UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

JODY MAICK ARAUJO DE MATOS

MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA OFF-LINE DE PLANEJAMENTO DE ROTAS PARA AMBIENTES BIDIMENSIONAIS ESTÁTICOS

FEIRA DE SANTANA
2012

JODY MAICK ARAUJO DE MATOS

MODELAGEM E IMPLEMENTAÇÃO DE UM SISTEMA OFF-LINE DE PLANEJAMENTO DE ROTAS PARA AMBIENTES BIDIMENSIONAIS ESTÁTICOS

Trabalho de Conclusão de Curso apresentado ao Colegiado de Engenharia de Computação como requisito parcial para obtenção do grau de Bacharel no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana.

Orientador: Anfranserai Morais Dias

FEIRA DE SANTANA
2012

TERMO DE APROVAÇÃO

Jody Maick Araujo de Matos

Trabalho de Conclusão de Curso apresentado ao Colegiado de Engenharia de Computação como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação pela Universidade Estadual de Feira de Santana, e julgada em ____/_____/_____ perante a banca examinadora:

Universidade Estadual de Feira de Santana
Dr. Anfranserai Morais Dias

Universidade Estadual de Feira de Santana
Dr. Delmar Brolgio Carvalho

Universidade Estadual de Feira de Santana
Dr. Matheus Giovanni Pires

À minha amada esposa, Joseane de Matos: inspiração constante, ajudadora até o último momento e porto seguro sempre presente.

AGRADECIMENTOS

À Deus, primeiramente, "*porque dEle, por Ele e para Ele são todas as coisas*" (Romanos 11:32).

À minha esposa, amiga, companheira e dedicada ajudadora, Joseane Carvalho Santos de Matos, por todo o esforço desprendido para fazer dessa jornada a mais suave possível, regada de muito amor, carinho e compreensão.

Aos meus pais, Jorge Pereira de Matos e Tania das Graças Araujo de Matos, e ao meu irmão, Yuri Araujo de Matos, pelo apoio incondicional.

Aos amigos/irmãos, João Carlos Nunes Bittencourt e Claudia Asevedo Mattos Bittencourt, por compartilharem dos momentos mais importantes nesse período, dentro e fora do ambiente acadêmico. Um VIVA ao Quarteto Fantástico.

Aos Professores Dr. Anfranserai Morais Dias e Dr. Delmar Broglio Carvalho, pelas experiências vividas, ensinamentos incalculáveis, apoio e orientação em todo o processo.

Ao Professor Dr. Matheus Giovanni Pires, pela disponibilidade durante todo o projeto e auxílio no processo de correção da monografia, elementos essenciais para minha aprovação.

Aos amigos, Paulo Roberto Cerqueira de Carvalho e Melchizedec Santos Gonçalves Junior, por me apresentarem uma formação que não se encontra em nenhuma Instituição de Ensino Superior e me permitirem acessá-los independente de estrutura hierárquica. Os Srs. são pessoas excepcionais.

Aos amigos, Fernando Alberto, Marcelo Miranda, Nils Bergsten, Tiago Oliveira, Karine Almeida, Tassalon Ferreira e tantos outros, que por uma questão de tempo e espaço não foram citados, pela troca de experiências, excelentes discussões, horas de estudos e momentos inesquecíveis.

A esses e todos os demais que contribuiram para essa conquista e seguem contribuindo para que outras sejam alcançadas, meu MUITO OBRIGADO!

RESUMO

Este trabalho descreve as etapas de modelagem e desenvolvimento de um sistema *off-line* de planejamento de rotas para ambientes bidimensionais estáticos. Para a decomposição do espaço de trabalho, o projeto propõe utilizar a Diagramação de Voronoi, implementada a partir da Triangulação de Delaunay. Para a tarefa de busca no grafo resultante e escolha do melhor caminho, a abordagem proposta utiliza um algoritmo de inteligência artificial baseado na metodologia de Colônia de Formigas. Para uma melhor parametrização das análises, uma plataforma de testes foi elaborada e os resultados obtidos com a abordagem proposta foram comparados com resultados utilizando o Algoritmo de Dijkstra. Outras duas análises foram realizada, indicando os impactos no resultado final do planejamento com a variação da quantidade de formigas e de iterações do algoritmo. Os principais resultados indicam o sistema proposto como uma solução viável para a tarefa de planejamento de rotas, seja num escopo local, como uma aplicação *standalone*, ou escopos globais, inserido em projetos maiores. Pode, ainda, produzir rotas inteligentes, adaptáveis à cada aplicação, somente variando os parâmetros do algoritmo de Colônia de Formigas. O caminho produzido possui um número muito reduzido de nós em comparação com o Dijkstra, o que caracteriza uma vantagem significativa, pois optar por caminhos com uma quantidade menor de nós e, consequentemente, com menos curvas para um sistema robótico, pode representar uma maior autonomia energética na tarefa de movimentação no espaço de trabalho.

Palavras-chave: Robôs Autônomos. Planejamento de Rotas. Diagrama de Voronoi. Otimização por Colônia de Formigas

ABSTRACT

This paper describes the steps for modeling and development an off-line path planning system for two-dimensional and static environments. The project proposes a Voronoi Diagram approach for workspace decomposition task, implemented from the Delaunay Triangulation. The path is choosed using an artificial intelligence algorithm, based on Ant Colony Optimization, by searching the best one in a graph-based structure. A test platform was proposed to provide an analytical environment. The results were compared with results using the Dijkstra's Shortest Path Algorithm. Two further analyses was performed, indicating the impact of amount ants and iterations floating in the final results. The main results show the proposed system as a viable solution to the task of path planning, whether in a local scope, as a standalone application, or global scopes, inserted into larger projects. It can also produce intelligent routing, adaptable to each application, just changing the Ant Colony parameters.The resulting path has a very small number of nodes in comparison with the Dijkstra's approach, which can be seen as an advantage, since choose routes with a smaller number of nodes and thus with fewer turns to a robotic system may represent greater energy autonomy in moving task in the workspace.

Keywords: Autonomous Robots. Path Planning. Voronoi Diagram. Ant Colony Optimization

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IA	Inteligência Artificial
ACO	Ant Colony Optimization
TSP	Traveling Salesman Problem
AS	Ant System
MMAS	MAX-MIN Ant System
ACS	Ant Colony System
GEV	Grafo Estendido de Voronoi

SUMÁRIO

1 INTRODUÇÃO

Um dos principais objetivos na área de robótica é a criação de robôs capazes de realizar uma tarefa com o mínimo de interferência possível. A meta, então, é fazer com que um robô, que por ora pode ser interpretado como um dispositivo mecânico versátil equipado com sensores e atuadores, controlado por um sistema computacional, possa cumprir tarefas após ter sido indicado a ele apenas **o que** ele deverá fazer, sem a necessidade de especificar **como** fazer (LATOMBE, 2004; LAUMOND, 1998).

O desenvolvimento de pesquisas e novas tecnologias para robôs autônomos torna-se um empreendimento formidável para as mais diversas aplicações nos campos afins, com ramificações profundamente entrelaçadas com raciocínio automatizado, percepção e controle (LATOMBE, 2004). Segundo Laumond (1998), os sistemas de programação para robôs ainda têm uma capacidade de planejamento de movimento muito limitada, principalmente quando aliada ao crescimento das áreas de pesquisa em robótica, como aplicações em exploração espacial, trabalho submarino, intervenção em ambientes de risco, serviços robóticos, dentre outros. Isso conduz o problema de planejamento de movimentos a um dos principais componentes para a autonomia necessária aos robôs no mundo real.

A posição, ou *pose*, de um robô é normalmente descrita por um conjunto de variáveis. Para robôs móveis, as mais típicas são variáveis como localização baseada em coordenadas x e y num plano, e sua orientação naquele ponto. Para robôs articulados, como braços robóticos ou robôs manipuladores, as mais comuns são as posições das juntas e do manipulador no espaço, ou descritas pelo ângulo de rotação de cada junta em relação a um referencial. Um movimento para um robô pode, consequentemente, ser considerado como um caminho (ou uma rota) a partir de uma *pose* inicial até uma *pose* final (BERG *et al.*, 2008; LATOMBE, 2004).

Tendo em vista que no mundo real os ambientes estão constituídos de objetos, móveis, paredes ou quaisquer outras partes que possam estar inseridas no meio, o planejamento de rotas de um robô no seu espaço de trabalho deve limitar-se àqueles livres de colisões com essas partes (BERG *et al.*, 2008; LATOMBE, 2004). Esse problema já foi caracterizado por J. T. Schwartz e M. Sharir, em 1983, como "O Problema do Movimento de Pianos", e posteriormente generalizado por J. Reif como o "Problema do Móvel Generalizado".

Mais recentemente, dentre outras diversas possibilidades de classificação, os trabalhos sob o escopo do planejamento de rotas têm se dividido entre planejamentos *on-line* e *off-line*, os quais se caracterizam em quanto do espaço de trabalho se conhece previamente. No primeiro deles, o *on-line*, não há conhecimento prévio sobre o espaço de trabalho ou posicionamento de obstáculos no meio em que o robô está inserido, proporcionando um planejamento de rotas dinâmico; já no *off-line*, detalhes do espaço de trabalho e do posicionamento de obstáculos são

previamente conhecidos, possibilitando um planejamento estático da rota (BERG *et al.*, 2008; LAUMOND, 1998).

Há uma imensa gama de possibilidades quanto às abordagens utilizadas para solucionar problemas de planejamento de rotas. Porém, cada estratégia possui uma série de particularidades, como abordagens cujo espaço de trabalho seja, necessariamente, bidimensional; outras, onde os obstáculos sejam poligonais. Entretanto, as principais técnicas são baseados em abordagens gerais: *roadmap*, decomposição em células e campo potencial (BERG *et al.*, 2008; GAVRILOVA, 2008; KLEIN, 1989; LATOMBE, 2004). Tomando como base as abordagens encontradas na literatura, técnicas fundamentadas na Diagramação de Voronoi apresentam-se como uma solução interessante para o projeto proposto, pois caracterizam-se por produzir os caminhos mais seguros numa decomposição espacial (BARCLAY; GALTON, 2009; GOLD, 2009; HUMPHREY, 2009). Essa característica adequa-se à proposta de planejamento de rotas para sistemas robóticos, garantindo maior segurança e obtendo excelentes performances quando tratando-se de ambientes com muitos corredores.

Devido à possibilidade de encontrar mínimos locais ou soluções subótimas, os métodos mais tradicionais de busca em grafos podem não ser bons candidatos para a gama de problemas enfrentados em trabalhos de otimização, como o caso de planejamento de rotas. A aplicação de algoritmos com abordagens mais inteligentes pode proporcionar resultados mais eficientes, dirimindo as dificuldades enfrentadas nas abordagens convencionais. Uma análise bibliográfica mostra que, para os problemas tratados nesse projeto, melhores resultados são obtidos com algoritmos de otimização utilizando colônia de formigas (DORIGO; MANIEZZO; GAMBARDELLA, 1997; DORIGO; MANIEZZO; COLORNI, 1996; SERAPIAO, 2009).

Nesse sentido, este trabalho tem o objetivo de modelar e implementar um sistema de planejamento de rotas *off-line* para robôs móveis autônomos, inseridos em ambientes bidimensionais estáticos, aplicando uma abordagem baseada em Colônia de Formigas. A proposta é elaborar rotas inteligentes para a realização da tarefa de deslocamento. Primeiramente, o sistema fará a decomposição do espaço de trabalho em regiões de Voronoi utilizando a Triangulação de Delaunay. A partir da decomposição, é gerado um grafo das rotas possíveis dentro do espaço de trabalho livre de obstáculos. Por fim, é usada uma variação do *Ant Colony System* para implementar um sistema inteligente para busca de uma trajetória no grafo. O objetivo é utilizar da parametrização do algoritmo inteligente e propor rotas que se adequem à demanda do projetista, que pode necessitar ora do menor caminho, ora do caminho com uma menor quantidade de curvas, por exemplo. Deve, ainda, partir de uma *pose* inicial até uma final, quando possíveis, ou indicar uma negativa quando não houver caminho possível

entre as configurações sugeridas.

Neste trabalho, o Capítulo 2 aborda os principais conceitos e técnicas a serem utilizados no decorrer do projeto; no Capítulo 3 são detalhadas as etapas em que o projeto foi desenvolvido, seguido de suas respectivas descrições; o Capítulo 4 apresenta os resultados obtidos no desenvolvimento do trabalho; e uma discussão final sobre o projeto é realizada no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os principais conceitos envolvidos no desenvolvimento do trabalho. A Seção 2.1 apresenta a teoria relacionada ao Diagrama de Voronoi. Na Seção 2.2, os conceitos sobre Triangulação de Delaunay são discutidos. A Seção 2.3 aborda a relação existente entre as teorias de Voronoi e Delaunay. Na Seção 2.4, as teorias básicas sobre a abordagem de Inteligência de Artificial e o *Ant Colony System* são apresentadas.

2.1 DIAGRAMA DE VORONOI

Os conceitos abordados no Diagrama de Voronoi exploram um fato geométrico evidente que, para cada ponto num sistema de pontos, é sempre possível distinguir uma parte do espaço que está mais próxima de um ponto em detrimento dos demais (GAVRILOVA, 2008). Essa região é chamada de *Região de Voronoi*, e seu conjunto, formado por todos os pontos cobrindo os espaços sem intervalo e superposição, é chamado de *Diagrama de Voronoi*, ou também *Mosaico de Voronoi* (BERG *et al.*, 2008; GAVRILOVA, 2008; KLEIN, 1989).

Segundo Klein (1989), Okabe *et al.* (2000) e Laumond (1998), uma abordagem matemática dos conceitos pode ser obtida definindo $S = \{p_1, p_2, ..., p_n\}$ como um conjunto de n pontos diferentes no plano. Para p, q e $z \in S$, todos eles diferentes entre si, sejam:

$$B(p,q) = \{z \in \Re^2; |p-z| = |q-z|\} \tag{1}$$

$$D(p,q) = \{z \in \Re^2; |p-z| < |q-z|\} \tag{2}$$

$B(p,q)$ é o bissetor perpendicular do seguimento de reta que une p à q. Dos dois semiplanos separados por $B(p,q)$, $D(p,q)$ é aquele que contem p, como apresentado na Figura 1a.

