ROTEAMENTO DINÂMICO DE VEÍCULOS NÃO-HOLONÔMICOS PARA VISITA A REGIÕES

DOUGLAS GUIMARÃES MACHARET

ROTEAMENTO DINÂMICO DE VEÍCULOS NÃO-HOLONÔMICOS PARA VISITA A REGIÕES

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: MARIO FERNANDO MONTENEGRO CAMPOS

Belo Horizonte

Março de 2013

Macharet, Douglas Guimarães.

M149r Roteamento Dinâmico de Veículos Não-Holonômicos para Visita a Regiões / Douglas Guimarães Macharet. — Belo Horizonte, 2013.
xxxii, 144 f. : il. ; 29cm.

Tese (doutorado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação.

Orientador: Mario Fernando Montenegro Campos.

1. Computação - Teses. 2. Robótica - Teses. I. Orientador. II. Título.

CDU 519.6*82.9(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Roteamento dinâmico de veículos não-holonômicos para visita a regiões

DOUGLAS GUIMARAES MACHARET

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. MARIO FERNANDO MONTENEGRO CAMPOS - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. ANNA HELENA REALI COSTA
Departamento de Engenharia de Computação e Sistemas Digitais - USP

PROF. EDSON PRESTES E SILVA JÚNIOR
Departamento de Informática Teórica - UFRGS

PROF. GUILHERME AUGUSTO SILVA PEREIRA
Departamento de Engenharia Elétrica - UFMG

PROF. LUIZ CHAIMOWICZ
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 15 de março de 2013.

Dedico este trabalho aos meus pais, Roberto e Elisabete.

Agradecimentos

Primeiramente agradeço a Deus por essa grande oportunidade, e por ter me dado força e sabedoria para vencer mais essa etapa.

Agradeço ao meus pais, Roberto e Elisabete, e meus avós, Ary, Benildes e Edná, por sempre me apoiarem em todos momentos da minha vida e pelas palavras de incentivo, sem vocês nada disso seria possível. Agradeço também a Keyla e Timóteo por me ajudarem sempre que foi necessário.

Ao meu orientador, professor Mario F. M. Campos, pela paciência e por todos os ensinamentos. Todo o aprendizado obtido durante o processo foi de imensurável valor para minha formação acadêmica e profissional.

Aos membros da banca, professores Anna Helena Reali Costa (PCS/USP), Edson Prestes e Silva Júnior (INF/UFRGS), Guilherme Augusto Silva Pereira (DEE/UFMG) e Luiz Chaimowicz (DCC/UFMG), e também ao professor Geovany Araújo Borges (ENE/UnB) pela participação na banca de qualificação. Os comentários e sugestões levantados foram fundamentais para a consolidação e melhoria do trabalho.

Aos colegas do VeRLab, pelos momentos de descontração e importantes trocas de informações, além de sempre estarem dispostos a ajudar sempre quando precisei. Em especial Armando, grande colega e parceiro de publicações, sempre aberto a ouvir as lamentações frequentes em um processo desgastante como o desenvolvimento de uma tese, e Vilar, pela colaboração na escrita dos artigos relacionados à esse trabalho.

Aos demais colegas do DCC pelo companheirismo e pela troca de experiências e ideias, ao Synergia pela oportunidade e auxílio que permitiu que esse trabalho pudesse ser realizado.

A todos que de alguma forma participaram desse processo, muito obrigado.

Soli Deo gloria.

Resumo

Com o avanço no desenvolvimento e utilização de veículos autônomos, faz-se necessário a investigação de técnicas capazes de gerar soluções eficientes para os diversos desafios relacionados à utilização desses veículos. Dentre eles, a utilização de veículos com restrições de movimentos (não-holonômicos), a visita a posições não específicas (regiões), regiões de interesse determinadas dinamicamente ao longo do tempo e a utilização de múltiplos veículos.

Este trabalho aborda o problema do planejamento de caminhos eficientes para um grupo de veículos com restrições de movimentos cujo objetivo é visitar regiões de interesse determinadas dinamicamente no ambiente. Neste trabalho, assume-se que os veículos são homogêneos e se deslocam com velocidade constante. Dessa forma, o objetivo final consiste em minimizar o tempo total para se visitar todas as regiões.

A primeira parte da tese aborda o problema considerando o caso onde apenas um veículo é utilizado. Com base em determinadas características observadas nos melhores caminhos obtidos a partir de um algoritmo evolutivo proposto, é apresentada uma heurística determinística. Ao final, é discutida a estratégia *online* utilizada para tratar a inserção dinâmica de novas regiões ao caminho.

Em seguida, é apresentada a metodologia proposta para o caso com múltiplos veículos. Para o caso estático, são apresentados dois métodos, o primeiro consiste na utilização de duas técnicas já existentes na literatura, e o segundo é baseado em um algoritmo memético. Finalmente, todos os métodos discutidos são agregados a um algoritmo de leilão descentralizado.

Palavras-chave: Robótica Móvel, Veículos Não-Holonômicos, Problema de Roteamento Dinâmico de Veículos.

Abstract

With the advance in the development and use of autonomous vehicles, it is necessary the development of techniques capable of generating efficient solutions to the many challenges related to the use of such vehicles. Among them, we can mention the use of vehicles with constrained motion (nonholonomic), visit to non-specific positions (regions), regions of interest determined dynamically over time and the use of multiple vehicles.

This work addresses the problem of efficient planning paths for a group of vehicles with movement restrictions, whose goal is to visit certain regions of interest dynamically determined in the environment. It is assumed that all vehicles are homogeneous and move with constant velocity. Therefore, the ultimate goal is to minimize the total time necessary to visit all regions.

The first part addresses the problem by considering the case where only one vehicle is used. Based on certain characteristics observed in the best paths obtained from a proposed evolutionary algorithm, a deterministic heuristic is presented. Finally, we discuss the online strategy used to treat dynamic insertion of new regions to the path.

Then, the proposed methodology is presented for the case with multiple vehicles. For the static case two methods are shown, the first one comprised the use of two existing techniques from the literature, and the second one is based on a memetic algorithm. Finally, all methods discussed are aggregated to a decentralized auction algorithm.

Keywords: Mobile Robotics, Nonholonomic Vehicles, Dynamic Vehicle Routing Problem.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Acrônimos

AA	*Alternating Algorithm* (Algoritmo Alternante)
AAOG	Algoritmo Alternante Ótimo Generalizado
ACO	*Ant Colony Optimization* (Otimização por Colônia de Formigas)
ATSP	*Asymmetric TSP* (TSP Assimétrico)
DDTSPN	*Dynamic Dubins TSP with Neighborhoods* (TSP Dubins com Vizinhanças Dinâmico)
DTRP	*Dynamic Traveling Repairman Problem* (Problema do Restaurador Viajante Dinâmico)
DTSP	*Dubins TSP* (TSP Dubins)
DTSPN	*Dubins TSP with Neighborhoods* (TSP Dubins com Vizinhanças)
EA	*Evolutionary Algorithm* (Algoritmo Evolutivo)
EDAs	*Estimation of Distribution Algorithms* (Algoritmos de Estimação de Distribuição)
EGTSP	*Euclidean Group TSP* (TSP para Grupo Euclidiano)
ETSP	*Euclidean TSP* (TSP Euclidiano)
GA	*Genetic Algorithm* (Algoritmo Genético)
k-DDTSPN	*k-Dynamic Dubins TSP with Neighborhoods* (*k*-TSP Dubins com Vizinhanças Dinâmico)
k-DTSPN	*k-Dubins TSP with Neighborhoods* (*k*-TSP Dubins com Vizinhanças)
k-TSP	*k*-Traveling Salesman Problem
k-TSPN	*k-TSP with Neighborhoods* (*k*-TSP com Vizinhanças)
MA	*Memetic Algorithm* (Algoritmo Memético)
mDTRP	*m-Vehicle DTRP* (DTRP Multi-Veículo)
mTSP	*Multiple TSP* (TSP Múltiplo)
OA	Orientações Aleatórias
PFIH	*Push-forward Insertion Heuristic* (Heurística de Inserção Adiante)
PTAA	*Polynomial-Time Approximation Algorithm* (Algoritmo de Aproximação em Tempo Polinomial)

PTAS	*Polynomial-Time Approximation Scheme* (Esquema de Aproximação em Tempo Polinomial)
PVDTSP	*Polygon-Visiting DTSP* (TSP Dubins para Visita a Polígonos)
RRT	*Rapidly-Exploring Random Tree* (Árvore Aleatória de Exploração Rápida)
TSP	*Traveling Salesman Problem* (Problema do Caixeiro Viajante)
TSPA	*Angle-TSP* (TSP Angular)
TSPN	*TSP with Neighborhoods* (TSP com Vizinhanças)
UAV	*Unmanned Aerial Vehicle* (Veículo Aéreo Não Tripulado)
VRP	*Vehicle Routing Problem* (Problema de Roteamento de Veículos)

Lista de Símbolos

$\lvert \cdot \rvert$	Cardinalidade de um conjunto
$\lVert \cdot \rVert$	Norma euclidiana em $\mathbb{R}^2$
α	Ângulo formado pelos segmentos $\overline{\mathbf{p}_i\mathbf{p}_j}$ e $\overline{\mathbf{p}_i\mathbf{p}_k}$ do triângulo $\mathbf{p}_i\mathbf{p}_j\mathbf{p}_k$, utilizado no cálculo de η
β	Ângulo formado pelos segmentos $\overline{\mathbf{p}_j\mathbf{p}_k}$ e $\overline{\mathbf{p}_k\mathbf{p}_i}$ do triângulo $\mathbf{p}_i\mathbf{p}_j\mathbf{p}_k$, utilizado no cálculo de η
γ	Direção de movimento utilizada na etapa de atualização da posição dos pontos de visita (Algoritmo Evolutivo)
δ	Número aleatório com distribuição uniforme no intervalo $[-\pi/2, \pi/2]$
η	Direção de movimento utilizada na etapa de otimização do posicionamento dos pontos de visita (Heurística)
θ_t	Orientação em $\mathbb{S}^1$ de um veículo no instante de tempo t
κ	Curvatura máxima dos veículos, definido como $1/\rho$
λ	Taxa média de inserção de regiões, utilizada no processo de Poisson
ν	Velocidade linear de todos os veículos
ρ	Raio mínimo de curvatura de todos os veículos
ϱ	Número aleatório com distribuição normal de média μ e variância σ^2
τ_i	i-ésima rota de $\mathcal{T}$
ψ_i	Orientação em $\mathbb{S}^1$ associada ao i-ésimo ponto de visita
$\varphi_{i,j,k}$	Desvio angular entre os segmentos $\overline{\mathbf{q}_i\mathbf{q}_j}$ e $\overline{\mathbf{q}_j\mathbf{q}_k}$
ϕ	Ângulo de esterçamento das rodas direcionais de um veículo tipo *car-like*
ω	Velocidade angular de um veículo
Σ	Sequência (permutação) de um conjunto
Ψ	Conjunto de orientações em $\mathbb{S}^1$ associadas a cada ponto de visita
$\mathbf{\Omega}_i$	i-ésimo indivíduo (cromossomo) de $\mathcal{G}$

$\mathbf{b}_i$	i-ésimo lance de $\boldsymbol{\mathcal{B}}$
$c_{\max}$	Maior distância euclidiana entre o ponto escolhido como base ($\mathbf{q}_0$) e todos os demais pontos de visita do caminho
$\mathrm{dir}(\cdot)$	Função que retorna a direção (argumento) do vetor unitário considerando dois pontos em $\mathbb{R}^2$
k	Número total de veículos em $\boldsymbol{\mathcal{V}}$, ou seja, $k = \lvert\boldsymbol{\mathcal{V}}\rvert$
k	Constante do limite superior de uma curva de Dubins, $\mathsf{k} \in [2,657\ ,\ 2,658]$
$\mathbf{n}_i$	Posição em $\mathbb{R}^2$ do centroide da região $\mathcal{H}_i$
$\mathbf{p}_i$	Posição em $\mathbb{R}^2$ do i-ésimo ponto de visita
$\mathbf{q}_i$	Configuração completa $\mathbf{q}_i = \langle \mathbf{p}_i, \psi_i \rangle$ do i-ésimo ponto de visita
r_i	Comprimento do raio da região $\mathcal{H}_i$
$r_{\min}$	Menor raio dentre todas as regiões de $\boldsymbol{\mathcal{H}}$
$\mathbf{s}_t$	Configuração completa $\mathbf{s}_t = \langle \mathbf{v}_t, \theta_t \rangle$ de um veículo no instante de tempo t
$\mathbf{v}_t$	Posição em $\mathbb{R}^2$ de um veículo no instante de tempo t
$\boldsymbol{\mathcal{B}}$	Conjunto de lances de um leilão
$\mathcal{D}_\rho(\cdot)$	Função comprimento da menor curva de Dubins entre duas configurações
$\boldsymbol{\mathcal{E}}$	Ambiente convexo onde as regiões de interesse serão determinadas
$\mathcal{F}(\cdot)$	Função de aptidão (*fitness*) do Algoritmo Memético, retorna o comprimento da maior rota do conjunto $\boldsymbol{\mathcal{T}}$
$\boldsymbol{\mathcal{G}}$	Conjunto de indivíduos (população)
$\mathcal{H}_i$	i-ésima região de $\boldsymbol{\mathcal{H}}$
$\mathcal{H}_{\text{new}}$	Região determinada mais recentemente no ambiente
$\boldsymbol{\mathcal{H}}$	Conjunto de regiões de interesse a serem visitadas
I	Número total de indivíduos em $\boldsymbol{\mathcal{G}}$, ou seja, $I = \lvert\boldsymbol{\mathcal{G}}\rvert$
L	Distância entre eixos para um veículo do tipo *car-like*
$\mathcal{L}_\rho(\cdot)$	Função comprimento do circuito composto por curvas de Dubins
$\mathcal{L}_\rho^{\text{veic}}$	Comprimento do caminho formado por curvas de Dubins do ponto de visita inicial ($\mathbf{q}_0$) até a posição atual do veículo ($\mathbf{q}_{\text{veic}}$), a distância percorrida
$\mathcal{L}_\rho^{\text{rem}}$	Comprimento do caminho restante (ainda não percorrido) a partir da posição atual do veículo ($\mathbf{q}_{\text{veic}}$), ou seja, $\mathcal{L}_\rho^{\text{rem}} = \mathcal{L}_\rho - \mathcal{L}_\rho^{\text{veic}}$
$\mathcal{M}$	Matriz de custo das curvas de Dubins ligando todas as posições
N	Número total de regiões em $\boldsymbol{\mathcal{H}}$, ou seja, $N = \lvert\boldsymbol{\mathcal{H}}\rvert$
P	Número total de pontos de visita em $\boldsymbol{\mathcal{Q}}$, ou seja, $P = \lvert\boldsymbol{\mathcal{Q}}\rvert$
$\boldsymbol{\mathcal{P}}$	Conjunto das posições em $\mathbb{R}^2$ dos pontos de visita
$\boldsymbol{\mathcal{Q}}$	Conjunto das configurações (posição e orientação) dos pontos de visita
$\boldsymbol{\mathcal{Q}}_{\text{veic}}$	Conjunto de pontos de visita pertencentes à rota de um veículo específico

$\boldsymbol{\mathcal{Q}}_{\text{veic}}^{\text{ativos}}$	Conjunto de pontos de visita ativos (ainda não visitados) pertencentes à rota de um veículo específico
$\mathcal{R}_{ij}$	Área de interseção entre as regiões $\mathcal{H}_i$ e $\mathcal{H}_j$
$\boldsymbol{\mathcal{R}}$	Conjunto das áreas de interseção entre as regiões de $\boldsymbol{\mathcal{H}}$
$\boldsymbol{\mathcal{T}}$	Conjunto de rotas
U	Número aleatório com distribuição uniforme no intervalo $[0, 1]$
$\mathcal{V}_i$	i-ésimo veículo de $\boldsymbol{\mathcal{V}}$
$\boldsymbol{\mathcal{V}}$	Conjunto de veículos
$\mathcal{X}$	Comprimento da maior rota presente em $\boldsymbol{\mathcal{T}}$

Sumário

Capítulo 1

Introdução

É notável o avanço no desenvolvimento e utilização de veículos autônomos. Entretanto, assim como em todos os sistemas, existem diferentes custos relacionados à utilização desses veículos. Por exemplo, o gasto energético total está diretamente relacionado ao tempo necessário para se percorrer um determinado caminho.

Devido a isso, faz-se necessário o desenvolvimento de técnicas capazes de gerar soluções eficientes para os diversos desafios relacionados à utilização de veículos autônomos, aumentando a aplicabilidade dos mesmos nas mais diversas áreas.

Este trabalho apresenta a concepção, desenvolvimento e análise de um sistema de planejamento de caminhos para veículos autônomos com restrições de movimentos para visita a regiões determinadas dinamicamente no ambiente.

1.1 Contextualização

A Robótica ganhou notoriedade nas últimas décadas, principalmente, devido à utilização de manipuladores, ou braços robóticos, em linhas de montagem industriais [Siegwart et al., 2011]. Entretanto, uma área que tem se destacado e obtido grandes avanços em anos mais recentes é a Robótica Móvel, responsável por estudar robôs capazes de locomoverem-se no ambiente em que estão inseridos [Choset et al., 2005; Thrun et al., 2005], denominados robôs móveis.

Uma das linhas de pesquisa fundamentais na área da Robótica Móvel tem como objetivo o desenvolvimento de métodos que permitam prover maior autonomia aos robôs, principalmente em questões ligadas à sua movimentação pelo ambiente. Dessa forma, é possível desenvolver sistemas que necessitarão cada vez menos de alguma interferência humana para auxiliar ou acompanhar a realização das tarefas designadas aos robôs.

Para que um robô seja realmente autônomo, se diz de maneira informal, que ele deve ser capaz de responder a três perguntas básicas: "***Onde Estou?***" (localização), "***Aonde Vou?***" (objetivo) e "***Como Vou?***" (estratégia). Planejadores de caminhos estão relacionados à terceira pergunta, ou seja, qual deve ser o plano (rota) a ser seguido pelo robô para realização de determinada tarefa. Esses métodos são responsáveis por calcular um caminho factível no espaço livre do ambiente ligando, no mínimo, duas dadas posições. A qualidade do caminho gerado pode ser avaliada considerando-se diferentes métricas como o comprimento (mais usual), tempo de execução, gasto energético, entre outras.

Além disso, também tem crescido o interesse no desenvolvimento de técnicas para o problema de geração dinâmica de caminhos para múltiplos robôs. Uma das principais razões está ligada ao fato de que soluções para esse problema têm apresentado bons resultados, permitindo assim uma maior utilização desses robôs na realização de tarefas geradas dinamicamente e distribuídas pelo ambiente.

1.2 Motivação

A coordenação de vários agentes móveis (em nosso caso, veículos ou robôs), é tema de grande importância na Robótica Móvel, sendo o foco de diversos trabalhos na literatura [Gerkey & Matarić, 2004; Dias et al., 2006; Farinelli et al., 2004; Parker, 2008]. Isso se deve ao fato de que a utilização de múltiplos agentes traz diversas vantagens em comparação a sistemas compostos por um único agente. Por exemplo, um sistema formado por diversos agentes é capaz de executar uma tarefa de maneira mais eficiente, permitindo delegar funções a cada agente. Além disso, o sistema se torna mais robusto a falhas individuais dos agentes.

É interessante observar que a maioria dos veículos presentes em nosso cotidiano, por exemplo, automóveis, motocicletas, aviões e navios, possuem determinadas restrições associadas aos movimentos que esses podem realizar. Veículos que possuem essa característica são denominados veículos não-holonômicos. Ao utilizar-se esse tipo de veículo para realização de uma tarefa, o custo associado à execução do caminho poderá ser bem maior do que a simples distância euclidiana linear entre os pontos inicial e final, variando de acordo com as características específicas de cada veículo.

A Figura 1.1 apresenta diferentes exemplos de veículos com aplicações militares ou comerciais que possuem restrições não-holonômicas e poderiam se beneficiar das contribuições deste trabalho.

Destaca-se o veículo apresentado na Figura 1.1(c), que possui um raio de curvatura de aproximadamente 1,6 km (em condições climáticas favoráveis) [Clarkson, 2010]. Além disso, são necessários cerca de 9 km para realizar-se uma parada completa. Dessa forma, é possível observar a importância de se planejar previamente o caminho a ser realizado.

(a) Predator

(b) Caterpillar 797B

(c) Knock Nevis

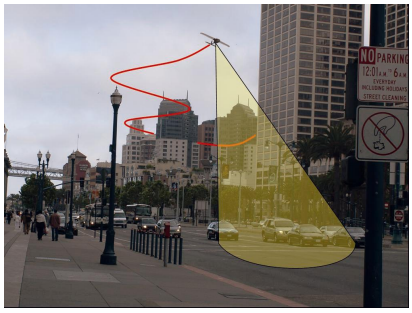
Figura 1.1. Exemplos de veículos atuais que possuem restrições não-holonômicas. (a) UAV militar utilizado em missões de reconhecimento. (b) Veículo de carga utilizado em minas a céu aberto. (c) Navio petroleiro de maior comprimento atualmente em utilização.

O interesse no desenvolvimento de veículos autônomos tem crescido bastante, tendo como principais objetivos a redução dos custos envolvidos em diversos atividades e uma menor exposição de pessoas desnecessariamente a situações consideradas de risco. Uma ampla gama de aplicações se beneficiariam da utilização de diversos agentes móveis autônomos. Podemos citar, por exemplo, monitoramento ambiental, tarefas de busca e salvamento, exploração de regiões, coleta de dados em redes de sensores sem fio, entre outras [Siciliano & Khatib, 2008] (Figura 1.2).

Como pode ser observado, este trabalho foca principalmente na utilização desses veículos em ambientes abertos (sem obstáculos), por exemplo um campo de plantação ou no espaço aéreo. Portanto, a existência de possíveis obstáculos não será considerada durante o planejamento do caminho.

(a) Agricultura de precisão

(b) Monitoramento ambiental



(c) Vigilância/Exploração de regiões

Figura 1.2. Exemplos de possíveis tarefas que não demandam a visita a uma posição exata no ambiente e que normalmente são realizadas utilizando-se veículos com restrições de movimento.

Além disso, em diversas tarefas é comum que novos objetivos sejam adicionados enquanto o agente ainda está realizando o planejamento inicial. Por exemplo, durante a exploração de um ambiente, novas regiões de interesse podem ser informadas. Como consequência, é necessário, também, que os caminhos sejam calculados dinamicamente, de tal maneira que as mudanças no ambiente sejam incorporadas à missão.

1.3 Problema

Um dos grandes desafios associados ao planejamento de caminhos está fortemente relacionado à liberdade de movimento do veículo, ou seja, à existência de determinadas limitações que restringem os comandos que podem ser executados pelo veículo. Um automóvel típico, por exemplo, não pode realizar deslocamentos laterais, ou seja, mover-se instantaneamente de maneira ortogonal ao plano de suas rodas. Esse veículo normalmente realiza movimentos para frente (ou para trás) e realiza curvas de acordo com o esterçamento das rodas direcionais (frontais).

Esse tipo de limitação relacionada ao movimento do veículo é conhecida como restrição não-holonômica, e, conforme dito anteriormente, veículos que a possuem são denominados veículos não-holonômicos. Uma ou mais restrições podem estar associadas ao movimento desses veículos. Um avião, por exemplo, possui como algumas de suas restrições não-holonômicas a curvatura máxima e o ângulo máximo de subida [LaValle, 2006].

Formalmente, veículos não-holonômicos são aqueles em que o número de graus de liberdade controláveis é menor que o seu número total de graus de liberdade [Siegwart et al., 2011]. Logo, uma premissa fundamental ao se calcular caminhos para esse tipo de veículo é que o caminho seja realizável, ou seja, o caminho planejado deve ser passível de execução pelo veículo considerando as suas restrições específicas.

Outra questão relevante na geração do caminho e que também deve ser considerada é a energia gasta para sua realização. Dessa forma, é importante que o comprimento do caminho também seja considerado, dando-se prioridade à geração de caminhos de menor comprimento (no caso planar), permitindo assim gastar-se menos energia e tempo para realização do mesmo. É importante destacar que o gasto energético de um veículo pode envolver vários aspectos (velocidade, características do ambiente, etc.). Entretanto, neste trabalho considera-se que apenas o comprimento do caminho é um fator de influência.

Características relacionadas ao ambiente também são relevantes ao problema. Em diversos trabalhos que abordam o problema de roteamento de veículos, a localização de todos os pontos a serem visitados é previamente conhecida. O problema resume-se, então, em encontrar um caminho que otimize uma determinada métrica (geralmente o comprimento do caminho) passando por esses pontos. Assim, a dificuldade de se encontrar uma solução para o problema aumenta quando novos pontos a serem visitados são adicionados ao ambiente após o início da realização do caminho inicial. É necessário, então, que esses pontos sejam incorporados de forma eficiente ao caminho que o veículo deve percorrer.

A modelagem desse problema, geralmente, é feita considerando-se a visita à posição exata dos pontos. Entretanto, isso não representa bem grande parte dos casos encontradas em situações reais. Como exemplo, suponhamos que um veículo aéreo que tem como tarefa fotografar certas regiões em um ambiente. Não é necessário que esse veículo passe em uma posição específica, mas a tarefa poderá ser executada a partir do momento em que o sensor possa captar adequadamente a região do objetivo. Portanto, é interessante modelar o problema de forma que o veículo possua uma região de interesse a ser visitada, e não apenas um ponto específico.

Finalmente, a modelagem deve considerar o número de veículos que será utilizado na execução da tarefa, ou seja, se um veículo ou múltiplos veículos. A utilização de múltiplos veículos levanta a necessidade de haver algum tipo de coordenação entre eles, inicialmente sendo realizada uma etapa onde cada veículo será informado de quais partes do problema ele é responsável.

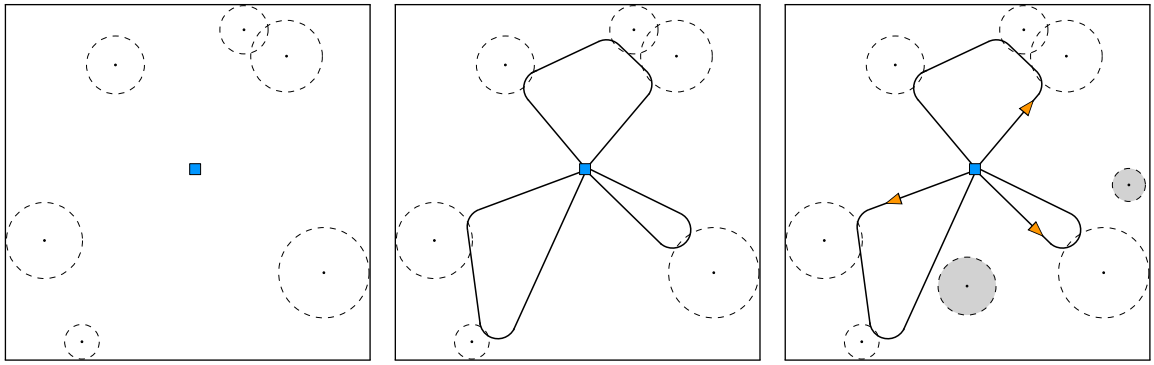
É necessário, então, que uma solução para o problema de geração de caminhos e para a alocação de tarefas seja realizada em conjunto. Pois, conforme visto, utilizar apenas a informação da distância euclidiana entre os pontos alvo dados como entrada para a alocação de tarefas pode gerar resultados inadequados, já que as restrições não-holonômicas dos veículos possuem impacto significativo no comprimento final do caminho gerado.

