. Esses dois estimadores permitem avaliar as diversas arquiteturas do ponto de vista da eficiência versus variabilidade da solução.

Evidentemente, com a variação de diversas características como a topologia e a distância média entre os vértices, o intervalo dos valores calculados dos estimadores será diferente para grafos distintos rodando em uma mesma arquitetura, no entanto, considerando que as estratégias de atribuição dos vértices entre os agentes e os algoritmos de seqüenciamento manterão um comportamento semelhante mesmo para grafos diferentes é possível fazer uma comparação avaliando a eficiência e variabilidade das soluções propostas. Serão apresentados os valores dos estimadores nas simulações com seis, quatro e dois agentes, respeitando esta ordem. Os resultados apresentados a seguir só exibem o valor do TSP com visitas locais para o caso onde $\mathcal{CH}_{VL} = 5\%$, existem, portanto, variações desse algoritmo que obtiveram menores médias do tempo de espera, e outros onde o desvio-padrão foi maior. Para cada um dentre os algoritmos de seqüenciamento foi proposta uma modificação buscando um acréscimo da variabilidade, essas soluções foram destacadas com a inclusão de uma marca *(V)* após o nome do método original no qual se baseiam. Ambos os estimadores, média e desvio padrão, são apresentados seguindo a unidade de medida padrão, ou seja, o número de iterações da simulação. De forma que, a média, por exemplo, representa o número médio de iterações que se passaram entre duas visitas ao nodo alvo.

Estratégia de Distribuição	**Algoritmo de Seqüenciamento**	**Tempo Médio de Espera**	**Desvio Padrão Médio**
Raízes Randômicas	Randômico Local	829	811
Raízes Randômicas	Randômico Global	666	652
Raízes Randômicas	TSP - Visitas Locais	197	174
Raízes Randômicas	TSP - Visitas Locais (V)	189	158
Raízes Randômicas	TSP - Permutações	163	138
Raízes Randômicas	TSP - Permutações (V)	180	169
Raízes Randômicas	TSP - Rank	148	126
Raízes Randômicas	TSP - Rank (V)	158	146
Raízes Randômicas	TSP	135	105
Raízes Randômicas	TSP (V)	135	193
Raízes Equidistantes	Randômico Local	826	807
Raízes Equidistantes	Randômico Global	666	651
Raízes Equidistantes	TSP - Visitas Locais	197	173
Raízes Equidistantes	TSP - Visitas Locais (V)	189	158
Raízes Equidistantes	TSP - Permutações	163	139
Raízes Equidistantes	TSP - Permutações (V)	180	170
Raízes Equidistantes	TSP - Rank	148	126
Raízes Equidistantes	TSP - Rank (V)	158	146
Raízes Equidistantes	TSP	135	108
Raízes Equidistantes	TSP (V)	135	195

Tabela 4.1: Tabela comparativa da média e desvio padrão do tempo de espera das extensões multiagentes sem particionamento, para todos os algoritmos de seqüenciamento, em uma execução com 6 agentes.

Estratégia de Particionamento	**Algoritmo de Seqüenciamento**	**Tempo Médio de Espera**	**Desvio Padrão Médio**
Hierárquico	Randômico Local	593	562
Hierárquico	Randômico Global	509	480
Hierárquico	TSP - Visitas Locais	230	75
Hierárquico	TSP - Visitas Locais (V)	211	69
Hierárquico	TSP - Permutações	246	69
Hierárquico	TSP - Permutações (V)	279	83
Hierárquico	TSP - Rank	212	21
Hierárquico	TSP - Rank (V)	211	63
Hierárquico	TSP	194	0
Hierárquico	TSP (V)	193	177
Multilevel	Randômico Local	855	791
Multilevel	Randômico Global	733	664
Multilevel	TSP - Visitas Locais	438	141
Multilevel	TSP - Visitas Locais (V)	419	136
Multilevel	TSP - Permutações	491	158
Multilevel	TSP - Permutações (V)	539	186
Multilevel	TSP - Rank	426	61
Multilevel	TSP - Rank (V)	422	130
Multilevel	TSP	400	0
Multilevel	TSP (V)	399	357
K-Means	Randômico Local	379	353
K-Means	Randômico Global	334	308
K-Means	TSP - Visitas Locais	179	56
K-Means	TSP - Visitas Locais (V)	169	53
K-Means	TSP - Permutações	201	61
K-Means	TSP - Permutações (V)	224	73
K-Means	TSP - Rank	172	19
K-Means	TSP - Rank (V)	171	49
K-Means	TSP	160	0
K-Means	TSP (V)	159	141

Tabela 4.2: Tabela comparativa da média e desvio padrão do tempo de espera das extensões multiagentes com particionamento, para todos os algoritmos de seqüenciamento, em uma execução com 6 agentes.

Estratégia de Distribuição	**Algoritmo de Seqüenciamento**	**Tempo Médio de Espera**	**Desvio Padrão Médio**
Raízes Randômicas	Randômico Local	1240	1206
Raízes Randômicas	Randômico Global	1000	975
Raízes Randômicas	TSP - Visitas Locais	296	243
Raízes Randômicas	TSP - Visitas Locais (V)	252	195
Raízes Randômicas	TSP - Permutações	245	190
Raízes Randômicas	TSP - Permutações (V)	270	228
Raízes Randômicas	TSP - Rank	223	171
Raízes Randômicas	TSP - Rank (V)	237	195
Raízes Randômicas	TSP	203	145
Raízes Randômicas	TSP (V)	202	282
Raízes Equidistantes	Randômico Local	1242	1207
Raízes Equidistantes	Randômico Global	1002	975
Raízes Equidistantes	TSP - Visitas Locais	296	243
Raízes Equidistantes	TSP - Visitas Locais (V)	252	195
Raízes Equidistantes	TSP - Permutações	245	191
Raízes Equidistantes	TSP - Permutações (V)	271	228
Raízes Equidistantes	TSP - Rank	222	173
Raízes Equidistantes	TSP - Rank (V)	237	198
Raízes Equidistantes	TSP	203	143
Raízes Equidistantes	TSP (V)	202	282

Tabela 4.3: Tabela comparativa da média e desvio padrão do tempo de espera das extensões multiagentes sem particionamento, para todos os algoritmos de seqüenciamento, em uma execução com 4 agentes.

Estratégia de Particionamento	Algoritmo de Seqüenciamento	Tempo Médio de Espera	Desvio Padrão Médio
Hierárquico	Randômico Local	1426	1364
Hierárquico	Randômico Global	1185	1134
Hierárquico	TSP - Visitas Locais	429	144
Hierárquico	TSP - Visitas Locais (V)	375	132
Hierárquico	TSP - Permutações	420	99
Hierárquico	TSP - Permutações (V)	475	119
Hierárquico	TSP - Rank	363	35
Hierárquico	TSP - Rank (V)	365	111
Hierárquico	TSP	332	0
Hierárquico	TSP (V)	330	316
Multilevel	Randômico Local	1269	1193
Multilevel	Randômico Global	1087	1007
Multilevel	TSP - Visitas Locais	540	179
Multilevel	TSP - Visitas Locais (V)	504	169
Multilevel	TSP - Permutações	587	164
Multilevel	TSP - Permutações (V)	651	193
Multilevel	TSP - Rank	507	65
Multilevel	TSP - Rank (V)	504	160
Multilevel	TSP	472	0
Multilevel	TSP (V)	470	441
K-Means	Randômico Local	683	652
K-Means	Randômico Global	582	548
K-Means	TSP - Visitas Locais	269	86
K-Means	TSP - Visitas Locais (V)	247	80
K-Means	TSP - Permutações	290	79
K-Means	TSP - Permutações (V)	324	94
K-Means	TSP - Rank	249	25
K-Means	TSP - Rank (V)	248	74
K-Means	TSP	228	0
K-Means	TSP (V)	227	210

Tabela 4.4: Tabela comparativa da média e desvio padrão do tempo de espera das extensões multiagentes com particionamento, para todos os algoritmos de seqüenciamento, em uma execução com 4 agentes.

Estratégia de Distribuição	Algoritmo de Seqüenciamento	Tempo Médio de Espera	Desvio Padrão Médio
Raízes Randômicas	Randômico Local	2490	2418
Raízes Randômicas	Randômico Global	2002	1928
Raízes Randômicas	TSP - Visitas Locais	591	383
Raízes Randômicas	TSP - Visitas Locais (V)	484	325
Raízes Randômicas	TSP - Permutações	489	288
Raízes Randômicas	TSP - Permutações (V)	542	323
Raízes Randômicas	TSP - Rank	445	259
Raízes Randômicas	TSP - Rank (V)	470	307
Raízes Randômicas	TSP	406	201
Raízes Randômicas	TSP (V)	403	502
Raízes Equidistantes	Randômico Local	2482	2372
Raízes Equidistantes	Randômico Global	2004	1939
Raízes Equidistantes	TSP - Visitas Locais	593	385
Raízes Equidistantes	TSP - Visitas Locais (V)	485	328
Raízes Equidistantes	TSP - Permutações	490	289
Raízes Equidistantes	TSP - Permutações (V)	542	322
Raízes Equidistantes	TSP - Rank	446	257
Raízes Equidistantes	TSP - Rank (V)	470	302
Raízes Equidistantes	TSP	406	207
Raízes Equidistantes	TSP (V)	403	506

Tabela 4.5: Tabela comparativa da média e desvio padrão do tempo de espera das extensões multiagentes sem particionamento, para todos os algoritmos de seqüenciamento, em uma execução com 2 agentes.

Estratégia de Particionamento	Algoritmo de Seqüenciamento	Tempo Médio de Espera	Desvio Padrão Médio
Hierárquico	Randômico Local	3311	3169
Hierárquico	Randômico Global	2692	2539
Hierárquico	TSP - Visitas Locais	843	280
Hierárquico	TSP - Visitas Locais (V)	710	260
Hierárquico	TSP - Permutações	740	137
Hierárquico	TSP - Permutações (V)	825	166
Hierárquico	TSP - Rank	660	57
Hierárquico	TSP - Rank (V)	668	207
Hierárquico	TSP	601	0
Hierárquico	TSP (V)	596	586
Multilevel	Randômico Local	2501	2349
Multilevel	Randômico Global	2065	1948
Multilevel	TSP - Visitas Locais	779	259
Multilevel	TSP - Visitas Locais (V)	688	242
Multilevel	TSP - Permutações	757	166
Multilevel	TSP - Permutações (V)	845	198
Multilevel	TSP - Rank	664	75
Multilevel	TSP - Rank (V)	667	217
Multilevel	TSP	610	0
Multilevel	TSP (V)	606	593
K-Means	Randômico Local	2005	1919
K-Means	Randômico Global	1668	1585
K-Means	TSP - Visitas Locais	577	195
K-Means	TSP - Visitas Locais (V)	499	179
K-Means	TSP - Permutações	556	124
K-Means	TSP - Permutações (V)	634	156
K-Means	TSP - Rank	480	46
K-Means	TSP - Rank (V)	484	156
K-Means	TSP	437	0
K-Means	TSP (V)	435	420

Tabela 4.6: Tabela comparativa da média e desvio padrão do tempo de espera das extensões multiagentes com particionamento, para todos os algoritmos de seqüenciamento, em uma execução com 2 agentes.

Os dados apresentados nas tabelas evidenciam diversos padrões de comportamento relevantes, dentre os quais destacam-se:

- A solução do TSP original sem quaisquer modificações, embora seja uma solução estática, e portanto, sem nenhuma variabilidade, nos casos específicos das soluções cíclicas (sem particionamento), em que todos os agentes ficam responsáveis por patrulhar todos os nodos do grafo, apresentou desvio padrão diferente de zero, isso se deve à variação da distância entre os agentes no ciclo do TSP. Essa fonte de variação decorrente da extensão multiagente utilizada contribuiu também para um acréscimo no desvio padrão de todos os demais algoritmos de seqüenciamento.