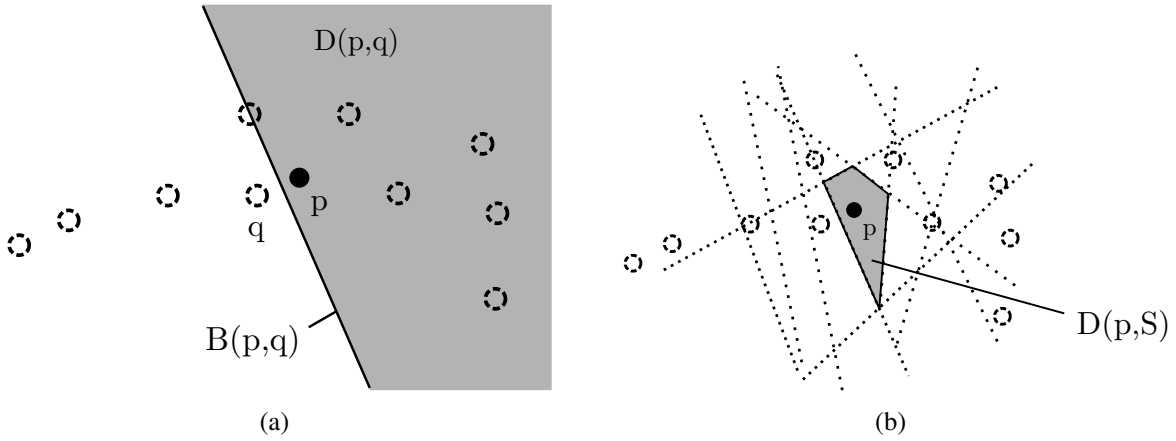


Figura 1: Representação da Região de Voronoi.

O conjunto:

$$D(p,S) = \bigcap_{q \in S, q \neq p} D(p,q) \tag{3}$$

de todos os pontos z que estão mais próximos de p do que de qualquer outro elemento de S é chamado de *região de Voronoi de p em relação à S*, como apresentado na Figura 1b. Nessa memsa Figura, o ponto p é também chamado de sítio de Voronoi.

A união:

$$V(S) = \bigcup_{p \in S} \partial D(p,S) \tag{4}$$

de todas as regiões limítrofes é chamada de *Diagrama de Voronoi* de S, apresentado como $V(S)$ na Figura 2. O limite comum de duas regiões de Voronoi é uma *aresta de Voronoi*. Duas arestas se encontram num *vértice de Voronoi*.

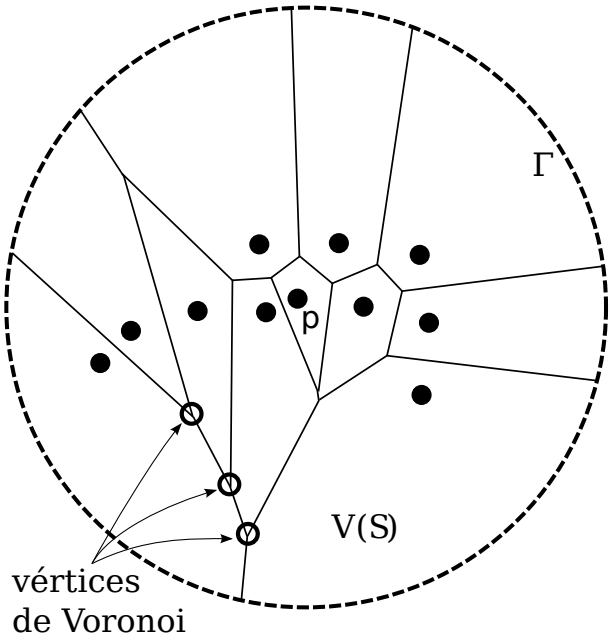


Figura 2: Representação do Diagrama de Voronoi.

2.2 TRIANGULAÇÃO DE DELAUNAY

Seja P um conjunto de pontos no plano:

1. três pontos p_i, p_j, e $p_k \in P$ são vértices da mesma face de um elemento da triangulação de Delaunay se, e somente se, a circunferência através desses pontos não contém pontos de P em seu interior;

2. dois pontos p_i, e $p_j \in P$ formam uma aresta de Delaunay de P se, e somente se, existir um círculo fechado C que contenha p_i e p_j sobre sua circunferência que não contenha nenhum outro ponto de P em seu interior;

Dessa forma, tem-se que uma triangulação de P, denotada como $T_D(P)$, é uma *Triangulação de Delaunay* se, e somente se, as circunferências que circunscrevem os triângulos de $T_D(P)$ não contêm pontos de P em seu interior. A Figura 3 apresenta um exemplo de Triangulação de Delaunay.

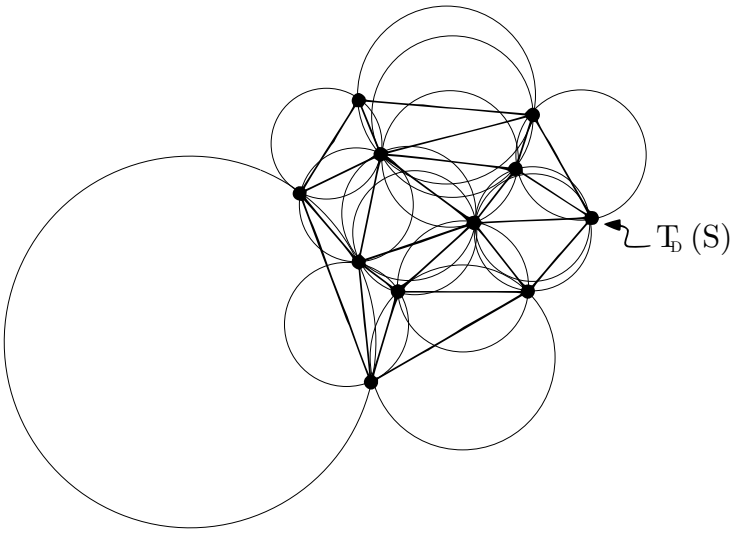


Figura 3: Exemplo da Triangulação de Delaunay.

2.3 DUALIDADE VORONOI-DELAUNAY

O "particionamento" proporcionado pela abordagem de Voronoi define um mosaico dual em relação ao próprio Diagrama de Voronoi. Esse mosaico consiste das estruturas mais simples possíveis considerando a dimensão sob o escopo em que se aborda o problema, como um tetraedro de qualquer forma num espaço 3D, ou triângulo num espaço 2D. A Figura 4 ilustra essas estruturas, que são conhecidas como *Simplex de Delaunay*, doravante tratadas por *simplex* (GAVRILOVA, 2008).

Um Simplex de Delaunay forma, então, um aglomerado estrutural, cujos vértices são os centros atômicos dos objetos no espaço (GAVRILOVA, 2008). Sua dualidade com o Diagrama de Voronoi se dá pelo fato de que esses centros atômicos de Delaunay são os sítios de Voronoi, enquanto que os vértices de Voronoi são os centros das circunferências que circunscrevem os triângulos de Delaunay, num caso específico de espaço bidimensional, podendo ainda ser generalizado para espaços de ordem maior que dois (OKABE *et al.*, 2000; GAVRILOVA, 2008). A Figura 5 apresenta uma representação gráfica da sobreposição dos diagramas de Voronoi, $V(S)$ na Figura, e Delaunay, $T_D(S)$, encontrados na literatura como *Mosaico Delaunay-Voronoi*.

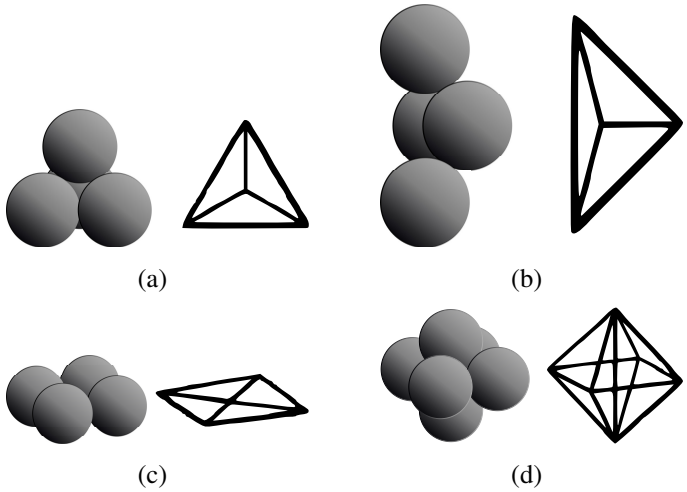


Figura 4: Exemplos de Simplex de Delaunay no espaço.

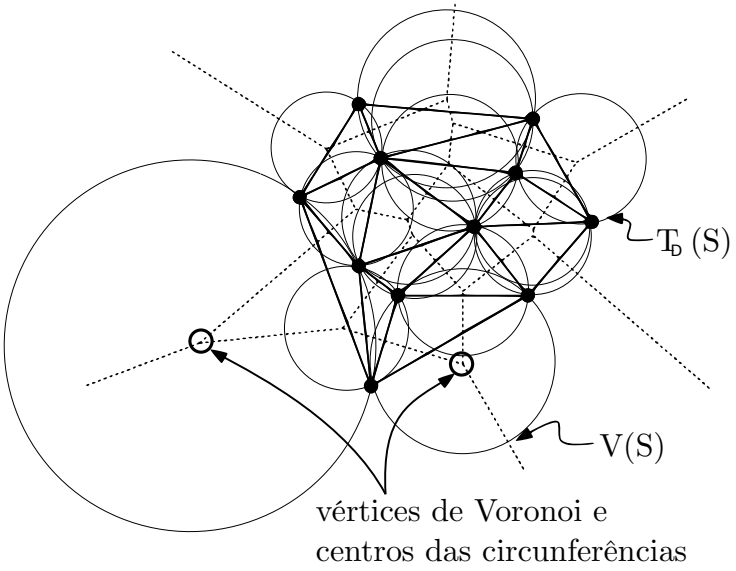


Figura 5: Exemplo de Mosaico Delaunay-Voronoi.

2.4 INTELIGÊNCIA ARTIFICIAL E OTIMIZAÇÃO COM COLÔNIA DE FORMIGAS

O primeiro trabalho oficialmente reconhecido como sendo de Inteligência Artificial (IA) foi realizado por Warren McCulloch e Walter Pitts, em 1943. Posteriormente, em 1949, Donald Hebb propôs uma rede de neurônios artificiais capaz de aprender e de processar informações, cujo modelo continua a ser influente até os dias de hoje (RUSSEL; NORVIG, 2004). Reconhecida como ciência somente em 1987, com a inserção de rigorosos experimentos empíricos e análise estatística dos resultados, trata-se, então, do estudo do comportamento inteligente, seja em humanos, animais ou máquinas, e tenta encontrar formas nas quais cada comportamento poderia projetar qualquer tipo de artefato ou trabalhar na resolução de problemas (RUSSEL; NORVIG, 2004; WHITBY, 2009).

Neste sentido, diversos algoritmos bioinspirados têm sido recorrentemente utilizados na resolução dos mais variados problemas. O uso de procedimentos baseados em populações e metaheurísticas recebem atenção especial nas aplicações de problemas de busca e otimização em vários domínios, cujas soluções robustas são difíceis ou impossíveis de encontrar usando abordagens tradicionais, como a programação matemática. Com base em processos que conduzem uma estimativa de população inicial em direção a uma solução ótima, esses algoritmos são conhecidos como algoritmos de computação evolutiva (BONABEAU; DORIGO; THERAULAZ, 1999; WHITBY, 2009).

Um exemplo onde essas abordagens alcançam resultados promissores são casos onde

> não-linearidades e interações complexas entre variáveis de projeto (x) e variáveis operacionais ($g(x) \leq 0,\ h(x) = 0$) em problemas de engenharia (f) formam um espaço de busca (S) que pode conter várias soluções ótimas ($x^* | f(x^*) < f(x), \forall x \in S \subset \Re^n$), como no caso de um problema de minimização. Devido a possibilidade de encontrar mínimos locais ou soluções subótimas, os métodos baseados em gradiente podem não ser bons candidatos como algoritmos de otimização eficientes quando aplicados a uma ampla gama de projetos de engenharia e de problemas operacionais (SERAPIAO, 2009).

As abordagens evolutivas podem ainda ter outro destaque numa classe diferente de algoritmos: a inteligência de enxame, ou *Swarm Intelligence*. Seus trabalhos têm sido inspirados no estudo do comportamento de insetos sociais, como formigas, abelhas, cupins e vespas. (BONABEAU; DORIGO; THERAULAZ, 1999; RUSSEL; NORVIG, 2004; SERAPIAO, 2009).

Fechando o escopo sobre o comportamento social de formigas, percebe-se que elas são capazes de encontrar o menor caminho entre o seu ninho e uma fonte de alimento sem o uso de trilhas visuais, fazendo-o com a exploração de informações de feromônio. Enquanto andam, elas depositam essa substância por onde passam e, reproduzindo um comportamento probabilístico considerando o feromônio previamente depositado por outras formigas, escolhem qual o melhor caminho a ser seguido (BECKERS; DENEUBOURG; GOSS, 1992; GOSS *et al.*, 1989). A Figura 6 ilustra o processo de tomada de decisão com base no feromônio, utilizado pelas formigas observadas no experimento aplicado por Goss *et al.* (1989).

Inicialmente, cada formiga escolhe aleatoriamente um dos dois caminhos (Figura 6a). Com o passar do tempo, a aleatoriedade na escolha inicial do caminho é bem reduzida, além de um fator importante se mostrar bastante decisivo: as formigas que escolhem o caminho mais curto são as primeiras a alcançar a fonte de comida (ou o ninho, a depender do sentido do trecho), como na Figura 6b. Como esse caminho recebe mais feromônio em menos tempo, há maior probabilidade que as formigas façam essa escolha em detrimento do caminho mais longo, como nas Figuras 6c e 6d (DORIGO; BIRATTARI; STUTZLE, 2006; GOSS *et al.*, 1989).