Uma definição, ainda que superficial, para o problema abordado neste trabalho, com o principal objetivo de introduzir a ideia geral, é apresentada a seguir. Os conceitos envolvidos e uma descrição formal para o problema serão apresentados no Capítulo 3.

***Roteamento Dinâmico de Veículos Não-Holonômicos para Visita a Regiões**: Dado um grupo de veículos com restrições de movimentos e regiões de interesse a serem visitadas determinadas dinamicamente no ambiente ao longo do tempo. Calcular um conjunto de caminhos de forma que todas as regiões sejam visitadas por pelo menos um veículo e o tempo total de visita seja minimizado.*

Conforme pode ser observado, diversos desafios estão diretamente ligados ao problema de geração de caminhos e precisam ser adequadamente abordados de modo a permitir que os veículos naveguem da forma mais eficiente possível. A Figura 1.3 apresenta uma ilustração para o problema que será tratado neste trabalho, permitindo uma melhor visualização das questões descritas anteriormente.

A Figura 1.3(a) apresenta a situação inicial do ambiente, onde algumas regiões de interesse são representadas por círculos tracejados, além disso, é possível também visualizar a posição base onde os veículos devem iniciar e terminar a navegação. Em seguinda, a Figura 1.3(b) apresenta os caminhos iniciais, considerando as demandas já conhecidas. Finalmente, a Figura 1.3(c) representa um instante de tempo em que os veículos (triângulos laranja) já se encontram em movimento e novas regiões de interesse são selecionadas (destacadas em cinza). Logo, surge a necessidade de se alterar o planejamento inicial de forma que ao final da execução todas as regiões tenham sido visitadas.



(a) (b) (c)

Figura 1.3. Ilustração do problema de roteamento dinâmico de veículos não-holonômicos para visita a regiões. (a) Disposição inicial de algumas regiões (círculos tracejados) e da base (quadrado azul). (b) Possíveis rotas para as regiões previamente conhecidas. (c) Os veículos (triângulos laranja) já estão executando as rotas quando novas regiões (destacadas em cinza) são inseridas no ambiente.

Portanto, é necessário o desenvolvimento de técnicas de roteamento de veículos específicas e que considerem os desafios envolvidos na utilização de veículos não-holonômicos para a visita a regiões geradas dinamicamente.

1.4 Contribuições

A seguir são apresentadas as principais contribuições obtidas a partir deste trabalho, que estão agrupadas de acordo com cada um dos diferentes desafios relacionados ao problema.

- **Planejamento de Caminhos para Um Veículo**:
  - **Caso estático**: Apresentação de dois métodos para o problema da geração de caminhos eficientes para veículos não-holonômicos. Inicialmente é proposta a utilização de um Algoritmo Evolutivo, onde a posição e orientação dos pontos de visita, além da sequência de visita, são otimizados de uma forma combinada em cada iteração. O segundo método consiste em uma heurística determinística desenvolvida com base em determinadas características observadas nos melhores caminhos "evoluídos" resultantes do método anterior. Ambos os métodos apresentaram resultados melhores que a solução trivial (utilizando o Algoritmo Alternante [Savla et al., 2005b]). Além disso, a heurística obteve resultados melhores que o estado-da-arte [Isaacs et al., 2011] em aproximadamente 70% dos casos.

  - **Caso dinâmico**: Desenvolvimento de uma estratégia *online* para a inserção de novas regiões ao caminho. O método classifica as novas regiões de acordo com a necessidade de se alterar ou não o caminho para visitá-las. O método apresentou uma razão de competitividade em relação ao método *offline* (heurística), obtida empiricamente, com valor 3,21, entretanto, possuindo um valor médio bem inferior, 2,08.

- **Planejamento de Caminho para Múltiplos Veículos**:

  - **Caso estático**: São apresentados dois métodos: o primeiro método, denominado *k*-DSPLITOUR, consiste na utilização de duas técnicas já existentes na literatura (*k*-SPLITOUR [Frederickson et al., 1978] e Algoritmo Alternante [Savla et al., 2005b]). O segundo método é baseado em um Algoritmo Memético. Ambos os métodos apresentaram resultados (comprimento da maior rota) aproximadamente 30 % maiores que o método DGPTour [Bhadauria et al., 2011] (solução para o *k*-TSPN, ou seja, as restrições do veículo não são consideras).
  - **Caso dinâmico**: É apresentado um algoritmo descentralizado baseado no mecanismo de leilão. O valor do lance efetuado por cada veículo é calculado a partir dos algoritmos apresentados para o caso dinâmico utilizando apenas um veículo. O método apresentou uma razão de competitividade em relação ao método *offline* (MA), obtida empiricamente, com valor 1,56, entretanto, possuindo um valor médio bem inferior, 1,17.

Durante o desenvolvimento do projeto, partes da metodologia proposta, além de problemas correlatos, foram apresentados em artigos que se encontram publicados em conferências internacionais.

- Douglas G. Macharet, Armando A. Neto, Vilar F. C. Neto e Mario F. M. Campos. Nonholonomic Path Planning Optimization for Dubins' Vehicles. *International Conference on Robotics and Automation (ICRA)*, 2011. [Qualis-CC A1]

- Douglas G. Macharet, Armando A. Neto, Vilar F. C. Neto e Mario F. M. Campos. Data Gathering Tour Optimization for Dubins' Vehicles. *IEEE Congress on Evolutionary Computation (CEC)*, 2012. [Qualis-CC A2]

- Douglas G. Macharet, Armando A. Neto, Vilar F. C. Neto e Mario F. M. Campos. An Evolutionary Approach for the Dubins' Traveling Salesman Problem with Neighborhoods. *Genetic and Evolutionary Computation Conference (GECCO)*, 2012. [Qualis-CC A1]

- Paulo L. J. Drews Jr., Douglas G. Macharet e Mario F. M. Campos. A Terrain-Based Path Planning for Mobile Robots with Bounded Curvature. *Latin American Robotics Symposium (LARS)*, 2012. [Qualis-CC B4]

- Douglas G. Macharet, Armando A. Neto, Vilar F. C. Neto e Mario F. M. Campos. Efficient Target Visiting Path Planning for Multiple Vehicles with Bounded Curvature. *International Conference on Intelligent Robots and Systems (IROS)*, 2013. [Qualis-CC A1]

1.5 Organização do Texto

O Capítulo 2 apresenta uma revisão dos principais trabalhos encontrados na literatura relacionados ao tema aqui abordado. No Capítulo 3 o problema desta tese e suas restrições são formalizados. No Capítulo 4 é detalhada a metodologia desenvolvida para o caso onde apenas um veículo é utilizado, considerando inicialmente o caso estático e em seguida o caso dinâmico. Ao final são apresentados os experimentos realizados para validação e avaliação dos métodos propostos. No Capítulo 5 é apresentada a metodologia proposta para solução do problema considerando o caso para múltiplos veículos. Ao final são apresentados os experimentos realizados para validação e avaliação dos métodos propostos. Finalmente, o Capítulo 6 sumariza e discute os resultados obtidos e traça possíveis direções a serem tomadas para extensão do projeto.

Capítulo 2

Trabalhos Relacionados

Nesta seção são apresentadas e discutidas as principais contribuições realizadas por trabalhos encontrados na literatura que estão relacionados ao problema abordado neste trabalho. O problema de planejamento de caminhos para veículos autônomos é fundamental e de grande importância, especialmente para a Robótica. Dessa forma, é possível encontrar diversos trabalhos sobre o tema na literatura [Choset et al., 2005; LaValle, 2006].

Uma possível taxonomia das diferentes abordagens do problema pode agrupar os trabalhos considerando-se, por exemplo, a quantidade (um ou vários) ou tipo (holonômico ou não-holonômico) de veículo envolvido. Características do ambiente também podem ser consideradas, como por exemplo, a presença ou não de obstáculos ou se o ambiente é dinâmico ou estático.

O Problema do Caixeiro Viajante (*Traveling Salesman Problem*, ou TSP) é um problema de otimização combinatória fundamental e amplamente estudado na computação [Applegate et al., 2007]. O problema consiste em determinar o menor caminho (sequência de visita) que passa por um conjunto de pontos (cidades) informados previamente, iniciando em um ponto qualquer e retornando ao ponto de partida após visitar todos os pontos uma única vez. De maneira mais formal, consiste em se determinar o menor ciclo hamiltoniano [Rosen, 2012].

Por se tratar de um problema NP-difícil, as soluções tipicamente adotadas são constituídas por heurísticas. Isso deve-se ao fato de que, devido à sua complexidade, o tempo de cálculo para instâncias com algumas dezenas de pontos já pode se tornar proibitivo caso a aplicação demande um resultado em um curto intervalo de tempo. Dessa forma, é possível escolher entre qualidade de solução ou eficiência. Dentre as principais e mais utilizadas heurísticas pode-se citar os algoritmos de Christofides [Christofides, 1972] e Lin-Kernighan [Lin & Kernighan, 1973].

Uma das principais deficiências ao se utilizar a modelagem proposta pelo TSP em sistemas robóticos está no fato dessa modelagem não incorporar informações relativas aos alvos a serem visitados ou às restrições dos veículos utilizados, por exemplo, o raio mínimo de curvatura. Devido a isso, diversos trabalhos propõem generalizações do TSP incorporando certas restrições relativas, por exemplo, ao veículo ou ao ambiente, tornando o problema mais próximo a cenários reais. Parte das possíveis variações para o TSP que se aplicam ao problema tratado nesse trabalho serão discutidas nas próximas seções.

2.1 Roteamento de Veículos Não-Holonômicos

Como mencionado anteriormente, um grande número de veículos presentes em nosso cotidiano possuem restrições em seus movimentos, e podem ser classificados como não-holonômicos. Esses veículos possuem uma ou mais restrições associadas a seu movimento, dentre as quais pode-se citar curvatura, torção, ângulo de subida, entre outras. Dessa forma, ao se utilizar esses tipos de veículos, por exemplo um veículo com geometria de Ackerman ou um Veículo Aéreo Não Tripulado (*Unmanned Aerial Vehicle*, ou UAV) de asa-fixa, é importante considerar essas restrições e utilizar métodos diferentes para se planejar e estimar o custo de um caminho.

A geração de caminhos para veículos não-holonômicos é um tema de grande importância na Robótica [Siegwart et al., 2011; LaValle, 2006]. Entretanto, apesar de diversos trabalhos na literatura abordarem o tema, uma parte deles possui como foco principal apenas a geração de caminhos "realizáveis" (caminhos que respeitam as restrições de movimento do veículo), desconsiderando o comprimento que esse caminho possuirá.

Diversos trabalhos tratando desse problema fazem uso dos chamados algoritmos probabilísticos. Em [Alves Neto et al., 2010] foi proposta uma metodologia para a geração de trajetórias (o tempo é considerado) para UAVs de asa-fixa em ambientes com obstáculos baseada no uso da Árvore Aleatória de Exploração Rápida (*Rapidly-Exploring Random Tree*, ou RRT). Em [Macharet et al., 2010] a geração de caminhos (o tempo não é considerado) é realizada utilizando-se Algoritmos Genéticos (*Genetic Algorithms*, ou GAs) e curvas de Bézier. Esse tipo de algoritmo (probabilístico) é utilizado em diversos trabalhos [Kuwata et al., 2008; Karaman & Frazzoli, 2010], uma vez que permitem representar de forma mais simplificada as restrições envolvidas no problema, por exemplo, a curvatura máxima do veículo.

Ao se considerar apenas a restrição de curvatura, grande parte dos planejadores que focam em gerar caminhos de comprimento mínimo para esse tipo de veículo utilizam as curvas de Dubins para a modelagem dos caminhos, ao invés de unicamente linhas retas. Em seu trabalho seminal, Dubins demonstrou uma maneira de calcular o menor caminho no espaço bidimensional entre dois pontos com orientação e restrição de curvatura [Dubins, 1957]. O caminho resultante é composto por segmentos de reta e arcos cujo raio respeitam a curvatura máxima do veículo.

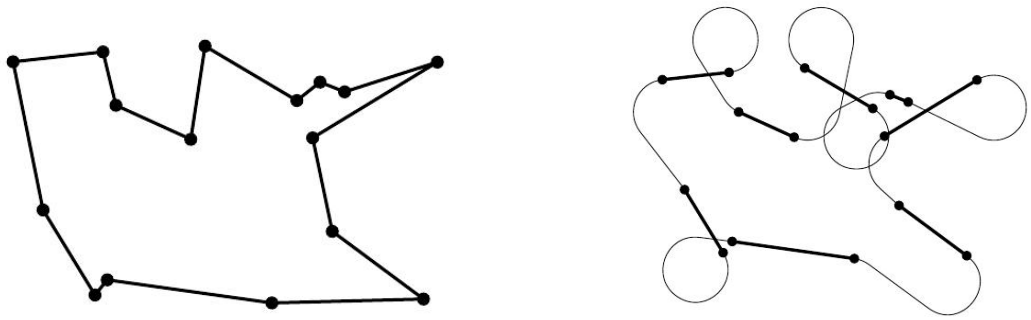
As curvas de Dubins possuem basicamente seis possibilidades de ligação entre os segmentos de reta e os arcos. Uma classificação abordando essas diferentes possibilidades foi apresentada em [Shkel & Lumelsky, 2001], tornando-se possível encontrar o caminho ótimo por meio de uma manipulação lógica das curvas candidatas, sem a necessidade de se calcular todos os seis caminhos possíveis.

Fazendo uso das curvas de Dubins, o problema de se gerar um circuito mínimo de visita e que atende a restrição de curvatura do veículo foi inicialmente introduzido em [Savla et al., 2005b] e recebeu o nome de TSP Dubins (*Dubins TSP*, ou DTSP). O DTSP é uma generalização para o TSP na qual o caminho será composto por curvas de Dubins. Esse problema possui dois requisitos básicos: (i) o caminho a ligar quaisquer dois pontos deve ser uma curva de Dubins; (ii) as curvas de Dubins que incidem sobre um mesmo ponto devem possuir orientações iguais.

Em [Savla et al., 2005b] o caminho é calculado em três etapas. Inicialmente a sequência de visita é obtida resolvendo-se o TSP Euclidiano (*Euclidean TSP*, ou ETSP) para o conjunto de pontos, em seguida devem ser geradas as curvas de Dubins ligando os pontos na ordem definida. Entretanto, para se gerar as curvas é necessário primeiramente que os pontos possuam uma orientação associada a eles (o cálculo dessas orientações é um dos desafios do problema). É então proposta uma heurística simples para gerar orientações denominada Algoritmo Alternante (*Alternating Algorithm*, ou AA), onde os pontos são alternadamente conectados por segmentos de reta e curvas de Dubins.

A Figura 2.1 apresenta uma comparação dos circuitos gerados utilizando-se apenas a métrica euclidiana (Figura 2.1(a)) e considerando-se um veículo com restrições de curvatura (Figura 2.1(b)). Conforme pode ser observado, dependendo da restrição de curvatura e da distribuição dos pontos, o impacto no comprimento pode ser significativo.

Grande parte dos trabalhos na literatura seguem essas mesmas etapas para geração dos caminhos, ou seja, utilizam a sequência de visita obtida a partir do ETSP e, em seguida, os pontos são conectados com curvas de Dubins, dentre os quais pode-se citar [Savla et al., 2005b; Ma & Castañón, 2006].



(a) ETSP (b) DTSP

Figura 2.1. Comparação dos circuitos resultantes obtidos utilizando-se o (a) ETSP e o (b) DTSP com orientações determinadas pelo AA [Savla et al., 2005b].

Entretanto, considerar apenas a métrica da distância euclidiana para determinar a ordem de visita não necessariamente garantirá bons resultados ao se utilizar curvas de Dubins para ligar os pontos. Dessa forma, fazendo com que o veículo tenha que realizar muitas manobras (principalmente nos casos em que os pontos estão dispostos muito próximos uns dos outros).

Em [Tang & Özgüner, 2005], é proposto um método que não se baseia na sequência obtida pelo ETSP. A técnica consiste em inicialmente calcular o centro geométrico do conjunto de pontos. Em seguida, as orientações são calculadas de acordo com o ângulo existente entre o ponto e o centro geométrico. A sequência de visita é então obtida a partir da ordenação crescente dos pontos considerando-se as orientações atribuídas (Figura 2.2).

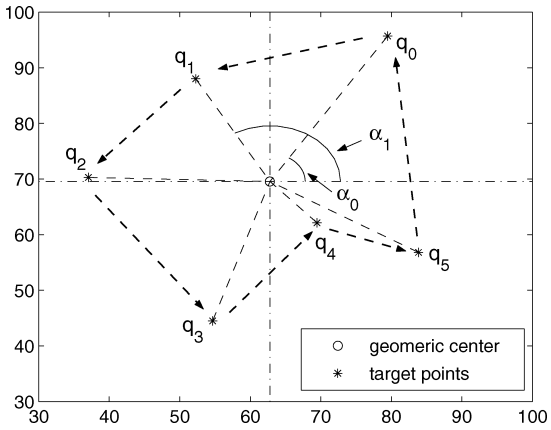


Figura 2.2. Determinação da sequência de visita de acordo com o ângulo existente entre os pontos e o centro geométrico [Tang & Özgüner, 2005].

Conceito similar é utilizado em [Medeiros & Urrutia, 2010], onde a sequência de visita é feita determinando-se o circuito com o menor custo angular. A determinação do circuito com o menor custo angular também corresponde à uma generalização para o TSP denominado TSP Angular (*Angle-TSP*, ou TSPA) [Aggarwal et al., 1997]. O custo angular de um circuito corresponde ao somatório de todas as mudanças de direção (curvas) que o veículo deve realizar. A principal idéia é que circuitos que possuem poucas curvas, ou seja, são mais lineares, tendem a gerar curvas de Dubins menores.

Já em [Le Ny & Feron, 2005], a sequência é obtida utilizando-se diretamente o comprimento das curvas de Dubins entre os pontos. As orientações de todos os pontos são definidas inicialmente como zero e todas as $n(n-1)$ curvas entre todos os pontos são calculadas. Em seguida, é utilizado o TSP Assimétrico (*Asymmetric TSP*, ou ATSP) para se calcular o menor caminho no grafo obtido.

Conforme demonstrado em [Le Ny et al., 2007], o DTSP também é NP-difícil. É importante ressaltar que, diferente de outras variações para o TSP, não é possível formular o DTSP como um problema em um grafo finito. Logo, esse fato impede a aplicação de técnicas já bem estabelecidas em otimização combinatória.

2.2 Geração de Caminhos para Visita a Regiões

Outra generalização para o TSP é conhecida como TSP com Vizinhanças (*TSP with Neighborhoods*, ou TSPN) e foi apresentada em [Arkin & Hassin, 1994]. Nesse problema o caixeiro continua possuindo um circuito de visitas a serem realizadas, entretanto, nessa formulação do problema cada ponto está associado a uma vizinhança. Assim, tanto o caixeiro quanto o possível comprador a ser visitado podem se encontrar em qualquer lugar dessa vizinhança. Logo, o problema é definido em como gerar o menor caminho que intercepte todas as vizinhanças pelo menos uma vez.

O TSPN é um problema NP-Difícil, mais especificamente APX-difícil (aceita um Algoritmo de Aproximação em Tempo Polinomial (*Polynomial-Time Approximation Algorithm*, ou PTAA) com razão de aproximação limitada por uma constante), ou seja, não pode ser aproximado por um fator $2-\varepsilon$, com $\varepsilon > 0$, a menos que P=NP [Safra & Schwartz, 2006].

Entre os primeiros trabalhos que abordam o caso geral para esse problema pode-se citar [Mata & Mitchell, 1995; Gudmundsson & Levcopoulos, 1999]. Nesses trabalhos foram apresentados algoritmos com fator de aproximação $O(\log n)$, onde n é o número de regiões.

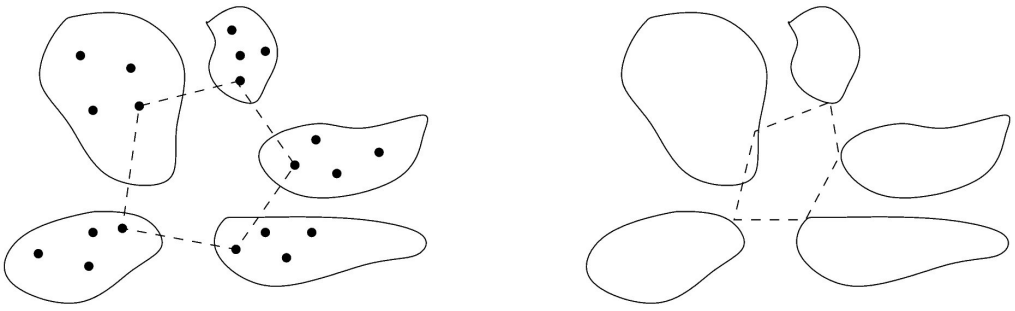
Em [Dumitrescu & Mitchell, 2003] é apresentado um algoritmo para o TSPN onde as regiões são discos uniformes e inicialmente sem interseção. Para o caso geral onde as regiões podem possuir interseção, primeiramente é calculado um conjunto máximo de regiões que não se interceptam. Em seguida, é construído um circuito que visite os centros das regiões desse novo conjunto e que passe pelo perímetro visitando discos que não fazem parte do conjunto.

Um Esquema de Aproximação em Tempo Polinomial (*Polynomial-Time Approximation Scheme*, ou PTAS) para o problema é apresentado em [Mitchell, 2007; Dumitrescu & Mitchell, 2003], especificamente para o caso de regiões convexas que não possuem interseção. A solução apresentada em [de Berg et al., 2005] considera essas mesmas restrições, entretanto, já é tratado o caso em que as regiões possuem tamanho variado.

O algoritmo apresentado em [Elbassioni et al., 2006] considera a utilização de regiões circulares de diferentes tamanhos, entretanto, é necessário que essas possuam diâmetros iguais (ou comparáveis). O circuito está restrito a passar por cada região apenas através de um conjunto finito de pontos previamente definidos. É demonstrado que o algoritmo também possui fator de aproximação constante.

Um problema similar relacionado ao TSPN é denominado na literatura de TSP para Grupo Euclidiano (*Euclidean Group TSP*, ou EGTSP) [Elbassioni et al., 2005]. Nessa instância os pontos de interesse são agrupados em regiões, e o problema consiste em gerar o menor caminho que passe por pelo menos um dos pontos dispostos em cada região. É importante ressaltar a diferença para o TSPN, onde a região precisa ser visitada (caso contínuo), e não a posição exata de um ponto (caso discreto).

A Figura 2.3 exemplifica dois circuitos resultantes gerados considerando o EGTSP e o TSPN.



(a) EGTSP (b) TSPN

Figura 2.3. Comparação dos circuitos resultantes obtidos utilizando-se o (a) EGTSP e o (b) TSPN [Elbassioni et al., 2005].

O problema de geração de caminhos para um sorvedouro (*sink*) móvel para coleta de dados em uma rede de sensores sem fio é um dos principais problemas que fazem uso da modelagem proposta pelo TSPN. O raio de comunicação do sensor representa a vizinhança do pontos, e o agente móvel deve apenas estar nessa região para conseguir se comunicar com o sensor.

Em [Yuan et al., 2007b], a sequência de visita é obtida resolvendo-se o TSP baseado no centro das regiões (pontos de coleta), e em seguida três algoritmos evolucionários são utilizados para otimizar o caminho movimentando-se esses pontos de dentro das regiões (Figura 2.4).

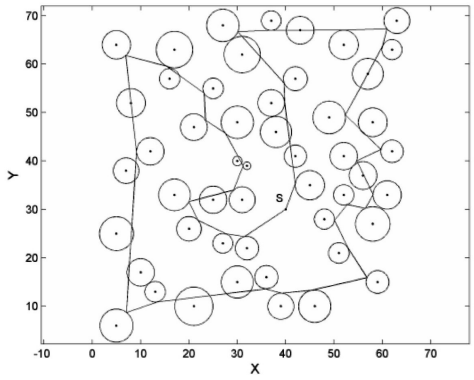


Figura 2.4. Exemplo da solução obtida a partir de um Algoritmo Genético para o TSPN aplicado à geração de caminhos para coleta de dados em uma rede de sensores sem fio [Yuan et al., 2007b]. As regiões circulares representam o raio de comunicação dos sensores.

Em [Comarela et al., 2011] é apresentado um algoritmo baseado na técnica de Otimização por Colônia de Formigas (*Ant Colony Optimization*, ou ACO), onde a principal diferença para o trabalho citado anteriormente se encontra no fato de a permutação também estar embutida no processo de otimização. É importante ressaltar que em ambos os trabalhos as regiões não possuem interseção.

O EGTSP também serve de inspiração para a modelagem do problema de coleta de dados. Em [Wu et al., 2009], um conjunto de nós sensores são agrupados em diferentes subgrupos, e deve-se minimizar o caminho de coleta onde pelo menos a posição exata de um nó sensor em cada subgrupo deve ser visitada. Em [Valle et al., 2008] também é utilizada uma modelagem que remete ao EGTSP, entretanto, após o agrupamento dos nós sensores, o caminho gerado não necessariamente passa pela posição de um nó, mas em pontos virtuais calculados no centro dos grupos previamente gerados.

O TSPN também foi abordado considerando a distribuição de sensores no espaço tridimensional (problema existente principalmente em cenários subaquáticos). Em [Yuan et al., 2007a] é apresentada uma solução para geração de caminhos curtos que passem por diversas regiões esféricas dispostas no espaço utilizando Algoritmos de Estimação de Distribuição (*Estimation of Distribution Algorithms*, ou EDAs).

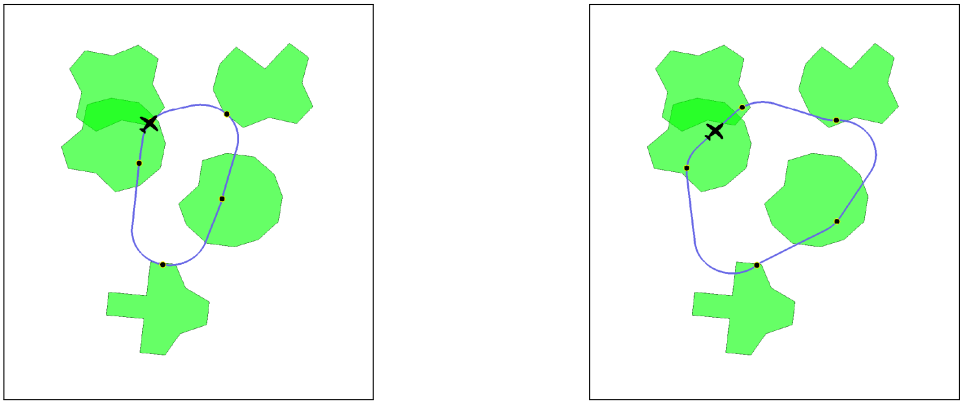
Em [Di Francesco et al., 2011] é apresentada uma revisão geral dos desafios e principais técnicas existentes para o problema da utilização de elementos móveis (veículos) para a coleta de dados em redes de sensores sem fio.