- Conforme o esperado, o tempo de espera médio é muito inferior para todos os algoritmos de seqüenciamento baseados no ciclo do TSP, quando comparados com os algoritmos fundamentalmente randômicos. Por outro lado, a variabilidade dos algoritmos aleatórios é muito superior àquela obtida pelos algoritmos baseados no TSP.

- Para a maioria dos casos, a modificação dos algoritmos de seqüenciamento buscando o acréscimo da variabilidade nas soluções resultou em um desvio padrão maior. No caso específico do TSP com visitas locais, no entanto, observa-se o contrário, uma diminuição do desvio padrão médio. Isso ocorre porque ao atribuir deterministicamente que vértice será escolhido para receber uma visita local, geralmente um pequeno grupo de vértices é beneficiado; esse grupo seleto tem seu desvio-padrão aumentado, mas a variação decresce para o grafo como um todo. Como o valor apresentado nas tabelas refere-se exclusivamente a um único nodo - o vértice alvo das invasões - se esse vértice não for um dos nodos beneficiados com o comportamento determinístico o valor do seu desvio padrão naturalmente será inferior. É importante ressaltar, entretanto, que a média do tempo de espera também é inferior, e portanto, a solução embora não possua uma variação maior é mais eficiente que o algoritmo original. Outra distinção importante, ainda com relação ao TSP com visitas locais modificado, surge do fato de que esse é o único algoritmo de seqüenciamento que leva em consideração apenas aspectos locais, avaliando somente o impacto da escolha com relação aos nodos adjacentes ao vértice atual; todos os demais algoritmos de seqüenciamento com alterações para o acréscimo da variabilidade levam em consideração todos os vértices do grafo, e portanto, buscam a melhor escolha do ponto de vista global.

- Os valores obtidos no caso de raízes equidistantes e raízes randômicas são muito semelhantes, com diferenças quase imperceptíveis do ponto de vista desses dois estimadores, a principal razão para isso vem do fato de que o impacto da escolha

dos nodos onde cada agente irá iniciar a simulação é muito baixo no resultado da simulação como um todo. Já o comportamento padrão seguido pelos algoritmos de seqüenciamento durante o restante da execução determinam de fato os valores desses dois estimadores, e portanto minimizam drasticamente as diferenças entre as duas abordagens, do ponto de vista da média e desvio padrão do tempo de espera entre as visitas de agentes.

- De forma geral as extensões multiagente que apresentaram o menor tempo médio de espera foram aquelas sem particionamento, seguidas pelo *K-means*, particionamento *Aglomerativo Hierárquico* e particionamento *Multilevel*.

4.3 Estudo comparativo das Soluções Propostas

Além de definir três critérios de avaliação que permitem a comparação das diversas soluções baseando-se em perspectivas diferenciadas é importante especificar uma metodologia de comparação adequada para os resultados obtidos em cada um dos cenários propostos. Para criar tal metodologia é preciso identificar os objetivos primordiais do patrulhamento, quantificá-los e por fim comparar os diversos valores obtidos para cada uma das combinações de soluções propostas em cada um dos cenários de avaliação. O objetivo fundamental do patrulhamento consiste em garantir a segurança do ambiente patrulhado, portanto, o agente patrulhador deverá ser capaz de reduzir ao máximo o número de ataques, inibindo a possibilidade de invasões, e tornando, se possível, o ambiente em um local livre de invasões. No entanto, como a eliminação completa de invasões em geral não pode ser alcançada, torna-se imprescindível que o patrulhador seja capaz de impedir que um ataque seja bem sucedido quando ele não pode ser prevenido, assegurando que o invasor seja encontrado antes de concluir com sucesso seu ataque. Dessa forma o objetivo da ação de patrulhamento pode ser sumarizado como: prevenir o ataque e impedir que ataques iniciados sejam bem sucedidos. Esses dois objetivos podem ser facilmente expressos em termos de probabilidades; prevenir um ataque consiste em minimizar a probabilidade de um ataque acontecer $P(ataque)$, enquanto impedir ataques já iniciados refere-se a maximizar a probabilidade de que o invasor seja descoberto durante seu ataque $P(impedir|ataque)$. Para facilitar a apresentação dos cálculos, um ataque será representado como o evento $\mathcal{A}$, enquanto a ação de impedir um ataque será apresentado como $\mathcal{I}$. Considere $\mathcal{N}_{MAX\_ataques}$ o número máximo de ataques que o nodo-alvo poderá receber. Para o intruso randômico esse valor será fixo, já que o número de máximo de ataques é definido por meio de uma constante que também expressa o número de tentativas de ataques. É importante ressaltar que conforme descrito anteriormente, as tentativas de invasão para os invasores estatístico e observador acontecem sempre imediatamente

após a visita de um agente, isso porque dessa forma é possível maximizar o tempo do qual um intruso dispõem para a sua invasão. Portanto, o número máximo de ataques a serem realizados pelo intruso observador ou estatístico fica limitado ao número de visitas de agentes patrulhadores ao nodo alvo, ou seja, $\mathcal{N}_{MAX\_ataques} = \mathcal{N}_{visitas}$; $\mathcal{N}_{tentativas}$ refere-se ao número de tentativas de invasão realizadas pelo invasor. No caso específico do atacante randômico e observador não existe possibilidade de que um ataque seja prevenido, de forma que, o número de tentativas será igual ao número máximo de ataques, e portanto, a probabilidade de um ataque ser realizado é igual a um, ou seja, $P(\mathcal{A}) = 1$. Por fim $\mathcal{N}_{ataques\_impedidos}$ refere-se ao número de invasões que foram interceptadas por patrulhadores. Dessa forma, o cálculo da probabilidade de um ataque é dado por:

$$P(\mathcal{A}) = \frac{\mathcal{N}_{tentativas}}{\mathcal{N}_{MAX\_ataques}}$$

já a probabilidade de impedir que um ataque seja bem sucedido depois que ele foi iniciado será calculado da seguinte maneira:

$$p(\mathcal{I}|\mathcal{A}) = \frac{P(\mathcal{I} \cap \mathcal{A})}{P(\mathcal{A})} = \frac{\frac{\mathcal{N}_{ataques\_impedidos}}{\mathcal{N}_{MAX\_ataques}}}{\frac{\mathcal{N}_{tentativas}}{\mathcal{N}_{MAX\_ataques}}} = \frac{\mathcal{N}_{ataques\_impedidos}}{\mathcal{N}_{tentativas}}$$

A métrica utilizada para a comparação das soluções propostas será dada pela multiplicação da média das duas probabilidades para todos os grafos da simulação, com a ressalva de que para o cálculo da probabilidade de impedir um ataque iniciado serão considerados apenas os grafos onde o número de tentativas de ataque foi maior que zero. Além disso para que o cálculo busque maximizar o resultado em ambos os casos, inverte-se o valor da probabilidade de um ataque, portanto determina-se a chance de que um ataque seja prevenido. Dessa forma tem-se:

$$\mathcal{M} = \overline{[1 - P(\mathcal{A})]} \times \overline{P(\mathcal{I}|\mathcal{A})}$$

Como a métrica é resultado da multiplicação de dois valores entre um e zero, naturalmente

$$0 \leq \mathcal{M} \leq 1$$

O objetivo de cada uma das soluções é maximizar o valor obtido pela métrica, assegurando que a estratégia é eficaz em prevenir ataques e eficiente em conter eventuais atacantes ativos.

Para melhor avaliar a diferença entre cada uma das arquiteturas levando em consideração todos os resultados obtidos nos diversos intervalos de intrusão testados, as tabelas foram ordenadas de acordo com o valor da área do gráfico obtido na relação entre a constante do intervalo de intrusão pelo valor da métrica resultante da execução daquela solução

em particular, naturalmente, quanto maior a área desse gráfico melhor foi o desempenho da arquitetura em análise. A figura 4.1 ilustra o gráfico que obteve o melhor desempenho para o patrulhamento de um ambiente considerando um intruso randômico e uma simulação com dois agentes patrulhadores. Embora o gráfico apresente um de seus eixos usando uma escala logarítmica, o cálculo da área considera os valores sem qualquer transformação. O valor máximo que a área do gráfico pode assumir é dado por:

$$\mathcal{A}_{MAX} = Lado \times Altura = \left(2 - \frac{1}{8}\right) \times 100 = 187.5$$

Para determinar a estratégia que obteve o melhor desempenho foi calculado a área destacada do gráfico para cada uma das soluções propostas, e de acordo com esse critério todas as soluções foram ordenadas.

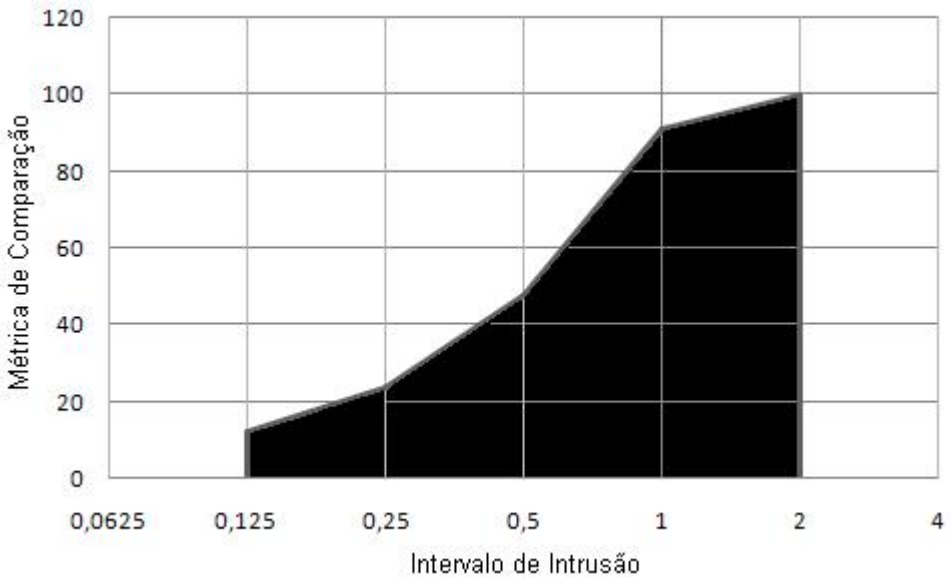


Figura 4.1: Gráfico demonstrativo da área resultante de uma simulação. O eixo das ordenadas refere-se ao valor obtido pela métrica de comparação em porcentagem, enquanto, o eixo das abscissas contém a variação do valor da constante utilizada no cálculo do intervalo de intrusão e é apresentado em escala logarítmica.

Foram simuladas todas as combinações de extensões multiagentes com os diversos algoritmos de seqüenciamento propostos. Os principais resultados são apresentadas nas seguintes tabelas, onde constam os valores referentes ao melhor resultado obtido para cada uma das extensões multiagentes, assim como, o maior valor obtido para cada um dos algoritmos de seqüenciamento apresentados anteriormente.