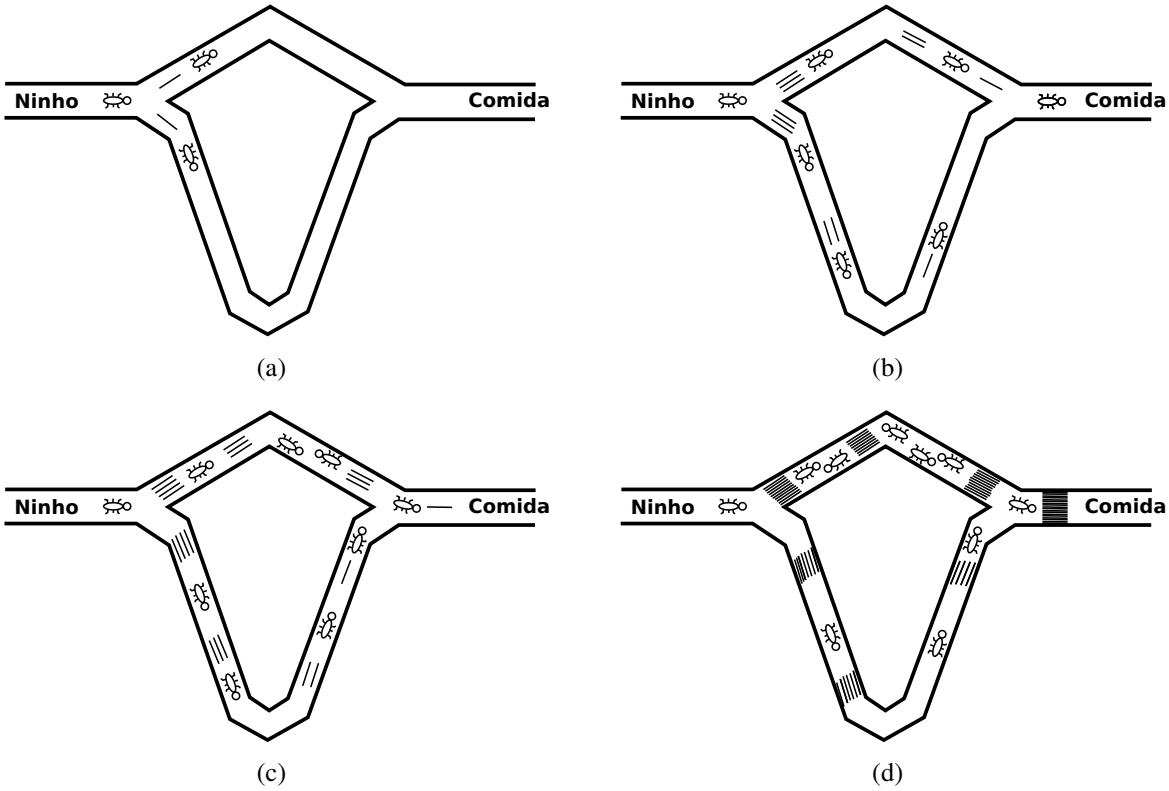


Figura 6: Comportamento das formigas na escolha de caminhos.

Essa característica inspirou uma classe de algoritmos de inteligência artificial, chamada de *Ant Colony Optimization* (ACO), ou Otimização por Colônia de Formigas, que vem obtendo bons resultados na resolução de problemas de otimização combinatorial, como o conhecido *Travelling Salesman Problem* (TSP), ou Problema do Caixeiro Viajante (COLORNI; DORIGO; MANIEZZO, 1991a, 1991b; DORIGO; MANIEZZO; GAMBARDELLA, 1997), além do Problema de Atribuição Quadrática (MANIEZZO; COLORNI; M, 1994) e dos problemas de *Multicasting*[1] e *Unicasting*[2] numa rede de computadores (CHIRICO, 2012; DORIGO; MANIEZZO; GAMBARDELLA, 1997).

A proposta dessa abordagem é, então, caracterizada enquanto uma meta-heurística da inteligência de enxames, baseada no comportamento de agentes, com as seguintes características:

(i) não-determinismo, inspirado no comportamento de formigas para a determinação de caminhos a partir de suas colônias para procura eficiente de fontes de comida;

(ii) algoritmo paralelo e adaptativo, pois trata-se de uma população onde os agentes se movem simultaneamente e sem um controle ou supervisão central;

[1] *Multicasting*, em rede de computadores, trata-se da entrega de uma mensagem a um grupo de destino com uma única transmissão.

[2] *Unicasting*, em rede de computadores, trata-se da entrega de uma mensagem a único destino.

(iii) algoritmo cooperativo, onde cada agente escolhe um caminho com base na informação deixada por outros agentes que tenham selecionado previamente o mesmo caminho;

Um fator interessante apontado por essas características é a autocatálise provocada por feromônios que se formam no próprio sistema reativo, considerando que o aumento da probabilidade de escolha de um determinado agente está vinculada às escolhas prévias que foram feitas indicando aquele mesmo caminho (COELHO; NETO, 2004; DORIGO; MANIEZZO; COLORNI, 1991).

2.4.1 Ant System

O primeiro algoritmo ACO proposto foi o *Ant System* (AS) (DORIGO; BIRATTARI; STUTZLE, 2006; DORIGO; MANIEZZO; COLORNI, 1991; DORIGO; MANIEZZO; GAMBARDELLA, 1997). Seguindo a abordagem de Dorigo, Maniezzo e Gambardella (1997), a explanação do algoritmo será feita tomando como referência o TSP, embora o modelo possa ser usado para resolver outros problemas de otimização.

No TSP, um conjunto de cidades é dado e as distâncias entre cada uma delas é conhecida. O objetivo é fornecer o menor caminho para visitação de todas as cidades, sem repetir nenhuma delas. Modelando as cidades como nós e os caminhos como arestas de um grafo, seja, então, $b_i(t)$ o número de formigas na cidade i no instante de tempo t; n o número total de cidades; e $m = \sum_{i=1}^{n} b_i(t)$ o número total de formigas. Seja, ainda, $\tau_{ij}(t)$ a intensidade da trilha (quantidade de feromônio) na aresta (i, j) no instante t.

Considerando uma iteração do AS como um movimento realizado por cada uma das m formigas do instante t ao instante $t+1$, então, após todas as n iterações do algoritmo, fechando um ciclo, cada formiga completa uma trajetória. Nesse ponto, as intensidades das trilhas são atualizadas de acordo com a seguinte equação:

$$\tau_{ij}(t+n) = \rho * \tau_{ij}(t) + \Delta\tau_{ij} \tag{5}$$

onde: ρ é um coeficiente em que $(1-\rho)$ represente a evaporação do feromônio entre t e $t+n$;

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

em que $\Delta\tau_{ij}^{k}$ é a quantidade de feromônio deixada na aresta (i, j) pela k-ésima formiga entre t e $t+n$, dada por

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, & \text{se a } k\text{-ésima formiga usa a aresta } (i,j) \text{ no seu caminho} \\ 0, & \text{caso contrário} \end{cases}$$

sendo, ainda, Q uma constante e L_k o comprimento do caminho da k-ésima formiga, até então.

Considerando a visibilidade η_{ij} como sendo o inverso da distância entre as cidades i e j, é possível definir a função de probabilidade de transição da cidade i para cidade j pela k-ésima formiga como

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\Sigma[\tau_{ik}(t)]^{\alpha} * [\eta_{ik}]^{\beta}}, & \text{se } k \in \text{“permitidas}_k\text{”} \\ 0, & \text{caso contrário} \end{cases} \tag{6}$$

onde “permitidas$_k$” é a relação de cidades que ainda podem ser visitadas pela k-ésima formiga.

Na Equação 6, os parâmetros α e β controlam a importância relativa entre a trilha e a visibilidade. Dessa forma, a probabilidade de transição é compensada, de acordo com a intenção do projetista do algoritmo, entre a preferência de escolha de cidades mais próximas ou por caminhos onde há maior tráfego.

2.4.2 MAX-MIN Ant System

Conhecido também como MMAS e apresentado na literatura como uma melhoria do AS (DORIGO; BIRATTARI; STUTZLE, 2006; STUTZLE; HOOS, 1997, 2000), nesse algoritmo somente a formiga com o melhor resultado do ciclo atualiza a trilha de feromônios, além do valor do feromônio depositado ser obrigatório e limitado. Sua atualização de feromônio é dada por

$$\tau_{ij}(t+n) = \left[\rho * \tau_{ij}(t) + \Delta\tau_{ij}^{\text{melhor}}\right]_{\tau_{min}}^{\tau_{max}} \tag{7}$$

onde τ_{max} e τ_{min} são, respectivamente, os limites superior e inferior impostos ao feromônio; o operador $[x]_b^a$ é definido como

$$[x]_b^a = \begin{cases} a, & \text{se } x > a \\ b, & \text{se } x < b \\ x, & \text{caso contrário} \end{cases}$$

e $\Delta\tau_{ij}^{\text{melhor}}$ é

$$\Delta\tau_{ij}^{\text{melhor}} = \begin{cases} 1/L_{\text{melhor}}, & \text{se } (i,j) \text{ pertence ao melhor caminho} \\ 0, & \text{caso contrário} \end{cases}$$

com L_{melhor} sendo o comprimento do caminho da formiga com melhor resultado. Este último pode ser, inclusive, a depender da decisão do projetista do algoritmo, tanto o melhor caminho da iteração atual, o melhor desde o início até o momento, ou a combinação de ambas.

2.4.3 Ant Colony System

O *Ant Colony System* (ACS) introduz o conceito de atualização local do feromônio, além da atualização realizada no fim do processo. Essas atualizações locais são realizadas por todas as formigas depois de cada passo dado no caminho. Cada formiga atualiza, então, somente a última aresta utilizada, com base na seguinte equação:

$$\tau_{ij}(t+1) = (1-\varphi) * \tau_{ij}(t) + \varphi * \tau_0 \tag{8}$$

onde $\varphi \in (0,1]$ é o coeficiente de decaimento do feromônio e τ_0 é valor inicial do feromônio.

A atualização do feromônio ao final do processo, agora chamada de *off-line* ou global, ocorre similarmente ao MMAS, com uma ligeira diferença na equação:

$$\tau_{ij}(t+n) = \begin{cases} \rho * \tau_{ij}(t) + \rho * \Delta\tau_{ij}^{\text{melhor}}, & \text{se } (i,j) \text{ pertencer ao melhor caminho} \\ \tau_{ij}(t+n), & \text{caso contrário} \end{cases} \tag{9}$$

Outra diferença importante entre o ACS e AS está na regra de decisão usada pela formiga durante o processo. No ACS, uma regra pseudoaleatória proporcional é usada, onde a probabilidade para uma formiga se mover do nó i ao nó j depende de uma variável aleatória q uniformemente distribuída sobre $[0,1]$ e uma parâmetro de *threshold* q_0, como se segue:

$$j = \begin{cases} max\{\tau_{il}, \eta_{il}^{\beta}\}, & \text{se } q <= q_0 \\ \text{Equação 6}, & \text{caso contrário} \end{cases} \tag{10}$$

2.4.4 Considerações Finais

Tomando como base os conceitos apresentados, é possível verificar a aplicabilidade das técnicas discutidas para realização das tarefas propostas por este trabalho. A decomposição do espaço de trabalho utilizando a diagramação de Voronoi apresenta-se como uma proposta promissora, por produzir uma segmentação do ambiente de forma a viabilizar caminhos seguros entre obstáculos. Abordar essa tarefa utilizando a Triangulação de Delaunay permite obter os mesmos resultados, porém inserido no contexto de cálculos trigonométricos, o que tende e minimizar o alto custo algorítmico das comparações ponto-à-ponto demandadas pela abordagem de Voronoi e justifica, assim, sua proposta para desenvolvimento neste trabalho.

Os algoritmos de IA baseados na metodologia de Colônia de Formigas são identificados como instrumentos metodológicos capazes de viabilizarem o planejamento de rotas para sistemas robóticos. Inserem, ainda, no contexto da realização dessa tarefa, a possibilidade de otimização dos parâmetros de suas equações para adequação à cada aplicação. A proposta

é utilizar da parametrização do algoritmo inteligente para propor rotas que se adequem às demandas do projetista, como citado anteriormente. O ACS, em especial, apresenta uma evolução em relação às demais técnicas apresentadas, como a lista candidata e a atualização local de feromônio, o que motiva sua utilização para implementação deste projeto.

Os Capítulos a seguir apresentam uma proposta de implementação em *software* que possibilite o planejamento de rotas para sistemas robóticos móveis utilizando Diagrama de Voronoi e Otimização por Colônia de Formigas. Serão explicitadas a metodologia adotada, o desenvolvimento da solução sugerida, as restrições do modelo projetado e uma análise dos resultados obtidos. Para sua validação, os resultados serão comparados com o Algoritmo de Dijkstra, uma abordagem comumente utilizada para problemas que envolvem rotas por sempre encontrar o menor caminho entre dois pontos num grafo (LAFORE, 2003).

3 METODOLOGIA

Para o desenvolvimento do trabalho, o mesmo foi dividido em cinco etapas principais, tal qual apresentado na Tabela 1. Uma descrição detalhada das ações envolvidas em cada etapa será apresentada nas seções seguintes.

Tabela 1: Etapas para o Desenvolvimento do Projeto.