2.3 Visita a Regiões com Veículos Não-Holonômicos

Apenas em trabalhos recentes, a união das restrições impostas pelos problemas do DTSP e TSPN começaram a ser consideradas em um único problema, denominado TSP Dubins com Vizinhanças (*Dubins TSP with Neighborhoods*, ou DTSPN).

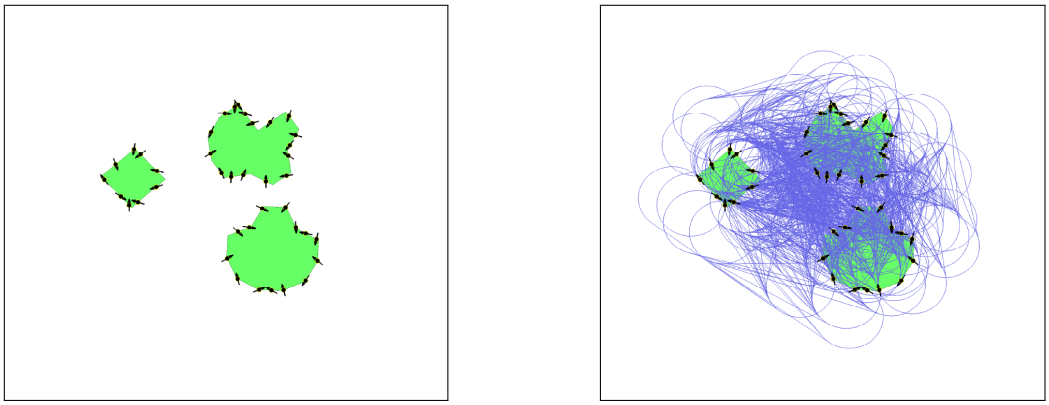
O primeiro trabalho a abordar o DTSPN foi [Obermeyer, 2009], onde foi utilizada a nomenclatura TSP Dubins para Visita a Polígonos (*Polygon-Visiting DTSP*, ou PVDTSP). A modelagem foi utilizada para representar o problema onde um UAV em missão de reconhecimento deve sobrevoar determinadas regiões e fotografar todos os alvos no menor tempo possível. Como os alvos são estáticos e o veículo possui velocidade constante, a função de custo utilizada é constituída apenas pelo fator referente à minimização do comprimento do caminho. O trabalho apresenta uma solução utilizando Algoritmos Genéticos. O cromossomo é composto por uma posição e uma orientação dentro de cada região a ser visitada. O algoritmo é avaliado a partir de uma série de simulações numéricas e os resultados são comparados com os obtidos por um algoritmo de busca aleatória. A Figura 2.5 apresenta dois resultados obtidos e utilizados para avaliação do método apresentado em [Obermeyer, 2009].

Em um trabalho subsequente [Obermeyer et al., 2010], foi apresentada uma abordagem baseada em amostragem (*sampling-based roadmap*). Diversas configurações (posição e orientação) são amostradas nos limites das regiões. Essas amostras são denominadas posições de entrada na região, uma vez que todas as amostras estão orientadas em direção ao interior da região ou são paralelas à borda. Após realizada a amostragem, são calculados os caminhos de todas as amostras de cada região para todas as outras amostras das demais regiões. Em seguida, faz-se uso do ATSP para o cálculo do circuito. A Figura 2.6 exemplifica a disposição das amostras para uma instância do DTSPN e o grafo final após conectar todas as amostras.



(a) Algoritmo Genético (b) Busca Aleatória

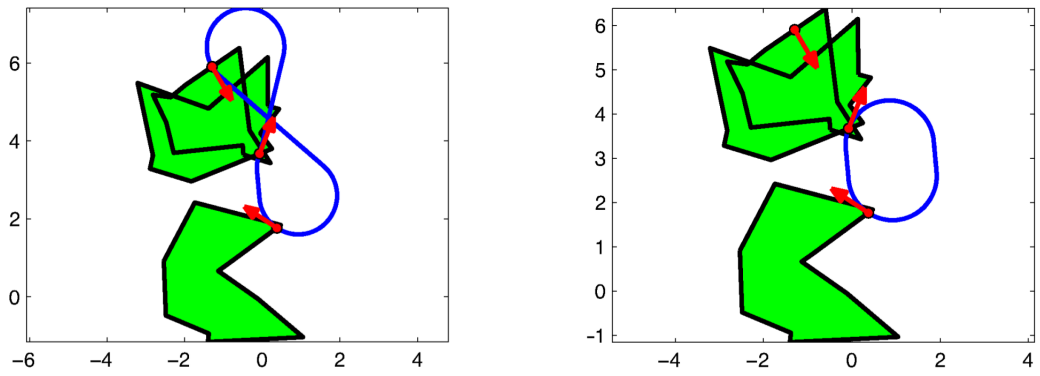
Figura 2.5. Comparação entre os caminhos resultantes para o DTSPN utilizando-se um (a) Algoritmo Genético e (b) Busca Aleatória. [Obermeyer, 2009].



(a) (b)

Figura 2.6. Instância exemplo para o DTSPN. (a) Exemplo de amostras sobre as bordas das regiões e (b) caminhos Dubins conectando todas as amostras entre as regiões [Obermeyer et al., 2010].

O método mais recente encontrado na literatura apresentado em [Isaacs et al., 2011] consiste em uma extensão de [Obermeyer et al., 2010], e também faz uso de uma técnica baseada na amostragem de pontos de visita. A principal diferença é que as amostras não precisam ter apenas "orientações de entrada" (podem assumir qualquer orientação). Além disso, a metodologia consegue tirar vantagem das amostras que são alocadas em regiões de interseção, dessa forma gerando caminhos mais curtos que os propostos na técnica anterior. A Figura 2.7 permite a comparação dos caminhos resultantes obtidos por ambas as técnicas.



(a) (b)

Figura 2.7. Comparação entre os caminhos resultantes para o DTSPN a partir das técnicas de (a) Amostragem [Obermeyer et al., 2010] e (b) Amostragem considerando regiões de interseção [Isaacs et al., 2011].

2.4 Geração de Caminhos para Múltiplos Veículos

Nos problemas discutidos nas seções anteriores o resultado consiste em um único circuito. Uma extensão do TSP para o caso onde deve-se gerar caminhos para mais de um agente é denominado *k*-Traveling Salesman Problem (*k*-TSP). O *k*-TSP pode ser descrito, então, como o problema de gerar caminhos para *k* caixeiros que iniciam em uma determinada cidade e retornam à essa cidade após cada uma das demais cidades terem sido visitadas por exatamente um dos caixeiros. O objetivo nesse caso pode ser a minimização da soma do comprimento de todos os caminhos gerados (redução do gasto energético) ou minimização do maior circuito (redução do tempo de execução).

Um problema bastante similar ao *k*-TSP geralmente é denominado TSP Múltiplo (*Multiple TSP*, ou mTSP). Entretanto, diferentemente do problema anterior, nesse caso normalmente a exigência de que todos os circuitos possuam uma cidade base em comum não é considerada [Sofge et al., 2002].

O *k*-TSP é uma instância do problema mais genérico conhecido como Problema de Roteamento de Veículos (*Vehicle Routing Problem*, ou VRP) [Dantzig & Ramser, 1959; Toth & Vigo, 2001]. O objetivo no caso do VRP clássico é gerar rotas para um conjunto de veículos a partir de um ponto inicial (depósito) de forma a suprir demandas dispostas no espaço bidimensional, otimizando o custo de acordo com uma determinada métrica de qualidade (na maioria dos casos o comprimento do circuito ou o tempo de espera para se atender as demandas).

Enquanto para o VRP consideram-se veículos com uma determinada capacidade de atendimento às demandas, o *k*-TSP é uma instância onde os veículos possuem capacidade ilimitadas. Esse problema também é NP-difícil, e um resumo das possíveis formulações e abordagens utilizadas são apresentadas em [Bektas, 2006].

A Figura 2.8 apresenta uma instância exemplo utilizada como entrada e o resultado esperado após encontrar-se uma solução para o VRP.

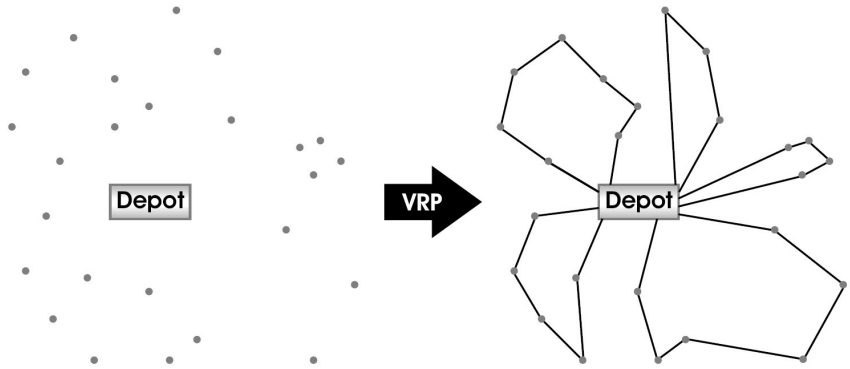


Figura 2.8. Exemplo de uma possível solução para o VRP com um único depósito considerando uma instância aleatória [Alba & Dorronsoro, 2004].

Em [Rathinam et al., 2007] é apresentado um planejador de caminhos para múltiplos veículos Dubins. Assume-se que todos os pontos a serem visitados possuem uma separação mínima entre si de pelo menos duas vezes o tamanho do raio mínimo de curvatura do veículo. Essa é uma das principais deficiências do método, uma vez que a métrica Dubins exercerá maior influência no comprimento do caminho justamente nos casos em que os pontos estão próximos. Além disso, os circuitos não necessariamente possuem ponto de partida e chegada iguais (Figura 2.9).

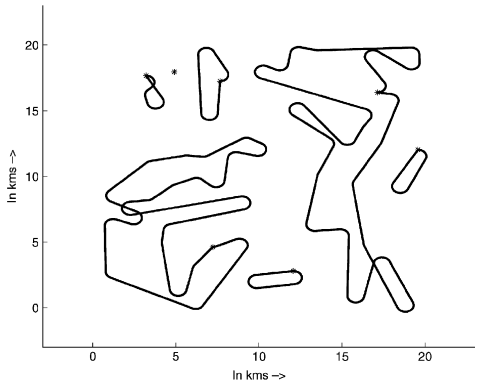


Figura 2.9. Geração de múltiplos caminhos considerando-se veículos não-holonômicos com restrição de curvatura [Rathinam et al., 2007]. O objetivo consiste em minimizar a soma das distâncias percorridas por todos os veículos.

O *k*-TSP também foi utilizado para modelar a coleta de dados em redes de sensores sem fio utilizando-se múltiplos robôs móveis [Tekdas et al., 2009; Bhadauria et al., 2011]. Embora o raio de comunicação seja inicialmente mencionado em [Tekdas et al., 2009], apenas em [Bhadauria et al., 2011] o mesmo é considerado na geração do caminho final. Além disso, em ambos os trabalhos as restrições do veículo também não são consideradas pela metodologia (Figura 2.10). Esses trabalhos consistem em uma generalização para o TSPN, formalizando então o *k*-TSP com Vizinhanças (*k-TSP with Neighborhoods*, ou *k*-TSPN).

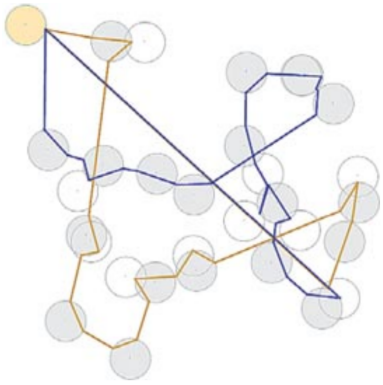


Figura 2.10. Geração de caminhos para coleta de dados em redes de sensores sem fio utilizando múltiplos veículos [Bhadauria et al., 2011]. O objetivo consiste em minimizar o tempo total de coleta.

O problema de geração de caminhos para mais de um veículo também possui relação com o problema de alocação de tarefas (ou alvos). Por exemplo, em [Moore & Passino, 2007; Arslan et al., 2007] são apresentados algoritmos distribuídos para alocação de demandas previamente conhecidas (caso estático) a diferentes veículos. Apesar de se considerar comunicação limitada entre os veículos, as restrições de movimento do veículo e a geração de caminhos para se atender às demandas alocadas a ele não são tratados.

Apesar da restrição de curvatura do veículo (são utilizados UAVs) ser considerada em [Shima et al., 2007], o problema de alocação de alvos e geração de caminhos são tratados separadamente. A alocação é feita a partir da ordenação dos alvos de acordo com a distância euclidiana, e a geração do caminho é realizada, então, utilizando-se a métrica Dubins.

Em geral, grande parte dos trabalhos encontrados na literatura que abordam o problema de roteamento de veículos foca principalmente em ambientes estáticos, ou seja, não tratando o caso onde novas demandas são inseridas ao longo do tempo.

2.5 Roteamento Dinâmico de Veículos

Nesta seção será abordado e discutindo o problema do roteamento dinâmico de veículos. Diferentemente dos casos apresentados anteriormente, o caso dinâmico considera a situação onde não se tem toda a informação relativa ao ambiente (mais especificamente os pontos a serem visitados) previamente à geração do caminho. Dessa forma, é necessário que se faça um replanejamento sempre que uma nova informação se tornar disponível.

Esse problema foi introduzido por [Psaraftis, 1988], onde as demandas estavam dispostas em um grafo, e recebeu o nome de Problema do Restaurador Viajante Dinâmico (*Dynamic Traveling Repairman Problem*, ou DTRP).

O primeiro trabalho que tratou o problema de demandas dispostas no plano euclidiano foi [Bertsimas & van Ryzin, 1991], onde apenas um veículo era utilizado para atender às demandas. Em um trabalho seguinte [Bertsimas & van Ryzin, 1993], a técnica foi estendida para o caso de múltiplos veículos com capacidade limitada, sendo também denominado de DTRP Multi-Veículo (*m-Vehicle DTRP*, ou mDTRP).

O problema de roteamento dinâmico possui diferentes aspectos que devem ser considerados durante o desenvolvimento de novas técnicas, por exemplo, a métrica utilizada pela função de custo, tipo de arquitetura (centralizada/descentralizada), características do veículo (restrições de movimento, quantidade), entre outras.

A Figura 2.11 exemplifica a característica dinâmica do problema com a inserção de novas demandas no ambiente enquanto o veículo já está executando o caminho inicialmente planejado.

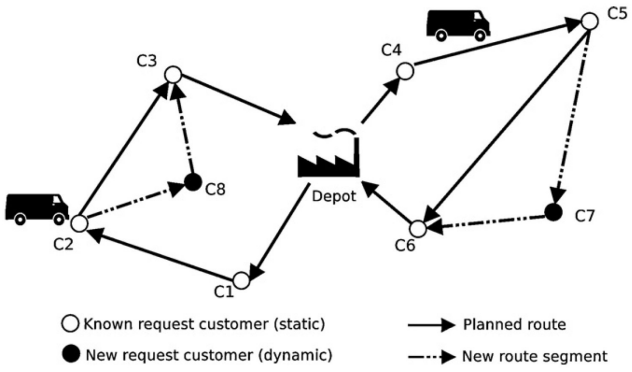


Figura 2.11. Exemplo do problema do roteamento dinâmico de veículos [Khouadjia et al., 2012]. São apresentados os caminhos para dois veículos e as alterações necessárias para se visitar duas novas demandas inseridas após o início da execução.

Em relação à métrica sendo utilizada pela função de custo, pode-se citar como exemplos de possíveis objetivos a minimização da energia gasta pelo veículo, minimização do tempo de espera para se atender uma demanda, minimização do comprimento do caminho e minimização do número de veículos sendo utilizados.

Grande parte das metodologias desenvolvidas abordando o DTRP são constituídas por políticas centralizadas, ou seja, uma entidade central é responsável por calcular todos os caminhos e informar à cada veículo sua respectiva rota. Entretanto, ao se utilizar diversos veículos, é interessante utilizar-se técnicas distribuídas, aumentando assim a escalabilidade e também tornando o sistema mais robusto à possíveis falhas dessa única entidade controladora. Entre os trabalhos que apresentaram formas distribuídas de solução pode-se citar [Frazzoli & Bullo, 2004; Arsie et al., 2009; Pavone et al., 2011].

A literatura na área de Otimização para o DTRP normalmente lida com o problema de um modo mais abstrato, não tratando diretamente de questões que podem surgir em uma instanciação real, como por exemplo, as limitações físicas do veículo. O primeiro trabalho a considerar a restrição de curvatura do veículo foi [Savla et al., 2005a], onde o veículo descreve um padrão de movimento em ziguezague por todo o ambiente. Esse ziguezague ocorre pois o ambiente é dividido em pequenas partes seguindo um formato que lembra uma gota, devido ao formato característico do caminho realizado por veículos com restrições de curvatura. Devido à isso o algoritmo recebeu o nome de *Bead-Tiling*.

Em [Enright et al., 2009] outras soluções são apresentadas. A primeira consiste em dividir o ambiente em subregiões de acordo com um diagrama de Voronoi, onde os veículos devem permanecer no centroide de cada da região até o aparecimento de uma nova demanda, quando só então deslocam-se até ela (Política *Median Circling*). A solução seguinte divide o ambiente em faixas horizontais de movimento (corredores), e os veículos cobrem todo o ambiente se locomovendo por esses corredores (Política *Strip Loitering*).

Apesar dessas soluções mencionadas apresentarem bons resultados quando o objetivo é a minimização do tempo de espera das demandas, o gasto energético é grande, uma vez que os veículos estão em movimento durante todo o tempo (mesmo quando não existem novas demandas). Esse problema poderia ser minimizado se os veículos retornassem para uma base em comum. A Figura 2.12 apresenta os caminhos previamente citados.

Uma abordagem bastante utilizada para se lidar com problemas dinâmicos são os chamados Algoritmos *Online*. O termo *online* aqui se refere à característica dos métodos de lidar com problemas onde a solução deve ser calculada de forma

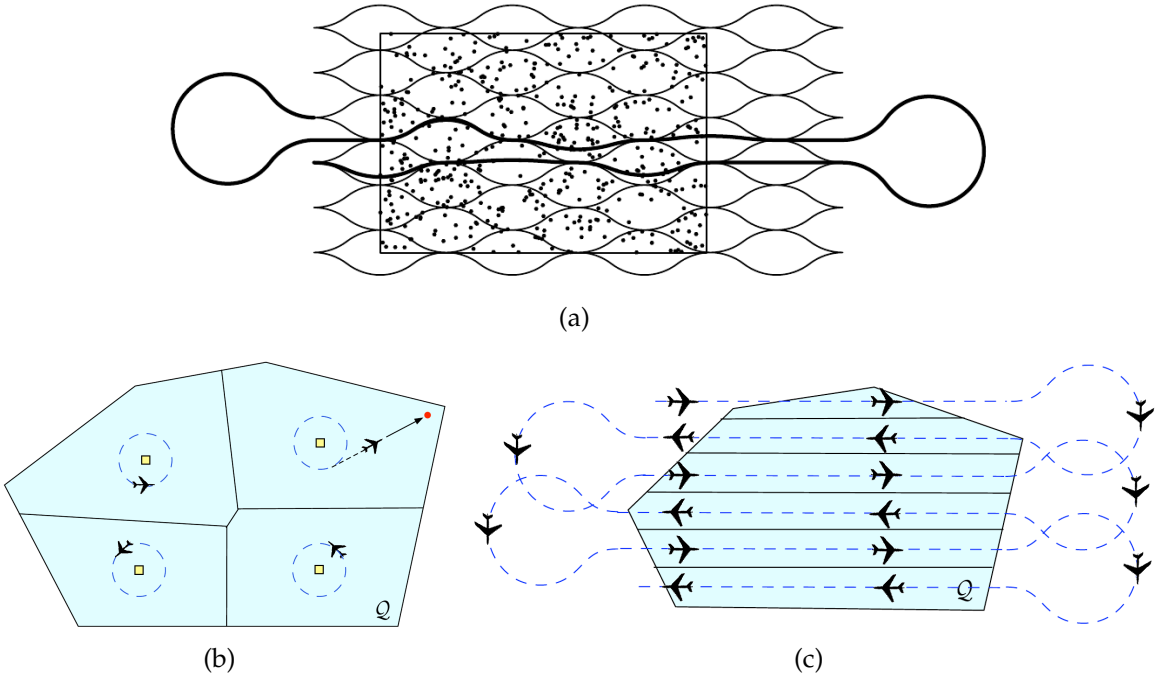


Figura 2.12. Exemplos de caminhos padronizados para solução do DTRP. (a) Algoritmo *Bead-Tiling* [Savla et al., 2005a], (b) Política *Median Circling* e (c) Política *Strip Loitering* [Enright et al., 2009].

incremental, uma vez que não se tem todo conhecimento da instância *a priori*. O principal foco dessa abordagem está no desenvolvimento de algoritmos (em sua maioria heurísticas) eficientes, ou seja, técnicas capazes de gerar boas soluções de forma rápida, entretanto, sem garantias de desempenho [Jaillet & Wagner, 2008].

O primeiro trabalho que abordou a versão *online* do TSP para um espaço métrico geral foi [Ausiello et al., 2001], onde é apresentado um algoritmo ótimo. A versão *online* para o caso assimétrico (ATSP) também já foi foco de estudo [Ausiello et al., 2008]. Em [Jaillet & Wagner, 2008] é apresentada uma pequena revisão da literatura relacionada ao problema de roteamento *online*.

Apesar da característica dinâmica do problema, geralmente considera-se que após a inserção de uma nova demanda em uma determinada posição do ambiente, a mesma permanece nessa posição até ser atendida por um veículo. Em [Bopardikar et al., 2010; Smith et al., 2009], é tratado o caso onde as demandas mudam de posição ao longo do tempo. Nesses trabalhos as demandas são adicionadas sobre o eixo de uma reta e se distanciam dessa em uma trajetória perpendicular e com velocidade constante (Figura 2.13).

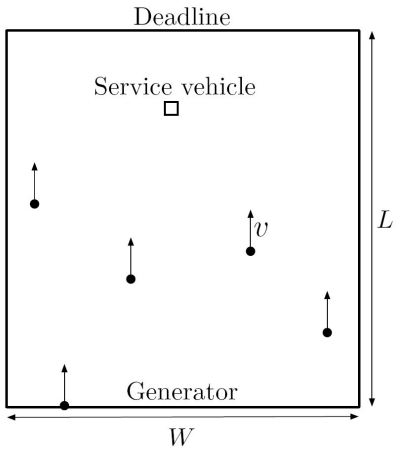


Figura 2.13. Exemplo do roteamento dinâmico de veículos com demandas móveis [Smith et al., 2009]. Todas as demandas são criadas sobre a reta inferior (*generator*) e se deslocam em direção à reta superior (*deadline*). Elas devem ser atendidas pelo veículo (quadrado) antes de alcançarem o final.

Diversas outras particularidades também podem ser analisadas no contexto de roteamento dinâmico, por exemplo, restrições relativas ao instante (janela de tempo) de visita à uma determinada demanda [Pavone et al., 2009; Pavone & Frazzoli, 2010], demandas com diferentes prioridades de atendimento [Smith et al., 2010] e veículos com um conhecimento limitado do ambiente [Enright & Frazzoli, 2006]. Em [Bullo et al., 2011] é apresentada uma revisão geral dos desafios e principais técnicas existentes para o problema do roteamento dinâmico de veículos.

2.6 Contextualização do Trabalho

Conforme apresentado nas seções anteriores, abordar separadamente cada uma das características mencionadas (veículos com restrições não-holonômicas, roteamento a regiões e não pontos específicos, regiões de interesse geradas dinamicamente e a utilização de múltiplos veículos) já representa um grande desafio. Neste trabalho, além de serem apresentadas contribuições para cada uma dessas etapas, será a primeira vez que todas elas são consideradas em conjunto como um único problema, o que aumenta ainda mais o grau de dificuldade.

A Tabela 2.1 apresenta um resumo dos principais trabalhos encontrados na literatura relacionados ao problema, destacando quais características específicas abordam, e ao final contextualizando o problema abordado neste trabalho.

Tabela 2.1. Principais trabalhos encontrados na literatura e as características por eles abordadas, contextualizando o problema proposto neste trabalho.

	Veículos com Restrições	**Visita a Regiões**		**Múltiplos Veículos**	**Problema Dinâmico**
		Convexas	Não-Convexas		
[Savla et al., 2005b]	•				
[Ma & Castañón, 2006]	•				
[Le Ny et al., 2007]	•				
[Elbassioni et al., 2006]		•			
[Yuan et al., 2007b]		•			
[Obermeyer et al., 2010]	•		•		
[Isaacs et al., 2011]	•		•		
[Rathinam et al., 2007]	•			•	
[Shima et al., 2007]	•			•	
[Tekdas et al., 2009]		•		•	
[Bhadauria et al., 2011]		•		•	
[Ausiello et al., 2001]					•
[Savla et al., 2005a]	•				•
[Arsie et al., 2009]				•	•
[Pavone et al., 2011]				•	•
[Enright et al., 2009]	•			•	•
Este Trabalho	•	•		•	•

Capítulo 3

Fundamentos e Formalização

Conforme mencionado nas seções anteriores, este trabalho aborda o problema de geração de rotas para veículos não-holonômicos para visitação a regiões determinadas dinamicamente no ambiente ao longo do tempo. Este capítulo possui como objetivo formalizar os conceitos e definições relativas ao problema. Inicialmente define-se as restrições relacionadas ao modelo dos veículos, em seguida, descreve-se as características das regiões de interesse e das curvas de Dubins, finalmente, a parte dinâmica do problema é formalizada.

3.1 Veículos

Os veículos são modelados a partir da definição clássica de veículos Dubins no plano [Dubins, 1957]. Ou seja, veículos não-holonômicos restritos a se deslocarem por um caminho de curvatura máxima κ. Considerando a cinemática relacionada ao sistema, a curvatura é definida como sendo diretamente proporcional à aceleração lateral do veículo no plano de movimento. A curvatura máxima κ é inversamente proporcional ao raio mínimo de curvatura ρ.

Suposição 1: *Todos os veículos que fazem parte de $\boldsymbol{\mathcal{V}}$ se deslocam com velocidade linear v constante.*

Dado um conjunto $\boldsymbol{\mathcal{V}} = \{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ de veículos disponíveis. O estado do i-ésimo veículo em um determinado instante de tempo é dado por uma configuração $\mathbf{s}_i \in \mathbb{R}^2 \times \mathbb{S}^1 = \mathrm{SE}(2)$, onde $\mathbf{s}_i = \langle \mathbf{v}_i, \theta_i \rangle$. A variável $\mathbf{v}_i = (x_i, y_i)$ representa a posição do veículo no plano dos eixos XY, e θ_i a orientação do veículo em relação ao eixo X.