4.3.1 Intruso Randômico

Extensão Multiagente	Algoritmo de Seqüenciamento	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
K-Means	TSP	11,6	22,9	44,9	83,6	99,8	134,5
K-Means	TSP - Visitas Locais	11,2	22,1	44,0	81,3	99,6	132,1
K-Means	TSP - Rank	10,7	20,9	42,1	78,4	99,6	128,9
Hierárquico	TSP	11,7	23,1	44,7	73,9	95,7	125,1
Raízes Randômicas	TSP	12,3	22,9	41,6	67,4	91,9	117,2
Raízes Equidistantes	TSP	12,3	23,2	41,1	66,8	90,9	116,1
K-Means	TSP - Permutações	9,4	18,2	35,3	66,4	95,9	114,9
K-Means	TSP(V)	10,4	17,4	29,6	51,2	87,1	96,9
K-Means	Randômico Global	6,5	12,4	23,4	39,8	61,8	72,2
Multilevel	TSP	4,5	9,1	17,5	34,0	66,8	67,4
K-Means	Randômico Local	5,6	10,9	20,5	36,0	56,9	65,6
K-Means	TSP - Visitas Locais(V)	0,0	0,0	0,0	0,1	0,3	0,2
K-Means	TSP - Rank(V)	0,0	0,0	0,0	0,1	0,3	0,2
K-Means	TSP - Permutações(V)	0,0	0,0	0,0	0,1	0,2	0,2

Tabela 4.7: Melhores resultados obtidos na simulação com 6 agentes no caso de um intruso randômico, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
K-Means	TSP	11,6	23,5	46,3	87,3	99,9	137,9
K-Means	TSP - Visitas Locais	11,2	22,5	44,7	83,9	99,6	134,4
K-Means	TSP - Rank	11,0	21,3	42,3	81,1	99,8	131,3
Raízes Equidistantes	TSP	12,4	22,8	42,0	68,6	94,2	119,4
Raízes Randômicas	TSP	12,4	23,2	41,6	68,5	94,0	119,1
K-Means	TSP - Permutações	9,3	17,9	35,6	69,0	97,7	117,9
Hierárquico	TSP	11,7	22,7	41,5	66,8	92,6	117,0
K-Means	TSP(V)	9,6	15,9	27,2	50,6	89,5	96,5
Multilevel	TSP	5,6	11,0	21,4	42,5	84,2	84,4
K-Means	Randômico Global	5,3	10,0	19,0	33,9	55,1	62,3
K-Means	Randômico Local	4,3	8,5	15,9	29,6	49,0	54,6
K-Means	TSP - Visitas Locais(V)	0,0	0,0	0,0	0,1	0,4	0,3
K-Means	TSP - Rank(V)	0,0	0,0	0,0	0,1	0,4	0,3
K-Means	TSP - Permutações(V)	0,0	0,0	0,0	0,1	0,2	0,2

Tabela 4.8: Melhores resultados obtidos na simulação com 4 agentes no caso de um intruso randômico, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
K-Means	TSP	12,0	23,7	47,5	90,8	99,7	141,0
K-Means	TSP - Visitas Locais	11,3	22,2	43,8	84,5	99,7	134,5
K-Means	TSP - Rank	11,0	21,7	43,1	83,6	99,7	133,5
Raízes Randômicas	TSP	12,2	23,7	44,1	74,6	99,7	127,5
Raízes Equidistantes	TSP	12,1	23,2	43,3	74,4	99,7	127,0
K-Means	TSP - Permutações	9,3	18,7	36,8	71,7	99,0	121,2
Hierárquico	TSP	12,2	22,9	39,6	67,0	99,6	120,0
Multilevel	TSP	8,3	16,4	32,8	65,4	99,6	114,7
K-Means	TSP(V)	7,9	13,7	25,6	49,6	91,7	95,8
K-Means	Randômico Global	3,2	6,5	12,6	23,4	40,9	44,1
K-Means	Randômico Local	2,8	5,3	10,4	19,6	34,7	37,1
K-Means	TSP - Visitas Locais(V)	0,0	0,0	0,1	0,2	0,8	0,6
K-Means	TSP - Rank(V)	0,0	0,0	0,1	0,2	0,7	0,5
K-Means	TSP - Permutações(V)	0,0	0,0	0,0	0,1	0,6	0,4

Tabela 4.9: Melhores resultados obtidos na simulação com 2 agentes no caso de um intruso randômico, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

4.3.2 Intruso Observador

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes	TSP(V)	31,6	54,2	61,4	69,3	81,1	127,7
Raízes Randômicas	TSP(V)	31,7	54,2	61,1	69,1	80,9	127,3
Raízes Equidistantes	TSP	8,2	18,3	35,0	59,6	86,7	105,1
Raízes Randômicas	TSP	8,2	17,8	34,3	58,9	87,0	104,4
Hierárquico	TSP(V)	26,3	47,0	50,6	54,2	68,0	104,1
K-Means	TSP(V)	24,3	44,8	50,2	51,1	67,8	101,0
Raízes Randômicas	TSP - Rank	9,2	17,9	32,8	56,1	83,4	100,0
Raízes Equidistantes	TSP - Visitas Locais	9,1	17,5	32,6	56,1	83,4	99,8
Raízes Randômicas	TSP - Rank(V)	10,7	19,9	34,9	57,5	77,4	99,3
Raízes Randômicas	TSP - Visitas Locais(V)	7,7	14,8	27,8	51,2	85,0	94,5
Raízes Equidistantes	TSP - Permutações	8,3	16,1	30,0	52,3	79,9	93,9
Raízes Equidistantes	TSP - Permutações(V)	11,3	20,3	34,4	51,8	70,6	91,6
Multilevel	TSP(V)	4,8	15,8	37,9	50,6	52,0	81,4
K-Means	Randômico Global	2,2	6,0	15,7	34,9	58,8	62,8
K-Means	Randômico Local	1,3	4,6	14,1	31,5	54,2	56,9

Tabela 4.10: Melhores resultados obtidos na simulação com 6 agentes no caso de um intruso observador, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes	TSP(V)	44,6	55,5	59,8	67,4	79,9	126,1
Raízes Randômicas	TSP(V)	44,2	55,3	59,8	67,5	79,8	126,1
Hierárquico	TSP(V)	40,5	49,8	51,1	55,6	65,1	105,3
Raízes Randômicas	TSP	7,6	16,5	32,3	57,8	87,8	103,0
K-Means	TSP(V)	38,4	49,7	50,2	50,4	68,8	102,7
Raízes Equidistantes	TSP - Visitas Locais	8,4	16,4	30,8	54,1	83,4	97,4
Raízes Equidistantes	TSP - Rank	8,4	16,3	30,6	54,0	83,5	97,3
Raízes Equidistantes	TSP - Rank(V)	8,7	16,3	30,2	56,2	76,6	95,4
Raízes Equidistantes	TSP - Visitas Locais(V)	20,3	27,2	33,7	47,9	78,0	93,9
Raízes Equidistantes	TSP - Permutações	7,6	14,8	28,1	50,3	79,6	91,3
Multilevel	TSP(V)	14,8	35,6	50,1	51,5	51,6	90,8
Raízes Randômicas	TSP - Permutações(V)	9,3	17,0	29,8	47,8	69,2	85,4
K-Means	Randômico Global	2,2	5,3	14,2	30,2	52,8	55,5
K-Means	Randômico Local	1,1	3,8	12,0	26,2	47,2	48,5

Tabela 4.11: Melhores resultados obtidos na simulação com 4 agentes no caso de um intruso observador, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes	TSP(V)	51,9	53,8	57,2	63,5	75,6	120,2
Raízes Randômicas	TSP(V)	51,8	53,6	56,9	63,2	75,4	119,8
K-Means	TSP(V)	49,8	50,3	50,3	50,5	68,2	103,4
Hierárquico	TSP(V)	50,1	50,8	52,3	54,8	59,1	102,9
Raízes Equidistantes	TSP	6,4	13,3	26,6	49,7	99,8	100,1
Raízes Equidistantes	TSP - Visitas Locais(V)	6,9	11,4	22,4	49,6	97,8	97,1
Multilevel	TSP(V)	47,7	51,0	51,1	51,2	51,2	95,7
Raízes Randômicas	TSP - Rank	6,0	11,8	23,1	45,6	90,3	90,6
Raízes Randômicas	TSP - Visitas Locais	5,9	11,9	23,6	45,8	89,3	90,4
Raízes Randômicas	TSP - Rank(V)	7,6	13,2	24,0	49,0	79,9	88,7
Raízes Equidistantes	TSP - Permutações	5,5	10,7	21,3	41,8	81,8	82,6
Raízes Randômicas	TSP - Permutações(V)	6,0	10,3	17,6	34,9	65,6	67,9
K-Means	Randômico Global	1,8	4,2	10,5	21,4	39,3	40,6
K-Means	Randômico Local	1,0	3,2	8,8	18,3	34,1	34,7

Tabela 4.12: Melhores resultados obtidos na simulação com 2 agentes no caso de um intruso observador, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

4.3.3 Intruso Estatístico

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes	TSP(V)	13,8	28,1	45,8	83,0	100,0	135,6
Raízes Equidistantes	TSP - Visitas Locais	7,8	15,3	18,6	100,0	100,0	135,4
Raízes Randômicas	TSP(V)	14,0	21,4	27,1	58,6	100,0	109,0
Raízes Equidistantes	TSP - Permutações	6,7	11,7	22,1	56,0	100,0	102,9
Raízes Randômicas	TSP - Visitas Locais(V)	6,2	13,3	21,3	47,5	100,0	96,5
Raízes Randômicas	TSP - Rank	5,2	10,9	17,1	26,8	100,0	78,9
Raízes Equidistantes	TSP - Rank(V)	2,0	4,0	9,2	11,8	100,0	63,2
Raízes Randômicas	TSP	0,7	0,4	0,5	0,0	100,0	50,3
K-Means	Randômico Local	1,3	4,8	13,8	29,3	42,0	49,1
K-Means	Randômico Global	1,8	6,9	14,9	23,9	43,7	46,8
Hierárquico	Randômico Global	1,7	6,0	11,0	21,4	36,7	39,7
Raízes Randômicas	TSP - Permutações(V)	4,1	7,2	8,8	9,2	50,0	36,8
Multilevel	Randômico Global	1,1	2,1	4,8	14,4	26,9	26,5

Tabela 4.13: Melhores resultados obtidos na simulação com 6 agentes no caso de um intruso estatístico, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes	TSP - Visitas Locais(V)	21,4	29,5	38,0	53,8	100,0	111,5
Raízes Randômicas	TSP - Visitas Locais(V)	19,1	24,7	33,6	34,0	100,0	93,9
Raízes Equidistantes	TSP - Visitas Locais	6,2	10,9	25,6	41,3	100,0	93,0
Raízes Equidistantes	TSP - Permutações	4,2	7,9	12,6	17,6	100,0	69,7
Raízes Equidistantes	TSP(V)	4,8	6,0	7,5	20,5	100,0	69,6
Raízes Equidistantes	TSP - Rank	3,7	7,1	8,5	13,4	100,0	64,8
K-Means	TSP - Visitas Locais(V)	1,0	4,7	13,9	30,8	69,4	64,0
K-Means	Randômico Global	2,2	3,8	14,3	26,9	48,1	50,4
Raízes Equidistantes	TSP	0,0	0,0	0,1	0,0	100,0	50,0
K-Means	Randômico Local	0,7	5,3	14,7	31,9	34,0	47,5
Hierárquico	TSP - Visitas Locais(V)	6,4	10,1	15,1	26,5	34,2	44,9
Multilevel	TSP - Visitas Locais(V)	1,9	5,6	10,2	19,6	37,2	38,3
Raízes Equidistantes	TSP - Permutações(V)	2,6	4,3	5,6	6,6	11,4	13,7
Raízes Equidistantes	TSP - Rank(V)	1,4	2,8	4,2	4,4	0,0	5,5

Tabela 4.14: Melhores resultados obtidos na simulação com 4 agentes no caso de um intruso estatístico, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Extensão Multiagente	**Algoritmo de Seqüenciamento**	$\mathcal{M}$ (%)					$\mathcal{A}$
		$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
K-Means	TSP - Visitas Locais(V)	9,0	11,5	24,4	30,7	100,0	84,9
Raízes Randômicas	TSP - Visitas Locais(V)	12,2	18,0	29,6	39,5	75,1	82,4
Raízes Equidistantes	TSP - Visitas Locais(V)	12,7	17,9	27,7	35,5	56,9	69,6
Multilevel	TSP - Visitas Locais(V)	0,4	0,6	1,6	16,0	100,0	62,7
K-Means	TSP - Visitas Locais	0,0	0,0	0,2	11,5	100,0	58,7
K-Means	TSP - Permutações	0,0	0,0	0,1	7,1	100,0	55,4
Raízes Randômicas	TSP(V)	2,7	1,0	0,4	0,5	100,0	50,9
K-Means	TSP - Rank	0,0	0,0	0,0	1,0	100,0	50,7
Hierárquico	TSP	0,0	0,0	0,0	0,0	100,0	50,0
K-Means	Randômico Local	2,1	3,7	9,7	21,7	30,5	36,0
K-Means	Randômico Global	1,0	3,4	9,8	18,8	33,5	35,2
Raízes Equidistantes	TSP - Rank(V)	0,3	0,5	1,4	3,5	13,8	10,2
K-Means	TSP - Permutações(V)	0,0	0,0	0,1	0,3	2,5	1,5

Tabela 4.15: Melhores resultados obtidos na simulação com 2 agentes no caso de um intruso estatístico, considerando isoladamente cada algoritmo de seqüenciamento e cada extensão multiagente.