Etapa	Descrição	Ações
Etapa 1	Revisão bibliográfica e levantamento de requisitos	♦ Reflexão teórica sobre abordagens de decomposição do espaço de trabalho (ênfase no Diagrama de Voronoi e Triangulação de Delaunay); ♦ Análise teórica sobre otimização de planejamento de rotas (ênfase no ACS); ♦ Levantamento de Requisitos;
Etapa 2	Modelagem do sistema	♦ Modelagem do diagrama de blocos e caminho de dados do sistema proposto; ♦ Modelagem da Arquitetura de Software do Sistema;
Etapa 3	Implementação Delaunay/Voronoi	♦ Implementação e teste local da decomposição do espaço de trabalho;
Etapa 4	Implementação do ACS	♦ Implementação e teste local do ACS;
Etapa 5	Testes e análise de resultados	♦ Junção dos módulos e teste geral; ♦ Análise dos resultados obtidos;

3.1 REVISÃO BIBLIOGRÁFICA

Nessa etapa do projeto foi realizado o levantamento teórico envolvendo os principais conceitos a serem tratados no seu desenvolvimento. Essa análise cobriu um estudo sobre as principais abordagens de decomposição de células, a caracterização do Diagrama de Voronoi, definição da Triangulação de Delaunay e as características correlatas entre as duas abordagens. Foram analisados, ainda, os conceitos inerentes a otimização e planejamento inteligente de rotas, fechando um escopo sobre Inteligência Artificial de Enxames e o *Ant Colony System*.

Findado esse passo, uma análise da viabilidade do projeto e levantamento minucioso de requisitos subsidiaram o dispêndio no desenvolvimento do referido projeto.

3.2 MODELAGEM DO SISTEMA

O sistema desenvolvido conta com uma arquitetura em *software* modelada em camadas bem definidas. A Figura 7 apresenta uma diagramação da modelagem proposta, dando ênfase

em aspectos julgados de maior relevância.

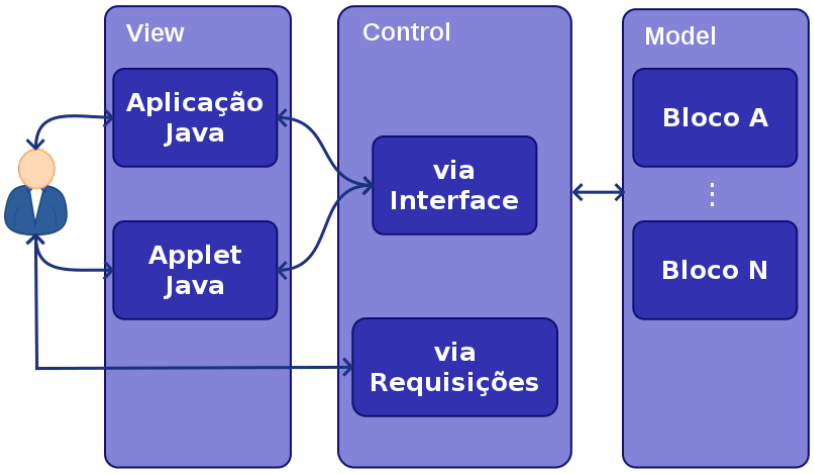


Figura 7: Arquitetura de Software do Sistema Proposto.

Nessa Figura, o ator que interage com o sistema pode ser abstraído de diferentes formas, a depender do escopo no qual o projeto está inserido. Há a possibilidade de interação utilizando uma *interface* com um usuário, seja enquanto aplicação *standalone* ou um Applet embarcado numa aplicação *Web*; ou, ainda, visualizando o projeto como um possível subsistema de um projeto maior, o ator pode passar a ser interpretado como um módulo desse macrossistema, que requisita o planejamento de uma rota fornecendo um mapa do espaço de trabalho como entrada, sem a necessidade de uma de interface gráfica.

Com o intuito de facilitar a visualização do caminho de dados no sistema proposto, o diagrama de blocos na Figura 8 apresenta o cenário no qual o ator fornece um mapa de entrada e requisita o planejamento de rotas, seja no contexto da aplicação *standalone*, quanto nos contextos do Applet ou requisição sem interface.

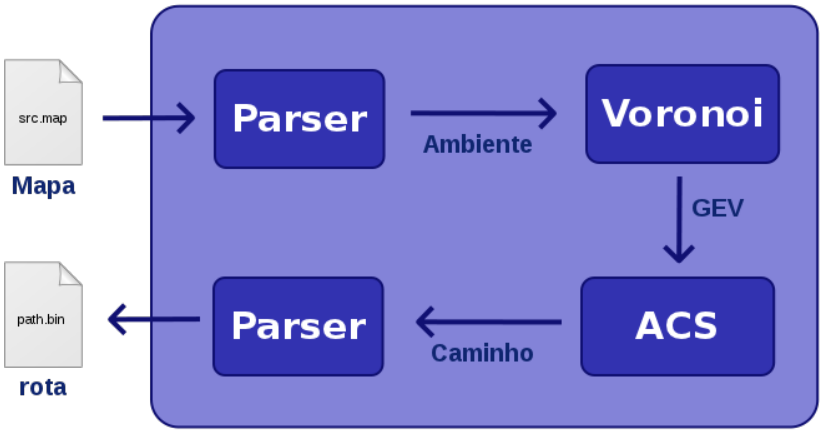


Figura 8: Diagrama de Blocos do Sistema Proposto.

Tomando como base a Figura 8, a proposta é iniciar o processo realizando um *parse*

do arquivo de entrada e fornecer as informações inerentes ao ambiente para um módulo de decomposição do espaço de trabalho. O referido módulo deve produzir um Grafo Estendido de Voronoi GEV (GAVRILOVA, 2008; KLEIN, 1989; OKABE *et al.*, 2000), para que o *Ant Colony System* possa ser aplicado. Então, após o término do processo do ACS, o caminho encontrado é submetido a um novo *parse* (agora de saída) e fornecido como a resposta do sistema à requisição inicial.

3.3 IMPLEMENTAÇÃO DELAUNAY/VORONOI

Utilizando-se das características de dualidade entre Voronoi e Delaunay, foi adotada uma abordagem de obtenção do diagrama e grafo de Voronoi empregando os resultados da Triangulação de Delaunay. Isso porque os resultados obtidos tendo como base uma análise trigonométrica do espaço de trabalho, que é o caso da triangulação, tornam-se menos onerosos computacionalmente do que aqueles adquiridos com uma abordagem bruta, de comparação ponto a ponto, obstáculo a obstáculo, que seria o caso da implementação direta do Voronoi.

Nessa etapa do desenvolvimento, foram realizadas análises, implementações e testes locais.

3.4 IMPLEMENTAÇÃO DO ACS

Para implementação e aplicação do ACS, foi utilizado como base um *framework* desenvolvido em Java, que visa fornecer ferramental para a implementação de sistemas baseado em colônias de formigas (CHIRICO, 2012). Esse *framework* propõe explorar as características de orientação a objetos para adaptar e reutilizar sua estutura, de maneira a facilitar a utilização dessa abordagem para a resolução de problemas específicos, como os já citados Problema do Caxeiro Viajante, o problema de *multicasting* numa rede ou o Problema de Steiner (BERG *et al.*, 2008).

A proposta adotada foi, então, adaptar o referido *framework* para implementar uma solução para o problema de melhor rota em *unicasting* como uma especificação do problema de *multicasting* numa rede de computadores, já tratado pelo *framework*. Por sua vez, esse problema, tem sua implementação tratada como uma especificação do TSP. Dessa forma, se modelada como um grafo, a arquitetura de rede considerada para elaboração do caminho sempre produz um grafo completo. A Figura 9 apresenta um modelo dessa arquitetura, onde cada ponto deve ser interpretado como um nó da rede e, consequentemente, do grafo, e cada aresta é um possível caminho de comunicação entre os nós da rede. No caso exemplificado, a rede conta com oito nós interconectados.

Partindo desse pressuposto, os algoritmos implementados no *framework* utilizado não

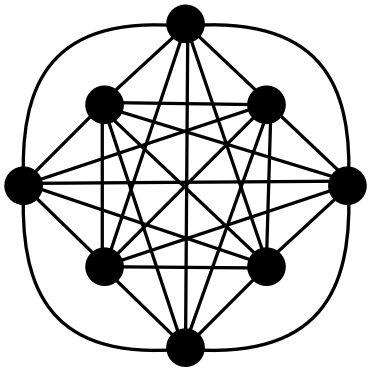


Figura 9: Representação de um grafo completo.

consideram a possibilidade de uma formiga não conseguir alcançar o nó final, pois não há caminhos sem saída.

De uma forma relativamente oposta, um Grafo Estendido de Voronoi (GEV), produzido a partir da decomposição de um espaço de trabalho, possui a característica comum de ser um grafo esparso. A Figura 10 apresenta uma representação do modelo de grafo geralmente extraído na aplicação da abordagem proposta.

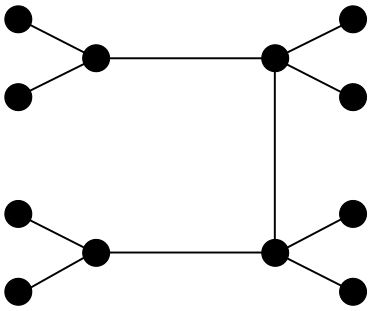


Figura 10: Representação de um grafo esparso.

Assim, como a abordagem base do *framework* tem inspiração no TSP e considerando que uma formiga não possa voltar a visitar um nó que já foi visitado anteriormente, não é difícil projetar possibilidades em que uma tomada de decisão de uma formiga, num dado nó, conduza-a para um ponto no grafo onde ela não possa sair.

Nesse sentido, trabalhar o problema de *unicasting* como especificação do problema de *multicasting* envolve alterar o algoritmo utilizado, permitindo que uma formiga possa retroceder no caminho percorrido caso alcance um nó onde não haja possibilidade de prosseguir.

Para tanto, à cada tomada de decisão quanto ao próximo nó no caminho de uma formiga, uma lista de nós visitados é alimentada para identificação do caso especial de impossibilidade de prosseguimento. Caso os únicos nós acessíveis forem nós pertencentes a essa lista, a formiga poderá temporariamente retornar por pontos já visitados até que encontre uma possibilidade de caminho não visitado até então. Para evitar que essa mesma formiga siga o mesmo caminho

novamente, uma lista de nós pertencentes a um caminho sem saída é alimentada à cada retrocesso. A Figura 11 apresenta um exemplo de aplicação dessa abordagem, onde os pontos representam os nós do grafo, dos quais, os verdes ilustram nós sendo visitados, os amarelos, já visitados anteriormente, e os vermelhos, aqueles pertencentes a caminhos sem saída.

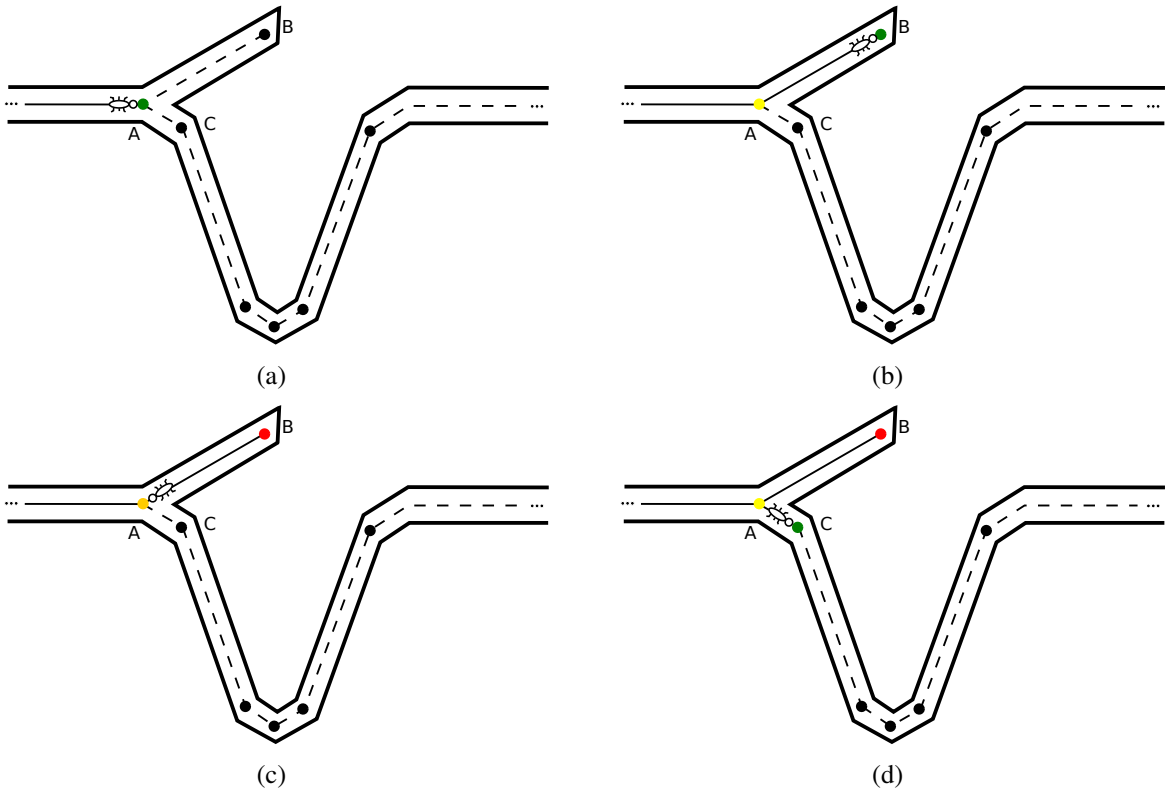


Figura 11: Exemplo de comportamento em caso de caminho sem saída.

A Figura 11a ilustra a chegada de uma formiga a um nó onde a tomada de decisão possa inferir num caminho sem saída. Ao chegar ao nó "A", a possibilidade de escolher o nó "B" como próximo nó é adotada na Figura 11b. Nesse momento, o único nó vizinho a "B" é o nó "A", que já foi visitado no passo anterior, o que caracteriza um caminho sem saída. Seguindo o algoritmo proposto, excepcionalmente por essa condição, a formiga poderá voltar para o nó "A", como na Figura 11c, para que possa tentar um caminho diferente. Como existe nova possibilidade de caminho partindo do nó "A", a formiga pode, então, escolher esse novo nó. A Figura 11d exemplifica esse passo, onde a formiga sai do nó "A" e passa para o nó "C". Se, porventura, não houvesse essa possibilidade de caminho, a formiga continuaria retrocedendo até que um nó não visitado fosse alcançado, ou então não houvesse um caminho possível.