Suposição 2: *Todos os veículos que fazem parte de $\mathcal{V}$ são homogêneos, ou seja, possuem as mesmas características de raio mínimo de curvatura (ρ) e velocidade linear (v).*

A Figura 3.1 apresenta as variáveis previamente definidas em um modelo de um automóvel típico. Além disso, também são apresentadas L, relativo à distância entre eixos, e ϕ, o ângulo de esterçamento das rodas direcionais.

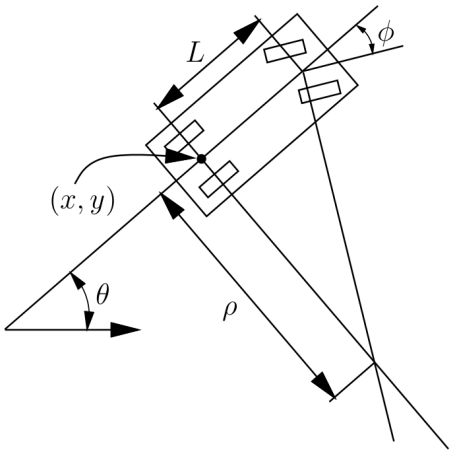


Figura 3.1. Ilustração indicando os diversos parâmetros associados ao modelo cinemático de um carro com direção de Ackerman [LaValle, 2006].

O modelo cinemático para um determinado veículo i é dado por:

$$\dot{\mathbf{s}}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} v\ \cos(\theta_i) \\ v\ \sin(\theta_i) \\ \omega_i \end{bmatrix}, \tag{3.1}$$

onde v ($v \in \mathbb{R}^+$) denota a velocidade linear e ω a velocidade angular ($\omega \in \{-v/\rho,\ 0,\ v/\rho\}$).

Este modelo é bastante utilizado na literatura para representar uma vasta classe de veículos não-holonômicos, dentre os quais destacamos veículos que possuem a forma de direção utilizada em automóveis (*car-like*, ou Ackerman *steering*) [Dudek & Jenkin, 2010] e UAVs de asa-fixa [Shanmugavel et al., 2005].

3.2 Regiões

Conforme mencionado anteriormente, o problema aqui abordado está relacionado ao TSPN, e por clareza repetimos a definição para esse problema.

Dado um conjunto $\boldsymbol{\mathcal{H}} = \{\mathcal{H}_1, \ldots, \mathcal{H}_N\}$ de regiões convexas a serem visitadas, cada uma localizada em uma determinada posição (centro da região) $\mathbf{n}_i = (x_i, y_i)$ (com $1 \leq i \leq N$) em um ambiente convexo $\boldsymbol{\mathcal{E}} \subset \mathbb{R}^2$ e sem a presença de obstáculos. O objetivo do TSPN é determinar um conjunto $\boldsymbol{\mathcal{P}}$ de P posições de visita, assim como também uma permutação Σ, ou seja, a ordem de visita $\boldsymbol{\mathcal{P}}_\Sigma = \langle \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_P \rangle$, que satisfaz as seguintes restrições:

1. cada região $\mathcal{H}_i$ deve possuir pelo menos uma posição de visita; e
2. uma função de custo pré-determinada (*e.g.* comprimento total do caminho) deve ser otimizada.

O TSP clássico é o caso particular em que $\mathcal{H}_i = \{\mathbf{n}_i\}$ e $P = N$, para o qual Σ é uma permutação de $\{\mathbf{n}_1, \ldots, \mathbf{n}_N\}$.

Assume-se que a região associada a uma determinada posição de interesse é definida como uma região circular com centro sobre a posição do alvo, ou seja:

$$\mathcal{H}_i = \{\mathbf{p} \in \mathbb{R}^2 : \|\mathbf{p} - \mathbf{n}_i\| \leq r_i\}, \tag{3.2}$$

onde $\|\cdot\|$ denota a norma euclidiana em $\mathbb{R}^2$, e r_i representa o comprimento do raio da região. Escolheu-se a utilização de regiões circulares por questão de simplicidade, não influenciando a aplicação da técnica no cenário mais geral, ou seja, qualquer região convexa.

É interessante observar que de acordo com a definição acima, uma única posição de visita pode simultaneamente servir como posição de visita para mais de uma região, caso essas regiões possuam uma área de interseção. Formalmente temos, que se $\mathcal{R}_{ij} = \mathcal{H}_i \cap \mathcal{H}_j \neq \emptyset$ para um dado $i \neq j$, então qualquer posição de visita $\mathbf{p}_p \in \mathcal{R}_{ij}$ atende à primeira restrição definida para $\boldsymbol{\mathcal{P}}$. De maneira mais geral, uma única posição de visita pode ser definida de forma a atender diversas vizinhanças que possuam áreas de interseção ($\mathcal{H}_i \cap \mathcal{H}_j \cap \mathcal{H}_k \cap \ldots$). A Figura 3.2 é apresentada com o intuito de permitir um melhor entendimento de todas as variáveis relacionadas às regiões aqui definidas.

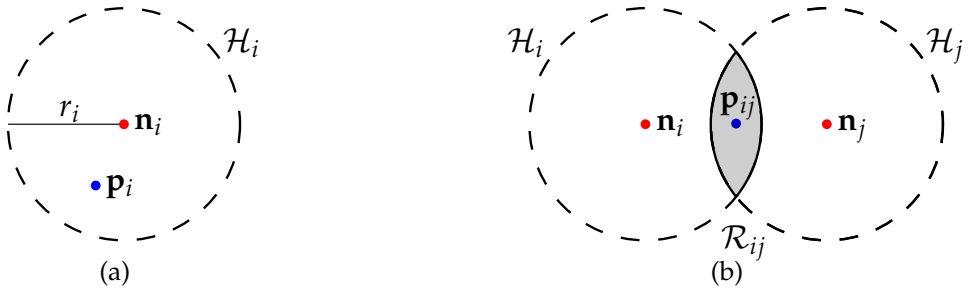


Figura 3.2. Exemplos apresentando todas as variáveis relacionadas às regiões. (a) A região não possui interseção com outras regiões. Nesse caso a posição de visita ($\mathbf{p}_i$) pode assumir qualquer localização dentro da região. (b) Caso onde ocorre a interseção entre duas regiões. Nesse caso é interessante utilizar apenas uma posição de visita ($\mathbf{p}_{ij}$) que atenda às duas regiões simultaneamente.

Finalmente, considerando o modelo cinemático do veículo que será utilizado (visto na seção anterior), não apenas a posição de visita é importante, mas também a orientação que o veículo deve assumir no ponto. Logo, cada posição de visita $\mathbf{p}_i$ também deve ter associada a si uma determinada orientação ψ_i. Dessa forma, define-se de maneira mais geral um ponto de visita como uma configuração $\mathbf{q}_i = \langle \mathbf{p}_i, \psi_i \rangle$, que também irá representar o estado completo do veículo naquele ponto, ou seja, $\mathbf{s}_i = \mathbf{q}_i$.

3.3 Curvas de Dubins

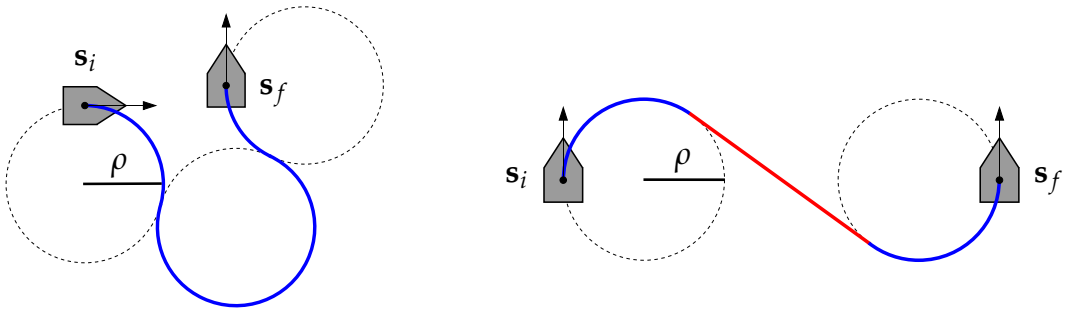
As curvas de Dubins representam o menor caminho entre dois pontos com orientação no espaço bi-dimensional, considerando-se um veículo com um determinado raio mínimo de curvatura [Dubins, 1957].

Os caminhos gerados utilizando-se curvas de Dubins são formados a partir da ligação de segmentos de reta (S) e arcos de circunferência (C) de raio igual ao raio mínimo de curvatura do veículo. Os caminhos gerados utilizando-se curvas de Dubins podem ser classificados basicamente em dois tipos fundamentais: (i) caso curto (CCC) e (ii) caso longo (CSC), possuindo um total de seis possibilidades.

O caso curto agrupa os caminhos compostos pela ligação consecutiva de três arcos de raio ρ para a esquerda (L) ou para direita (R), dessa forma, possuindo as seguintes formações $\{LRL, RLR\}$, uma vez que arcos consecutivos devem possuir movimentos contrários (ou seja, caminhos LL ou RR não são permitidos).

O caso longo representa os caminhos onde existe um segmento de reta tangente entre os arcos, e possuindo as seguintes configurações admissíveis $\{LSL, RSR, LSR, RSL\}$. É interessante ressaltar que dentro de um mesmo grupo poderá existir mais de um caminho (combinação) entre as diferentes posições que deseja-se conectar, entretanto, apenas um terá o comprimento ótimo.

A Figura 3.3 exemplifica os caminhos gerados ligando duas configurações para ambos os casos (curto e longo), as partes destacadas em azul representam os arcos e em vermelho o segmento de reta.



(a) Caso curto (RLR) (b) Caso longo (RSL)

Figura 3.3. Exemplos de curvas de Dubins entre duas configurações ($\mathbf{s}_i$ e $\mathbf{s}_f$), abordando as principais situações características: (a) caso curto e (b) caso longo.

A determinação do caso a ser escolhido para a geração do caminho é feita baseada nas configurações inicial ($\mathbf{s}_i$) e final ($\mathbf{s}_f$) desejadas. Assim, temos que se $d > d_{\min}$ o caminho será determinado por um caso longo, caso contrário, o caso curto deverá ser aplicado. As variáveis d e $d_{\min}$ são obtidas pelas Equações 3.3 e 3.4, respectivamente, onde θ_i e θ_f representam as orientações das configurações já modificadas para estarem sobre o mesmo eixo X do sistema de coordenadas.

$$d = \frac{\|\mathbf{v}_i - \mathbf{v}_f\|}{\rho}, \tag{3.3}$$

$$d_{\min} = \sqrt{4 - (|\cos\theta_i| + |\cos\theta_f|)^2} + |\sin\theta_i| + |\sin\theta_f|. \tag{3.4}$$

Essa validação foi proposta por [Shkel & Lumelsky, 2001] como forma de simplificar a geração dos caminhos, evitando-se o cálculo de todas as possibilidades desnecessariamente. O próximo passo consiste no cálculo das curvas admissíveis de acordo com o tipo do caso determinado.

Dessa forma, o comprimento da curva que atende a Equação 3.1, a partir de uma configuração inicial $\mathbf{q}_1$ até uma configuração final $\mathbf{q}_2$, será representado por $\mathcal{D}_\rho(\mathbf{q}_1, \mathbf{q}_2) : \mathrm{SE}(2) \times \mathrm{SE}(2) \to \mathbb{R}^+$. O comprimento final da curva é calculado então a partir da soma de três parâmetros normalizados pelo raio mínimo de curvatura do veículo, ou seja:

$$\mathcal{D}_\rho(\cdot) = (a + b + c)\rho. \tag{3.5}$$

As equações utilizadas para o cálculo dos parâmetros a, b e c para cada um dos casos são apresentadas a seguir e foram adaptadas a partir de [Shkel & Lumelsky, 2001; Dubins, 1957]. Conforme mencionado anteriormente, essas equações também já consideram que as configurações foram transformadas para um sistema de coordenadas onde uma configuração está sobre a origem e ambas estão sobre o mesmo eixo X.

- **Caso LSL**:

$$a_{\mathrm{lsl}} = \left(-\theta_i + \arctan\left(\frac{\cos\theta_f - \cos\theta_i}{d + \sin\theta_i - \sin\theta_f}\right)\right) \mod 2\pi$$

$$b_{\mathrm{lsl}} = \sqrt{2 + d^2 - 2\cos(\theta_i - \theta_f) + 2d(\sin\theta_i - \sin\theta_f)}$$

$$c_{\mathrm{lsl}} = \left(\theta_f - \arctan\left(\frac{\cos\theta_f - \cos\theta_i}{d + \sin\theta_i - \sin\theta_f}\right)\right) \mod 2\pi$$

- **Caso RSR**:

$$a_{\mathrm{rsr}} = \left(\theta_i - \arctan\left(\frac{\cos\theta_i - \cos\theta_f}{d - \sin\theta_i + \sin\theta_f}\right)\right) \mod 2\pi$$

$$b_{\mathrm{rsr}} = \sqrt{2 + d^2 - 2\cos(\theta_i - \theta_f) + 2d(\sin\theta_f - \sin\theta_i)}$$

$$c_{\mathrm{rsr}} = \left((\theta_f \mod 2\pi) + \arctan\left(\frac{\cos\theta_f - \cos\theta_i}{d + \sin\theta_i - \sin\theta_f}\right)\right) \mod 2\pi$$

- **Caso RSL**:

$$a_{\mathrm{rsl}} = \left(\theta_i - \arctan\left(\frac{\cos\theta_i + \cos\theta_f}{d - \sin\theta_i - \sin\theta_f}\right) - \arctan\left(\frac{2}{b_{\mathrm{rsl}}}\right)\right) \mod 2\pi$$

$$b_{\mathrm{rsl}} = \sqrt{d^2 - 2 + 2\cos(\theta_i - \theta_f) - 2d(\sin\theta_i + \sin\theta_f)}$$

$$c_{\mathrm{rsl}} = \left((\theta_f \mod 2\pi) - \arctan\left(\frac{\cos\theta_i + \cos\theta_f}{d - \sin\theta_i - \sin\theta_f}\right) - \arctan\left(\frac{2}{b_{\mathrm{rsl}}}\right)\right) \mod 2\pi$$

- **Caso LSR**:

$$a_{\text{lsr}} = \left(-\theta_i + \arctan\left(\frac{\cos\theta_i - \cos\theta_f}{d + \sin\theta_i + \sin\theta_f}\right) - \arctan\left(\frac{-2}{b_{\text{lsr}}}\right)\right) \mod 2\pi$$

$$b_{\text{lsr}} = \sqrt{d^2 - 2 + 2\cos(\theta_i - \theta_f) + 2d(\sin\theta_i + \sin\theta_f)}$$

$$c_{\text{lsr}} = \left(-(\theta_f \mod 2\pi) + \arctan\left(\frac{\cos\theta_i - \cos\theta_f}{d + \sin\theta_i + \sin\theta_f}\right) - \arctan\left(\frac{-2}{b_{\text{lsr}}}\right)\right) \mod 2\pi$$

- **Caso RLR**:

$$a_{\text{rlr}} = \left(\theta_i + \arctan\left(\frac{\cos\theta_i - \cos\theta_f}{d - \sin\theta_i + \sin\theta_f}\right) - \frac{b_{\text{rlr}}}{2}\right) \mod 2\pi$$

$$b_{\text{rlr}} = \left(-\arccos\left(\frac{6 - d^2 + 2\cos(\theta_i - \theta_f) + 2d(\sin\theta_i - \sin\theta_f)}{8}\right)\right) \mod 2\pi$$

$$c_{\text{rlr}} = \left(\theta_i - \theta_f - a_{\text{rlr}} + b_{\text{rlr}}\right) \mod 2\pi$$

- **Caso LRL**:

$$a_{\text{lrl}} = \left(-\theta_i + \arctan\left(\frac{-\cos\theta_i + \cos\theta_f}{d + \sin\theta_i - \sin\theta_f}\right) - \frac{b_{\text{lrl}}}{2}\right) \mod 2\pi$$

$$b_{\text{lrl}} = \left(-\arccos\left(\frac{6 - d^2 + 2\cos(\theta_i - \theta_f) + 2d(-\sin\theta_i + \sin\theta_f)}{8}\right)\right) \mod 2\pi$$

$$c_{\text{lrl}} = \left(\theta_f - \theta_i - a_{\text{lrl}} + b_{\text{lrl}}\right) \mod 2\pi$$

3.4 Dinamicidade

O DTRP pode ser interpretado como um problema de enfileiramento de demandas distribuídas espacialmente. Portanto, em vez das demandas chegarem em uma fila localizada em uma posição previamente definida, elas são adicionadas por todo o espaço. A principal diferença para um problema de fila simples está no fato de que nesse caso não necessariamente as demandas serão atendidas de forma *first-in-first-out*, mas é necessário determinar de forma eficiente a ordem de atendimento.

As demandas serão adicionadas dinamicamente em um ambiente convexo $\mathcal{E}$ de acordo com um processo de Poisson espaço-temporal homogêneo. Esse processo foi escolhido por ser um dos mais utilizados para representação de problemas de chegadas em fila, além de possuir algumas características interessantes como aleatoriedade espacial completa (as demandas são independentes e identicamente distribuídas) e apresentar um modelo analiticamente tratável [Gelfand et al., 2010].

A frequência de inserção de novas regiões de interesse será definida pela taxa $\lambda > 0$, que representa uma estimativa da quantidade de novas regiões que devem ser criadas em uma determinada unidade de tempo. A função de distribuição acumulada de um processo de Poisson pode ser representada por uma distribuição exponencial [Papoulis & Pillai, 2002]:

$$F_z(z) = 1 - e^{-\lambda z}, \qquad z > 0. \tag{3.6}$$

O tempo decorrido até a próxima inserção de uma região no ambiente será obtido a partir do método da transformada inversa [Knuth, 1997]. Esse método gera amostras na transformada obtida a partir da função de distribuição acumulada. Para o caso da função exponencial previamente citada temos

$$T = \frac{-\ln U}{\lambda}, \tag{3.7}$$

onde $U \in [0, 1]$ é um número real aleatório com distribuição uniforme.

A disposição no ambiente será determinada por uma distribuição espacial de acordo com uma função densidade de probabilidade f. Assumimos que $f : \boldsymbol{\mathcal{E}} \rightarrow \mathbb{R}^+$, onde toda a probabilidade relacionada ao posicionamento de uma nova região encontra-se no ambiente, ou seja

$$\int_{\boldsymbol{\mathcal{E}}} f(x)\mathrm{d}x = 1. \tag{3.8}$$

3.5 Formalização do Problema

Um vez discutidos todos os conceitos e variáveis envolvidas é possível, então, apresentar a definição formal do problema abordado neste trabalho.

De acordo com a quantidade de veículos (um ou múltiplos) sendo considerada, o problema possuirá características bem distintas (inclusive o objetivo final que se deseja alcançar). Dessa forma, os diferentes problemas sendo considerados, e dependentes do número de veículos, são formalizados a seguir.

3.5.1 Planejamento de Caminhos para Um Veículo

Conforme mencionado anteriormente, assume-se que o veículo se desloca com velocidade constante, logo, a minimização do comprimento do caminho também corresponderá à redução no tempo total de visita.

Problema I: ***Planejamento Dinâmico de Caminho para um Veículo Não-Holonômico para Visita a Regiões****. Seja* $\boldsymbol{\mathcal{Q}}_\Sigma = \langle \boldsymbol{\mathcal{P}}_\Sigma, \Psi_\Sigma \rangle$ *uma determinada permutação* Σ *de certas configurações* $\mathbf{q}$*, cada uma relativa a pelo menos uma das* N *regiões* $\mathcal{H}$*, onde* $\boldsymbol{\mathcal{P}}_\Sigma$ *representa o vetor relativo às informações de posição e* Ψ_Σ *o vetor de orientações dos pontos de visita dessa permutação. O veículo é modelado como um veículo Dubins, possuindo um raio mínimo de curvatura* ρ*. Seja também* $\mathcal{J} : SE(2) \to \mathbb{R}^2$ *uma função de projeção que transforma uma determinada configuração em um ponto no plano do ambiente de trabalho, i.e.* $\mathcal{J}(\mathbf{q}) = [x \quad y]^T$*. Dessa forma, o problema consiste em otimizar (minimizar) o comprimento total do circuito,* $\mathcal{L}_\rho(\cdot)$*, formado por curvas de Dubins. Formalmente:*

$$\begin{aligned} &\underset{\boldsymbol{\mathcal{P}},\, \Psi,\, \Sigma}{\arg\min}\, \mathcal{L}_\rho(\boldsymbol{\mathcal{Q}}_\Sigma) \\ &\textit{sujeito à } \mathcal{J}(\mathbf{q}_i) \in \mathcal{H}_i,\ i = 1, \ldots, N \end{aligned} \tag{3.9}$$

com

$$\mathcal{L}_\rho(\boldsymbol{\mathcal{Q}}_\Sigma) = \sum_{i=1}^{P-1} \mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{i+1}) + \mathcal{D}_\rho(\mathbf{q}_P, \mathbf{q}_1), \tag{3.10}$$

onde $\mathcal{D}_\rho : SE(2) \times SE(2) \to \mathbb{R}^+$ *é a função que retorna o comprimento do menor caminho entre duas configurações para um veículo Dubins com raio mínimo de curvatura* ρ*.*

Logo, o problema consiste em determinar um conjunto de pontos de visita, de forma que todas as regiões de interesse possuam pelo menos uma configuração dentro de seus limites. Considerando que o veículo se desloca com velocidade constante, para se minimizar o tempo total de visita basta determinar o menor circuito possível. Circuito esse que será minimizado escolhendo-se boas posições e orientações para o conjunto de pontos de visita.

3.5.2 Planejamento de Caminho para Múltiplos Veículos

Para se manter o objetivo inicial (minimização do tempo total de visita), o problema de planejamento de caminhos, agora considerando múltiplos veículos, não mais necessariamente terá como objetivo apenas a redução do comprimento (ou da soma dos comprimentos) de todos os caminhos (rotas) envolvidos.

Uma das principais vantagens em se utilizar múltiplos veículos (agentes) para a execução de determinadas tarefas está no fato de se poder distribuir entre eles partes de uma tarefa, e com isso tornar a execução geral mais eficiente.

Logo, considerando o conjunto $\boldsymbol{\mathcal{T}} = \{\tau_1, \tau_2, \ldots, \tau_k\}$ de rotas (caminhos) associadas a k diferentes veículos, de forma que cada rota inicia e termina em um ponto em comum (base). Onde uma determinada rota τ_j é definida como uma permutação (sequência) $\Sigma_j = \langle 0, 1, 2, \ldots, 0 \rangle$ de configurações (pontos de visita), ou seja:

$$\tau_j = \langle \mathbf{q}_0,\ \mathbf{q}_1,\ \mathbf{q}_2, \ldots,\ \mathbf{q}_0 \rangle.$$

Exceto pela base ($\mathbf{q}_0$), cada ponto de visita associado a uma região de $\boldsymbol{\mathcal{H}}$ pertence a uma, e somente uma, rota de $\boldsymbol{\mathcal{T}}$ e

$$\boldsymbol{\mathcal{Q}} \subset \bigcup^{k} \tau_j, \tag{3.11}$$

onde $\boldsymbol{\mathcal{Q}} = \{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_P\}$, com $\mathbf{q}_i \subset \mathrm{SE}(2)$, representa o conjunto de todas os pontos de visita (configurações).

Pelas definições anteriores é interessante observar que não necessariamente uma determinada rota possuirá outros pontos além da própria base. Ou seja, uma possível solução para o problema também não obrigatoriamente utilizará todos os k veículos disponíveis. Por exemplo, caso o objetivo da missão seja consumir a menor quantidade possível de energia, utilizar todos os veículos pode não ser a melhor solução. Em contrapartida, caso o objetivo seja alcançar todos os alvos o mais rápido possível, o uso de mais veículos pode gerar resultados mais satisfatórios. Conforme mencionado anteriormente, o objetivo deste trabalho é executar a tarefa (visitar todas as regiões) no menor tempo possível.

Dessa forma, dada uma determinada rota τ_j para o j-ésimo veículo de um conjunto $\boldsymbol{\mathcal{V}}$ de veículos disponíveis, é necessário definir uma configuração $\mathbf{q}_i$ interior a cada região $\mathcal{H}_i$ que pertence à sequência de visita Σ_j. Cada uma dessas configurações deve ser determinada de forma a minimizar o custo (comprimento) total $\mathcal{L}_\rho^j$ da rota à qual pertence. Em outras palavras, deve-se calcular os conjuntos de configurações $\boldsymbol{\mathcal{Q}}_{\Sigma_j}$ para $1 \leq j \leq k$ compostos por $\boldsymbol{\mathcal{P}}_{\Sigma_j} = \langle \mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_0 \rangle$ (um subconjunto de todas as posições possíveis) e $\Psi_{\Sigma_j} = \langle \psi_0, \psi_1 \ldots, \psi_0 \rangle$ (o subconjunto das orientações correspondentes a cada posição) que produzem circuitos hamiltonianos mínimos que passam pela base para cada um dos veículos.

Suposição 3: *Não são considerados os possíveis casos de colisões durante a execução dos caminhos. Assume-se que veículos aéreos podem se locomover em diferentes planos de altitude, enquanto veículos terrestres ou aquáticos podem interromper seu movimento a qualquer momento, dando passagem a outro veículo.*

Problema II: ***Roteamento Dinâmico de Veículos Não-Holonômicos para Visita a Regiões****. Seja k o número de veículos disponíveis para executar a tarefa de se vistar todas as regiões em* $\boldsymbol{\mathcal{H}}$*. Os veículos são modelados como veículos Dubins, possuindo um raio mínimo de curvatura* ρ*. Seja também* $\mathcal{J} : SE(2) \rightarrow \mathbb{R}^2$ *uma função de projeção que transforma uma determinada configuração em um ponto no plano do ambiente de trabalho, i.e.* $\mathcal{J}(\mathbf{q}) = [x \quad y]^T$*. Seja também* $\mathcal{X} : SE(2)^P \rightarrow \mathbb{R}^+$ *obtida por*

$$\mathcal{X}(\boldsymbol{\mathcal{T}}) = \max_j \left[\mathcal{L}_\rho^j(\boldsymbol{\mathcal{Q}}_{\Sigma_j}) \right], \; j = 1, \ldots, k, \tag{3.12}$$
$$\textit{sujeito à } \mathcal{J}(\mathbf{q}_i) \in \mathcal{H}_i, i = 1, \ldots, P,$$

o comprimento da maior rota (circuito Dubins) presente em $\boldsymbol{\mathcal{T}}$*, e calculada como*

$$\mathcal{L}_\rho^j(\boldsymbol{\mathcal{Q}}_{\Sigma_j}) = \sum_{i=1}^{P-1} \mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{i+1}) + \mathcal{D}_\rho(\mathbf{q}_P, \mathbf{q}_1) \qquad \forall \mathbf{q}_i \in \tau_j, \tag{3.13}$$

onde $\mathcal{D}_\rho : SE(2) \times SE(2) \rightarrow \mathbb{R}^+$ *é a função que retorna o comprimento do menor caminho entre duas configurações de uma rota para um veículo Dubins com raio mínimo de curvatura* ρ*. Logo, a função objetivo consiste em minimizar o comprimento da maior rota Dubins:*

$$\arg\min_{\boldsymbol{\mathcal{Q}}} \; \mathcal{X}(\boldsymbol{\mathcal{T}}) \tag{3.14}$$
$$\textit{sujeito à Equação 3.11.}$$

Dessa forma, considerando as definições anteriores, pode-se definir o problema sendo abordado nesta etapa como encontrar circuitos hamiltonianos que passam por um conjunto de regiões para um grupo de veículos com restrições de curvatura modelados como veículos Dubins. O objetivo principal que se deseja alcançar é a redução do comprimento da maior rota, dessa forma distribuindo-se de maneira mais uniforme os pontos de visita entre os veículos. Como se considera que todos os veículos deslocam-se com velocidade constante, a redução da maior rota também produzirá uma redução no tempo total de visita das regiões.