Os resultados apresentados indicam que para invasores que atacam de maneira aleatória os esquemas tradicionais de particionamento são mais efetivos do que o uso de estratégias sem particionamento. Algumas dentre as principais razões para isso são: Conforme indicam os dados expostos na seção anterior, embora as estratégias sem particionamento quando comparadas aos demais métodos, mantêm uma média do tempo de espera inferior, o alto valor do desvio-padrão indica que as visitas dos agentes foram distribuídas de forma a não manterem um intervalo de tempo constante entre si. No caso de intrusos puramente randômicos, conforme mencionado anteriormente, o fator mais importante é garantir a eficiência do método em diminuir o tempo de espera entre visitas aos nodos, e além disso é desejável que esse intervalo entre as visitas seja o mais próximo possível de um valor constante, isso porque assim é possível normalizar ao máximo a probabilidade de impedir ataques. Quando o intervalo de espera varia muito, alguns intervalos serão menores e portanto, a probabilidade de conter um ataque entre as duas visitas definidas pelo intervalo é alta, no entanto, quanto menor for esse intervalo, menor será a chance de que um intruso randômico escolherá atacar durante esse intervalo. Como a variação é alta existirão intervalos onde a espera entre duas visitas é grande, e nesses casos a chance de que o intruso disponha de tempo suficiente para concretizar seu ataque é maior, além disso como esse tempo de espera é maior, aumentam também as chances de que um intruso

randômico escolha atacar durante esse intervalo, e dessa forma seja com maior probabilidade bem sucedido. De forma que, no caso específico do atacante randômico o ideal é que a estratégia seja eficiente e além disso mantenha intervalos constantes entre as visitas de agentes.

Ainda no que diz respeito aos invasores randômicos, o algoritmo de seqüenciamento que obteve os melhores resultados foi conforme o esperado o uso do ciclo do TSP sem quaisquer modificações. A segunda melhor alternativa faz uso do TSP com visitas locais ($\mathcal{CH}_{VL} = 1\%$), seguido do rank de soluções do TSP, essa ordem corresponde precisamente a ordenação das métricas com relação ao tempo de espera, excetuando-se os casos onde a variação é muito alta, como no caso do TSP com acréscimo de variabilidade.

Diferentemente do atacante randômico, quando o invasor dispõe de informações a respeito do patrulhamento da área onde deseja atacar, como por exemplo, um histórico de visitas de um local específico, ou mesmo no caso em que o atacante estiver apto a apenas descobrir quando houve uma visita de um patrulhador ao local alvo de sua intrusão, para esses casos as estratégias sem particionamento apresentaram melhores resultados. Os resultados obtidos quando foram utilizados o particionamento *Hierárquico Aglomerativo* ou o *K-means* são muito semelhantes e podem ser considerados equivalentes, no entanto, conforme declarado anteriormente eles são significativamente menos eficientes que as estratégias sem particionamento, esse fato esta diretamente relacionado com o maior desvio-padrão obtido com o uso dessas extensões multiagentes. Ao contrário do invasor aleatório, os invasores que consideram o uso de informações relativas ao patrulhamento no planejamento de seus ataques requerem estratégias com alta variabilidade e que mantenham ainda assim a eficiência do patrulhamento.

No caso específico do atacante escondido, o algoritmo de seqüenciamento com melhores resultados foi o TSP com aumento de variabilidade, fica bastante evidente o altíssimo desempenho desse método mesmo diante de intervalos de intrusão muito pequenos, isso ocorre evidentemente porque o patrulhador sempre faz duas visitas ao mesmo nodo com intervalo de tempo muito pequeno, voltando a visitar o nodo somente depois que visitar todos os outros vértices do grafo duas vezes. Esse método, embora, muito eficiente para conter atacantes escondidos no caso de intervalos de intrusão bastante reduzidos, é menos eficiente do que outros algoritmos quando o tempo para o ataque for maior.

Para uma avaliação consistente do atacante estatístico é necessário levar em consideração alguns fatores adicionais. De forma geral, conforme o esperado, nos casos em que o intervalo de intrusão é uma ou duas vezes o tamanho do ciclo do TSP dividido pelo número de agentes, o atacante estatístico opta por realizar pouquíssimas tentativas de invasão e por isso a taxa de susceptibilidade dos patrulhadores é bastante elevada. Além disso para o caso específico dos algoritmos de seqüenciamento fundamentalmente randômicos, não é possível para o atacante fazer uma predição adequada já que o inter-

valo de confiança esta diretamente relacionada com o valor do desvio-padrão e no caso dos patrulhadores randômicos esse valor é demasiadamente alto, no entanto, como o tempo de espera médio é muito elevado, quando o atacante faz uma tentativa de invasão ele quase que invariavelmente obtém sucesso. Embora sejam facilmente predizíveis quando tomados isoladamente, a estratégia de percorrer o ciclo do TSP continuamente, bem como o algoritmo do TSP com acréscimo de variabilidade, tornam-se métodos de difícil previsão quando realizados por uma grande quantidade de agentes patrulhadores simultâneamente. Dessa forma existe uma variação nos resultados obtidos quando a simulação conta com seis, quatro e dois agentes. Para o caso com seis agentes, o TSP com acréscimo de variabilidade apresenta o melhor resultado dentre todos os métodos, quando existem apenas quatro ou dois agentes o TSP com visitas locais ($\mathcal{CH}_{VL} = 15\%$) obteve os melhores resultados. Com quatro e dois agentes o uso do TSP com acréscimo de variabilidade resultou em valores drasticamente inferiores se comparados aos melhores métodos nesses cenários.

Outro aspecto relevante indicado nos resultados, que corrobora com o princípio de que os atacantes são de fato mais eficazes com o acréscimo de seu grau de complexidade, é a ordem decrescente assumida pelos valores obtidos na avaliação dos patrulhadores quando comparam-se os três modelos de atacantes. No caso onde existem apenas dois agentes patrulhadores, por exemplo, o melhor resultado do atacante randômico revela que o patrulhador é quase 20% mais eficiente em sua patrulha a esse modelo de ataque quando comparado com um atacante escondido. A diferença fica ainda mais perceptível quando o atacante randômico é comparado com o intruso estatístico, o patrulhador é no geral, cerca de 65% mais eficiente em conter os ataques do invasor randômico.

4.4 Determinação do modelo apropriado para um Cenário Específico

Ao longo deste capítulo foram apresentados os diversos resultados referentes à simulação dos vários modelos propostos - extensões multiagentes e algoritmos de seqüenciamento - em cada um dos cenários de avaliação considerando as variações dos intervalos de intrusão e do número de agentes envolvidos no patrulhamento. Foi proposto também uma métrica para quantificar e comparar a eficiência das várias soluções, e para a ordenação do conjunto de modelos foi proposto o uso da área do gráfico referente a variação do tamanho do intervalo de intrusão pelo resultado da métrica de comparação para cada um dos casos. No entanto, as variações significativas dos resultados para os diferentes critérios de avaliação nas diversas condições estudadas impede que seja definido de maneira categórica uma métrica que seja a melhor para todas as situações. Portanto, apresenta-se nesta seção um breve estudo a respeito de como determinar um modelo composto por um algoritmo

de seqüenciamento e uma extensão multiagente, que seja adequado para um ambiente específico.

O primeiro fator a ser determinado é o conjunto de modelos de atacantes que poderão tentar invadir o ambiente. Ainda mais importante é determinar a probabilidade com a qual ocorrerão ataques por parte de cada um dos modelos de invasores. Pode-se, por exemplo, determinar-se que para o ambiente em estudo a chance de que o atacante seja randômico é de 20%, enquanto os 80% restantes referem-se a possibilidade de que o invasor seja estatístico. Determinar a probabilidade da ocorrência de determinados modelos de atacantes é um processo bastante complexo, durante o qual será necessário a avaliação de diversos fatores, dentre os quais destacam-se:

1. A possibilidade de atacantes randômicos figurarem em tentativas de invasão está diretamente relacionada com algumas características inerentes ao ambiente. Se, por exemplo, os ganhos resultantes de uma invasão a esse ambiente são reduzidos, esse local com alta probabilidade será alvo de intrusões desprovidas de um planejamento árduo, figurando portanto, com maior probabilidade a ocorrência de atacantes randômicos. Existem também ambientes, onde uma observação dos vigilantes é muito difícil, por exemplo, locais onde o intruso não pode ficar escondido enquanto aguarda o momento apropriado para o seu ataque, nesses casos, novamente o atacante poderá optar por agir de acordo com o modelo randômico. De modo geral, no entanto, a atuação dos atacantes randômicos é mais comum em locais onde o nível de patrulhamento é muito baixo ou inexistente, em geral, atacantes não optarão por uma invasão em um momento decidido de maneira aleatória quando sabem que o ambiente é patrulhado.

2. De forma geral, a ação de atacantes escondidos estão relacionados a ataques que possam ser facilmente divididos sem deixar vestígios no alvo da intrusão, por exemplo, tentar todas as combinações de um cofre, até encontrar aquela que seja a adequada. Esse tipo de ataque não deixa qualquer marca no ambiente, e pode ser facilmente reduzido a um conjunto bem definido de passos com um intervalo de intrusão muito pequeno. Outra característica que poderá aumentar a probabilidade da ocorrência desse tipo de ataque é o conhecimento prévio do atacante de que aquele ambiente seja patrulhado, de forma que, ele será inibido a tentar uma invasão em um momento aleatório. Sabendo que o ambiente é patrulhado o invasor, se possível, esperará uma visita de um agente, para assim maximizar o tempo do qual irá dispor em sua tentativa de invasão. Dessa forma, a ação desse modelo de intrusão esta associada a locais onde os atacantes sabem que o nível de patrulhamento é intermediário.

3. Por fim a ação de atacantes estatísticos esta associada, naturalmente, a locais onde uma invasão poderá surtir ganhos financeiros elevados, e portanto, ambientes com

patrulhamento intensificado. De forma que, os atacantes irão reconhecer a necessidade de um planejamento prévio ao definir o momento em que deverão iniciar o ataque, para isso terão de coletar e avaliar informações a respeito do patrulhamento atual e então definir quando o ataque será realizado.