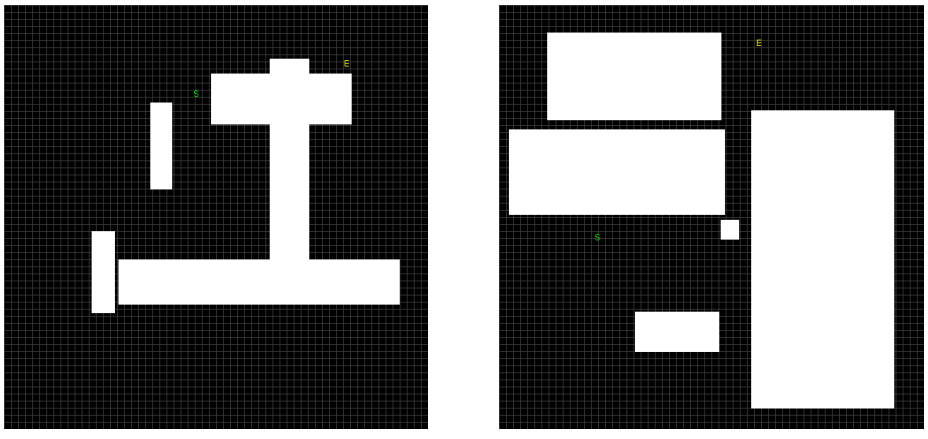
Essa abordagem sugere uma metodologia diferente do *framework* utilizado quanto ao tratamento dos nós "permitidos$_k$", citados na Equação 6, Capítulo 2 deste trabalho.

3.5 CENÁRIOS DE TESTE

Nessa etapa do projeto, os resultados obtidos localmente nas etapas anteriores conduziram para uma implementação de integração dos módulos. Para verificação e validação da implementação, bem como para gerir a análise de resultados, uma plataforma de testes foi elaborada de forma a facilitar esse processo.

Para tanto, um conjunto de regras foi elaborado com o intuito de parametrizar os modelos de testes e possibilitar análises qualitativas e quantitativas. A proposta é viabilizar a geração aleatória de mapas de entrada, considerando um conjunto de regras que venham a representar casos de testes interessantes para o sistema. Dentro de uma vasta gama de possibilidades de regras que poderiam ser utilizadas, um conjunto contendo seis regras foi considerado suficiente e adequado para a análise proposta, tendo em vista representar de forma apropriada os principais cenários cujos sistemas robóticos estão frequentemente submetidos, tais como ambientes com obstáculos de tamanhos fixos ou variados; conectados entre si ou desconexos; ou situações de corredores e portas. Esse conjunto de regras tem, então, suas especificações caracterizadas por regras de tamanho e posicionamento dos obstáculos no espaço de trabalho. As especificações adotadas são apresentas à seguir e os respectivos exemplos são ilustrados nas Figuras 12, 13 e 14. Em todos os casos, considera-se que os obstáculos encontram-se expandidos.

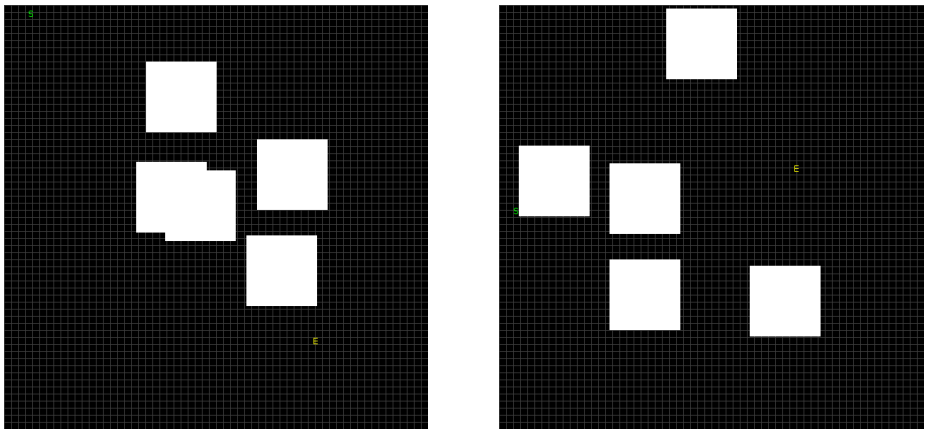
- **obstáculos sobrepostos e de tamanho dinâmico:** mapas cujos obstáculos podem se posicionar uns sobre os outros e têm seu tamanho definido em tempo de execução (Figura 12a);
- **obstáculos desconectados e de tamanho dinâmico:** mapas cujos obstáculos não podem se sobrepor e têm seu tamanho definido em tempo de execução (Figura 12b);



(a) (b)

Figura 12: Exemplos de mapas de entrada com obstáculos de tamanho dinâmico.

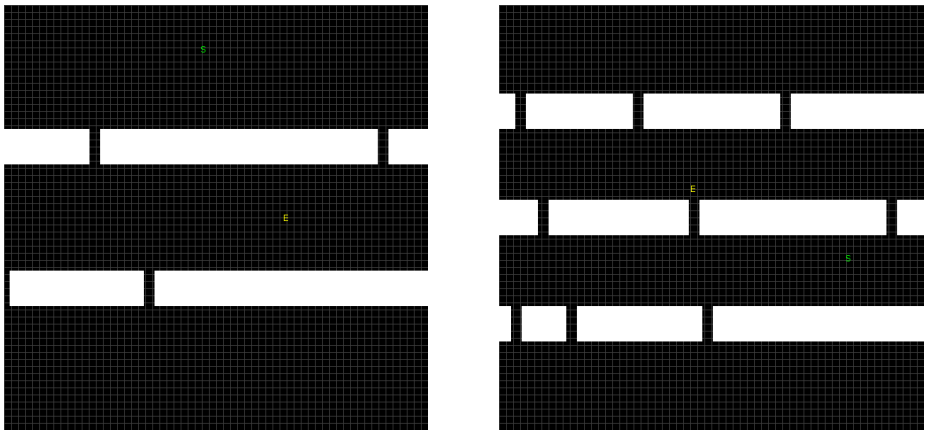
- **obstáculos sobrepostos e de tamanho fixo:** mapas cujos obstáculos podem se posicionar uns sobre os outros, porém todos terão um tamanho definido na plataforma de testes (Figura 13a);
- **obstáculos desconectados e de tamanho fixo:** mapas cujos obstáculos não podem se sobrepor, porém todos terão um tamanho fixo definido na plataforma de testes (Figura 13b);



(a) (b)

Figura 13: Exemplos de mapas de entrada com obstáculos de tamanho fixo.

- **duas paredes com dois acessos livres:** mapas cujos obstáculos formam três corredores, com duas possibilidades de acessos entre cada corredor (Figura 14a);
- **três paredes com três acessos livres:** mapas cujos obstáculos formam quatro corredores, com três possibilidades de acessos entre cada corredor (Figura 14b);



(a) (b)

Figura 14: Exemplos de mapas de entrada que modelam corredores.

4 RESULTADOS

Este Capítulo apresenta os principais resultados obtidos no projeto proposto. Para isso, as Seções seguintes estão organizadas de forma a especificar os resultados em cada um dos módulos desenvolvidos. A Seção 4.1 apresenta os resultados relacionados à *interface* de entrada do sistema proposto, tanto utilizando a aplicação/applet com entrada em ambiente gráfico, quanto utilizando uma requisição direta à camada de controle. Na Seção 4.2 são apresentados os resultados da implementação Delaunay/Voronoi, para decomposição do espaço de trabalho, utilizando como entrada o ambiente modelado na etapa anterior. A Seção 4.3 traz os resultados obtidos na etapa de planejamento de rotas com ACS. Na Seção 4.4 encontram-se os resultados quanto a *interface* de saída do sistema.

4.1 ENTRADA DO SISTEMA

A Figura 15 apresenta a tela de inicialização do sistema no caso de interação com um usuário através de uma aplicação/applet Java.

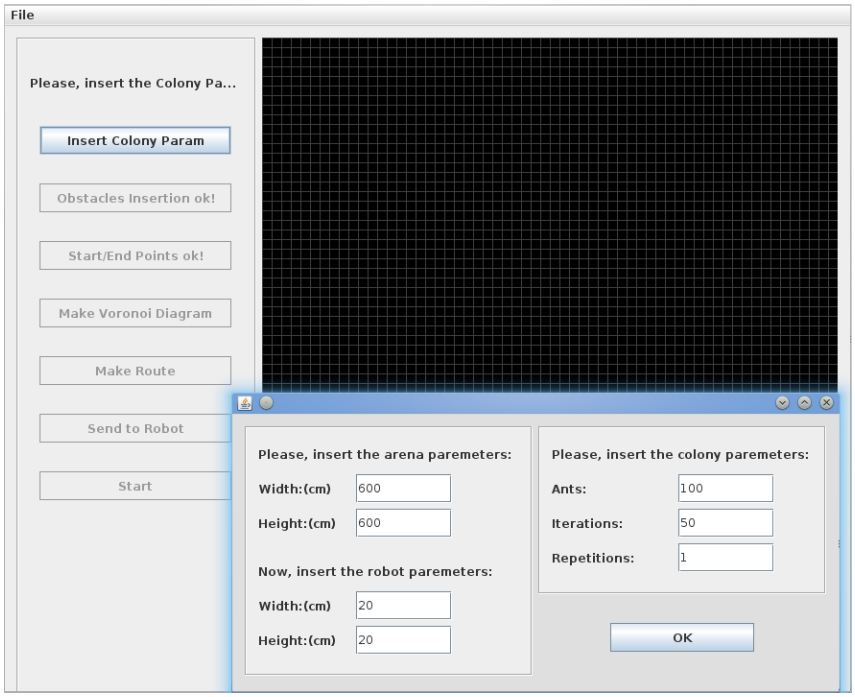


Figura 15: Interface gráfica com o usuário.

Nessa tela, o usuário pode escolher entre carregar um mapa salvo, acessando o *menu* **File**, ou prosseguir manualmente para a indicação dos parâmetros do ACS, utilizando formulário específico, e a descrição do ambiente na área reservada para exibição dos obstáculos. Após esse processo, seja utilizando o módulo *parser* do mapa indicado ou da modelagem feita pelo usuário na interface gráfica, o ambiente modelado é exibido, como na Figura 16.

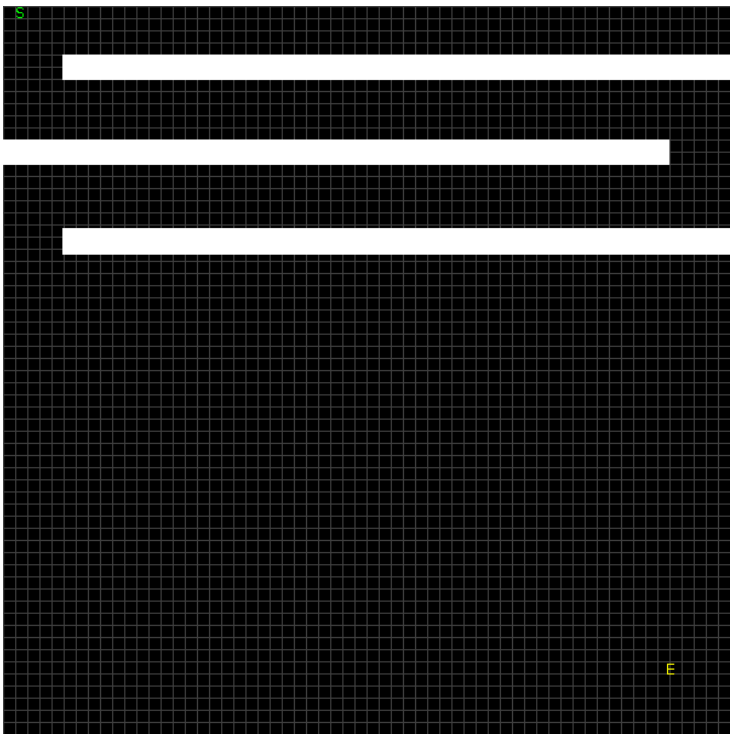


Figura 16: Exibição do ambiente modelado.

Objetivando uma melhor escalabilidade e flexibilidade do projeto, o módulo *parser* realiza a leitura de arquivos com extensão *.map*, o qual é o mesmo formato dos arquivos utilizados como entrada nos *softwares* Mobile Robot, Mapper 3, MobileSim e outros aplicativos de interação e simulação de sistemas robóticos móveis. Um exemplo de mapa de entrada é apresentado na Listagem 1.

Com o intuito de fornecer parâmetros adequados ao projeto, como número de formigas, iterações e repetições, algumas *tags* próprias foram inseridas no arquivo modelado para a aplicação proposta, porém sem comprometer seu funcionamento com os demais *softwares*. Tomando a própria Listagem 1 como base, as *tags* inseridas são os pontos inicial e final da rota a ser planejada (linhas 8 e 9 da Listagem), largura e comprimento do robô (linha 10) e as características do algoritmo do ACS a ser utilizado, como número de formigas na colônia, número de iterações de busca e de repetições do processo (linhas 12 à 15 da Listagem).

A linha 17 da Listagem é um exemplo de especificação de uma linha que compõe um obstáculo no ambiente modelado pelo mapa, a qual inicia no ponto ($x_1 = 400$, $y_1 = 50$) e termina no ponto ($x_2 = 450$, $y_2 = 50$). Dessa forma, os obstáculos são modelados enquanto retângulos formados por conjuntos de quatro linhas consecutivas no mapa de entrada.