Capítulo 4

Planejamento de Caminho para Um Veículo

Nesta seção é apresentada e discutida a metodologia proposta para o problema de planejamento de caminho de visita a regiões para o caso onde é utilizado apenas um veículo não-holonômico modelado como um veículo Dubins.

Primeiramente é abordado o caso onde todas as regiões são conhecidas e permanecem inalteradas durante a execução (caso estático). Em seguida, o problema é estendido para o caso onde após o início da execução pelo veículo do caminho inicial, novas regiões são inseridas no ambiente (caso dinâmico).

4.1 Caso Estático

Inicialmente é apresentado um algoritmo evolutivo, onde a posição e orientação dos pontos de visita, além da sequência de visita, são otimizados de uma forma combinada a cada iteração. Esse algoritmo possui como principal objetivo a análise do comportamento e principais características de boas soluções para o problema.

Em seguida, é proposta uma heurística determinística desenvolvida com base em determinadas características observadas nos melhores caminhos obtidos pelo algoritmo evolutivo.

4.1.1 Abordagem Evolutiva

A literatura a respeito dos Algoritmos Evolutivos (*Evolutionary Algorithms*, ou EAs) é bastante vasta, uma vez que existem diversos algoritmos com diferentes características. Entretanto, a ideia principal utilizada por todos esses métodos é a mesma:

dada uma determinada população de indivíduos, a pressão do ambiente demanda uma seleção natural (sobrevivência dos mais aptos) e consequentemente aumentando a aptidão (*fitness*) geral da população. Esses algoritmos são utilizados para resolver, principalmente, problemas que envolvem grandes e complexos espaços de busca baseando-se na simulação dos processos descritos pela Teoria da Evolução [Darwin, 1859].

Os EAs são compostos por duas etapas básicas: (i) etapa de variação, responsável por criar a diversidade necessária e dessa forma facilitar a inovação; (ii) etapa de seleção, responsável por escolher os melhores indivíduos e assim melhorar a qualidade da população [Eiben & Smith, 2007].

A abordagem evolutiva proposta inicialmente para abordar o DTSPN no caso estático é composta por três passos básicos durante o processo de otimização. Esses passos em conjunto têm o objetivo de solucionar ambas as partes (combinatória e contínua) do problema de uma maneira única. Cada um desses passos será melhor detalhado nas seções subsequentes, mas uma visão geral é apresentada a seguir:

1. **Atualização da posição**: Desloca o ponto de visita $\mathbf{q}_i$ dentro dos limites da região $\mathcal{H}_i$, dessa forma determinando o posicionamento que melhor irá contribuir para o processo de otimização descrito pela Equação 3.9;

2. **Atualização da orientação**: Modifica o valor de orientação ψ_i atribuído a cada ponto de visita $\mathbf{q}_i$, permitindo assim gerar caminhos Dubins menores conectando esses pontos;

3. **Atualização da sequência de visita**: Verifica se existe uma sequência de visita Σ dos pontos que descreve um circuito de menor comprimento do que o já existente.

O algoritmo proposto aqui utiliza para o processo de otimização uma técnica evolutiva dentre as mais simples encontradas na literatura, denominada Estratégia Evolutiva (1+1) [Schwefel, 1995]. As principais características dessa técnica são: (i) é mantido um único indivíduo na população ao longo das gerações, (ii) um novo indivíduo candidato é gerado por meio de um processo de mutação baseado em uma distribuição Gaussiana.

Os passos previamente descritos utilizados no processo de otimização são então agrupados sequencialmente e formam o Algoritmo 1.

A entrada fornecida para o algoritmo é composta por um conjunto de pontos de visita $\mathcal{P}$, onde cada ponto $\mathbf{p}_i$ está inicialmente localizado na posição central $\mathbf{n}_i$

Algoritmo 1 DTSPN-Evolutivo($\boldsymbol{\mathcal{P}}$)

$\Sigma, \Psi \leftarrow \text{AA}\ (\boldsymbol{\mathcal{P}})$
$\mathcal{L}_\rho^* \leftarrow \mathcal{L}_\rho(\boldsymbol{\mathcal{P}}_\Sigma, \Psi_\Sigma)$
while critério de parada não atendido **do**
    $\hat{\boldsymbol{\mathcal{P}}} \leftarrow \text{atualizaçãoPosição}(\boldsymbol{\mathcal{P}})$
    if $\mathcal{L}_\rho(\hat{\boldsymbol{\mathcal{P}}}_\Sigma, \Psi_\Sigma) < \mathcal{L}_\rho^*$ **then**
        $\boldsymbol{\mathcal{P}} \leftarrow \hat{\boldsymbol{\mathcal{P}}}$
        $\mathcal{L}_\rho^* \leftarrow \mathcal{L}_\rho(\hat{\boldsymbol{\mathcal{P}}}_\Sigma, \Psi_\Sigma)$
    end if
    $\hat{\Psi} \leftarrow \text{atualizaçãoOrientação}(\Psi)$
    if $\mathcal{L}_\rho(\boldsymbol{\mathcal{P}}_\Sigma, \hat{\Psi}_\Sigma) < \mathcal{L}_\rho^*$ **then**
        $\Psi \leftarrow \hat{\Psi}$
        $\mathcal{L}_\rho^* \leftarrow \mathcal{L}_\rho(\boldsymbol{\mathcal{P}}_\Sigma, \hat{\Psi}_\Sigma)$
    end if
    $\hat{\Sigma} \leftarrow \text{ATSP}(\mathcal{M})$
    if $\mathcal{L}_\rho(\boldsymbol{\mathcal{P}}_{\hat{\Sigma}}, \Psi_{\hat{\Sigma}}) < \mathcal{L}_\rho^*$ **then**
        $\Sigma \leftarrow \hat{\Sigma}$
        $\mathcal{L}_\rho^* \leftarrow \mathcal{L}_\rho(\boldsymbol{\mathcal{P}}_{\hat{\Sigma}}, \Psi_{\hat{\Sigma}})$
    end if
end while
return $\boldsymbol{\mathcal{P}}, \Psi, \Sigma$

de uma determinada região $\mathcal{H}_i$ de interesse, com $i = 1, \ldots, P$. O primeiro passo do algoritmo consiste em determinar a permutação (sequência de visita) Σ inicial, assim como o conjunto Ψ composto pelos ângulos de orientação em cada ponto de visita. Dessa forma, é possível determinar um circuito inicial que passa por todas as regiões e é realizável pelo veículo.

A sequência de visita e as orientações são calculadas utilizando-se uma técnica clássica denominada Algoritmo Alternante (*Alternating Algorithm*, ou AA) [Savla et al., 2005b] (Algoritmo 2), que consiste em uma heurística bem simples. Em suma, é construído um circuito de forma que pares consecutivos de pontos de visita são ligados utilizando-se segmentos de reta. Baseando-se nas orientações previamente atribuídas (após a ligação dos pontos por segmentos de reta), as demais ligações (entre os pares) são obtidas utilizando-se curvas de Dubins.

A função $\text{dir}(\mathbf{p}_m, \mathbf{p}_n)$ é definida como a direção (argumento) do vetor unitário a partir de um determinado ponto de visita para outro, ou seja

$$\text{dir}(\mathbf{p}_m, \mathbf{p}_n) \triangleq \arg\left(\frac{\mathbf{p}_n - \mathbf{p}_m}{\|\mathbf{p}_n - \mathbf{p}_m\|}\right). \tag{4.1}$$

Algoritmo 2 Algoritmo Alternante($\mathcal{P}$) [Savla et al., 2005b]

$\Sigma \leftarrow$ ETSP($\mathcal{P}$)
$\psi_1 \leftarrow \text{dir}(\mathbf{p}_1, \mathbf{p}_2)$
for $i = 2$ to $P - 1$ **do**
    if i é par **then**
        $\psi_i \leftarrow \psi_{i-1}$
    else
        $\psi_i \leftarrow \text{dir}(\mathbf{p}_i, \mathbf{p}_{i+1})$
    end if
end for
if P é par **then**
    $\psi_P \leftarrow \psi_{P-1}$
else
    $\psi_P \leftarrow \text{dir}(\mathbf{p}_P, \mathbf{p}_i)$
end if
return Σ, Ψ

A função de custo sendo otimizada corresponde ao comprimento total do circuito $\mathcal{L}_\rho$ formado por curvas de Dubins e calculado de acordo com a Equação 3.10.

Enquanto um certo critério de parada não é atendido, o Algoritmo 1 itera executando os três passos básicos previamente mencionados, otimizando os conjuntos contendo os valores de posição ($\mathcal{P}$) e orientação (Ψ) dos pontos de visita, além da sequência de visita (Σ). A variável $\mathcal{L}_\rho^*$ é utilizada para representar o melhor caminho (mais curto) encontrado até a atual iteração.

É importante destacar que, neste trabalho, é proposta uma abordagem evolutiva diferente dos métodos comumente encontrados na literatura. Conforme pode ser observado, a avaliação de uma nova solução candidata é realizada após cada passo do processo de otimização ser executado, e não apenas no momento final após a geração completa de uma nova população. Dessa forma, se uma solução candidata for melhor que a melhor solução encontrada até o momento, as alterações realizadas que propiciaram essa melhora já serão incorporadas à melhor solução. A razão fundamental para essa abordagem é evitar a *perda* de uma iteração completa nos casos em que uma solução candidata pode se tornar pior como resultado de apenas um dos passos (não importando a influência positiva dos demais passos). Essa decisão é muito importante, uma vez que cada passo que compõe o algoritmo pode promover um grande impacto no resultado final.

Nas próximas seções serão descritos, em detalhes, os passos básicos da abordagem proposta e apresentada no Algoritmo 1.

4.1.1.1 Atualização da Posição

O principal objetivo deste passo no processo de otimização é alterar a posição dos pontos de visita, movimentando-os dentro de suas próprias regiões. Dessa forma, é possível encontrar o posicionamento que melhor contribuirá para a redução do comprimento total do caminho.

De acordo com o exemplo apresentado na Figura 2.3(b), é possível inferir que uma boa solução candidata para o TSPN tem uma grande chance de possuir os pontos de visita na borda das regiões. Além disso, esses pontos devem estar mais próximos uns dos outros (de forma a reduzir o comprimento do caminho que os liga), o que os levaria também a um maior proximidade de um ponto central (centroide) de todas as regiões.

Considerando a suposição acima apresentada, é proposta uma heurística para melhorar a solução inicial que posiciona o ponto de visita no centro da região. Em cada iteração do algoritmo, cada ponto $\mathbf{p}_i$ irá se deslocar em direção a uma área central relativa a todas as regiões presentes no conjunto $\boldsymbol{\mathcal{H}}$. A nova posição $\hat{\mathbf{p}}_i$ de um ponto de visita será calculada de acordo com a Equação 4.2:

$$\hat{\mathbf{p}}_i = \begin{cases} \mathbf{p}_{\text{aux}} & \text{, se } \|\mathbf{p}_{\text{aux}} - \mathbf{n}_i\| \leq r_i \\ \mathbf{p}_i & \text{, caso contrário.} \end{cases} \tag{4.2}$$

O valor de $\mathbf{p}_{\text{aux}}$ é dado por

$$\mathbf{p}_{\text{aux}} = \mathbf{p}_i + \begin{bmatrix} |\varrho| \cos(\gamma_i) \\ |\varrho| \sin(\gamma_i) \end{bmatrix}, \tag{4.3}$$

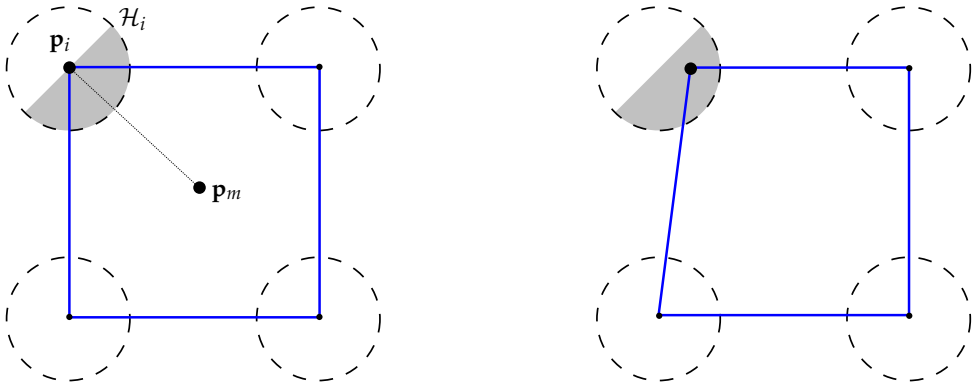
onde ϱ é uma variável aleatória obtida a partir de uma distribuição normal com média μ e variância σ^2, e γ_i determina a direção do ponto de visita, e é obtida a partir de

$$\gamma_i = \text{dir}(\mathbf{n}_i, \mathbf{p}_m) \; + \; \delta, \tag{4.4}$$

com $\delta \in [-\pi/2, \pi/2]$ sendo uma variável aleatória uniformemente distribuída. O ponto $\mathbf{n}_i$ representa o ponto central da região $\mathcal{H}_i$, e $\mathbf{p}_m$ é a posição do centroide considerando-se os pontos centrais das regiões.

A Figura 4.1 exemplifica esta etapa apresentando uma instância de exemplo no início da execução ($\mathbf{p}_i = \mathbf{n}_i$) e ao final. Os círculos pontilhados representam as regiões que devem ser visitadas, o ponto no centro da imagem corresponde ao

centroide. A seta indica a direção base de movimento (em direção ao centroide, obtida por dir$(\mathbf{n}_i, \mathbf{p}_m)$). A direção final de movimento, γ_i, é definida apenas após a incorporação de δ. A área em cinza representa o espaço de busca pelo qual o ponto de visita pode ser movimentado na busca por uma solução melhor (caminho mais curto).



(a) Antes (b) Depois

Figura 4.1. Atualização da posição: a posição de ponto de visita $\mathbf{p}_i$ pertecente à uma região $\mathcal{H}_i$ é variada deslocando o ponto em direção à área central interna ao conjunto de regiões $\mathcal{H}$. A área destacada em cinza representa o espaço de busca disponível para a movimentação do ponto $\mathbf{p}_i$.

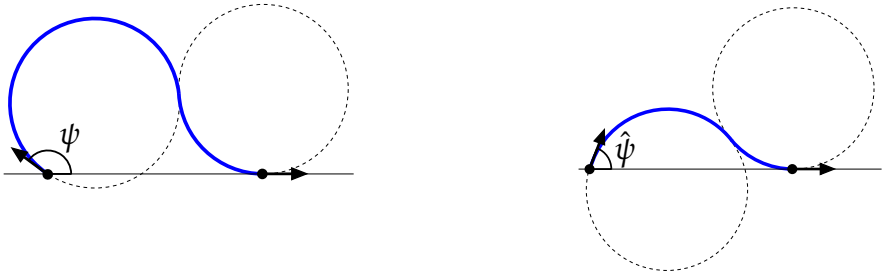
4.1.1.2 Atualização da Orientação

Conforme já mencionado em seções anteriores, o primeiro passo do algoritmo utiliza o Algoritmo Alternante como forma de definir as orientações iniciais relativas a cada ponto de visita. Entretanto, as orientações atribuídas usualmente se encontram distantes de valores que permitiriam encontrar o caminho mínimo pelo dado conjunto de pontos.

Esse fato é muito importante, especialmente ao lidar-se com veículos que possuem restrições de curvatura intrínsecas ao seu movimento. Uma pequena variação no valor da orientação pode ter um grande impacto no comprimento final do caminho, conforme pode ser observado no exemplo apresentado na Figura 4.2.

Dessa forma, o principal objetivo durante essa etapa do algoritmo é atualizar os ângulos de orientação inicialmente obtidos a partir do AA, reduzindo o comprimento das curvas entre os pontos, e consequentemente o comprimento total do caminho. Cada ângulo de orientação é modificado de acordo com:

$$\hat{\psi}_i = \psi_i + \varrho, \tag{4.5}$$



(a) Antes (b) Depois

Figura 4.2. Atualização da orientação: exemplo do impacto do ângulo de orientação no comprimento do caminho gerado utilizando-se uma curva Dubins.

onde ϱ corresponde a uma variável aleatória obtida a partir de distribuição normal com média μ e variância σ^2.

4.1.1.3 Atualização da Sequência de Visita

Apesar de a sequência obtida utilizando-se a métrica euclidiana durante a execução do ETSP internamente ao AA representar uma boa estimativa inicial, essa sequência não necessariamente produzirá bons resultados ao se considerar a métrica que utiliza curvas de Dubins para geração do caminho.

O objetivo deste último passo é determinar a melhor sequência atual de visita considerando-se as posições ($\hat{\mathcal{P}}$) e orientações ($\hat{\Psi}$) dos pontos de visita decorrentes das modificações que podem ter sido feitas nos passos anteriores.

A sequência da visita também desempenha um papel importante no processo de otimização. Apesar da distância a ser percorrida entre dois pontos (sem orientação) manter-se a mesma não importando o ponto inicial (considerando-se a métrica euclidiana), o mesmo fato não necessariamente será válido para o caso de pontos que possuem orientação. A Figura 4.3 apresenta um exemplo com as curvas de Dubins geradas para um mesmo par de pontos de visita, porém com sequências diferentes de visita $\mathbf{q}_i \rightarrow \mathbf{q}_j$ e $\mathbf{q}_j \rightarrow \mathbf{q}_i$.

Assim, este passo da abordagem será implementado fazendo-se uso do TSP Assimétrico (*Asymmetric TSP*, ou ATSP), técnica especialmente desenvolvida para casos em que o custo das ligações entre dois pontos é assimétrico, ou seja, a ordem de visita importa. O ATSP já foi previamente aplicado ao problema envolvendo custos relativos a curvas de Dubins. Entretanto, em nenhum caso o método foi utilizado em um contexto de otimização conforme propomos aqui, onde todas as variáveis envolvidas (posição, orientação e sequência de visita) são otimizadas em conjunto.

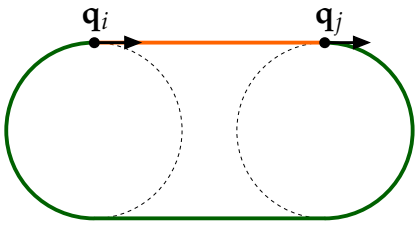


Figura 4.3. Atualização da sequência de visita: exemplo das curvas de Dubins geradas para um mesmo par de pontos, porém com sequências diferentes de visita $\mathbf{q}_i \rightarrow \mathbf{q}_j$ e $\mathbf{q}_j \rightarrow \mathbf{q}_i$.

Este passo é solucionado a partir da criação de uma matriz de custos $\mathcal{M}$ preenchida com os valores das $N(N-1)$ curvas de Dubins necessárias para ligar todos os pares de configurações. Recordando que nesse caso específico temos $P = N$. Em seguida, essa matriz de custos pode ser utilizada como entrada para qualquer resolvedor do ATSP.

4.1.1.4 Análise de Complexidade

Nesta seção é apresentada a análise da complexidade computacional do algoritmo evolutivo proposto para solução do DTSPN. Inicialmente é abordado o comportamento assintótico relativo ao tempo de execução do algoritmo, em seguida, é discutido o limite superior para o comprimento do caminho.

Os dois primeiros passos da metodologia proposta (atualização da posição e orientação) possuem tempo de execução de ordem linear $O(n)$, onde n representa o número de pontos de visita. É importante ressaltar que neste caso o número de pontos de visita é igual ao número de regiões ($P = N$). O terceiro passo é o mais custoso em relação a tempo computacional, uma vez que na primeira etapa o cálculo da matriz de custos é feito em $O(n^2)$. A complexidade para encontrar-se uma solução exata para o ATSP é exponencial, uma vez que existem $(n-1)!$ possíveis rotas. Entretanto, considerando-se possíveis heurísticas temos que o custo é $O(n^3)$ no pior caso [Frieze et al., 1982] (versão modificada para o algoritmo de Christofides [Christofides, 1972]).

O critério de parada utilizado no algoritmo foi determinado como um valor constante β de gerações (iterações), não influenciando o custo computacional, concluindo-se então que o método terá complexidade final $O(n^3)$ (considerando a utilização de uma heurística para resolução do ATSP).

O comprimento do caminho gerado pela abordagem apresentada possui o mesmo limite superior do caminho resultante obtido pelo Algoritmo Alternante, uma vez que esse algoritmo é utilizado na etapa de inicialização do método e o

resultado atual só é atualizado caso uma solução candidata melhor seja encontrada.

De acordo com o Teorema 3.4 de [Savla et al., 2008], o comprimento de uma curva Dubins ligando duas configurações possui um comprimento máximo de

$$\mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_j) \leq \|\mathbf{p}_i - \mathbf{p}_j\| + \mathsf{k}\rho\pi, \tag{4.6}$$

onde $\mathsf{k} \in [2,657\ ,\ 2,658]$. Considerando esse limite aplicado a cada par de pontos, é imediato observar que um limite superior não firme do comprimento do caminho considerando a entrada do problema (regiões de interesse) é

$$\mathcal{L}_\rho^{\text{evolutivo}} \leq \text{ETSP}(\boldsymbol{\mathcal{P}}_{\text{cent}}) + \left\lceil \frac{N}{2} \right\rceil \mathsf{k}\rho\pi, \tag{4.7}$$

onde $\boldsymbol{\mathcal{P}}_{\text{cent}}$ representa a posição inicial dos pontos de visita (no centro das regiões), e considerando-se o caso em que $N > 2$ e $\rho > 0$. Como a orientação inicial é determinada pelo Algoritmo Alternante, metade dos caminhos conectando dois pontos será um segmento de reta (primeiro termo da equação) e a outra metade será formada por curvas de Dubins (segundo termo da equação). A solução inicial representa o limite superior, já que uma nova solução será aceita apenas se for melhor que a melhor solução encontrada até o momento.

4.1.2 Heurística Determinística

Conforme visto na seção anterior, os algoritmos evolutivos são inspirados nos mecanismos de seleção natural encontrados na natureza. Dessa forma, é possível observar determinadas características que auxiliam na busca de boas soluções. Essas características são decorrentes do processo evolutivo que ocorre a cada iteração.

Dentre as características observadas que permitiriam a obtenção de melhores soluções pode-se citar (i) a utilização de pontos de visita nas regiões de interseção, uma vez que foi observada uma grande quantidade de áreas sobrepostas; (ii) não necessariamente o deslocamento de todos os pontos de visita em direção ao centroide das regiões irá beneficiar na redução do caminho e (iii) o caminho obtido pode ser reduzido caso a orientação do ponto de visita seja atribuída de forma com que ele esteja no centro da curva necessária para se ir até o ponto de visita seguinte.

A Figura 4.4 apresenta um exemplo de solução obtido a partir do algoritmo evolutivo onde todas as características mencionadas podem ser observadas.

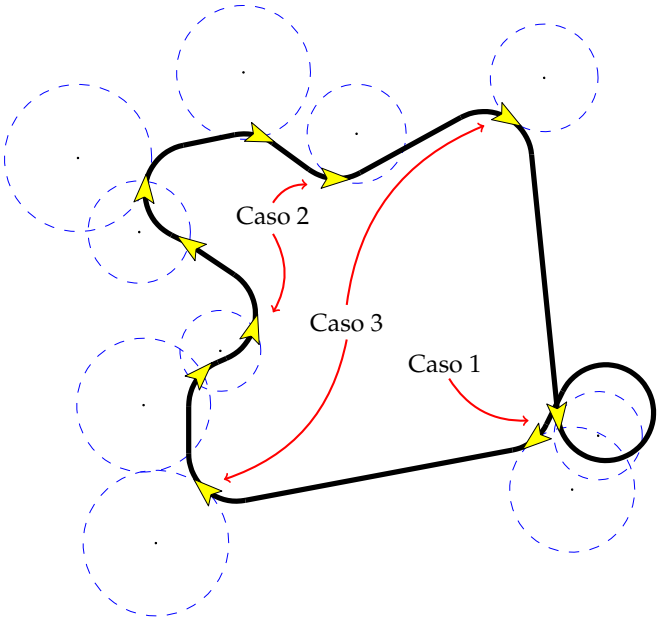


Figura 4.4. Características observadas nas soluções do algoritmo evolutivo que permitiriam a obtenção de melhores soluções. Caso 1: melhor utilização das regiões de interseção; Caso 2: não deslocar os pontos unicamente em direção ao centroide e Caso 3: atribuição de orientação no centro da curva.

Além disso, apesar do método evolutivo apresentar bons resultados na média, não existem garantias que em determinada execução uma boa solução será encontrada, uma vez que se trata de um método probabilístico. Dessa forma, nesta seção é apresentada uma heurística determinística desenvolvida com base em determinadas características observadas nos melhores caminhos obtidos pelo algoritmo evolutivo previamente apresentado.

4.1.2.1 Determinação dos Pontos de Visita

O primeiro passo de nossa metodologia consiste em determinar os pontos que serão utilizados como possíveis posições que o robô deverá alcançar. Dessa forma, a quantidade de informação sendo manipulada é reduzida, reduzindo o tempo real de execução (não a complexidade) de algoritmos que dependem da quantidade de pontos de visita.

Apesar de a tarefa de identificação de regiões em comum ser matematicamente simples, a seleção da região que será utilizada não é. Como forma de exemplificar esse problema, uma possível situação conflitante é apresentada na Figura 4.5. Conforme pode ser observado, as possíveis escolhas de pontos de visita $\mathcal{P} = \langle \mathbf{p}_1 \in (\mathcal{H}_1 \cap \mathcal{H}_2), \mathbf{p}_2 \in (\mathcal{H}_3 \cap \mathcal{H}_4) \rangle$ ou $\mathcal{P}' = \langle \mathbf{p}'_1 \in (\mathcal{H}_1 \cap \mathcal{H}_3), \mathbf{p}'_2 \in (\mathcal{H}_2 \cap \mathcal{H}_4) \rangle$ seriam aceitáveis.