Embora conhecendo o comportamento dos atacantes seja possível estimar com certa precisão a probabilidade da ocorrência de cada um dos modelos, é importante notar que a chance de um determinado atacante atuar está diretamente relacionado com o comportamento dos patrulhadores. Dessa forma, ao determinar um modelo apropriado para o cenário atual é preciso considerar o fato de que essa decisão por si só irá mudar o cenário atual, alterando a perspectiva de atuação dos invasores no ambiente. Por isso a escolha do modelo de patrulhamento deverá ser uma atividade contínua, levando em consideração toda e qualquer mudanças significativa do ambiente.

Outro fator imprescindível que deverá ser estimado enquanto investiga-se qual é o modelo apropriado para o cenário em questão refere-se ao número de agentes patrulhadores. Novamente, são muitos os detalhes que influenciam diretamente nessa decisão, destacam-se nesse ínterim os seguintes fatores:

1. O primeiro fator a ser considerado na determinação do número adequado de agentes refere-se, evidentemente, ao tamanho do ambiente onde será efetuado o patrulhamento. Existem diversas métricas que poderão auxiliar o processo de determinar quantos vigilantes o tamanho do ambiente exige, por exemplo, o ciclo do TSP que permite uma comparação precisa com ambientes que já possuam patrulhamento.

2. Não basta, no entanto, levar em consideração o tamanho do ambiente, é preciso também avaliar a conectividade dos locais a serem patrulhados. Ambientes com várias áreas de baixa conectividade em geral irão demandar uma quantidade maior de agentes.

3. Embora este trabalho não faça distinção das regiões de patrulhamento no sentido de dedicar maior prioridade a determinadas áreas, é importante levar em consideração o impacto de uma invasão, ambientes que estejam relacionados a valores financeiros elevados certamente demandarão um patrulhamento mais eficiente e portanto, uma quantidade maior de agentes.

4. Outro aspecto muito importante diz respeito ao modelo dos possíveis invasores, por exemplo, ambientes que sejam alvo direto de intrusos estatísticos ou escondidos, que podem coordenar seus ataques de forma a explorar falhas no patrulhamento, irão demandar mais agentes para aumentar a tolerância a erros, garantindo a integridade do ambiente sob vigilância.

O terceiro parâmetro a ser estimado é o intervalo provável de tempo que um ataque poderá levar. Para facilitar a comparação com outros cenários, aconselha-se que a medida de tempo seja calculada em termos do tamanho do ciclo do TSP dividido pelo número de agentes.

Evidentemente, a maneira mais eficiente de avaliar as soluções propostas para o ambiente no cenário específico é rodar a simulação considerando as características levantadas. No entanto, para os casos onde não é possível fazê-lo, propõe-se o uso de uma métrica simples que apontará de forma geral as melhores soluções para o cenário de acordo com os parâmetros passados:

$$\mathcal{F} = P_{Rand} \times \mathcal{A}_{Rand} + P_{Escondido} \times \mathcal{A}_{Escondido} + P_{Estat} \times \mathcal{A}_{Estat}$$

onde P_{Tipo_Ataque} e A_{Tipo_Ataque} referem-se, respectivamente, à probabilidade da ocorrência de um tipo de ataque, e a área obtida pela simulação para aquele tipo de ataque considerando o número de agentes e o intervalo estimado do tempo de ataque. Para casos onde o intervalo de tempo estimado é diferente para cada um dos atacantes será preciso normalizar o valor calculado em cada uma das situações.

Para ilustrar o uso da métrica foram especificados dois cenários, ambos consideram os intervalos de intrusão utilizados ao longo de todo o trabalho ($\{\frac{1}{8}, ..., 2\}$). O primeiro cenário, avalia as soluções para o caso onde a probabilidade de um atacante randômico, escondido ou estatístico são respectivamente, 20%, 30% e 50%. Para o segundo caso de estudo apresentado foram considerados os seguinte valores: 50%, 30% e 20%.

Número de Agentes	**Extensão Multiagente**	**Algoritmo de Seqüenciamento**	$\mathcal{F}$
6	Raízes Equidistantes	TSP (V)	122,7
6	Raízes Equidistantes	TSP - Visitas Locais	119,5
6	Raízes Randômicas	TSP (V)	109,4
4	Raízes Equidistantes	TSP (V)	89,1
4	Raízes Equidistantes	TSP - Visitas Locais	88,6
4	Raízes Randômicas	TSP - Visitas Locais	88,3
2	Raízes Equidistantes	TSP (V)	78,3
2	Raízes Randômicas	TSP (V)	78,3
2	K-Means	TSP - Visitas Locais	75,0

Tabela 4.16: Melhores resultados obtidos durante uma simulação na qual considerou-se que a probabilidade de um intruso randômico, observador ou estatístico valem respectivamente 20%, 30% e 50%.

Para este primeiro cenário, por conta da alta probabilidade de um atacante estatístico ou escondido, as soluções sem particionamento obtiveram melhores resultados, além disso, o melhor algoritmo de seqüenciamento em todos os casos foi o TSP com acréscimo de variabilidade.

Já para o segundo cenário, com o acréscimo da probabilidade de um ataque randômico, as soluções, naturalmente, foram as mais eficientes, como percorrer o ciclo original do TSP, ou o TSP com visitas locais.

Número de Agentes	**Extensão Multiagente**	**Algoritmo de Seqüenciamento**	$\mathcal{F}$
6	Raízes Equidistantes	TSP - Visitas Locais	111,8
6	Raízes Equidistantes	TSP (V)	106,9
6	Raízes Randômicas	TSP - Visitas Locais	105,5
4	Raízes Equidistantes	TSP	100,5
4	Raízes Randômicas	TSP	100,4
4	Raízes Randômicas	TSP - Visitas Locais	100,0
2	K-Means	TSP	102,2
2	K-Means	TSP - Visitas Locais	98,6
2	K-Means	TSP - Visitas Locais	96,1

Tabela 4.17: Melhores resultados obtidos durante uma simulação na qual considerou-se que a probabilidade de um intruso randômico, observador ou estatístico valem respectivamente 50%, 30% e 20%.

Capítulo 5

Agente Corrompido

Tendo por base a idéia de incluir a tolerância a falhas na tarefa de patrulhar determinada área, foi implementada uma extensão aos critérios de avaliação - que constam na seção 3.2. Para isso, a intrusão foi remodelada considerando o corrompimento de um dos agentes patrulhadores, um agente corrompido, ou deturpado, deixa de realizar corretamente sua tarefa como patrulhador sem impedir que um ataque ocorra mesmo quando ele poderia fazê-lo. No entanto, o agente corrompido não influência diretamente no comportamento dos demais patrulhadores. Neste trabalho, foi considerado a possibilidade de apenas um dentre todos os agentes ser corrompido, no entanto, para permitir a comparação adequada é escolhido sempre um dentre os agentes responsáveis por patrulhar o nodo-alvo.

Em termos práticos a ocorrência de patrulhadores corrompidos está relacionada, por exemplo, a vigilantes humanos que foram subornados, de forma a não impedir um atacante. Outra possibilidade é no caso de um sistema de verificação automatizada de redes, um dos computadores onde é executado o serviço de avaliação da disponibilidade da rede, poderia ser invadido de forma a comprometer a atividade específica desempenhada por aquela instância de verificação de disponibilidade do serviço.

Este capítulo foi estruturado da seguinte maneira, inicialmente serão definidas as alterações nos *critérios de avaliação*. Redefini-se então a forma como os vértices serão distribuídos entre os diversos agentes, de forma a incluir características que garantirão a robustez do sistema diante do comprometimento de um agente responsável pelo patrulhamento. Essa redistribuição dos nodos entre os agentes será abordada na seção *extensão multiagente*. Por fim, serão apresentados os principais resultados experimentais obtidos.

5.1 Critérios de avaliação

De forma semelhante aos critérios de avaliação originais, o ataque será realizado somente a um nodo do grafo de patrulha, esse vértice será novamente escolhido de maneira aleatória

dentre todos os nodos do grafo. Suponha que v seja o nodo escolhido para ser o alvo dos ataques. A principal diferença no caso da extensão corrente, é que dentre todos os agentes envolvidos na simulação um deles será *corrompido*, de forma que, sua ação no patrulhamento ficará totalmente comprometida. Na prática isso significa que se o intruso estiver atacando o nodo v e receber uma visita do agente patrulhador corrompido $\mathcal{A}_\mathcal{C}$ ele continuará o ataque sem qualquer impedimento por parte do agente deturpado. A escolha do agente patrulhador a ser deturpado, será feita de forma a beneficiar ao máximo o intruso, portanto, o agente deturpado $\mathcal{A}_\mathcal{C}$ será aquele que dentre todos os agentes tiver visitado com mais freqüência o nodo alvo. Se houverem vários agentes patrulhadores com a mesma quantidade de visitas ao nodo alvo, um deles será escolhido de forma aleatória. Para permitir a seleção do agente mais freqüente no nodo alvo, a simulação foi dividida em dois processos, no primeiro é feita a execução por parte dos agentes, de acordo com o algoritmo de seqüenciamento e a forma de particionamento escolhida, quando o tempo limite é atingido a execução é considerada completa e esse passo findo. No segundo passo são feitas as simulações das intrusões. Serão apresentados os detalhes deste passo para cada um dos critérios de avaliação, individualmente:

No *intruso randômico* remodelado, novamente, o momento inicial dos ataques será escolhido de maneira aleatória. Escolhido o instante em que o ataque terá início, a invasão é realizada. O histórico de visitas dos agentes, obtido durante a execução do passo anterior da simulação, é então verificado avaliando se houveram visitas durante o intervalo da intrusão. Se o resultado for negativo, ou seja, não houveram quaisquer visitas ao nodo alvo desde o momento inicial até o término da intrusão, ou então, se apenas o agente corrompido tiver feito visitas durante este intervalo, então o ataque é considerado bem sucedido. Caso contrário, se algum agente, que não seja o agente deturpado houver visitado o vértice alvo, somente então, a intrusão será considerada impedida.

Para o caso do *intruso escondido (ou observador)*, o atacante permanece escondido até que seja feita uma visita por um patrulhador que não seja o agente corrompido, e somente então, o invasor fará uma tentativa de invasão. Novamente, se durante o intervalo de intrusão não houverem visitas de agentes, ou as visitas forem exclusivamente por parte do agente deturpado, o ataque será considerado bem sucedido, e portanto, o patrulhamento ineficiente. A escolha por desconsiderar as visitas do agente corrompido ao definir os momentos iniciais de ataque foi feita pois se a visita for feita por um agente que não seja o patrulhador deturpado, a probabilidade de que o próximo agente a visitar o nodo alvo será o agente corrompido aumenta.

De foma semelhante ao caso anterior o *intruso estatístico* aguarda até uma visita de um patrulhador que não seja o vigilante corrompido, neste momento o atacante irá avaliar seus dados e decidir se a invasão será realizada, ou não. Para a construção do histórico de visitas, no qual baseia-se a decisão de realizar uma tentativa de ataque, serão considerados

apenas as visitas dos agentes não corrompidos.

Evidentemente, considerando a possibilidade de existir um agente corrompido os modelos com particionamento terão de ser adaptados, isso porque nestes modelos cada vértice será patrulhado por apenas um agente, de forma que, se este agente for corrompido, então não haverá qualquer possibilidade de que o atacante seja impedido. Dessa forma as soluções abordadas na próxima seção tratam de alternativas para incluir redundância nos métodos de particionamento. As extensões multiagentes que consideravam a possibilidade do uso de todos os agentes para o patrulhamento de todos os vértices são métodos inerentemente redundantes e portanto serão consideradas como soluções alternativas para o problema, para estes casos não serão incluídas quaisquer alterações no método original.