Listagem 1: Exemplo de arquivo de mapa de entrada.

```
2D–Map
MinPos: 0 0
MaxPos: 0 0
NumPoints: 2097268
Resolution: 7602292
LineMinPos: −6219 −1720
LineMaxPos: 5040 3460
StartPoint: 10 10
EndPoint: 540 540
Robot: 20 20
NumLines: 4
COLONY
100
50
1
LINES
400 50 450 50
450 50 450 450
450 450 400 450
400 450 400 50
```

4.2 DECOMPOSIÇÃO DO ESPAÇO DE TRABALHO

A Figura 17 apresenta um exemplo do resultado da decomposição do espaço de trabalho, a qual ilustra a Triangulação de Delaunay, em linhas vermelhas, e o Grafo de Voronoi, em linhas verdes, produzido pela decomposição do mapa apresentado na Figura 16. É possível perceber que o grafo gerado contém uma quantidade de vértices de Voronoi menor que a quantidade de triângulos da Triangulação de Delaunay. Isso ocorre porque um pré-processamento é realizado nesse grafo antes de passá-lo para o módulo de planejamento de rotas, no intuito de retirar pontos desnecessários para esse processo, o que aumentaria a complexidade algorítmica envolvida nessa etapa do projeto.

4.3 PLANEJAMENTO DE ROTAS

Com o intuito de validar os resultados obtidos na etapa de planejamento de rotas utilizando o ACS, 100 mapas foram aleatoriamente gerados para cada uma das regras descritas na Seção 3.5, totalizando uma base de testes de 600 mapas. A proposta é, então, realizar uma análise comparativa entre os resultados obtidos com o ACS e com caminhos planejados utilizando o Algoritmo de Dijkstra (LAFORE, 2003), o qual é sempre capaz de encontrar o menor caminho entre dois pontos num grafo e, assim, o parâmetro ótimo para comparação com

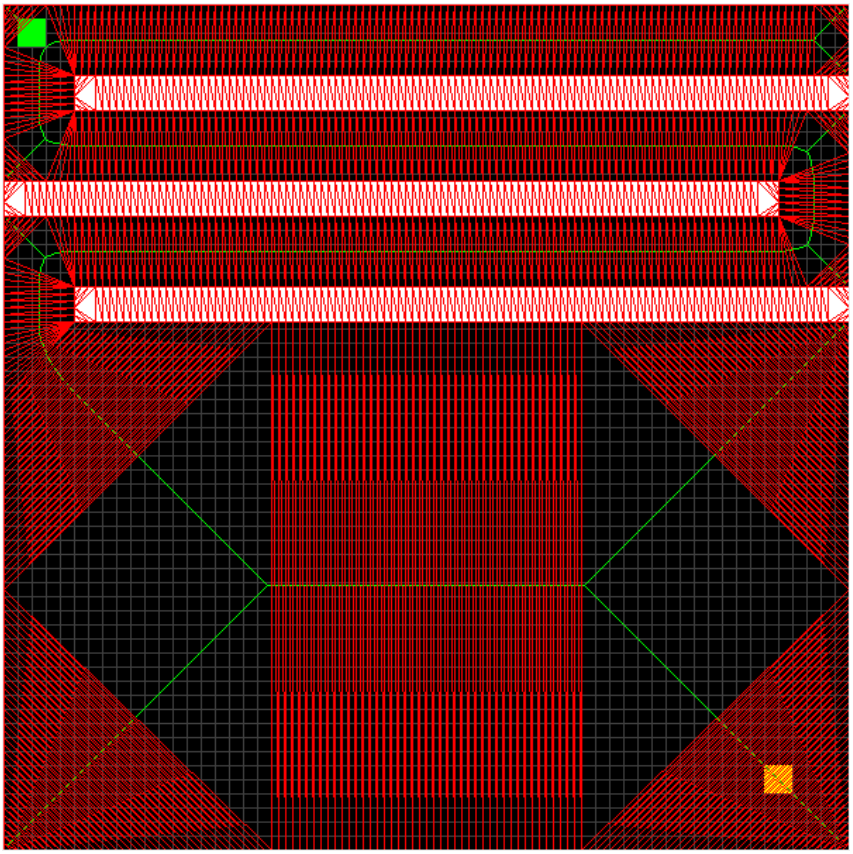


Figura 17: Exemplo de Voronoi/Delaunay produzido pelo sistema.

o Dijkstra.

Para tanto, três análises foram realizadas sob essa perspectiva: distância do caminho obtido, número de nós no grafo de melhor caminho e tempo de execução de ambas as propostas. Quatro ambientes de testes foram, então, elaborados, cujos resultados são apresentados nas subseções seguintes.

4.3.1 Ambiente de Testes 1

Nesse ambiente de testes, todos os seiscentos mapas foram observados quanto às três perspectivas de análise consideradas. O ACS foi executado utilizando os parâmetros apresentados na Tabela 2.

Tabela 2: Parâmetros do ACS para o Ambiente de Testes 1.

Parâmetro	**Valor**	**Parâmetro**	**Valor**
formigas	50	iterações	2
repetições	1	α	1
β	2	φ	0.1

A Tabela 3 apresenta um resumo dos resultados obtidos.

Tabela 3: Resultados do Ambiente de Testes 1.

	ACS		**Dijkstra**	
Análise	**mapas**	**média**	**mapas**	**média**
caminho mais curto	131	1674 $u.d.$	469	440 $u.d.$
menor número de nós no melhor caminho	495	470,6 nós	105	998 nós
menor tempo de execução	0	46730 ms	600	44,62 ms

Nessa Tabela, sob o escopo da distância dos caminhos gerados, é possível observar que os resultados obtidos com o ACS ficaram muito abaixo daqueles obtidos com o Algoritmo de Dijkstra. O fato de o ACS ter obtido um caminho mais longo em $78,2\%$ dos mapas, igualando-se ao Dijkstra somente em 131 dos 600 mapas, ainda não representa de forma clara o tamanho dessa perda. Os caminhos produzidos pelo ACS foram, em média, 280% mais longos que os caminhos do Dijkstra. No que tange o tempo de execução, os resultados com o ACS continuaram distantes dos resultados com o Dijkstra. O ACS obteve um tempo de execução mais longo em todos os seiscentos mapas, sendo, em média, 1046 vezes maior que o tempo de execução com o Dijkstra. Os resultados só se tornam melhores para o ACS quando o escopo muda para a análise de número de nós no caminho obtido, onde o ACS obtém menos nós no caminho final em $82,5\%$ dos casos de teste, tendo caminhos com uma média de $53,8\%$ menos nós que a abordagem utilizando Dijkstra.

As Figuras 18, 19 e 20 apresentam gráficos produzidos com os resultados desse Ambiente de Testes, considerando análises de distância dos caminhos obtidos, número de nós dos caminhos obtidos e tempo de execução, respectivamente. Essa análise gráfica possibilita a complementação das informações explicitadas na Tabela 3.

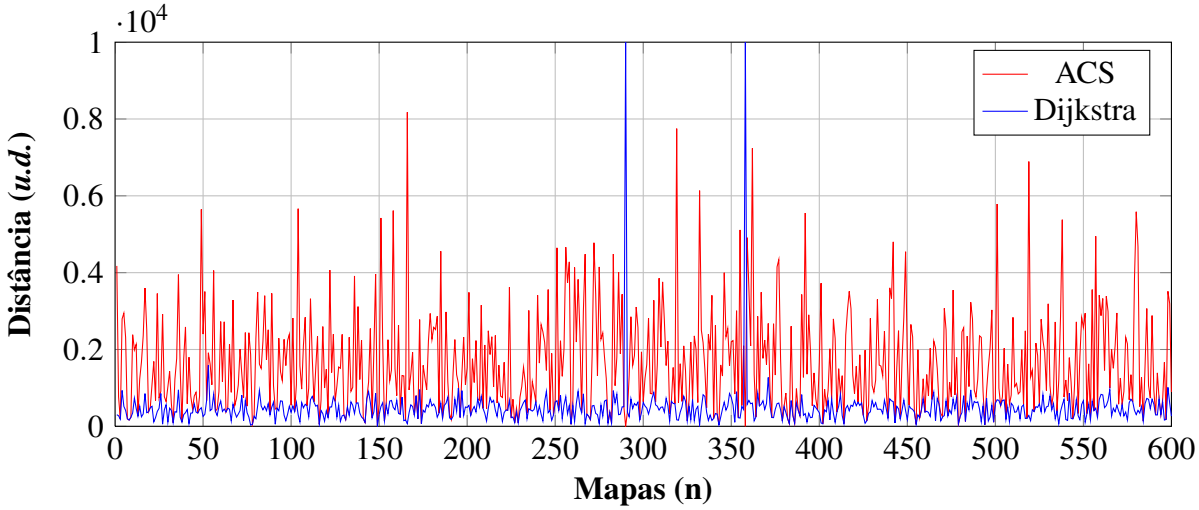


Figura 18: Gráfico comparativo das distâncias dos caminhos obtidos com o ACS e com Dijkstra.

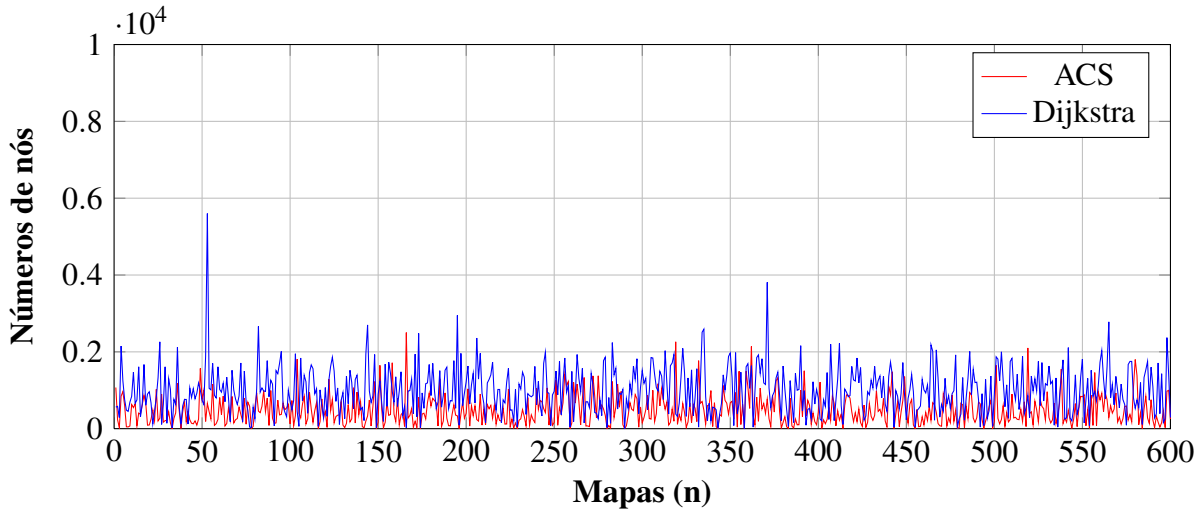


Figura 19: Gráfico comparativo do número de nós dos caminhos obtidos com o ACS e com Dijkstra.

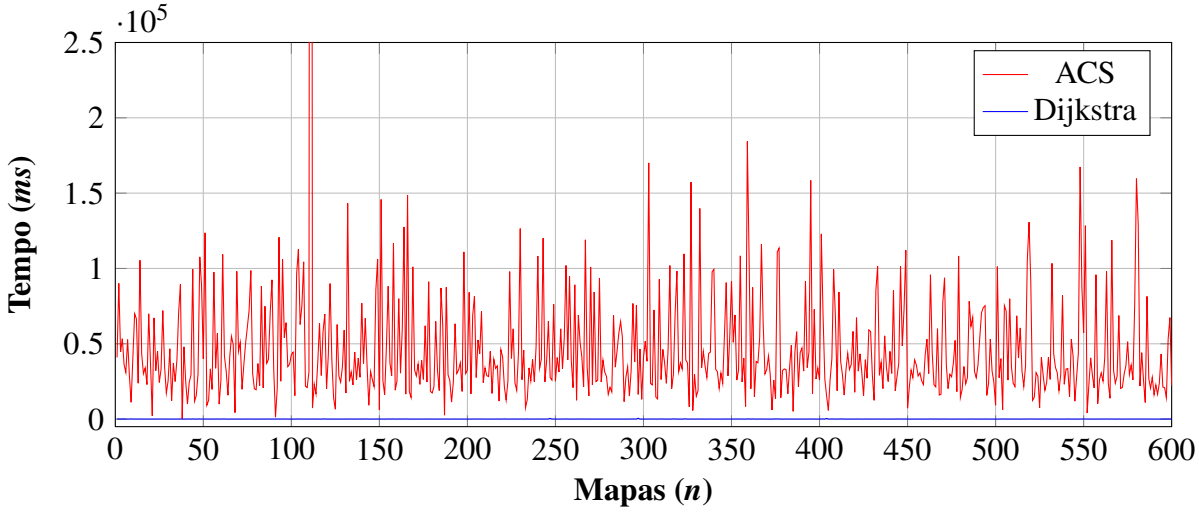


Figura 20: Gráfico comparativo do tempo para encontrar o caminho com ACS e com Dijkstra.

4.3.2 Ambiente de Testes 2

Nesse Ambiente de Testes, os mesmos parâmetros do ACS apresentados na Tabela 2 foram aplicados. Novos testes foram realizados sobre os mesmos seiscentos mapas, analisando as mesmas três perspectivas, alterando no ACS a visibilidade da lista de nós proibidos, alimentada pelas formigas à cada retrocesso num caminho sem saída. Essa lista, que outrora era visível apenas pela formiga que a alimentava, agora passa a ser uma única lista para todas as formigas da colônia.

A Tabela 4 apresenta um resumo dos resultados obtidos, indicando em quantos dos mapas testados cada abordagem obteve um melhor resultado, além do valor médio obtido em cada um dos parâmetros analisados.

Tabela 4: Resultados do Ambiente de Testes 2.