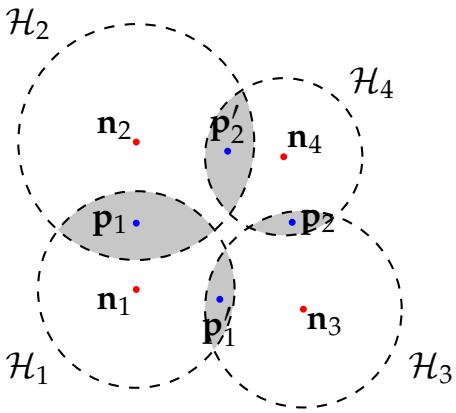


Figura 4.5. Instância que demonstra o caso em que existem múltiplas soluções para a escolha de pontos de visita em regiões em comum. Para o exemplo apresentado poderia-se escolher $(\mathbf{p}_1, \mathbf{p}_2)$ ou $(\mathbf{p}'_1, \mathbf{p}'_2)$.

Para solução desse problema é proposta uma heurística gulosa baseada em dois critérios: (i) Regiões que são comuns para o maior número de pontos de interesse são melhores; e (ii) regiões maiores são melhores. Regiões que atendem um maior número de pontos de interesse são interessantes pois permitem uma maior simplificação do problema inicial, uma vez que irão representar mais pontos. A preferência por regiões de maior área se deve ao fato de essas possibilitarem uma maior flexibilidade da posição do ponto de visita em seu interior. Baseando-se nesses critérios, o Algoritmo 3 apresenta os passos executados para a determinação dos pontos de visita.

Algoritmo 3 Determinação da Posição dos Pontos de Visita($\boldsymbol{\mathcal{H}}$)

1: $\boldsymbol{\mathcal{P}} \leftarrow \emptyset$
2: $\boldsymbol{\mathcal{R}} \leftarrow$ Criar o conjunto de áreas em comum a partir de $\boldsymbol{\mathcal{H}}$
3: **while** $\boldsymbol{\mathcal{H}}$ possuir regiões sem ponto de visita **do**
4: $\quad \mathcal{R} \leftarrow$ Selecionar a região em $\boldsymbol{\mathcal{R}}$ que atende ao maior número de pontos de interesse simultaneamente e que possui a maior área
5: $\quad \boldsymbol{\mathcal{P}} \leftarrow \boldsymbol{\mathcal{P}} \bigcup$ Novo ponto de visita relativo a $\mathcal{R}$
6: $\quad$ Marcar que foi atribuído um ponto de visita à cada região de $\mathcal{R}$
7: $\quad$ Remover de $\boldsymbol{\mathcal{R}}$ a área em comum $\mathcal{R}$
8: **end while**
9: **return** $\boldsymbol{\mathcal{P}}$

A partir do cálculo inicial de todas as áreas que pertencem a mais de uma região simultaneamente, o algoritmo itera selecionando aquelas em comum e que atendem ao maior número de pontos e que possuem a maior área. Determinada a região sendo manipulada na iteração atual, o próximo passo consiste em criar o novo ponto de visita que irá atender às regiões pertencentes à área de interseção (melhor detalhado abaixo). Em seguida, as regiões que compõem a região manipulada são marcadas como possuidoras de um ponto de visita e a essa então é retirada do conjunto de regiões em comum.

A partir do Algoritmo 3, fica claro que qualquer escolha de pontos de visita $\boldsymbol{\mathcal{P}} = \text{permutação}(\mathbf{p}_1, \ldots, \mathbf{p}_P)$, onde $\mathbf{p}_p \in \mathcal{R}_r$, é aceitável, no sentido em que todas regiões devem ser atendidas por um determinado ponto de visita.

Apesar de bem simples, esse algoritmo apresenta algumas dificuldades de caráter prático se abordado de forma puramente analítica sendo (i) como identificar a região em comum que atende à maior quantidade de pontos ao mesmo tempo e (ii) como avaliar o tamanho de uma determinada região em comum. Para lidar com os problemas mencionados, em nossa implementação adotamos uma solução simples inspirada em algoritmos clássicos de rasterização utilizados em Computação Gráfica. Os seguintes passos descrevem nossa abordagem:

1. O espaço bidimensional é representado por um *grid*, onde cada célula $c_{x,y}$ está relacionada a uma lista de pontos, e todas as células são inicializadas com $c_{x,y} \leftarrow \emptyset$;

2. Para cada ponto $\mathbf{n}_i$, todas as células que estiverem na região $\mathcal{H}_i$ associada ao ponto são atualizadas para $c_{x,y} \leftarrow c_{x,y} \cup \{\mathbf{n}_i\}$.

Após o *grid* ter sido completamente preenchido, os problemas previamente mencionados podem ser resolvidos mais facilmente: (i) As regiões que cobrem o maior número de pontos são aquelas cujo agrupamento de células possui o máximo $|c_{x,y}|$ e é relativo ao mesmo conjunto de nós $c_{x,y}$; (ii) a região com maior área é aquela composta pelo maior número de células (uma vez que essa quantidade é proporcional à área da região). É importante ressaltar que todos os cálculos envolvidos são aproximados (devido aos erros causados pela amostragem no *grid*), entretanto, diversos experimentos realizados mostraram que os resultados são consideravelmente precisos se for selecionado um tamanho de *grid* significativamente menor que o menor raio $r_{\min}$ dentre todas as regiões (por exemplo, $r_{\min}/50$).

Dessa forma, retornando à definição do Algoritmo 3, a posição de um novo ponto de visita pode ser determinado de duas formas: (i) Caso uma região não

possua interseção com nenhuma outra região no ambiente, o ponto de visita será posicionado no centro da região. (ii) Em caso de interseção, será selecionado um ponto central na área em comum de acordo com a média das coordenadas das células correspondentes.

4.1.2.2 Otimização do Posicionamento dos Pontos de Visita

De maneira similar ao passo de atualização da posição utilizado no algoritmo evolutivo, esta etapa também possui como objetivo determinar o melhor posicionamento do ponto de visita de forma a resultar em um caminho final menor.

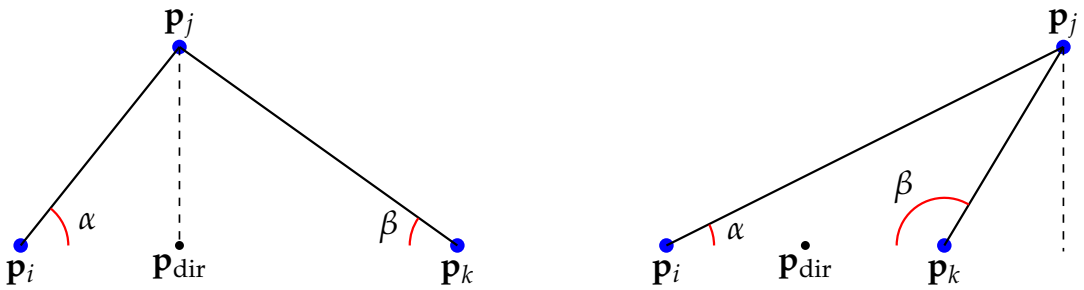
Apesar da heurística proposta na Seção 4.1.1.1 (deslocamento dos pontos de visita em direção ao centroide das regiões) apresentar bons resultados, observou-se que em determinados casos, dependendo da disposição dos pontos vizinhos, o ponto de visita deveria deslocar-se no sentido oposto para que houvesse efetivamente uma melhoria no resultado final.

Assim, uma questão fundamental para obtenção de melhores resultados no posicionamento dos pontos de visita está relacionada à determinação da direção com que o ponto deverá se movimentar. Para isso, é proposto um método que determina a direção de visita baseada na posição dos pontos vizinhos ao ponto que será deslocado. Dessa forma, considerando-se os três pontos envolvidos, realiza-se uma avaliação do triângulo formado por esses pontos, e a escolha da direção ocorrerá de acordo com dois casos distintos. (i) Caso a projeção da altura (o ponto que será movimentado) seja interior ao triângulo, a altura representa a menor distância a ser percorrida para que o três pontos sejam colineares, representando o menor caminho contendo os pontos. (ii) Caso a projeção da altura seja externa ao triângulo, o ponto deve deslocar-se em direção ao centro do triângulo, reduzindo ambos os segmentos (a base não sofre alteração) de forma igualitária.

A Figura 4.6 exemplifica o método. Para quaisquer três pontos de visita i, j e k, com coordenadas $\mathbf{p}_i$, $\mathbf{p}_j$ e $\mathbf{p}_k$, respectivamente, o objetivo é deslocar o ponto $\mathbf{p}_j$ de maneira a reduzir a distância entre ele e os pontos vizinhos.

Conforme mencionado anteriormente, a posição alvo que será utilizada no cálculo da orientação de movimento depende da disposição do ponto sendo avaliado em relação aos pontos vizinhos. Esse ponto será escolhido de acordo com:

$$\mathbf{p}_{\text{dir}} = \begin{cases} \text{Projeção de } \mathbf{p}_j \text{ sobre } \overline{\mathbf{p}_i\mathbf{p}_k} & \text{, se } \alpha, \beta \leq 90^\circ \\ \text{Ponto médio entre } \mathbf{p}_i \text{ e } \mathbf{p}_k & \text{, caso contrário.} \end{cases} \tag{4.8}$$



(a) Projeção da altura (b) Ponto médio

Figura 4.6. Técnica utilizada para determinação da direção de movimento para otimização do ponto de visita. (a) Caso a projeção da altura do triângulo formado pelos pontos vizinhos $\mathbf{p}_i, \mathbf{p}_j$ e $\mathbf{p}_k$ esteja sobre o segmento $\overline{\mathbf{p}_i\mathbf{p}_k}$, a direção de movimento aponta para o ponto $\mathbf{p}_{\text{dir}}$ relativo à projeção. (b) Caso a projeção seja externa ao triângulo, a direção de movimento aponta para o ponto médio $\mathbf{p}_{\text{dir}}$ do segmento $\overline{\mathbf{p}_i\mathbf{p}_k}$.

Logo, de maneira mais formal, a direção de movimento é dada por:

$$\eta = \text{dir}(\mathbf{p}_j, \mathbf{p}_{\text{dir}}). \tag{4.9}$$

Dessa forma, é apresentado o Algoritmo 4, que consiste em um algoritmo iterativo de otimização local. Cada ponto é avaliado e deslocado (caso necessário) de forma a minimizar a distância entre os vizinhos, e como consequência, também reduzindo o comprimento total do circuito.

Algoritmo 4 Heurística de Otimização da Posição dos Pontos de Visita($\mathcal{P}$)

```
1:  P_Σ ← ETSP(P)
2:  for i = 1 to P do
3:      η ← Determinar direção de movimento(p_{i−1}, p_i, p_{i+1})
4:      loop
5:          p̂_i ← Mover p_i de acordo com a direção η
6:          if Reduziu perímetro and Dentro dos limites das regiões then
7:              p_i ← p̂_i
8:          else
9:              break
10:         end if
11:     end loop
12: end for
13: return P̂_Σ
```

Inicialmente os pontos são ordenados de acordo com ETSP, e para cada ponto de visita (a partir de um ponto inicial qualquer) é determinada a melhor direção de

movimento (η). Em seguida, inicia-se um processo iterativo movendo-se o ponto nessa direção de acordo com um pequeno deslocamento (valor de passo). Caso a distância para os vizinhos (aqui denominada perímetro) tenha sido reduzida, e a nova posição seja válida (ou seja, continua atendendo às regiões atribuídas ao ponto de visita), essa nova posição é salva e uma nova iteração é iniciada. O algoritmo itera até o momento em que uma das condições não for atendida. Para simplificar a notação, uma vez que a sequência de pontos forma um circuito, assumimos que $\mathbf{p}_m \equiv \mathbf{p}_n$ se $m \equiv n \pmod{P}$.

Além de cumprir o objetivo principal, que consiste na redução do comprimento do circuito, essa heurística também beneficiará a etapa da geração efetiva do caminho de Dubins, apesar das orientações nos pontos ainda não serem tratadas, como será discutido a seguir.

Ao se utilizar veículos que possuem inércia não desprezível, o custo para se realizar rotações (curvas) pode ser maior que o custo relativo ao movimento de translação. Esse fato motivou [Aggarwal et al., 1997] a propor uma generalização para o TSP denominado TSPA. No caso do TSPA, além de se minimizar o comprimento do caminho, o custo total angular também deve ser minimizado. Esse custo angular é obtido somando-se as diferenças angulares entre segmentos de entrada e saída de um ponto.

Para quaisquer três pontos de visita i, j e k, com coordenadas $\mathbf{p}_i$, $\mathbf{p}_j$ e $\mathbf{p}_k$, respectivamente. Definimos a deflexão angular de i para k através de j, como o ângulo $\varphi_{i,j,k}$ entre os segmentos de reta $\overline{\mathbf{p}_i\mathbf{p}_j}$ e $\overline{\mathbf{p}_j\mathbf{p}_k}$ (Figura 4.7).

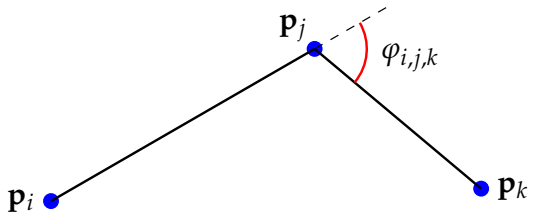


Figura 4.7. Representação do desvio angular $\varphi_{i,j,k}$ existente no caminho partindo do ponto de visita inicial $\mathbf{p}_i$ até o ponto final $\mathbf{p}_k$ passando por $\mathbf{p}_j$.

A escolha do circuito com menor custo angular como ferramenta auxiliar para a geração de caminhos Dubins mais curtos foi utilizada em [Medeiros & Urrutia, 2010]. Nesse trabalho é proposto um método baseado em otimização inteira onde a função de custo final é composta por um termo relativo à soma dos custos individuais de desvio angular e um termo referente ao custo euclidiano do TSP tradicional. Entretanto, por ser tratar de um método exato, sua utilização torna-se viável apenas para instâncias com poucos pontos.

Dessa forma, conforme pode ser observado, o método aqui proposto, além de reduzir o comprimento do circuito, também obterá circuitos com custo angular menor, uma vez que em cada iteração ocorre uma "linearização" das partes do circuito, removendo possíveis curvas. Assim, utilizando-se essa técnica, será possível encontrar uma sequência mais propícia para a geração de caminhos de Dubins menores, uma vez que sequências de pontos mais colineares (retilíneos) tendem a gerar caminhos Dubins mais curtos.

4.1.2.3 Atribuição das Orientações e Geração do Caminho

Após os pontos de visita e a sequência de visita terem sido determinados, o próximo passo consiste em atribuir uma certa orientação ψ_p à cada ponto p. A orientação de um determinado ponto i será denominado aqui de ψ_i, e o conjunto de orientações de todos os pontos de $\Psi = \langle \psi_1, \ldots, \psi_P \rangle$. Inicialmente, é apresentada uma pequena extensão para o método clássico encontrado na literatura, em seguida, é proposto um novo algoritmo para a determinação das orientações.

Conforme apresentado anteriormente, a primeira técnica proposta na literatura (e ainda uma das mais utilizadas) que abordam este problema é denominada Algoritmo Alternante [Savla et al., 2005b]. Trata-se de uma heurística bem simples, onde cada par de pontos da sequência é conectado utilizando-se segmentos de reta ou curvas de Dubins (Algoritmo 2). De forma simplificada, a orientação ψ_p de um determinado ponto de visita p, é determinada por:

$$\psi_p = \begin{cases} \text{dir}(\mathbf{p}_p, \mathbf{p}_{p+1}) & \text{, se } p \text{ é ímpar} \\ \text{dir}(\mathbf{p}_{p-1}, \mathbf{p}_p) & \text{, se } p \text{ é par} \end{cases} \quad 1 \leq p \leq P. \tag{4.10}$$

Entretanto, um dos principais problemas relacionados a essa técnica é que o circuito resultante, e consequentemente o comprimento total, depende fortemente da seleção do ponto inicial de execução do algoritmo. Por exemplo, $\mathcal{P}_\Sigma = \langle \mathbf{p}_1, \ldots, \mathbf{p}_P \rangle$ e $\mathcal{P}_{\Sigma'} = \langle \mathbf{p}_2, \ldots, \mathbf{p}_P, \mathbf{p}_1 \rangle$ representam exatamente o mesmo circuito, entretanto, dependendo da escolha inicial do ponto, os segmentos de reta e as curvas de Dubins serão alocados em posições diferentes. Além disso, se o número de pontos P for ímpar, então a ordem reversa do circuito (*i.e.*, $\mathcal{P}_\Sigma = \langle \mathbf{p}_P, \mathbf{p}_{P-1}, \ldots, \mathbf{p}_1 \rangle$) também resultará em uma solução distinta.

Um esforço inicial para se abordar esse problema foi apresentado em [Medeiros & Urrutia, 2009], onde se propôs gerar o caminho utilizando o AA variando o ponto inicial ao longo de todo o circuito. Entretanto, caso o número de pontos

seja ímpar, a direção em que o circuito é percorrido também terá influência no comprimento do caminho. Dessa forma, estendendo essa ideia para o caso geral, formalizamos abaixo a técnica que iremos denominar de Algoritmo Alternante Ótimo Generalizado (AAOG).

- P é *par*: Nesse caso, existem apenas duas possibilidades de circuitos distintos: (i) quando os segmentos de reta são alocados tendo como posição inicial um ponto em uma posição ímpar (a proposição original apresentada na Equação 4.10), e (ii) quando os segmentos de reta são iniciados em pontos de posição par;

- P é *ímpar*: Nesse caso, existirá um determinado ponto que não pertence à posição final de nenhum segmento de reta (uma vez que eles ligam pares de pontos adjacentes). Como esse ponto pode ser qualquer um dos P pontos, existem P soluções possíveis. Além disso, de acordo com a Equação 4.10, esse ponto "pendente" será orientado em direção ao vizinho seguinte. Logo, realizando-se uma análise do circuito com os pontos em ordem inversa (Figura 4.8), o ponto pendente não estará mais orientado em direção ao vizinho anterior, o que corresponderá a mais P soluções, resultado em um total de $2P$ possíveis soluções (circuitos distintos).

Todas as soluções devem ser analisadas, a que resultar na geração do menor caminho será escolhida como a solução final do método.

Como pode ser observado, o AA não utiliza nenhuma informação extra a respeito do circuito para melhorar o caminho resultante. É interessante considerar um método que utilize determinada característica do circuito de forma a atribuir orientações aos pontos que de alguma forma podem beneficiar a geração do caminho.

Uma vez que o caminho em si vai ser construído utilizando-se curvas de Dubins, a ideia principal do algoritmo aqui proposto possui dois objetivos fundamentais: (i) curvas do tipo CCC (caso curto) devem ser evitadas; e (ii) um determinado ponto de visita p deve estar preferencialmente alocado no meio do arco C que compõe a curva proveniente do ponto de visita anterior $(p-1)$ e vai em direção ao próximo ponto $(p+1)$.

Para se reduzir o número de curvas do caso curto no circuito, é interessante observar que esse tipo de curva ocorrerá no caso Dubins quando a distância euclidiana entre dois pontos de visita adjacentes for inferior à 2ρ. Apesar de não se poder evitar totalmente o caso curto em determinadas circunstâncias geométricas, neste trabalho propomos uma abordagem simples como forma de reduzir a utilização desse tipo

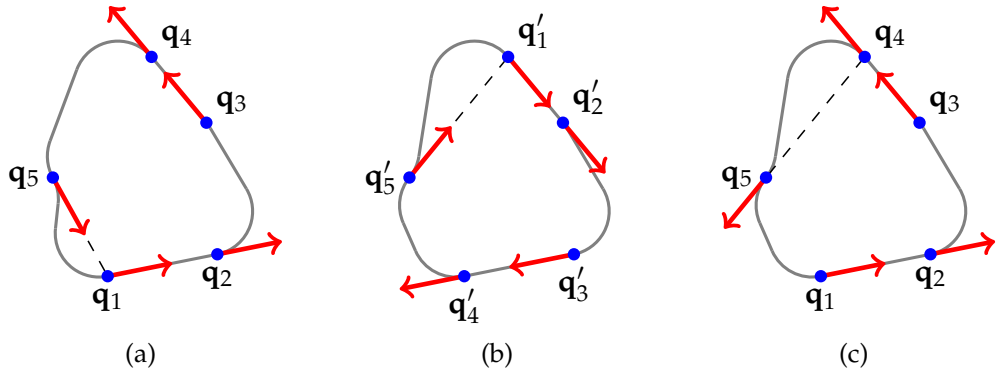


Figura 4.8. Exemplo dos diferentes caminhos de Dubins obtidos pelo Algoritmo Alternante para uma mesma instância. (a) Solução original: o último ("pendente") ponto de visita ($\mathbf{q}_5$) possuirá orientação em direção ao ponto inicial do circuito ($\mathbf{q}_1$); (b) Circuito reverso (ponto inicial escolhido de forma a se manter o mesmo ponto pendente, $\mathbf{q}'_5$, da situação anterior): Apesar das partes do circuito formadas por segmentos de reta e demais curvas intermediárias permanecerem iguais, a mudança de orientação no último ponto muda por completo o circuito na sua vizinhança; (c) Circuito reverso com orientações invertidas: Será equivalente ao circuito inicial, entretanto, nesse caso o ponto $\mathbf{q}_5$ encontra-se alinhado com o vizinho anterior e com orientação na direção oposta.

de curvas no caminho. Pontos adjacentes, cuja distância relativa for menor ou igual a 2ρ, devem ser conectados utilizando-se segmentos de reta, iniciando pelos pontos mais próximos. As orientações dos pontos restantes são definidas como o valor da orientação do vetor unitário médio dado por:

$$\psi_p = \arg\left(\frac{\mathbf{v}_1 + \mathbf{v}_2}{\|\mathbf{v}_1 + \mathbf{v}_2\|}\right), \tag{4.11}$$

onde $\mathbf{v}_1$ e $\mathbf{v}_2$ são os vetores formados com o ponto vizinho anterior e seguinte (Figura 4.9). Os vetores vizinhos são calculados de acordo com:

$$\mathbf{v}_1 = \frac{\mathbf{p}_p - \mathbf{p}_{p-1}}{\|\mathbf{q}_p - \mathbf{p}_{p-1}\|} \qquad \text{e} \qquad \mathbf{v}_2 = \frac{\mathbf{p}_{p+1} - \mathbf{p}_p}{\|\mathbf{p}_{p+1} - \mathbf{p}_p\|}. \tag{4.12}$$

A Figura 4.10 ilustra a execução das etapas do algoritmo para uma instância exemplo. A formalização do método é apresentada no Algoritmo 5.

De forma semelhante à utilizada na seção anterior, também assumimos aqui que $\mathbf{p}_m \equiv \mathbf{p}_n$ se $m \equiv n \pmod{P}$.

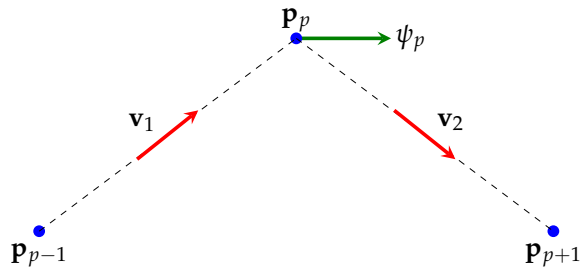


Figura 4.9. Cálculo da orientação de um determinado ponto de acordo com a disposição dos vizinhos. As setas em vermelho correspondem aos vetores que representam o fluxo direcional do circuito. A seta verde representa a orientação a ser atribuída ao ponto $\mathbf{p}_p$, referente ao vetor unitário resultante da soma de $\mathbf{v}_1$ e $\mathbf{v}_2$.

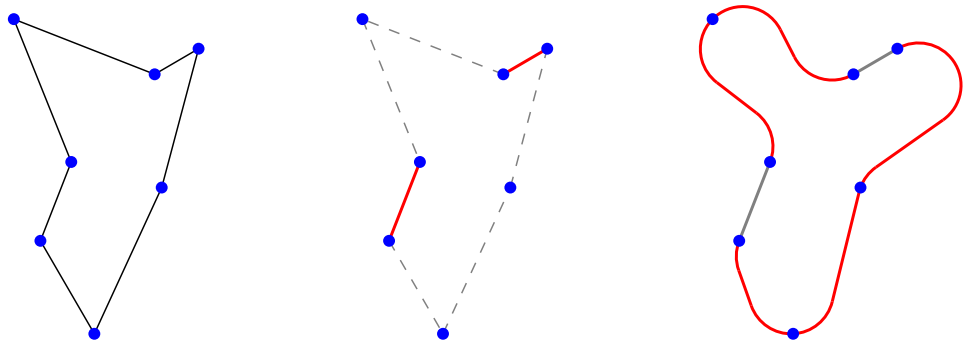


Figura 4.10. Etapas do algoritmo que atribui orientação aos pontos priorizando possíveis casos curtos de Dubins e ângulos médios nas curvas. (a) Pontos de visita fornecidos como entrada e sequência de visita obtida não considerando-se as restrições de curvatura do veículo (circuito euclidiano); (b) Inserção dos segmentos de reta conectado pares de pontos adjacentes com distância menor que 2ρ; (c) Ligação dos pontos restantes de acordo com o ângulo médio dos vetores vizinhos.

4.1.2.4 Análise de Complexidade

Nesta seção é realizada a análise da complexidade computacional da heurística determinística aqui proposta para o DTSPN. Inicialmente é avaliado o comportamento assintótico relativo à execução do algoritmo e, em seguida, é discutido o limite superior para o comprimento do caminho.

O método é composto por três etapas fundamentais: (i) determinação dos pontos de visita; (ii) otimização da posição dos pontos de visita e (iii) atribuição de orientações e cálculo das curvas de Dubins.

- **Determinação dos pontos de visita**: Em cada iteração do Algoritmo 3, pelo menos uma região de interesse receberá um novo ponto de visita, e o pior caso

Algoritmo 5 Determinar Orientações Priorizando Caso Curto e Ângulo Médio($\boldsymbol{\mathcal{P}}$)

$\Psi \leftarrow$ Inicializar todas as orientações $\psi_1, \ldots, \psi_P$ com valor *nulo*
while existir um par de pontos adjacentes com ambas as orientações *nulas* **and** a distância entre esses pontos é menor ou igual a 2ρ **do**
    Selecionar um par de pontos de acordo com

$$\arg\min_{\boldsymbol{\mathcal{P}}_\Sigma} \|\mathbf{p}_{p+1} - \mathbf{p}_p\|$$

    $\psi_p \leftarrow \mathrm{dir}(\mathbf{q}_p, \mathbf{q}_{p+1})$
    $\psi_{p+1} \leftarrow \psi_p$
end while
while existir um ponto de visita p com orientação ψ_p *nula* **do**
    Atribuir orientação ψ_p de acordo com a Equação 4.11
end while
$\boldsymbol{\mathcal{Q}} = \langle \boldsymbol{\mathcal{P}}, \Psi \rangle$
return $\boldsymbol{\mathcal{Q}}$

ocorrerá quando não houver interseção entre quaisquer regiões (obrigatoriamente percorrendo todas elas). Dessa forma, essa etapa possuirá um comportamento assintótico de $O(n)$, onde n corresponde à quantidade N de regiões.