5.2 Extensão Multiagente

5.2.1 Particionamento Reduzido

A solução mais natural ao incluir redundância no particionamento dos nodos entre os agentes consiste em reduzir a quantidade de partições existentes pela metade, garantindo assim que cada um dos vértices seja atribuído a dois agentes durante o particionamento. Dessa forma, para cada um dos métodos de particionamento apresentados na seção 3.4.2 ao invés de utilizar como número de partições a serem criadas a quantidade de agentes envolvidos no patrulhamento, esse número será dado por

$$k = \frac{\mathcal{N}_{agentes}}{2}$$

Para cada uma das partições obtidas serão atribuídos dois agentes, dessa forma, fica garantida a redundância na execução do patrulhamento. Considerando o fato de que durante as simulações um dos agentes da partição que possui o vértice alvo será corrompido, apenas um agente patrulhador será responsável por uma área que na média será duas vezes maior do que no caso original do problema do patrulhamento, sem agentes corrompidos. Portanto, o tempo de espera dos nodos aumentará significativamente, comprometendo seriamente a eficiência do patrulhamento. Conforme descrito na seção introdutória desse capítulo dentre todos os agentes responsáveis pelo patrulhamento do nodo alvo será escolhido como agente corrompido aquele que houver realizado o maior número de visitas, portanto, no caso específico desse método o patrulhador restante será o único responsável pelo patrulhamento do nodo alvo e como ele possivelmente realizou menos visitas a esse vértice do que o agente corrompido, o nodo alvo receberá freqüentemente menos da metade das visitas quando comparado com o caso original.

5.2.2 Particionamento Incremental

Outra possibilidade na alteração do particionamento, consiste em aumentar a área de cobertura dos agentes para além do conjunto de vértices atribuído por meio do particionamento. Isso será feito, incluindo aleatoriamente uma proporção constante ($\mathcal{INC}_{\%}$) de vértices em cada uma das partições originais. A proporção de nodos adicionais é fixa, e relativa à quantidade total de vértices do grafo completo, portanto, se $\mathcal{INC}_{\%} = 20\%$, cada partição será acrescida de $|V| \times 0.2$ vértices. A escolha dos nodos a serem incluídos nas partições, conforme mencionado anteriormente, será puramente randômica, sem levar em consideração quaisquer características, como a proximidade do vértice para com a partição. Isso porque, ao considerar um atributo, como a distância, alguns nodos, próximos de diversas partições, geralmente localizados em regiões mais centrais do grafo seriam privilegiados enquanto nodos periféricos poderiam ser prejudicados.

5.2.3 Fuzzy C-Means

Ao contrário da estratégia anterior, que não considerava qualquer característica na escolha dos nodos que seriam incluídos às partições, essa estratégia considera a minimização da distância quando atribui os nodos às diferentes partições.

O *Fuzzy C-Means* é um método de agrupamento de dados, e assim como os algoritmos do *K-means* e o agrupamento hierárquico é muito conhecido, e amplamente utilizado em diversos contextos. Em especial no particionamento *fuzzy*, assim como na lógica *fuzzy* tradicional, é atribuído a cada um dos pontos uma medida de participação (*degree of belonging*) com relação a cada uma das partições. Essa medida esta diretamente relacionada com a distância entre os pontos e os centros das partições, de forma que, pontos localizados nas extremidades da partição estarão relacionados a partição com um menor grau de participação do que pontos próximos ao centro da partição. Para cada um dos vértices (v) é calculado iterativamente um coeficiente que define o grau de participação desse nodo para com a i-ésima partição (u_{iv}). Tradicionalmente inclui-se uma restrição adicional de que a soma de todos os coeficientes para cada vértice deverá resultar em 1, possibilitando que os coeficientes sejam interpretados como a probabilidade do vértice v fazer parte da partição i:

$$\forall v \sum_{i=1}^{k} u_{iv} = 1$$

O método de agrupamento por meio do *Fuzzy C-Means* é bastante semelhante ao *K-Means*, apresentam-se agora de forma simplificada os principais passos:

1. Como no *K-Means*, escolhe-se aleatoriamente os valores iniciais, no caso do *Fuzzy C-Means* são escolhidos randomicamente os valores do coeficiente de participação

dos nodos para cada uma das partições (u_{iv}). Obedecendo, no entanto, a restrição de que a soma dos coeficientes para um único vértice é 1.

2. Calcula-se o valor do centróide das partições, fazendo uso das seguintes equações

$$\mathcal{C}_i.x = \frac{\sum_v (u_{iv})^m v.x}{\sum_v (u_{iv})^m}$$

$$\mathcal{C}_i.y = \frac{\sum_v (u_{iv})^m v.y}{\sum_v (u_{iv})^m}$$

onde, $\mathcal{C}_i.x$ refere-se à coordenada x do centróide da i-ésima partição, analogamente, $\mathcal{C}_i.y$ refere-se à coordenada y da mesma partição; $v.x$ e $v.y$ referem-se às coordenadas x e y do nodo v e m é uma constante real com valor maior que 1, quanto mais m se aproximar do valor 1 maior será a importância dada aos vértices próximos ao centro do cluster, de forma que aos nodos localizados perto do centróide serão atribuídos pesos muito maiores do que aos vértices periféricos, obtendo-se um resultado muito semelhante ao $k-means$. Quando $m = 2$ normaliza-se o coeficiente linearmente de forma que a soma deles torne-se 1. Para obedecer a restrição tradicional de que a soma dos coeficiente seja 1, em todos os experimentos foi utilizado o valor $m = 2$.

3. Para cada um dos pontos recalcula-se o coeficiente de participação nos clusters, do seguinte modo

$$u_{iv} = \frac{1}{\sum_j \left(\frac{dist(\mathcal{C}_i, v)}{dist(\mathcal{C}_j, v)}\right)^{\frac{2}{(m-1)}}}.$$

onde $dist(\mathcal{C}_i, v)$ refere-se à distância euclidiana do centróide da i-ésima partição para com o vértice v.

4. Se o número máximo de iterações ($\mathcal{MAX}_{ITER}$) ainda não foi alcançado e algoritmo não convergiu, então volta-se ao passo 2. Quando a diferença entre os coeficientes de participação de uma iteração para a próxima for menor do que uma constante pré-determinada (ϵ) então diz-se que o algoritmo convergiu.

5.2.4 Estratégias Cíclicas

Conforme mencionado anteriormente, as estratégias cíclicas naturalmente incluem redundância, isso porque todos os agentes compartilham a atribuição de patrulhar o conjunto completo de vértices. Sendo assim, mesmo que um dos agentes seja corrompido a redundância inerente a essa solução assegura a possibilidade de que o vértice seja patrulhado por um conjunto de agentes. Fica claro também, que essa estratégia assegura o

maior grau de redundância possível, assegurando assim, o menor impacto no patrulhamento no caso de existirem vigilantes corrompidos. Por exemplo, no caso de um único agente deturpado, para um conjunto de 6 agentes, o impacto no número de visitas recebidas por um dos vértices será de apenas $\frac{1}{6}$, enquanto nas demais estratégias um vértice potencialmente receberá menos da metade do número de visitas.

5.3 Resultados Experimentais

São apresentados a seguir os resultados obtidos com a simulação de cada uma das soluções propostas. Para cada uma das quatro propostas apresentadas na seção anterior serão exibidos os três melhores resultados obtidos. A ordenação dos valores para cada uma das estratégias, como no capítulo anterior, será dado por meio do cálculo da área do gráfico obtido na relação das constantes do intervalo de intrusão e o conjunto de resultados obtidos na simulação para cada uma dessas constantes. Na apresentação dos resultados o algoritmo de seqüenciamento do TSP com visitas locais foi abreviado para *TSP - VL*.

5.3.1 Intruso Randômico

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Randômicas		TSP	10.3	18.9	35.1	59.9	86.4	105.5
Raízes Equidistantes		TSP	9.9	19.2	34.9	58.8	85.6	104.2
Raízes Equidistantes		TSP - Rank	9.0	17.2	32.4	55.8	82.9	99.2
Part. Reduzido	K-Means	TSP	5.8	11.2	23.0	46.1	86.9	89.2
	K-Means	TSP - VL	5.3	10.9	22.1	43.3	82.4	84.3
	K-Means	TSP - Rank	5.2	10.4	21.1	41.8	80.4	81.7
Part. Incremental	Multilevel	TSP	6.9	13.5	25.7	46.0	72.5	83.3
	K-Means	TSP	7.0	13.9	25.6	45.5	72.7	83.2
	Hierárquico	TSP	6.7	13.5	25.4	45.2	72.4	82.5
Fuzzy C-Means		TSP	5.1	9.7	18.9	35.9	70.7	71.5
		TSP - VL	4.7	9.5	18.6	34.7	66.4	68.3
		TSP - Rank	4.8	8.9	17.3	33.5	64.4	65.8

Tabela 5.1: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso randômico em um contexto com 6 agentes patrulhadores, dentre os quais um foi corrompido.

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes		TSP	9.1	17.6	33.1	57.6	87.2	103.1
Raízes Randômicas		TSP	9.1	17.6	32.7	56.7	86.4	101.8
Raízes Randômicas		TSP - VL	8.3	16.2	30.1	53.6	82.7	96.4
Part. Reduzido	K-Means	TSP	6.0	11.8	23.7	47.3	90.8	92.3
	K-Means	TSP - VL	5.6	11.1	21.8	44.0	84.0	85.6
	K-Means	TSP - Rank	5.5	10.8	20.9	42.5	83.2	83.7
Fuzzy C-Means		TSP	5.9	11.3	22.3	43.7	85.5	86.4
		TSP - VL	5.3	10.6	20.8	41.0	79.4	80.6
		TSP - Rank	5.1	10.4	20.4	39.9	78.0	78.8
Part. Incremental	Multilevel	TSP	6.6	12.9	24.6	44.7	73.5	82.4
	K-Means	TSP	6.5	12.7	24.5	44.9	72.8	82.0
	Hierárquico	TSP	6.3	12.6	23.8	43.5	71.5	80.1

Tabela 5.2: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso randômico em um contexto com 4 agentes patrulhadores, dentre os quais um foi corrompido.

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Part. Reduzido	K-Means	TSP	6.2	12.3	25.0	49.7	98.8	98.7
	Multilevel	TSP	6.1	12.4	24.5	49.7	98.8	98.6
	Hierárquico	TSP	63.0	12.6	24.7	49.4	98.9	98.5
Raízes Randômicas		TSP	6.3	12.4	24.7	49.8	98.8	98.7
Raízes Equidistantes		TSP	6.2	12.5	24.7	49.5	98.8	98.5
Raízes Randômicas		TSP - Rank	5.7	11.2	22.7	45.0	89.2	89.3
Fuzzy C-Means		TSP	6.2	12.5	24.7	49.7	98.8	98.7
		TSP - Rank	5.7	11.1	22.4	44.8	89.1	89.0
		TSP - VL	5.5	11.2	22.4	44.2	88.5	88.2
Part. Incremental	Multilevel	TSP	6.0	12.2	24.1	48.6	96.9	96.6
	K-Means	TSP	5.7	11.4	22.7	45.3	90.4	90.2
	Multilevel	TSP - Rank	5.6	11.1	22.1	43.8	87.3	87.2

Tabela 5.3: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso randômico em um contexto com 2 agentes patrulhadores, dentre os quais um foi corrompido.