	ACS		**Dijkstra**	
Análise	**mapas**	**média**	**mapas**	**média**
caminho mais curto	179	1034,8 *u.d.*	421	440 *u.d.*
menor número de nós no melhor caminho	540	227 nós	60	998 nós
menor tempo de execução	1	4740 *ms*	599	44,62 *ms*

Na Tabela 4, voltando a considerar Dijkstra como parâmetro de comparação, os resultados quanto às distâncias dos caminhos gerados foram, em média, 135% mais longos, fazendo-o obter resultados inferiores em $70,2\%$ dos mapas testados. Em se tratando do tempo de execução, a abordagem utilizada no ACS obtém seu caminho de forma mais lenta que Dijkstra em $99,8\%$ dos casos, tendo um tempo médio 105 vezes maior. Novamente, quando o resultado é analisado sob o escopo do número de nós no caminho gerado, o ACS obtém melhores números que Dijkstra, agora em 90% dos casos, obtendo um caminho com $77,25\%$ menos nós, em média.

As Figuras 21, 22 e 23 apresentam gráficos produzidos com os resultados desse Ambiente de Testes, considerando análises de distância dos caminhos obtidos, número de nós dos caminhos obtidos e tempo de execução, respectivamente.

4.3.3 Ambiente de Testes 3

Esse Ambiente de Testes tem o objetivo de subsidiar uma análise quanto aos impactos da variação da quantidade de formigas no resultado final do planejamento de rotas. A proposta é colher as informações necessárias para viabilizar essa análise, considerando seis condições de variação da quantidade formigas:

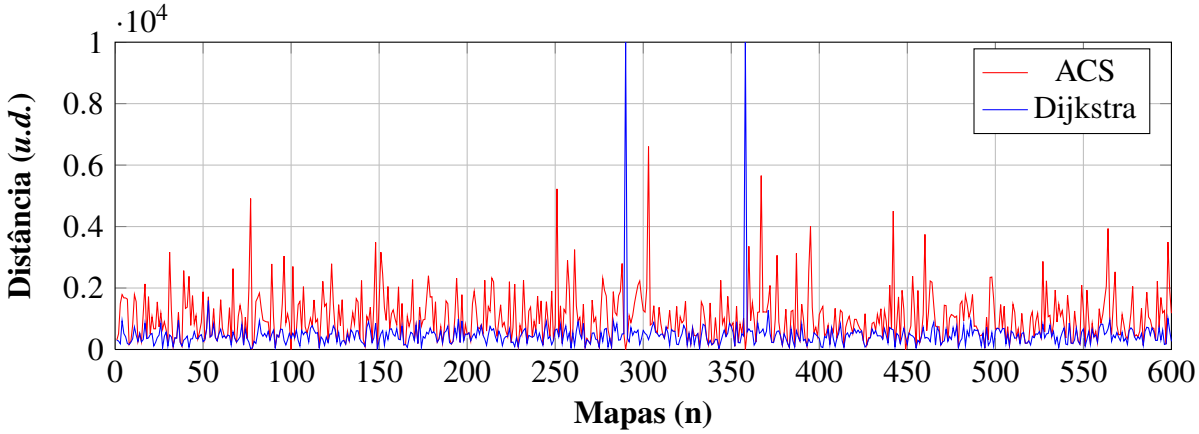


Figura 21: Gráfico comparativo das distâncias dos caminhos obtidos com o ACS e com Dijkstra.

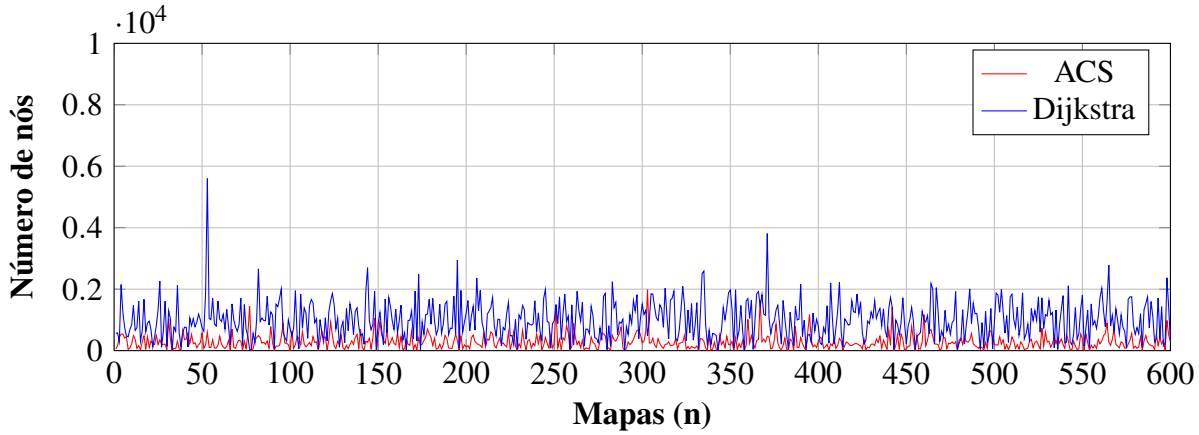


Figura 22: Gráfico comparativo do número de nós dos caminhos obtidos com o ACS e com Dijkstra.

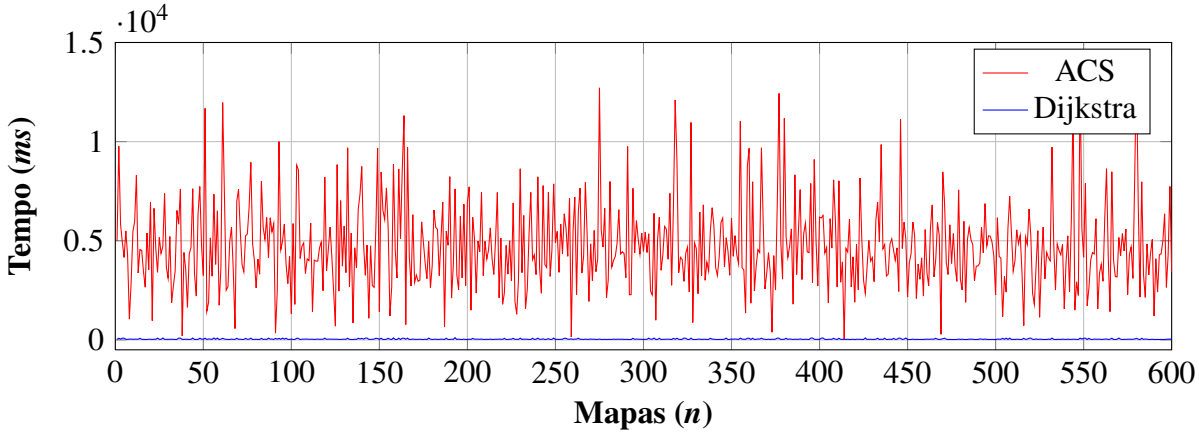


Figura 23: Gráfico comparativo do tempo para encontrar o caminho com ACS e com Dijkstra.

- **teste 1:** 5 formigas
- **teste 2:** 10 formigas
- **teste 3:** 20 formigas

- **teste 4:** 50 formigas
- **teste 5:** 100 formigas
- **teste 6:** 200 formigas

A Figura 25 apresenta um gráfico produzido com os resultados desse Ambiente de Testes, considerando uma análise das médias de distância dos caminhos obtidos, número de nós dos caminhos obtidos e tempo de execução, respectivamente.

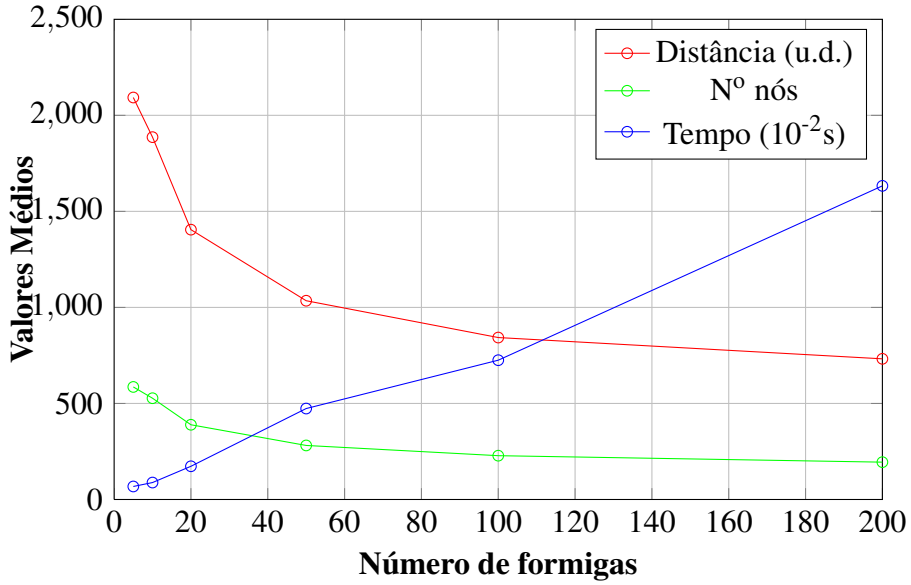


Figura 24: Gráfico comparativo das médias de distância, número de nós e tempo de execução com variação do número de formigas.

Com base nessa Figura, verifica-se para os casos testados que o aumento do número de formigas deve ser considerado até uma quantidade próxima a 100 formigas. Isso ocorre porque, enquanto as médias de distância e número de nós do caminho obtido comportam-se como uma exponencial decrescente e tendem a estabilizar depois desse valor, o tempo de execução tende a continuar crescendo linearmente.

4.3.4 Ambiente de Testes 4

Nesse Ambiente de Testes, o objetivo é viabilizar análises quanto aos impactos da variação de iterações no resultado final do planejamento de rotas. Assim como no Ambiente de Testes 3, a proposta é colher as informações necessárias para viabilizar essa análise, mantendo a mesma parametrização da Tabela 2, considerando seis condições de variação da quantidade de iterações:

- **teste 1:** 5 iterações
- **teste 2:** 10 iterações

- **teste 3:** 20 iterações
- **teste 4:** 50 iterações
- **teste 5:** 100 iterações
- **teste 6:** 200 iterações

A Figura 25 apresenta um gráfico produzido com os resultados desse Ambiente de Testes, considerando uma análise das médias de distância dos caminhos obtidos, número de nós dos caminhos obtidos e tempo de execução, respectivamente.

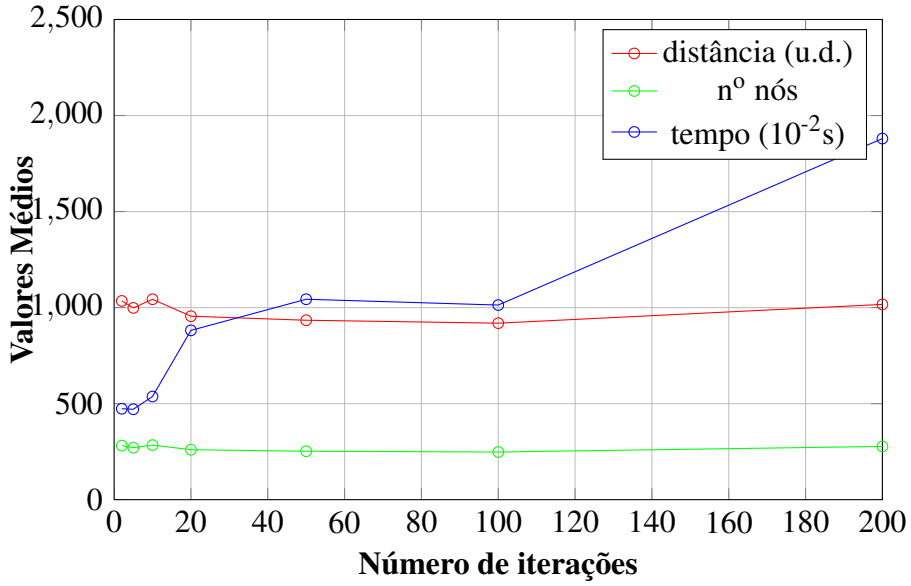


Figura 25: Gráfico comparativo das médias de distância, número de nós e tempo de execução com variação do Número de Iterações.

Analisando essa Figura, é possível perceber que os valores médios de distância e quantidade de nós dos caminhos obtidos tendem a não sofrerem alterações significativas com o aumento do número de iteações, ao passo que tempo médio de execução é quase quatro vezes maior.

4.4 SAÍDA DO SISTEMA

A Figura 26 apresenta uma tela de exemplo de exibição do caminho obtido, no caso de interação com um usuário através de uma aplicação/applet Java. Nessa Figura, os caminhos em verde indicam os possíveis caminhos, gerados pela abordagem Voronoi/Delaunay, e o caminho em vermelho indica o resultado do processamento do ACS.

Independente do uso da aplicação através dessa interface gráfica, um arquivo *.path* é gerado como uma saída do sistema. Um exemplo de mapa de saída é apresentado na Listagem 2.

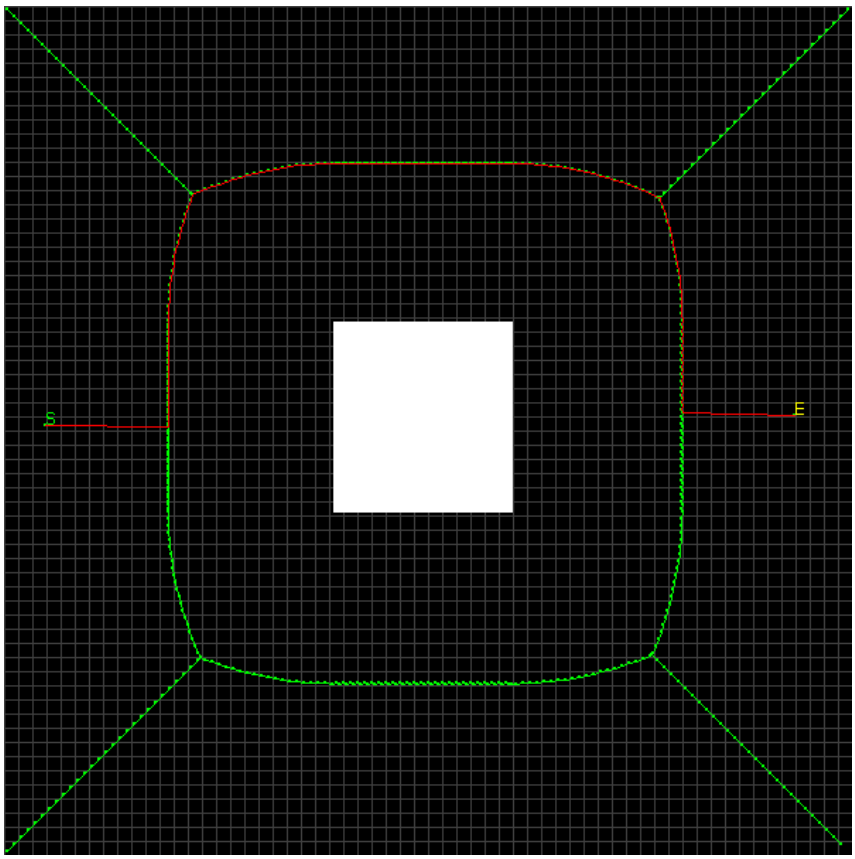


Figura 26: Exibição do caminho obtido com o ACS.