- **Otimização da posição dos pontos de visita**: O Algoritmo 4 percorre os P pontos calculados na etapa anterior, movendo-os no interior das regiões. A direção de movimento é obtida em tempo constante $O(1)$ e a otimização possui um custo linear $O(n)$, com n representando o número de pontos de visita. Entretanto, o primeiro passo do algoritmo consiste em encontrar uma solução para uma instância do ETSP, que é NP-difícil ($O(n!)$). Entretanto, é possível a utilização de heurísticas como Christofides [Christofides, 1972] ($O(n^3)$ e fator de aproximação 1.5) e Lin-Kernighan [Lin & Kernighan, 1973] ($O(n^{2.2})$, produzindo resultados a aproximadamente 5% do valor ótimo, porém sem garantias formais). Além disso, existem disponíveis na literatura bibliotecas capazes de encontrarem soluções exatas para instâncias com algumas centenas de pontos de forma eficiente, por exemplo a `Concorde` [Concorde, 2011], considerada uma das melhores [Hahsler & Hornik, 2007]. Dessa forma, considerando que a sequência de visita é inicialmente conhecida, a otimização proposta possuirá custo $O(n)$. Entretanto, o custo completo desse passo será exponencial, dada a necessidade de se encontrar uma solução para uma instância do ETSP.

- **Atribuição de orientações**: Por último, o Algoritmo 5 também deve percorrer todos os P pontos, inicialmente determinando a orientação de pontos próximos baseado em segmentos de reta e, em seguida, os pontos remanescentes recebem orientações médias de acordo com os vizinhos. O primeiro passo pode ser realizado com custo linear, percorrendo o circuito e criando uma lista com os pares de pontos com distância menor ou igual a 2ρ, em seguida, basta percorrer essa lista novamente calculando as orientações possíveis. O segundo passo também possui custo linear, uma vez que apenas deve percorrer os pontos atribuindo as orientações. Conforme visto anteriormente, as curvas de Dubins podem ser calculadas em tempo constante $O(1)$. Logo, essa etapa possuirá um custo total $O(n)$, onde n refere-se ao número P de pontos de visita.

Dessa forma, é possível concluir que no pior caso, o limite assintótico da heurística será exponencial, dada a necessidade de se encontrar uma solução para uma instância do ETSP na etapa de otimização da posição dos pontos de visita. Caso a sequência de visita já seja conhecida, o custo será será de $O(n)$, onde n representa o número de regiões, uma vez que $P \leq N$.

O comprimento do caminho gerado pela abordagem apresentada aqui possuirá um limite superior maior, no pior caso, que o do caminho resultante obtido pelo Algoritmo Alternante. Esse limite pode ser estimado de acordo com

$$\mathcal{L}_{\rho}^{\text{heuristica}} \leq \text{ETSP}(\boldsymbol{\mathcal{P}}_{\text{cent}}) + N\, \mathsf{k}\rho\pi, \tag{4.13}$$

onde $\boldsymbol{\mathcal{P}}_{\text{cent}}$ representa a posição inicial dos pontos de visita (no centro das regiões), e considerando-se o caso em que $N > 2$ e $\rho > 0$. Lembrando que $\mathsf{k} \in [2,657\ ,\ 2,658]$. Diferentemente do limite apresentado para o Algoritmo Evolutivo, no caso da heurística não existem garantias de que algum par de pontos adjacentes necessariamente será conectado por um segmento de reta, logo, todos os segmentos conectantes podem ser curvas de Dubins (segundo termo da equação).

Entretanto, apesar de resultar em um caminho maior no pior caso, é importante observar que, na média, os resultados obtidos possuem uma grande possibilidade de serem bem mais curtos que o limite teórico. Isso ocorrerá como resultado da otimização do posicionamento pontos de visita que, conforme mencionado na Seção 4.1.2.2, produz um alinhamento dos pontos. Esse fato, além de permitir a utilização de segmentos de reta em casos não previstos pelo Algoritmo 5 (pontos adjacentes com distância maior que 2ρ), também resultará em curvas de Dubins menores.

Além disso, o limite apresentado representa o caso onde não ocorrem interseções entre as regiões. Havendo interseções, o caminho também deverá ser consideravelmente menor, uma vez que a métrica utilizando-se curvas de Dubins satisfaz a desigualdade triangular [Yadlapalli et al., 2007], e nesse caso $P < N$.

4.1.3 Experimentos

Nesta seção são apresentados os resultados obtidos a partir de simulações numéricas utilizando-se a metodologia proposta para o caso onde é utilizado um veículo e as demandas são previamente conhecidas, não ocorrendo a inserção de novas demandas ao ambiente (caso estático).

Convenciona-se que, ao determinar as dimensões do ambiente simulado, essas delimitam apenas as posições em que o centro das regiões podem ocupar, sendo que tanto parte da região quanto do caminho podem se encontrar fora desses limites.

Para execução das simulações foi desenvolvido um simulador em Matlab, e todos os experimentos foram executados em um PC com um processador Intel Core i7-2720QM 2.20 GHz, 8 Gb de RAM e o sistema operacional Ubuntu 11.10 64-bit.

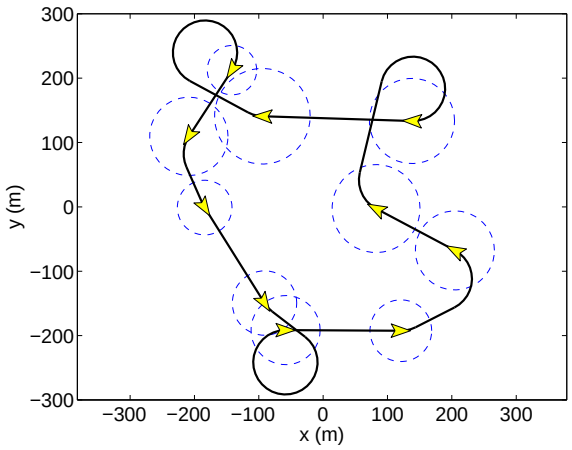
Todas as soluções para as instâncias relativas ao ETSP e ATSP foram calculadas à otimalidade utilizando-se a biblioteca `Concorde` [Concorde, 2011], um dos resolvedores mais conhecidos e utilizados em trabalhos da área. É utilizado um algoritmo exato baseado no método de planos de corte, que iterativamente soluciona as relaxações de Programação Linear do TSP [Applegate et al., 2007].

4.1.3.1 Abordagem Evolutiva

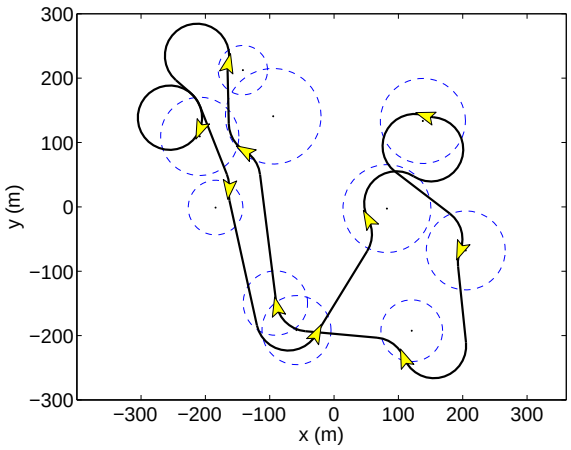
Nesta seção são apresentados e discutidos os experimentos e suas respectivas análises estatísticas, estimando-se a melhoria obtida pelo método evolutivo proposto quando comparado com os resultados obtidos por outras técnicas da literatura.

A Figura 4.11 apresenta os caminhos resultantes obtidos a partir das técnicas que serão utilizadas para comparação para uma instância exemplo, permitindo, assim, uma visão inicial do comportamento de cada uma. Nesse exemplo, é utilizado um ambiente com dimensões $500\,\mathrm{m} \times 500\,\mathrm{m}$ contendo 10 regiões (círculos tracejados azuis) aleatoriamente distribuídas de forma uniforme, e o veículo simulado utilizado possui raio mínimo de curvatura $\rho = 50\,\mathrm{m}$. O raio de cada região circular também foi determinado de forma aleatória dentro no intervalo $[25\,\mathrm{m}, 75\,\mathrm{m}]$. As setas amarelas representam a posição e orientação que o veículo deverá assumir no ponto de visita de determinada região.

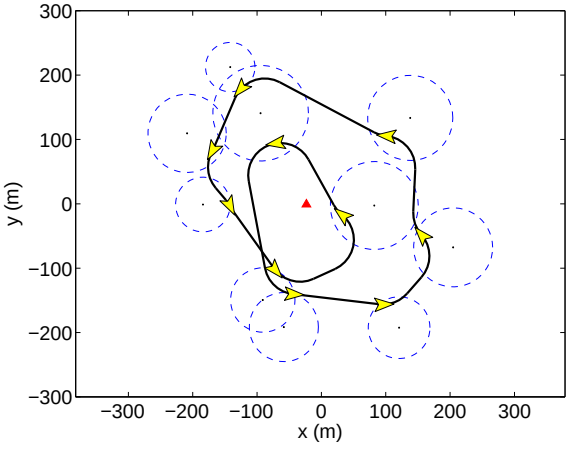
A Figura 4.11(a) apresenta a solução trivial para o problema, que consiste em ignorar as regiões e calcular o caminho utilizando o Algoritmo Alternante considerando apenas pontos de visita na posição central de cada região. A Figura 4.11(b) apresenta o caminho obtido a partir de uma técnica baseada em amostragem aleatória proposta em [Isaacs et al., 2011], e será referenciada aqui pelo termo de Aleatório. Finalmente, a Figura 4.11(c) é o resultado produzido pelo método proposto, referenciado aqui por Evolutivo.



(a) Algoritmo Alternante (AA) (2.335,20 m)



(b) Aleatório (2.813,00 m)



(c) Evolutivo (1.694,10 m)

Figura 4.11. Exemplo inicial apresentando possíveis caminhos resultantes obtidos pelas diferentes técnicas que serão analisadas nessa seção para uma instância exemplo: (a) Algoritmo Alternante (AA), (b) Aleatório e (c) Evolutivo (o triângulo vermelho representa o centroide da posição central de todas as regiões).

O efeito que cada etapa do algoritmo Evolutivo exerce sobre o resultado final pode ser claramente observado e identificado na Figura 4.11(c). Os pontos de visita foram deslocados dos centros de suas respectivas regiões em direção ao centroide

(triângulo vermelho) considerando-se todos os pontos centrais. Além disso, também pode ser verificadas as modificações realizadas nas orientações, assim como na sequência de visita. Nesse exemplo, foi utilizado como critério de parada o número limite de 100 iterações, valor consideravelmente baixo tratando-se de técnicas evolutivas, onde normalmente a execução é realizada por alguns milhares de iterações. Os demais parâmetros utilizados foram definidos com valores $\sigma = 1$ (atualização da posição) e $\sigma = 0{,}5$ (atualização da orientação), com $\mu = 0$ para ambos os passos. O exemplo inicial apresentou uma melhoria (redução do comprimento do caminho) de $\approx 27\%$ e $\approx 40\%$ quando comparados com os resultados do AA e do Aleatório, respectivamente.

Por se tratar de uma técnica probabilística, é importante a realização de uma análise estatística, verificado o comportamento do algoritmo para um número significativo de execuções. Dessa forma, foram executados 200 experimentos (diferentes instâncias) em um ambiente com dimensões $500\,\text{m} \times 500\,\text{m}$ contendo um total de 25 regiões distribuídas uniformemente de maneira aleatória, onde o raio de cada região circular também foi escolhido aleatoriamente no intervalo $[30\,\text{m}, 60\,\text{m}]$. O veículo simulado novamente possui um raio mínimo de curvatura de $\rho = 50\,\text{m}$, que representa um valor adequado para as simulações comparado às dimensões do ambiente. Para cada uma das instâncias, todas as técnicas sendo analisadas foram executadas 15 vezes, e o valor médio do comprimento do caminho encontrado foi utilizado para comparação. Os demais parâmetros permaneceram com os mesmos valores utilizados no exemplo inicial.

Para a avaliação estatística foi utilizado o teste de hipótese *Teste t* pareado, onde foram verificadas as seguintes hipóteses:

$$H_0 : \mu_{\text{EV}} - \mu_{\text{AA}} = 0 \quad \text{vs.} \quad H_1 : \mu_{\text{EV}} - \mu_{\text{AA}} < 0 \qquad \text{e} \tag{4.14}$$

$$H_0 : \mu_{\text{EV}} - \mu_{\text{AL}} = 0 \quad \text{vs.} \quad H_1 : \mu_{\text{EV}} - \mu_{\text{AL}} < 0, \tag{4.15}$$

onde μ_{AA}, μ_{AL} e μ_{EV} representam o comprimento médio do caminho obtido utilizando-se os métodos AA, Aleatório e Evolutivo, respectivamente. Ou seja, deseja-se verificar se a redução média do caminho possui significância estatística.

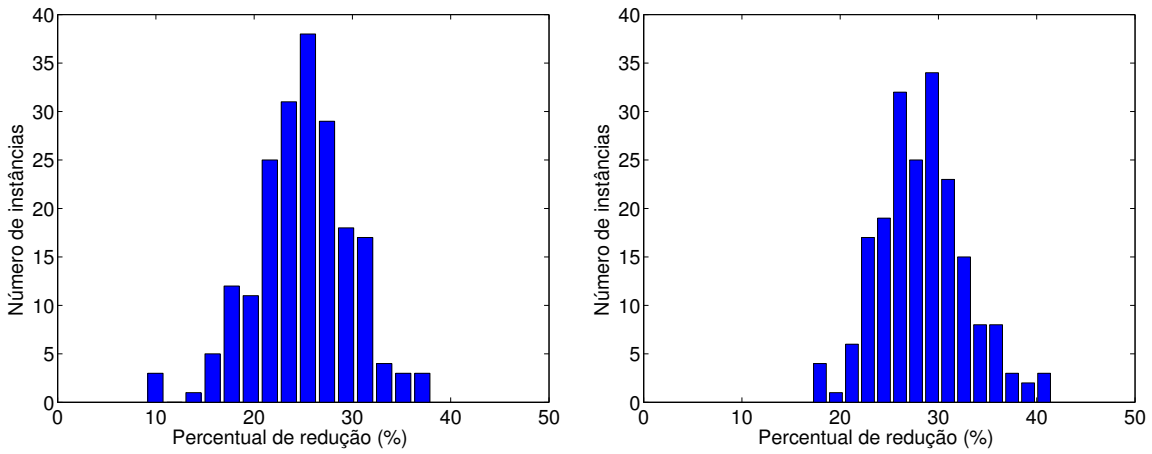
A Figura 4.12 apresenta os histogramas relativos ao percentual de redução do comprimento do caminho calculado pela metodologia proposta em relação à duas técnicas. Esse percentual é calculado a partir de

$$\text{Percentual de redução} = \left(\frac{A - B}{A}\right) * 100, \tag{4.16}$$

onde A representa o comprimento do caminho calculado pelo algoritmo sendo comparado e B o comprimento do caminho obtido a partir da metodologia proposta. Valores negativos significam que a metodologia encontrou um resultado pior que a técnica comparada.

A Figura 4.12(a) apresenta o histograma relativo ao percentual de redução do comprimento do caminho calculado pela metodologia proposta em comparação ao caminho resultante do AA. Conforme pode ser observado, foi possível obter uma redução média de $\approx 25\%$, com um desvio padrão de 4,9%. A redução proveniente do método Evolutivo, em comparação ao AA, possui significância estatística com $t(199) = 59{,}15$ e $p = 0$, ou seja, rejeitando a hipótese nula. O comprimento médio dos caminhos foi 5.078,05 m para o AA e 3.798,63 m para o Evolutivo, e é possível afirmar com 95% de confiança que a redução nesse caso encontra-se no intervalo (1.236,76, 1.322,07).

A Figura 4.12(b) apresenta o histograma relativo ao percentual de redução do comprimento do caminho calculado pela metodologia proposta em comparação ao caminho obtido pelo Aleatório. Para esse caso, a redução alcançou um valor médio de 28,3%, com um desvio padrão de 4,4%. De maneira semelhante ao caso anterior, a redução proveniente do método Evolutivo, em comparação ao Aleatório, também possui significância estatística com $t(199) = 86{,}22$ e $p = 0$, rejeitando a hipótese nula. O comprimento médio do caminho encontrado pelo Aleatório foi 5.302,40 m, e a redução encontra-se no intervalo (1.469,38, 1.538,17) com 95% de confiança.



(a) AA vs. Evolutivo (b) Aleatório vs. Evolutivo

Figura 4.12. Histogramas do percentual de redução do comprimento do caminho. (a) AA vs. Evolutivo e (b) Aleatório vs. Evolutivo.

Em seguida foi realizado um experimento com o objetivo de avaliar empiricamente o comportamento assintótico do tempo de execução do algoritmo. Foram realizadas simulações com a quantidade de regiões variando no conjunto $\{5, 10, 15, ..., 100\}$, e para cada valor 20 instâncias aleatórias foram geradas. Foi utilizado um ambiente com dimensões $500\,\text{m} \times 500\,\text{m}$ e o raio de cada região foi escolhido aleatoriamente no intervalo $[30\,\text{m}, 60\,\text{m}]$. O veículo simulado novamente possui um raio mínimo de curvatura de $\rho = 50\,\text{m}$. As demais variáveis envolvidas e aqui não mencionadas mantiveram o mesmo valor do experimento anterior.

A Figura 4.13 apresenta o valor médio mensurado. O tempo medido em cada experimento corresponde ao tempo total necessário para se executar as 100 iterações (critério de parada).

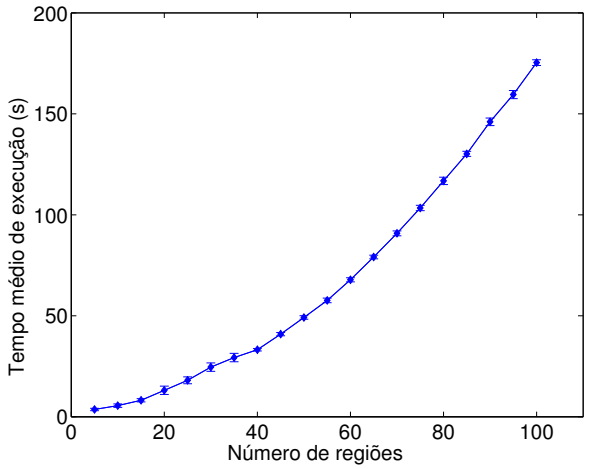


Figura 4.13. Tempo médio de execução do algoritmo evolutivo de acordo com o número de regiões no ambiente. O valor corresponde ao tempo total necessário para se executar 100 iterações (critério de parada).

Os valores encontrados foram satisfatórios, mantendo-se dentro dos limites aceitáveis de tempo inclusive para instâncias relativamente grandes (100 regiões). Também foi realizada uma comparação simples dos resultados com os valores apresentados em [Obermeyer, 2009], onde é utilizado um Algoritmo Genético. A maior instância avaliada possui 20 regiões e foi executada em 20,26 s (valor semelhante ao encontrado pela metodologia proposta). Além disso, é importante destacar que em [Obermeyer, 2009] a implementação foi feita utilizando-se a linguagem C++, enquanto neste trabalho, como já dito, foi utilizado Matlab (conhecido por não conseguir execuções muito eficientes como, por exemplo, C++).

O experimento seguinte consiste em avaliar a qualidade do algoritmo (comprimento do caminho final obtido) de acordo com a quantidade de regiões presentes no ambiente. Foram realizadas simulações com a quantidade de regiões variando

no conjunto $\{5, 10, 15, ..., 100\}$, e para cada valor 10 instâncias aleatórias foram geradas (para cada instância cada algoritmo foi executado um total de 15 vezes, sendo utilizada a média dos valores obtidos). Todas as variáveis envolvidas e aqui não mencionadas mantiveram o mesmo valor do experimento anterior.

A Figura 4.14 apresenta os resultados comparativos do comportamento efetivo encontrado para os três algoritmos. O limite assintótico inferior apresentado corresponde ao valor relativo para o DTSP apresentado em [Savla et al., 2008], sendo dado por $Cn^{2/3}$ para qualquer $\rho > 0$, onde

$$C = \left(\frac{3}{4}\right) \sqrt[3]{3\rho WH}, \tag{4.17}$$

onde W e H representam a largura e altura de um ambiente retangular, respectivamente. Considerando-se os parâmetros utilizados em nosso experimento obtemos $C = 251$. O limite inferior do DTSP nos informa qual é o menor caminho possível de ser gerado de acordo com as dimensões do ambiente e raio mínimo de curvatura do veículo para um determinado número de pontos. Logo, apesar de estarmos lidando com um problema diferente, esse valor pode ser utilizado para efeitos comparativos, uma vez que cada região possui apenas um único ponto de visita associado.

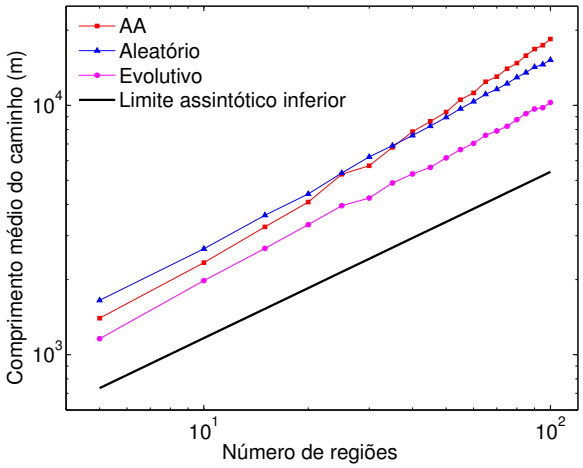


Figura 4.14. Comportamento assintótico do comprimento médio do caminho de acordo com o número de regiões para os algoritmos AA, Aleatório e Evolutivo, exibido em uma escala log-log.

Como pode ser observado, o algoritmo Evolutivo apresentou resultados melhores que os demais algoritmos comparados para todos os diferentes números de regiões utilizados no experimento. Um resultado interessante de ser destacado é o fato de o AA ter apresentado melhores resultados que o Aleatório para uma baixa de densidade de regiões, mas com o incremento na quantidade de regiões, após

um determinado valor (≈ 40) o Aleatório começa a apresentar melhores resultados (caminhos de menor comprimento).

Além de verificar-se o desempenho do algoritmo relativo à qualidade da solução, também é importante avaliar o comportamento assintótico. A Tabela 4.1 apresenta a taxa de crescimento calculada empiricamente para os diferentes algoritmos utilizando-se uma regressão linear com base nos dados apresentados na Figura 4.14. A metodologia proposta (Evolutivo) apresentou um comportamento assintótico significantemente abaixo da solução trivial obtida pelo AA, e ligeiramente superior ao limite inferior. Apesar de tratar-se de uma análise empírica, é possível concluir-se que com o aumento no número de regiões, o ganho obtido ao utilizar-se o Evolutivo tende a aumentar, uma vez que os outros métodos comparados possuem taxas de crescimento maiores.

Tabela 4.1. Taxas de crescimento do comprimento médio dos caminhos obtidos pelos métodos AA, Aleatório e Evolutivo. Valor calculado experimentalmente para regiões aleatórias distribuídas uniformemente em um ambiente com dimensões 500 m $\times$ 500 m.

Algoritmo	**Comprimento médio do caminho**
Aleatório	$6,2 \times n^{0,75}$
AA	$5,7 \times n^{0,88}$
Evolutivo	$5,9 \times n^{0,71}$
Limite inferior	$5,5 \times n^{0,67}$

Também foram executados experimentos com o objetivo de verificar se haveria ganho significativo no desempenho do algoritmo com o aumento no número de iterações. Embora seja intuitivo que executando por mais iterações serão obtidos melhores resultados, pôde-se concluir que a melhoria obtida não foi significativa. No entanto, cabe ao usuário decidir sobre optar entre o tempo de computação em detrimento da qualidade da solução, ajustando o número de iterações. Essa decisão pode ser tomada após experimentos iniciais, utilizando-se diferentes valores para os parâmetros.

4.1.3.2 Heurística Determinística

Nesta seção são apresentados e discutidos os experimentos e as respectivas análises estatísticas, verificando-se a melhoria obtida pela heurística determinística apresentada na Seção 4.1.2. A metodologia proposta (denominada Prioritário) será comparada inicialmente com o algoritmo clássico AA e com a melhoria proposta AAOG.

Em seguida, são realizados experimentos comparativos com o algoritmo Evolutivo, previamente proposto, e com a técnica mais recente da literatura abordando o DTSPN. Ao final, apresentamos uma análise geral do desempenho do método.

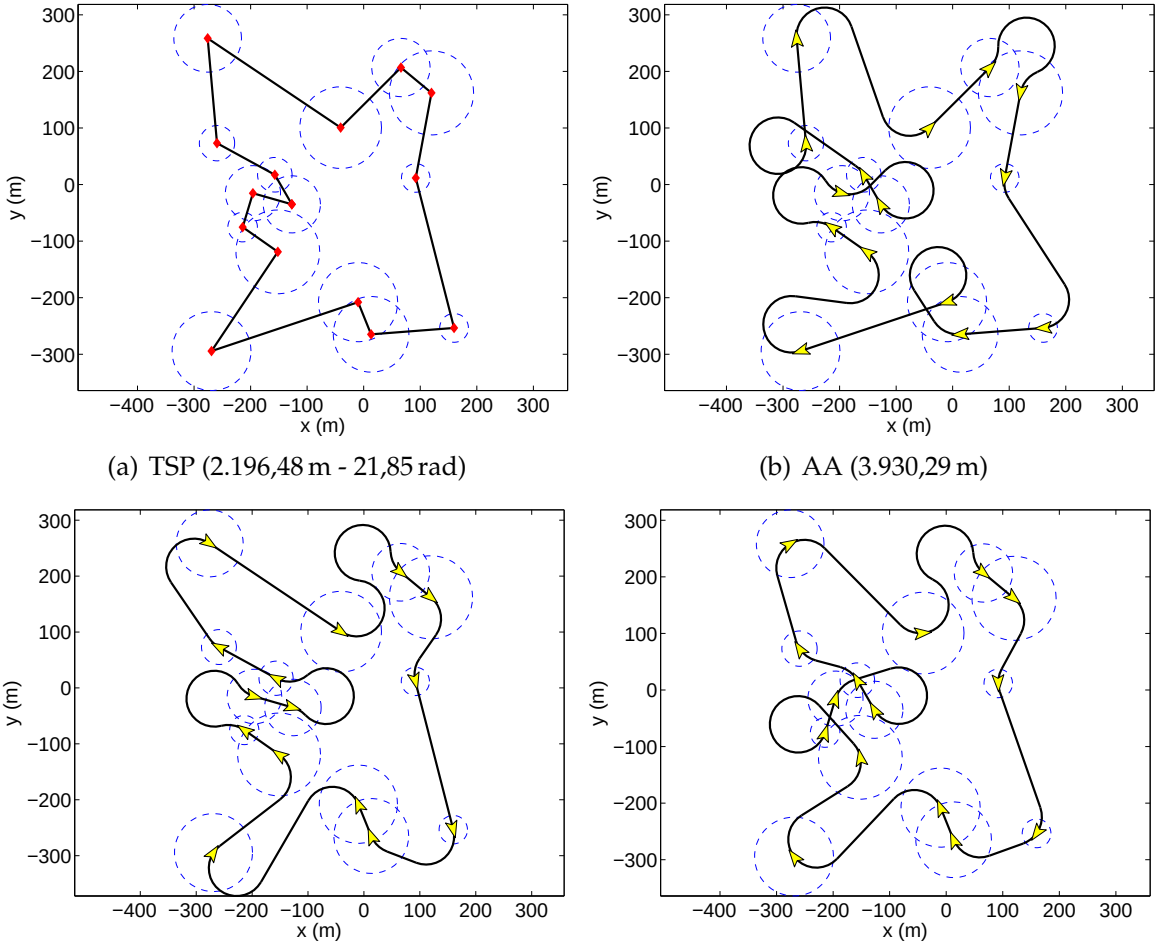
As Figuras 4.15 a 4.17 apresentam os caminhos resultantes obtidos a partir das técnicas que serão utilizadas para comparação para uma instância exemplo inicial. Nesse exemplo, é utilizado um ambiente com dimensões $600\,\mathrm{m} \times 600\,\mathrm{m}$ contendo 15 regiões aleatoriamente distribuídas de forma uniforme, e o veículo simulado utilizado possui raio mínimo de curvatura $\rho = 50\,\mathrm{m}$. O raio de cada região circular também foi determinado de forma aleatória dentro no intervalo $[25\,\mathrm{m}, 75\,\mathrm{m}]$.