5.3.2 Intruso Observador

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes		TSP (V)	30.4	52.9	59.5	66.6	77.0	122.6
Raízes Randômicas		TSP (V)	30.4	52.9	59.3	66.1	76.9	122.0
Raízes Equidistantes		TSP	7.2	15.9	30.9	52.6	79.3	94.1
Part. Incremental	K-Means	TSP (V)	24.4	46.1	55.2	59.7	67.1	109.2
	Multilevel	TSP (V)	23.0	45.1	55.1	59.7	67.0	108.8
	Hierárquico	TSP (V)	24.3	45.5	55.0	59.5	66.8	108.7
Part. Reduzido	Hierárquico	TSP (V)	26.2	46.7	50.3	51.3	55.8	95.6
	K-Means	TSP (V)	25.6	45.9	49.9	50.0	50.1	91.5
	Multilevel	TSP (V)	10.0	28.5	47.4	51.0	51.1	87.5
Fuzzy C-Means		TSP (V)	26.1	45.0	50.9	52.0	53.1	94.7
		TSP - VL (V)	9.6	16.3	19.1	22.5	26.5	41.0
		TSP - VL (V)	7.2	12.3	14.3	18.2	22.0	32.8

Tabela 5.4: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso observador em um contexto com 6 agentes patrulhadores, dentre os quais um foi corrompido.

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Randômicas		TSP (V)	43.1	53.9	57.3	63.0	72.6	117.9
Raízes Equidistantes		TSP (V)	43.3	53.8	57.0	62.6	72.6	117.4
Raízes Randômicas		TSP	5.5	12.6	24.6	45.1	73.4	82.4
Part. Incremental	Multilevel	TSP (V)	35.4	49.3	51.9	55.0	61.0	102.7
	K-Means	TSP (V)	36.8	49.2	51.8	55.0	61.0	102.7
	Hierárquico	TSP (V)	37.0	49.2	51.9	54.9	60.7	102.5
Part. Reduzido	Hierárquico	TSP (V)	40.5	49.7	50.4	51.9	54.4	96.9
	K-Means	TSP (V)	39.5	49.6	50.0	50.0	50.2	93.1
	Multilevel	TSP (V)	28.1	46.8	50.7	50.8	50.8	93.0
Fuzzy C-Means		TSP (V)	37.8	49.2	50.6	51.0	51.8	94.7
		TSP - VL (V)	15.1	19.9	21.9	23.9	26.3	44.0
		TSP - VL (V)	11.5	14.9	16.4	18.2	20.6	33.6

Tabela 5.5: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso observador em um contexto com 4 agentes patrulhadores, dentre os quais um foi corrompido.

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Part. Reduzido	Hierárquico	TSP (V)	49.9	50.0	50.0	50.0	50.0	93.8
	K-Means	TSP (V)	49.8	50.0	50.0	50.0	50.0	93.8
	Multilevel	TSP (V)	49.8	50.0	50.0	50.0	50.0	93.8
Raízes Randômicas		TSP (V)	49.8	50.0	50.0	50.0	50.0	93.8
Raízes Equidistantes		TSP (V)	49.8	50.0	50.0	50.0	50.0	93.8
Raízes Randômicas		TSP - VL (V)	20.0	20.8	21.0	21.3	21.7	39.8
Fuzzy C-Means		TSP (V)	49.8	50.0	50.0	50.0	50.0	93.8
		TSP - VL (V)	20.1	20.8	21.1	21.3	21.7	39.9
		TSP - VL (V)	14.3	14.7	14.9	15.0	15.2	28.2
Part. Incremental	Multilevel	TSP (V)	49.0	49.1	49.1	49.1	49.1	92.1
	K-Means	TSP (V)	45.5	45.8	45.8	45.8	45.8	85.8
	Hierárquico	TSP (V)	33.4	33.5	33.5	33.5	33.5	62.9

Tabela 5.6: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso observador em um contexto com 2 agentes patrulhadores, dentre os quais um foi corrompido.

5.3.3 Intruso Estatístico

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Equidistantes		TSP - VL (V)	4.5	11.2	20.6	87.5	100.0	125.7
Raízes Randômicas		TSP - VL	4.8	11.6	29.3	78.6	100.0	122.4
Raízes Randômicas		TSP - VL	6.7	10.6	22.4	80.0	100.0	120.8
Fuzzy C-Means		TSP - VL (V)	4.5	9.1	14.3	30.9	47.0	54.1
		TSP - VL (V)	1.5	3.1	5.8	19.0	30.0	32.1
		TSP - VL	0.5	1.6	5.3	16.5	31.9	30.6
Part. Incremental	Multilevel	TSP - VL (V)	3.5	6.0	11.4	21.4	36.6	40.0
	K-Means	TSP - VL (V)	3.0	6.1	12.0	19.9	34.5	38.0
	Multilevel	TSP - VL (V)	2.9	5.9	10.9	16.1	32.3	33.6
Part. Reduzido	K-Means	TSP - VL (V)	1.9	4.6	8.9	17.2	28.4	31.4
	Hierárquico	TSP - VL	0.9	2.5	6.4	15.7	27.7	28.6
	K-Means	TSP - VL	1.2	2.8	6.5	15.0	27.0	27.7

Tabela 5.7: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso estatístico em um contexto com 6 agentes patrulhadores, dentre os quais um foi corrompido.

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Raízes Randômicas		TSP - VL (V)	12.3	17.2	23.9	100.0	100.0	138.0
Raízes Randômicas		TSP - VL	2.1	4.6	11.0	90.0	100.0	122.6
Raízes Randômicas		TSP - VL (V)	14.8	20.6	28.4	70.0	100.0	117.9
Fuzzy C-Means		TSP - VL (V)	7.0	10.5	16.2	38.5	39.7	57.2
Fuzzy C-Means		TSP - VL (V)	2.2	4.0	8.1	27.1	36.0	42.2
Fuzzy C-Means		TSP - VL	0.8	2.4	8.0	19.5	37.8	37.0
Part. Incremental	Multilevel	TSP - VL (V)	5.5	9.7	15.1	26.8	42.7	49.3
Part. Incremental	K-Means	TSP - VL (V)	5.7	10.2	16.5	27.2	32.5	45.1
Part. Incremental	Hierárquico	TSP - VL (V)	5.3	9.3	15.2	25.6	32.7	43.3
Part. Reduzido	Multilevel	TSP - VL	1.2	2.8	8.8	20.0	33.8	35.8
Part. Reduzido	K-Means	TSP - VL (V)	3.2	6.8	10.8	18.4	32.4	35.5
Part. Reduzido	Hierárquico	TSP - VL	1.6	4.0	9.0	20.1	30.8	34.7

Tabela 5.8: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso estatístico em um contexto com 4 agentes patrulhadores, dentre os quais um foi corrompido.

Extensão Multiagente		**Algoritmo de Seqüenciamento**	Taxa de Captura (%)					$\mathcal{A}$
			$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	
Fuzzy C-Means		TSP - VL (V)	10.6	13.4	22.5	47.6	42.4	68.5
Fuzzy C-Means		TSP - VL (V)	3.1	4.6	10.2	33.2	30.7	45.2
Fuzzy C-Means		TSP - VL	1.6	5.2	12.5	22.7	34.7	40.1
Raízes Equidistantes		TSP - VL (V)	10.2	12.9	22.3	47.8	41.5	68.1
Raízes Randômicas		TSP - VL (V)	10.3	12.6	20.4	40.1	32.3	56.9
Raízes Equidistantes		TSP - VL (V)	3.1	4.6	10.3	32.2	28.2	43.2
Part. Incremental	Multilevel	TSP - VL (V)	10.9	14.0	22.6	43.4	42.3	65.5
Part. Incremental	K-Means	TSP - VL (V)	11.6	15.5	24.1	37.0	36.6	58.7
Part. Incremental	Hierárquico	TSP - VL	2.4	7.3	18.1	30.9	38.9	50.9
Part. Reduzido	Hierárquico	TSP - VL	2.3	7.3	18.1	30.9	38.9	50.9
Part. Reduzido	K-Means	TSP - VL	2.5	7.0	18.3	29.1	30.1	45.2
Part. Reduzido	Multilevel	TSP - VL	2.2	7.1	18.1	28.2	29.0	43.9

Tabela 5.9: Resultados Experimentais da simulação dos modelos propostos para o caso de um intruso estatístico em um contexto com 2 agentes patrulhadores, dentre os quais um foi corrompido.

Para todos os casos com 6 ou 4 agentes, os melhores resultados foram obtidos com o uso da solução cíclica como extensão multiagente, isso porque, conforme mencionado anteriormente, essa estratégia obtém a maior redundância possível ao incluir todos os agentes no patrulhamento de todos os vértices. É importante ressaltar que as diferenças nos resultados obtidos com o uso da estratégia cíclica para com os demais métodos é bastante significativa nesses dois casos. Em especial no caso do intruso estatístico a diferença torna-se ainda maior chegando a ser mais de três vezes maior do que os resultados obtidos com os outros métodos. No caso onde existem apenas 2 agentes patrulhadores, evidentemente, as soluções obtiveram resultados muito semelhantes, isso porque, como para a maioria dos métodos a redundância mínima permitida é a de dois agentes por vértice, novamente para esses casos todos os vértices serão responsabilidade de todos os agentes.

Para os atacantes randômico e observador, não houveram modificações significativa na escolha do melhor algoritmo de seqüenciamento se comparados com os mesmos modelos de atacantes no caso original, sem agentes corrompidos. Para o intruso randômico o uso do TSP original foi a melhor solução, já para o atacante observador a melhor solução encontrada foi obtida com o uso do TSP com alteração para o acréscimo da variabilidade. No caso específico do intruso estatístico com 6 agentes houve uma pequena alteração dos resultados se eles forem comparados com os valores obtidos no problema original, ao invés do TSP com acréscimo de variabilidade obter o melhor resultado, nesse caso, o TSP com visitas locais e acréscimo de variabilidade alcançou os melhores resultados.

Capítulo 6

Conclusão

O problema de patrulhar uma área orquestrando a ação de diversos agentes patrulhadores é bastante complexo e conforme demonstrado ao longo do trabalho, em muitos casos a solução ideal deverá integrar eficiência e impredictabilidade. A principal contribuição deste trabalho está associada a uma nova abordagem dada ao problema, movendo o foco no desenvolvimento do critério de avaliação da perspectiva exclusiva do agente patrulhador para uma avaliação baseada em modelos de intrusos. De forma que, foram criadas três novas métricas de avaliação, cada uma delas considera diferentes tipos de atacantes, partindo de um invasor com comportamento aleatório até um intruso que planeja seu ataque fazendo uso de métodos estatísticos. Baseando-se nessas métricas são propostas e posteriormente comparadas diversas arquiteturas, que envolvem a criação de algoritmos de seqüenciamento e extensões multiagentes.

As soluções propostas neste trabalho são fundamentalmente *homogêneas* e centralizadas. As soluções são ditas *homogêneas* porque no caso onde diversos agentes são responsáveis pelo patrulhamento todos eles utilizarão um único algoritmo de seqüenciamento. Do ponto de vista de sistemas multiagentes as soluções são ditas centralizadas porque um agente não influencia no comportamento dos demais patrulhadores. Por exemplo, no caso do TSP com visitas locais cada agente irá tomar sua decisão referente a escolha de realizar ou não uma visita a um dos nodos adjacentes, sem levar em consideração a decisão tomada por outros agentes. Existem, no entanto, algumas exceções, casos onde existe uma influência indireta no processo de tomada de decisão por outros agentes, isso acontece no caso exclusivo dos algoritmos de seqüenciamento com acréscimo de variabilidade (onde algumas decisões são determinísticas) para as extensões multiagentes sem particionamento. Nesses casos as soluções assumem um comportamento descentralizado.

Foram propostas também metodologias para a avaliação e comparação das arquiteturas fundamentando-se nos critérios de avaliação. Para permitir essa comparação propôs-se também um gerador de cenários experimentais, onde foram criados os ambientes que

fizeram parte da simulação.

Por fim, o trabalho apresenta uma extensão que considera o caso no qual um agente patrulhador foi corrompido, propondo novas extensões multiagentes que buscam minimizar o impacto do problema no patrulhamento. Novamente os resultados obtidos foram apresentados, avaliados e comparados.