Listagem 2: Exemplo de arquivo de mapa de saída.

```
1 numPoints: 6
2 25.0 283.0
3 25.0 282.5
4 25.0 277.5
5 25.0 272.5
6 25.0 267.5
7 25.0 262.5
```

Nessa Listagem, a linha 1 indica a quantidade de nós no caminho resultante e as demais linhas indicam as coordenadas *x* e *y* desses pontos no espaço de trabalho.

5 CONCLUSÕES

O presente projeto é fruto da disciplina de Introdução à Robótica, na qual foi proposto um sistema de planejamento de rotas utilizando uma abordagem com Quadtree para decomposição do espaço de trabalho (BERG *et al.*, 2008; LATOMBE, 2004; LAUMOND, 1998) e uma primeira adaptação do ACS para otimização do caminho obtido. Encontra-se, ainda, inserido no contexto de um projeto maior, intitulado Colônia de Robôs, cujo escopo é proporcionar uma experiência colaborativa com o uso dos mais variados sistemas robóticos para a obtenção de um objetivo comum. A proposta é explorar as características intrínsecas de cada sistema para que, autônoma e colaborativamente, uma colônia de robôs possa cumprir uma determinada meta.

Dessa forma, o projeto apresentado neste trabalho pode ser apontado como uma solução viável para a tarefa de planejamento de rotas, seja num escopo local, como aplicação *standalone*, ou escopos globais, como o caso do Colônia de Robôs.

Sua flexibilidade quanto à *interface* de entrada proporciona uma experiência diferenciada no que se refere à facilidade de interação do sistema com seus possíveis atores. Há a possibilidade de inseri-lo num contexto de uma aplicação *web*, funcionando enquanto Applet Java num navegador, ou, ainda, executá-lo localmente. Em ambos os casos, é possível optar por não utilizar uma *interface* gráfica, o que viabiliza uma interação direta com outras aplicações de forma mais simples. Outro fator de destaque quanto ao modelo de entrada do sistema foi a escolha dos arquivos de entrada de forma padronizada com a extensão *.map*, compatível com outras diversas aplicações voltadas para simulação e interação com sistemas robóticos.

A aplicação de uma abordagem baseada na Diagramação de Voronoi mostrou-se tanto eficiente quanto eficaz para o problema proposto. Explorar a dualidade Voronoi/Delaunay também pode ser apresentada como uma abordagem de ataque interessante para o tipo de problema enfrentado nesse projeto.

O uso do algoritmo de Colônia de Formigas apresentou resultados promissores para aplicação proposta. A possibilidade de otimizar os resultados para atender a demandas específicas de cada projeto fica explícita desde sua modelagem e equacionamento. É possível, por exemplo, propor soluções inteligentes com características específicas para cada aplicação. Ajustes nos parâmetros do próprio equacionamento, como o α ou o β (Equação 6, Capítulo 2), podem proporcionar resultados onde a quantidade de feromônio depositado numa aresta tem mais relevância do que a distância entre os nós, e vice-versa. Outros parâmetros podem, também, ser refinados para obtenção de resultados que se adequem melhor para o objetivo proposto, como variar a quantidade de formigas, iterações e/ou repetições do algoritmo.

A proposta de análise dos resultados em comparação com o algoritmo de Dijkstra foi

outro fator positivo da abordagem sugerida. Apesar dos resultados com o ACS parecerem momentânea e inicialmente inferiores aos obtidos com o Dijkstra, uma análise mais exaustiva pôde confirmar o potencial dessa abordagem e apresentar aspectos promissores. Um destaque é que, embora os caminhos obtidos com o ACS sejam mais longos que os obtidos com Dijkstra, possuem uma quantidade muito reduzida de nós, o que pode representar uma vantagem em se tratando de um sistema robótico, visto que muitos nós implicam em muitas curvas. Optar por caminhos com menos nós, e consequentemente com menos curvas, pode representar uma maior autonomia energética na tarefa de movimentação no espaço de trabalho.

O modelo de arquivo de saída também é outro fator positivo na arquitetura proposta, pois atende a todas as demandas de *interface* com o usuário suportadas pela aplicação. Devido a uma modelagem simples, pode, ainda, ser facilmente interpretado por quaisquer outras aplicações que as queiram utilizar.

De uma forma geral, a principal contribuição do trabalho proposto encontra-se na modificação sugerida ao *Ant Colony System*, de forma a adicionar uma nova informação ao processo de tomada de decisão: a lista de nós que conduzem a caminhos sem saída. Por se tratar de um dado importante desde a especificação do problema, o uso dessa informação, mesmo que apenas localmente, já apresentou resultados que justificam sua inserção nesse contexto. Quando considerados os avanços alcançados com seu compartilhamento com as demais formigas da colônia, tratando-o como uma informação global, fica mais evidente a importância dessa contribuição no cenário da aplicação.

Ainda assim, diversas dificuldades encontradas no processo de desenvolvimento devem ser pontuadas. Uma delas seria a adequação dos resultados obtidos com a Triangulação de Delaunay para obtenção do Grafo Estendido de Voronoi. A abordagem de Delaunay trata, intrinsecamente, de cenários onde os vértices de Voronoi são meramente pontuais, ou seja, no contexto da aplicação, o obstáculo deve ser modelado como um ponto, e não como um objeto bidimensional. Para alcançar os resultados apresentados, foi necessário modelar o obstáculo não como uma forma geométrica em si, mas, sim, como um conjunto de pontos sobre sua região limítrofe (ou área mais externa). Esse fator, apesar de resolver o problema citado, acaba inserindo pontos desnecessários ao grafo e, consequentemente, aumentando a complexidade algorítmica do ACS.

A característica de aleatoriedade e probabilidade do Colônia de Formigas proporcionou outro ponto crítico de desenvolvimento. Devido a essas características, por muitas vezes, o caminho resultante contém circuitos fechados. A proposta de algoritmo com uso compartilhado da lista ainda consegue dirimir esse problema, mas não é o suficiente para eliminá-lo. Esse fator também reflete na obtenção de caminhos resultantes mais longos.

Nesse sentido, uma série de trabalhos futuros pode ser proposta, de forma a otimizar os resultados obtidos e dirimir características que conduziram a dificuldades de desenvolvimento:

- otimizar os resultados na decomposição do espaço de trabalho, diminuindo a quantidade de pontos gerados pela abordagem Delaunay/Voronoi. Há ainda a possibilidade de remover nós desnecessários para o ACS, num pré-processamento posterior ao Voronoi;
- otimizar os resultados do ACS, removendo circuitos fechados existentes no caminho final;
- modelar o espaço de trabalho utilizando um conjunto mais variado de formas geométricas, sabendo que o uso restrito de formas retangulares não é suficiente para modelar todas as possibilidades de um espaço de trabalho no mundo real;
- prosseguir com uma análise de impacto e influência dos demais parâmetros do ACS, identificando como variações nos números de iterações e repetições, além de dos valores de α e β na lógica de transição de nós, alteram os resultados obtidos;
- realizar uma análise *trade-off* com outras abordagens de decomposição do espaço de trabalho, utilizando tanto abordagens com divisões de células inexatas, como o já citado quadtree, como outras abordagens de divisão de células exatas;
- realizar uma análise *trade-off* com outas abordagens de IA que possam se adequar ao problema proposto, como Algoritmos Genéticos;
- validar a implementação do projeto, embarcando os resultados obtidos em sistemas robóticos móveis, seja numa plataforma mais simples, como o NXT MindStorm, da Lego, ou em plataformas mais robustas, como o robô móvel Pioneer, da MobileRobots;
- integrar a versão final da aplicação no contexto do projeto da Colônia de Robôs.

REFERÊNCIAS

BARCLAY, M.; GALTON, A. **Comparison of region approximation techniques based on Delaunay Triangulations and Voronoi Diagrams**. 2009. Disponível em: <http://ncg.nuim.ie/gisruk/materials/proceedings/PDF/7A2.pdf>. Acesso em: 04 Ago. de 2012.

BECKERS, R.; DENEUBOURG, J. L.; GOSS, S. **Trails and U-turns in the selection of the shortestpath by the ant Lasius Niger**. 1992. Disponível em: <http://www.ulb.ac.be/sciences/use/publications/JLD/90.pdf>. Acesso em: 04 Ago. de 2012.

BERG, M. de *et al.* **Computational Geometry: Algorithms and Applications**. 3. ed. Nova York: Springer-Verlag Berlin Heidelberg, 2008.

BONABEAU, E.; DORIGO, M.; THERAULAZ, G. **Swarm Intelligence: from natural to artificial systems**. 1. ed. New York: Oxforf University Press, 1999.

CHIRICO, U. **A Java Framework for Ant Colony Systems**. 2012. Disponível em: <http://www.ugosweb.com/Download/JavaACSFramework.pdf>. Acesso em: 20 Nov. de 2012.

COELHO, L. dos S.; NETO, R. F. T. **Colônia de Formigas: Uma Abordagem Promissora para Aplicações de Atribuição Quadrática e Projeto de Layout**. 2004. Disponível em: <http://www.dep.ufscar.br/admin/upload/ARTIGO_1156534584.PDF>. Acesso em: 07 Ago. de 2012.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. **Distributed Optimization by Ant Colonies**. 1991. Disponível em: <ftp://iridia.ulb.ac.be/pub/mdorigo/conferences/IC.06-ECAL92.pdf>. Acesso em: 04 Ago. de 2012.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. **An investigation of some properties of an ant algorithm**. 1991. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.5444>. Acesso em: 04 Ago. de 2012.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. **Ant Colony Optimization: artificial ants as a computational intelligence technique**. 1. ed. New York: IEEE Computational Intelligence Magazine, 2006.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. **Positive feedback as a search strategy**. 1991. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.6342&rep=rep1&type=pdf>. Acesso em: 07 Ago. de 2012.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. **The Ant System:Optimization by a colony of cooperating agents**. 1996. Disponível em: <ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.pdf>. Acesso em: 07 Ago. de 2012.

DORIGO, M.; MANIEZZO, V.; GAMBARDELLA, L. M. **Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem**. 1997. Disponível em: <http://www.idsia.ch/ luca/acs-ec97.pdf>. Acesso em: 04 Ago. de 2012.

GAVRILOVA, M. L. **Generalized Voronoi Diagram: A Gerometry-Based Approach to Computational Intelligence**. 1. ed. Calgary: Springer-Verlag Berlin Heidelberg, 2008.

GOLD, C. M. **Advantages of the Voronoi Spatial Model**. 2009. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=3D10.1.1.88.6639>. Acesso em: 04 Ago. de 2012.

GOSS, S. *et al.* **Self-organized Shortcuts in the Argentine Ant**. 1989. Disponível em: <http://www.ulb.ac.be/sciences/use/publications/JLD/56.pdf>. Acesso em: 04 Ago. de 2012.

HUMPHREY, M. C. **Efficient Computation of Voronoi Diagrams**. 2009. Disponível em: <http://eprints.cs.vt.edu/archive/00000092/01/TR-88-07.pdf>. Acesso em: 04 Ago. de 2012.

KLEIN, R. **Concrete and Abstract Voronoi Diagrams**. 1. ed. Nova York: Springer-Verlag Berlin Heidelberg, 1989.

LAFORE, R. **Data structures & algorithms in Java**. 1. ed. Indianapolis, Ind.: Sams: Indianapolis, Ind.: Sams, 2003.

LATOMBE, J.-C. **Robot Motion Planning**. 8. ed. Norwell: Kluwer Academic Publishers, 2004.

LAUMOND, J.-P. **Robot Motion Planning and Control**. 1. ed. Nova York: Springer-Verlag Berlin Heidelberg, 1998.

MANIEZZO, M.; COLORNI, A.; M, D. **The ant system applied to the quadratic assignment problem,**. 1994. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.7683>. Acesso em: 04 Ago. de 2012.

OKABE, A. *et al.* **Spatial Tessellations: Concepts and Applications of Voronoi Diagrams**. 2. ed. Calgary: John Wiley and Sons LTD, 2000.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. 1. ed. Rio de Janeiro: Elsevier Editora Ltda., 2004.

SERAPIAO, A. B. de S. **Fundamentos de Otimização por Inteligência de Enxames: Uma Visão Geral**. 2009. Disponível em: <http://www.scielo.br/pdf/ca/v20n3/02.pdf>. Acesso em: 04 Ago. de 2012.

STUTZLE, T.; HOOS, H. H. **MAX-MIN Ant System and Local Search for the Traveling Salesman Problem**. 1997. Disponível em: <http://www.gta.ufrj.br/ensino/cpe717-2011/stutzle97-icec.pdf>. Acesso em: 07 Ago. de 2012.

STUTZLE, T.; HOOS, H. H. **MAX-MIN Ant System**. 2000. Disponível em: <http://os.cloudme.com/v1/webshares/12885457505/CloudComputing/CloudAcesso em: 07 Ago. de 2012.

WHITBY, B. **Artificial Intelligence**. 1. ed. Nova York: The Rosen Publishing Group, Inc., 2009.