6.1 Trabalhos Futuros

Baseando-se nas necessidades encontradas durante o desenvolvimento do trabalho obteve-se um conjunto de possíveis extensões ao trabalho atual, enunciadas a seguir:

- Modificações no gerador de cenários:
  - Incluir a possibilidade de tratar caminhos de sentido único, ou seja, modificar a perspectiva atual para uma que faz uso de grafos orientados;
  - Tratar o caso onde existem arestas que não são estáticas (e.g. obstáculos móveis);
  - Considerar o cenário de patrulhamento não-uniforme, que contém regiões com diferentes níveis de prioridade;
- *TSP relaxado* - Deixar de considerar a restrição de que durante o percurso de um ciclo completo no grafo pode-se passar apenas uma vez por cada um dos nodos. Embora essa restrição faça parte do TSP, ela não é imprescindível para o patrulhamento, portanto, pode ser removida sem qualquer prejuízo;
- Desenvolver metodologias para averiguar a quantidade necessária de agentes para patrulhar determinado ambiente;

De forma adicional às sugestões anteriores, destacam-se três propostas, que baseiam-se em extensões já implementadas mas que por conta de problemas encontrados durante seu desenvolvimento ainda não foram completamente finalizados e despontam, portanto, como possíveis trabalhos futuros:

6.1.1 *Vehicle Routing Problem* - VRP

Considerando as extensões multiagentes, uma das soluções de particionamento mais naturais para o problema de dividir os vértices entre os agentes consiste no uso da solução proveniente do problema de roteamento de veículos (*Vehicle Routing Problem* - VRP). O objetivo do VRP refere-se a definir as rotas ideais para um conjunto de veículos, tal que

todos os consumidores definidos no mapa tenham sua demanda atendida. Os veículos têm capacidade limitada e são provenientes de um ou mais depósitos. A soma do conjunto de rotas para todos os veículos deverá ser mínima. A similaridade com o problema de particionar o grafo no caso do patrulhamento multiagente é muito grande. Basta considerarmos que cada um dos vértices é um consumidor com uma demanda constante, por exemplo, igual a um, os número de veículos disponíveis no caso do patrulhamento referem-se precisamente aos agentes vigilantes. A capacidades dos patrulhadores, pode ser definida de forma a garantir que seja atribuído a cada um dos agentes uma quantidade razoável de agentes. Alterando-se a capacidade do patrulhador, por exemplo, pode-se garantir que será atribuído a cada um dos agentes no mínimo três vértices, ou então, pode-se fixar a capacidade como infinita, e nesse caso, naturalmente, o patrulhador poderá ficar responsável pelo conjunto quase completo dos vértices ($n_v - n_a$, onde n_v é o número de vértices e n_a é o número de agentes). No entanto, o problema do roteamento exige a definição de um ou mais depósitos, de onde, os veículos partem inicialmente. De forma que, o problema encontrado durante o uso do VRP na busca de um particionamento ideal dos vértices entre os agentes, foi que no caso específico do patrulhamento não existe um (ou mais) depósito inicial. Pode-se por exemplo criar um vértice adicional ao grafo, que seja vizinho de todos os vértices do grafo, e definir esse vértice como sendo o depósito inicial, no entanto, não foi possível determinar o valor correto a ser atribuído a cada uma das arestas entre os vértices do grafo e esse nodo artificial. Portanto, propõe-se como extensão ao trabalho buscar uma forma de solucionar o problema encontrado, de forma a permitir o uso do VRP no particionamento do grafo.

6.1.2 Ambientes Tridimensionais

O simulador criado para a execução dos experimentos suporta o uso de ambientes tridimensionais, adicionando ao gerador de grafos uma série de novos parâmetros:

- Número de andares ($\mathcal{N}_{andares}$);
- Deverá ser definida uma quantidade fixa de acessos entre os diversos níveis com um peso constante. Só existem acessos entre níveis que sejam subsequentes, por exemplo, do primeiro para o segundo nível, do segundo para o terceiro, e assim por diante, não existem portanto, acessos diretos do primeiro para o terceiro nível. Novamente os acessos são não orientados, e portanto poderão ser percorridos em qualquer sentido. O peso constante atribuído a cada um dos acessos será calculado fazendo-se o peso médio de todas as arestas do grafo dividido por uma constante, definida a priori.

- Os acessos entre níveis serão criados escolhendo-se de forma aleatória um nodo que pertença a cada um dos níveis, a única restrição utilizada é a de que um único nodo não faça parte de mais de um acesso, mesmo que para níveis diferentes.

Embora o simulador atual já ofereça suporte a ambientes tridimensionais, a utilização desse tipo de ambiente nos testes exige diversas modificações nas extensões multiagentes, isso porque, faz-se necessário, por exemplo, a redefinição da função de distância entre os vértices. Para o caso das extensões multiagentes sem particionamento, ou o K-means, é possível utilizar a distância calculada através do caminho mínimo entre todos os vértices, garantindo que a única forma de obter acesso a vértices em outros níveis seja através do acessos criados anteriormente. O particionamento hierárquico aglomerativo necessita de maiores modificações já que mesmo com o uso do caminho mínimo, em geral, as partições resultantes serão disconexas. Propõe-se então, como trabalho futuro, a alteração das extensões multiagentes buscando comportar as singularidades de ambientes tridimensionais.

6.1.3 Grafos não hamiltonianos

A abstração de muitos ambientes pode resultar em grafos não hamiltonianos, de forma que, esses grafos não possuem um ciclo-solução para o TSP. Como muitas dentre as soluções propostas neste trabalho baseiam-se precisamente no ciclo do TSP, grafos não-hamiltonianos foram desconsiderados. O uso de um *TSP Relaxado*, proposto anteriormente, no qual não existe a restrição de que o ciclo contenha apenas uma instância de cada vértice, é uma possível solução. Para o caso específico no qual o grafo possui arestas de corte e portanto, garantidamente, não possui um ciclo hamiltoniano, propõe-se uma solução alternativa: inicialmente as arestas de corte deverão ser identificadas, feito isso, os dois vértices incidentes a aresta de corte serão duplicados, mantendo para cada novo vértice a mesma lista de adjacência do vértice original, incluindo no entanto uma aresta entre os dois vértices duplicados, e uma aresta para cada um dos vértices, conectando o vértice original à sua cópia. Naturalmente, obtida a solução sempre que um dos vértices duplicados for visitado, essa visita será transformada em uma visita ao nodo original. Infelizmente, não existem garantias de que essa solução torne o novo grafo obtido com a duplicação dos vértices, hamiltoniano, no entanto, essa solução poderá ser bem sucedida, principalmente em casos mais simples, que incluem, por exemplo, grafos k-regulares, onde k seja um valor alto, com o acréscimo de alguns caminhos sem saída (vértices ligados a uma única aresta). Propõe-se, portanto, o estudo de soluções que viabilizem o uso das arquiteturas baseadas no ciclo do TSP mesmo em grafos não-hamiltonianos.

Referências Bibliográficas

[1] Alessandro Almeida, Geber Ramalho, Hugo Santana, Patricia Azevedo Tedesco, Talita Menezes, Vincent Corruble, and Yann Chevaleyre. Recent advances on multi-agent patrolling. In *SBIA '04: Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, volume 3171 of *Lecture Notes in Computer Science*, pages 474–483, São Luis, Maranhão, Brazil, September 2004. Springer.

[2] Greg Barnes and Uriel Feige. Short random walks on graphs. *SIAM Journal on Discrete Mathematics*, 9(1):19–28, 1996.

[3] Yann Chevaleyre, Francois Sempe, and Geber Ramalho. A theoretical analysis of multi-agent patrolling strategies. In *AAMAS '04: Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, volume 03, pages 1524–1525, Los Alamitos, CA, USA, July 2004. IEEE Computer Society.

[4] W Chin and S Ntafos. Optimum watchman routes. In *SCG '86: Proceedings of the second annual symposium on Computational geometry*, pages 24–33, New York, NY, USA, 1986. ACM.

[5] V. Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory*, 18:39–41, 1975.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, July 2001.

[7] Jacques Ferber. Multi-agent system: An introduction to distributed artificial intelligence. *Journal of Artificial Societies and Social Simulation*, 4(2), 2001.

[8] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand. Cooperative control for multiple autonomous uav's searching for targets. *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, 3:2823–2828, 2002.

[9] H. Gould and J. Tobochnik. *Introduction to Computer Simulation Methods: Applications to Physical Systems, Part 1 and 2*. Addison-Wesley, Reading, MA, USA, 1988.

[10] J. Grace and J. Baillieul. Stochastic strategies for autonomous robotic surveillance. *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 2200–2205, 2005.

[11] David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining (Adaptive Computation and Machine Learning)*. The MIT Press, August 2001.

[12] J.P. Hespanha, Hyoun Jin Kim, and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, 3:2432–2437, 1999.

[13] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[14] Roy Jonker and Ton Volgenant. Nonoptimal edges for the symmetric traveling salesman problem. *Operations Research*, 32:837–846, 1984.

[15] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[16] Aydano Machado, Geber Ramalho, Jean-Daniel Zucker, and Alexis Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In Jaime Simão Sichman, François Bousquet, and Paul Davidsson, editors, *MABS '02: Proceedings of the 3rd International Workshop on Multi-Agent-Based Simulation*, volume 2581 of *Lecture Notes in Computer Science*, pages 155–170, Bologna, Italy, July 2002. Springer.

[17] Aydano P. Machado. Patrulha multiagente: Uma análise empírica e sistemática. Master's thesis, UFPE: Universidade Federal de Pernambuco, 2002.

[18] Nimrod Megiddo, S. Louis Hakimi, M. R. Garey, David S. Johnson, and Christos H. Papadimitriou. The complexity of searching a graph. *Journal of the ACM*, 35(1):18–44, January 1988.

[19] Talita Menezes, Patrícia Tedesco, and Geber Ramalho. Negotiator agents for the patrolling task. In *IBERAMIA-SBIA '06: Proceedings of the 17th 2nd International Joint Conference, 10th Ibero-American Conference on AI, 18th Brazilian AI Symposium*, volume 4140 of *Lecture Notes in Computer Science*, pages 48–57, Ribeirão Preto, Brazil, October 2006. Springer.

[20] Simeon Ntafos and Markos Tsoukalas. Optimum placement of guards. *Inf. Sci. Inf. Comput. Sci.*, 76(1-2):141–150, 1994.

[21] Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987.

[22] Wei pang Chin and Simeon C. Ntafos. Shortest watchman routes in simple polygons. *Discrete & Computational Geometry*, 6:9–31, 1991.

[23] T. Parsons. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*, pages 426–441. Springer-Verlag, 1976.

[24] Gerhard Reinelt. *The Traveling Salesman, Computational Solutions for TSP Applications*, volume 840 of *Lecture Notes in Computer Science*. Springer, 1994.

[25] M. Riedmiller and A. Merke. *Using machine learning techniques in complex multi-agent domains*. Uknown, 2002.

[26] Hugo Santana, Geber Ramalho, Vincent Corruble, and Bohdana Ratitch. Multi-agent patrolling with reinforcement learning. In *AAMAS '04: Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, volume 03, pages 1122–1129, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[27] S.K. Subramanian and J.B. Cruz. Adaptive models of pop-up threats for multi-agent persistent area denial. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 1:510–515, 9-12 Dec. 2003.

[28] A. Volgenant and Roy Jonker. A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. *European Journal of Operational Research*, 9:83–89, 1982.

[29] A. Volgenant and Roy Jonker. The symmetric traveling salesman problem and edge exchanges in minimal 1-trees. *European Journal of Operational Research*, 12:394–403, 1983.