A Figura 4.15 apresenta a solução trivial para o problema utilizando as três técnicas. Neste caso, os caminhos são calculados utilizando-se apenas a informação da posição do centro das regiões (ou seja, não está sendo considerada a região como um todo). Na Figura 4.15(a) o valor dentro de parênteses corresponde ao custo angular do circuito. Nota-se que a escolha do ponto inicial possui um grande impacto no resultado final do AA, e a escolha do melhor caminho permitiu uma redução de aproximadamente 15% (AAOG). O algoritmo Prioritário apresentou, para essa instância específica, um resultado 6% menor que o AAOG. Chama-se a atenção para as extremidades do caminho à esquerda (superior e inferior), onde fica clara a redução proporcionada pela atribuição da orientação como o valor médio na curva.

A Figura 4.16 apresenta os caminhos resultantes após a determinação dos pontos de visita considerando-se, agora, as regiões de interseção. Conforme esperado, um menor número de pontos de visita também resulta em caminhos menores, uma vez que tanto a métrica euclidiana quanto a distância de Dubins satisfazem a desigualdade triangular.

Finalmente, a Figura 4.17 apresenta o resultado final após a execução de todos os passos da metodologia. A Figura 4.17(a) apresenta as novas posições dos pontos de visita. Como pode ser observado, a maior parte dos pontos foram alocados nas bordas das regiões. A redução do custo angular em cerca de 40% também fica clara. Por exemplo, uma grande curva na parte superior do caminho (Figura 4.16(a)) agora é uma linha reta (Figura 4.17(a)). Destaca-se, ainda, o fato de que o resultado final obtido pelo algoritmo Prioritário possuir um comprimento menor que o valor obtido na solução inicial para a métrica euclidiana.

Apesar do bom resultado obtido no exemplo inicial, é importante a realização de mais experimentos, verificado o comportamento estatístico do algoritmo para diferentes instâncias. Dessa forma, foram executados 200 experimentos em um ambiente com dimensões $800\,\mathrm{m} \times 800\,\mathrm{m}$ contendo um total de 30 regiões distribuídas uniformemente de maneira aleatória, onde o raio de cada região circular também



(a) TSP (2.196,48 m - 21,85 rad)

(b) AA (3.930,29 m)

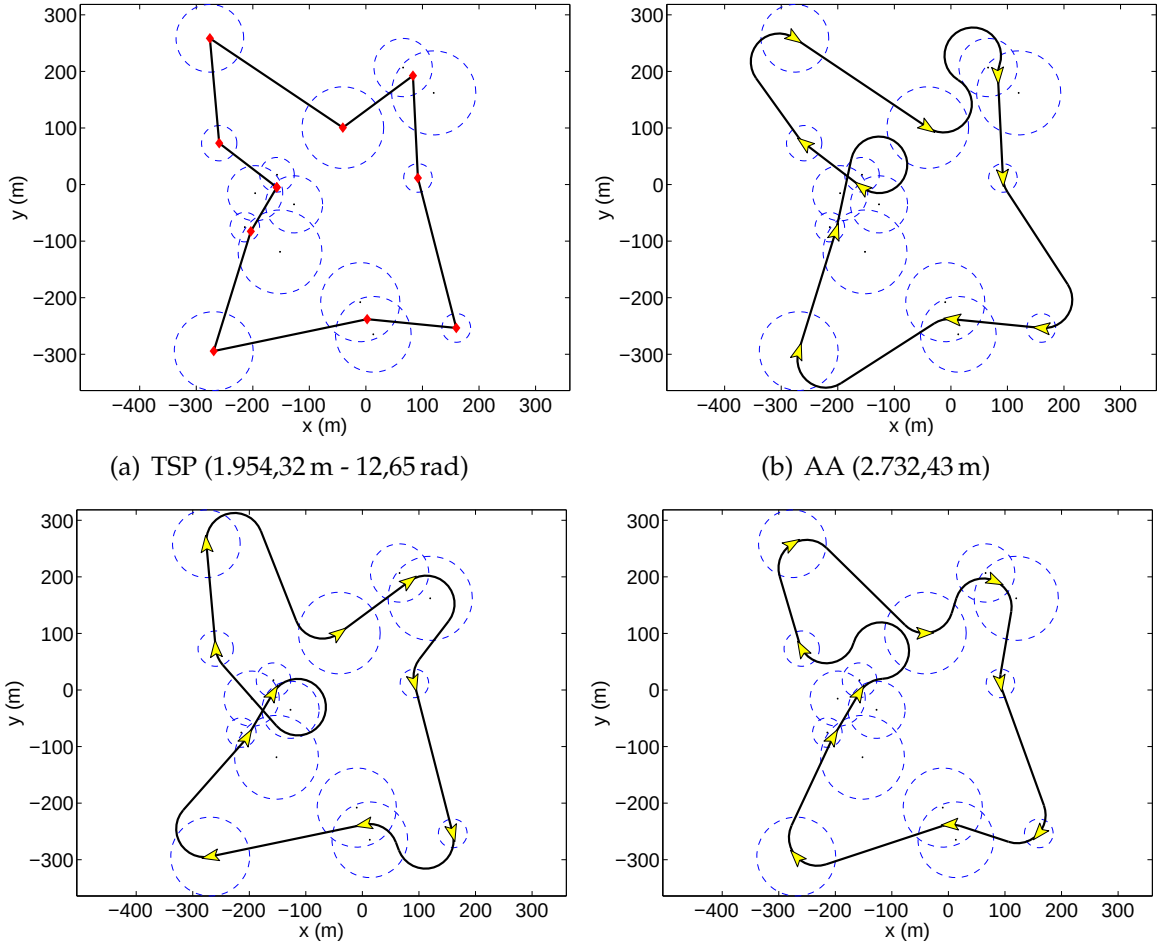
(c) AAOG (3.370,29 m)

(d) Prioritário (3.168,20 m)

Figura 4.15. Exemplo inicial: Caminhos resultantes obtidos pelas diferentes técnicas considerando os pontos de visita no centro das regiões. (a) TSP (o custo angular representa o somatório de das mudanças de direção), (b) AA, (c) AAOG e (d) Prioritário.

foi escolhido aleatoriamente no intervalo $[30\,\text{m}, 100\,\text{m}]$. O veículo simulado novamente possui um raio mínimo de curvatura de $\rho = 50\,\text{m}$. Para todos os casos foram comparados os caminhos gerados já utilizando os pontos de visitas nas regiões de interseção e com posições otimizadas.

É apresentado na Figura 4.18 um histograma com o percentual de redução obtido pela escolha do melhor circuito (AAOG) em vez da solução inicial (AA). Em $24,5\%$ dos casos não houve nenhuma melhoria, ou seja, o caminho inicial já correspondia ao melhor circuito. Nos demais casos, foi possível obter uma redução média de $11,3\%$ no comprimento do caminho (chegando a 36% para um caso específico). Por se tratar de uma verificação rápida, acredita-se ser interessante sempre utilizar o melhor caminho em vez da solução inicial, principalmente em aplicações onde o



(a) TSP (1.954,32 m - 12,65 rad)

(b) AA (2.732,43 m)

(c) AAOG (2.621,98 m)

(d) Prioritário (2.315,28 m)

Figura 4.16. Exemplo inicial: Caminhos resultantes obtidos pelas diferentes técnicas após a execução da etapa que determina os pontos de visitas nas regiões de interseção. (a) TSP, (b) AA, (c) AAOG e (d) Prioritário.

comprimento do caminho é crítico.

Para o restante da avaliação estatística, foi utilizado o teste de hipóteses *Teste t* pareado, onde foram verificadas as seguintes hipóteses:

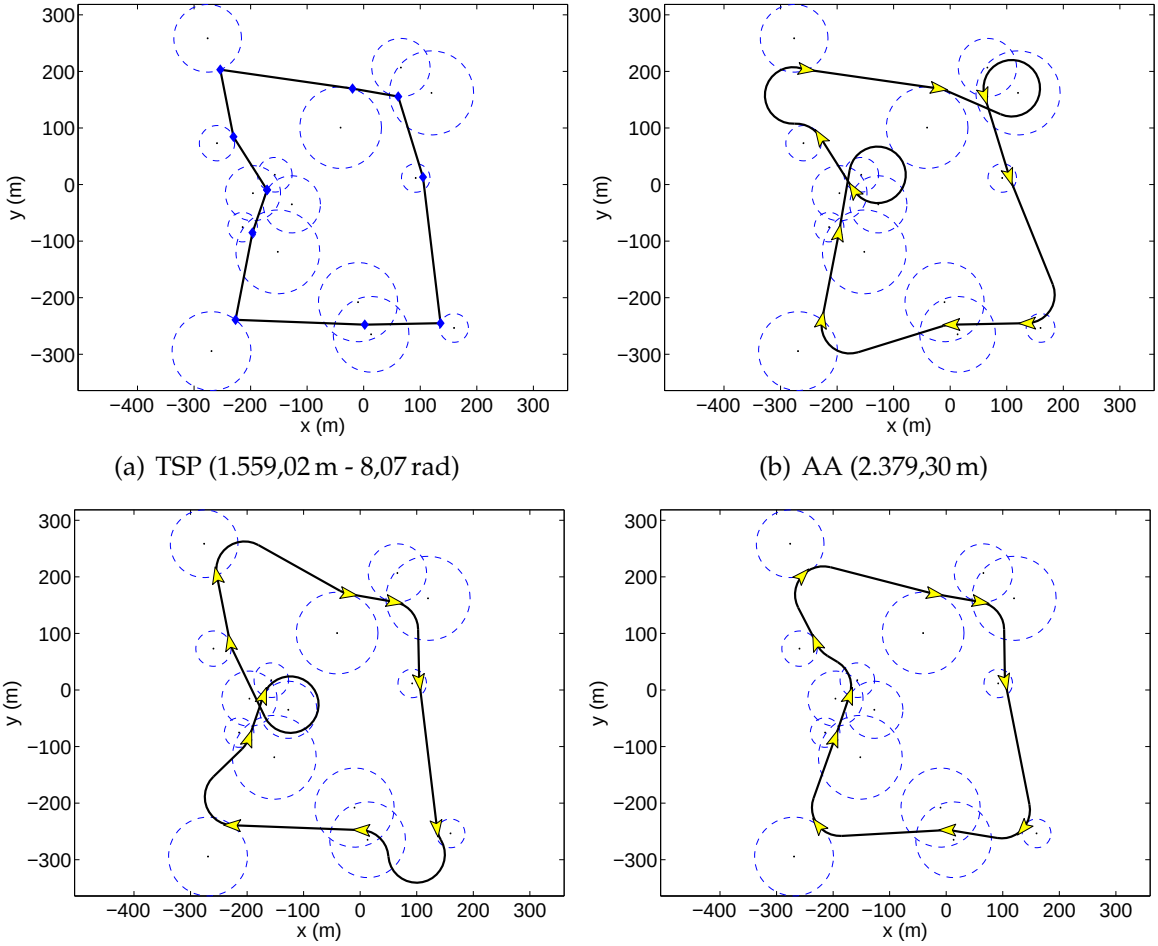
$$H_0 : \mu_{\text{PR}} - \mu_{\text{AA}} = 0 \quad \text{vs.} \quad H_1 : \mu_{\text{PR}} - \mu_{\text{AA}} < 0, \tag{4.18}$$

$$H_0 : \mu_{\text{PR}} - \mu_{\text{AAOG}} = 0 \quad \text{vs.} \quad H_1 : \mu_{\text{PR}} - \mu_{\text{AAOG}} < 0 \qquad \text{e} \tag{4.19}$$

$$H_0 : \mu_{\text{PR}} - \mu_{\text{OA}} = 0 \quad \text{vs.} \quad H_1 : \mu_{\text{PR}} - \mu_{\text{OA}} < 0, \tag{4.20}$$

onde μ_{AA}, μ_{AAOG}, μ_{OA} e μ_{PR} representam o comprimento médio do caminho obtido utilizando-se os métodos AA, AAOG, Orientações Aleatórias (OA)[1] e Prioritário,

[1]Similar ao algoritmo Aleatório utilizado na seção anterior. Entretanto, nesse caso apenas orientações aleatórias são geradas para os pontos de visitas previamente definidos, sendo executado em seguida o ATSP.



(a) TSP (1.559,02 m - 8,07 rad)

(b) AA (2.379,30 m)

(c) AAOG (2.137,34 m)

(d) Prioritário (1.624,50 m)

Figura 4.17. Exemplo inicial: Caminhos resultantes obtidos pelas diferentes técnicas após a execução da etapa que otimiza a posição dos pontos de visitas nas regiões de interseção. (a) TSP, (b) AA, (c) AAOG e (d) Prioritário.

respectivamente. É importante ressaltar que, neste caso, apenas os métodos de atribuição de orientações estão sendo avaliados, sendo fornecidos para cada método os mesmos pontos de visita previamente calculados pelas etapas iniciais da heurística proposta.

A Figura 4.19(a) apresenta o histograma relativo ao percentual de redução do comprimento do caminho calculado pela heurística proposta em comparação ao caminho resultante do AA. Conforme pode ser observado, foi possível obter uma redução no comprimento do caminho em 96, 5% dos casos, com um percentual médio de aproximadamente 15%. A redução proveniente do algoritmo Prioritário em comparação ao AA possui significância estatística com $t(199) = 21,63$ e $p = 0$, ou seja, rejeitando a hipótese nula.

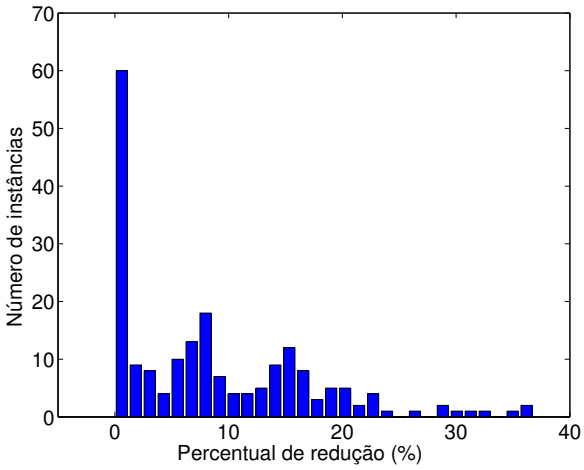
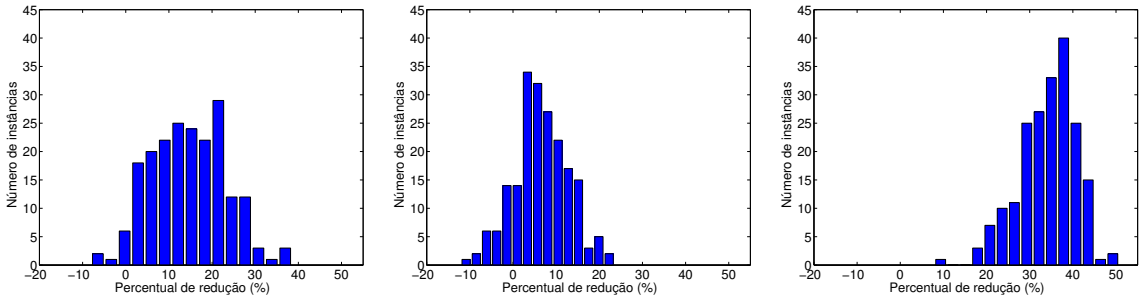


Figura 4.18. Histograma do percentual de redução do comprimento do caminho obtido pela escolha do melhor circuito (AAOG) em vez da solução inicial (AA). Em 24,5%, das 200 instâncias avaliadas, não houve nenhuma melhoria. Nos demais casos obteve-se uma redução média de 11,3%

A Figura 4.19(b) apresenta o histograma relativo ao percentual de redução do comprimento do caminho calculado pela heurística proposta em comparação ao caminho obtido pelo AAOG. Para esse caso, a redução alcançou um valor médio de aproximadamente 8%, reduzindo efetivamente o comprimento em 85,5% das instâncias. A redução proveniente do algoritmo Prioritário em comparação ao AAOG também possui significância estatística com $t(199) = 14,03$ e $p = 0$, rejeitando a hipótese nula.

A Figura 4.19(c) apresenta o histograma relativo ao percentual de redução do comprimento do caminho calculado pela heurística proposta em comparação ao caminho obtido pelo OA. Para esse caso, a redução alcançou um valor médio de aproximadamente 34%, reduzindo o comprimento em 100% das instâncias. De maneira semelhante ao caso anterior, a redução proveniente do algoritmo Prioritário em comparação ao OA também possui significância estatística com $t(199) = 58,93$ e $p = 0$, rejeitando a hipótese nula.

Conforme pode ser observado, diferentemente do algoritmo Evolutivo, o algoritmo Prioritário não possui garantia de redução do caminho em relação aos demais métodos. Entretanto, como foi validado, com significância estatística, o Prioritário apresentou resultados melhores para a maioria dos casos para as instâncias utilizadas. É possível observar que em determinadas circunstâncias (disposição dos pontos de visita), a atribuição do ângulo médio não é a melhor decisão, sendo melhor a ligação por um segmento de reta (conforme utilizado pelos métodos AA e AAOG). Esse fato ocorre principalmente quando vários pontos de visita estão muito próxi-



(a) AA vs. Prioritário (b) AAOG vs. Prioritário (c) OA vs. Prioritário

Figura 4.19. Histogramas do percentual de redução do comprimento do caminho. (a) AA vs. Prioritário, (b) AAOG vs. Prioritário e (c) OA vs. Prioritário.

mos.

Os resultados gerais obtidos durante o experimento podem ser vistos na Tabela 4.2, onde são apresentados os comprimentos médios dos caminhos obtidos pelas diferentes técnicas para cada etapa da metodologia. Conforme esperado, utilizando-se pontos de visita nas interseções e posteriormente otimizando a posição desses dentro da área em comum, é possível obter caminhos mais eficientes (menores). Além disso, o algoritmo Prioritário apresentou um comprimento médio menor que o AA e AAOG para todos os casos, independente da posição dos pontos de visita.

Tabela 4.2. Comprimento médio dos caminhos obtidos pelas diferentes técnicas para cada etapa da metodologia. Valores de comprimento apresentados em metros e de custo angular em radianos, entre parênteses os desvios padrão.

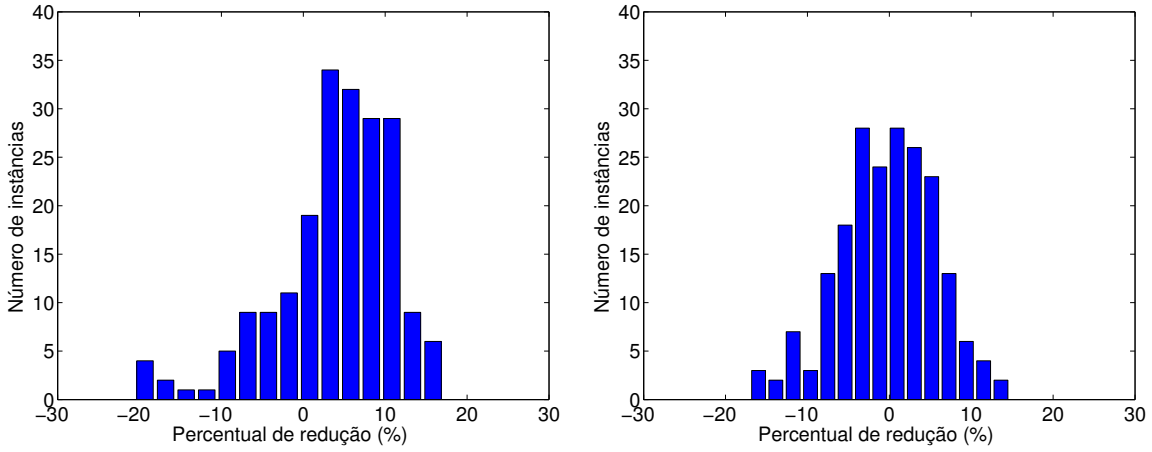
Algoritmo	**Pontos de Visita**		
	Centro	Interseção	Otimizado
TSP[1]	3.640,66 (236,63)	3.159,44 (249,22)	2.580,86 (211,55)
AA	6.371,42 (401,02)	4.078,08 (423,71)	3.555,46 (398,91)
AAOG	6.142,07 (339,09)	3.858,56 (370,46)	3.235,93 (345,19)
Prioritário	5.839,97 (488,29)	3.657,27 (398,16)	3.017,84 (312,74)
[1] Custo Angular	32,96 (2,69)	19,68 (2,77)	14,16 (1,78)

O método também foi comparado com a técnica mais recente encontrada na literatura abordando o DTSPN, disponível em [Isaacs et al., 2011], onde é apresentada uma abordagem baseada em amostragem (aqui referenciada como Amostragem). Determinadas orientações são amostradas nas bordas das regiões, em seguida é resolvido o problema que encontra o menor caminho que passa por uma amostra em cada região. Foram executados 200 experimentos comparativos, em cada experimento 15 regiões foram distribuídas uniformemente de maneira aleatória, os demais

parâmetros relacionados ao tamanho do ambiente, raio das regiões e raio mínimo de curvatura permaneceram iguais aos utilizados no experimento anterior.

A Figura 4.20(a) apresenta o histograma do percentual de redução do caminho proporcionado pelo algoritmo Prioritário quando comparado ao resultado obtido pelo algoritmo Amostragem. Foram obtidos resultados melhores em 77% das instâncias, com uma redução média de aproximadamente 7%. Foi verificado que esse resultado possui significância estatística com $t(199) = 7,75$ e $p = 0$. O comprimento médio dos caminhos, considerando todos os casos (ou seja, onde ocorreu redução ou aumento do caminho), foi de 2.342,13 m para o Amostragem e 2.248,69 m para o Prioritário.

O algoritmo Evolutivo também foi comparado com o algoritmo Amostragem, e o histograma do percentual de redução do caminho é apresentado na Figura 4.20(b). Os pontos de visita foram determinados de acordo com a Seção 4.1.2.1, para cada instância o algoritmo Evolutivo foi executado 15 vezes, e o valor médio obtido foi utilizado para comparação. O algoritmo Evolutivo obteve caminhos menores em 50,5% dos casos, e através do teste de hipóteses verificou-se que esse resultado não é estatisticamente significante, ou seja, pode-se considerar que ambos os algoritmos apresentam resultados compatíveis. O comprimento médio do caminho obtido pelo algoritmo Evolutivo foi de 2.347,29 m, ligeiramente maior que o obtido pelo Amostragem.



(a) Amostragem vs. Prioritário (b) Amostragem vs. Evolutivo

Figura 4.20. Histogramas do percentual de redução do comprimento do caminho. (a) Amostragem vs. Prioritário, (b) Amostragem vs. Prioritário, onde Amostragem refere-se ao mais recente método na literatura abordando o DTSPN e apresentado em [Isaacs et al., 2011].

As Figuras 4.21 e 4.22 apresentam os dois melhores e dois piores resultados

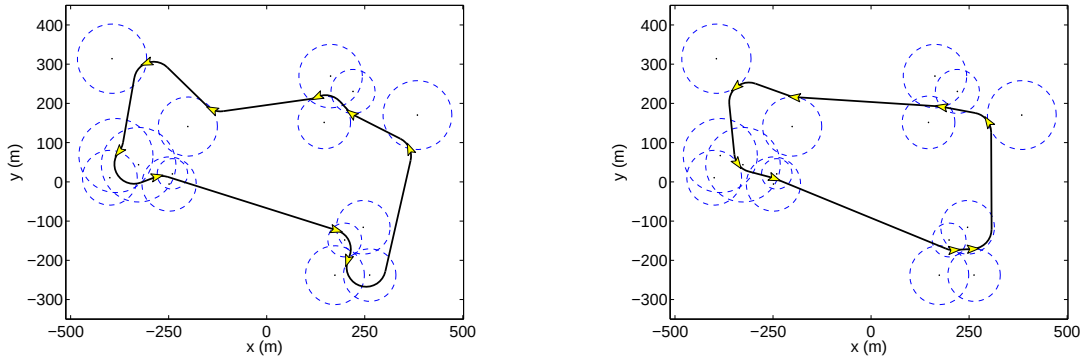
obtidos pelos algoritmos Prioritário e Evolutivo, respectivamente, quando comparados ao de Amostragem. Foi observado que em grande parte das instâncias os pontos e sequência de visita foram bem semelhantes (ou iguais), e o principal fator impactante foi a orientação em cada ponto. O algoritmo Prioritário apresentou um baixo desempenho principalmente no caso em que os pontos estão muito próximos (parte superior do caminho apresentado na Figura 4.21(d)), o que acaba por gerar vários caminhos "circulares". A mesma conclusão relativa ao algoritmo Evolutivo, discutida anteriormente, é novamente evidente aqui, ou seja, o movimento em direção ao centroide das regiões nem sempre é a melhor escolha. Esse fato é evidente na Figura 4.22(d), onde certamente ocorreria uma redução do caminho caso o ponto de visita quase disposto no centroide se movesse na direção oposta (chegando em uma posição similar à apresentada pelo Amostragem).

Em seguida foi realizado um experimento com o objetivo de avaliar empiricamente o comportamento assintótico do tempo de execução do algoritmo como um todo. Foram realizadas simulações com a quantidade de regiões variando no conjunto $\{5, 10, 15, ..., 200\}$, e para cada valor 30 instâncias aleatórias foram geradas. Foi utilizado um ambiente com dimensões $1.000\,\text{m} \times 1.000\,\text{m}$ e o raio de cada região circular também foi escolhido aleatoriamente no intervalo $[30\,\text{m}, 60\,\text{m}]$. O veículo simulado novamente possui um raio mínimo de curvatura de $\rho = 50\,\text{m}$. Os resultados são apresentados na Figura 4.23.

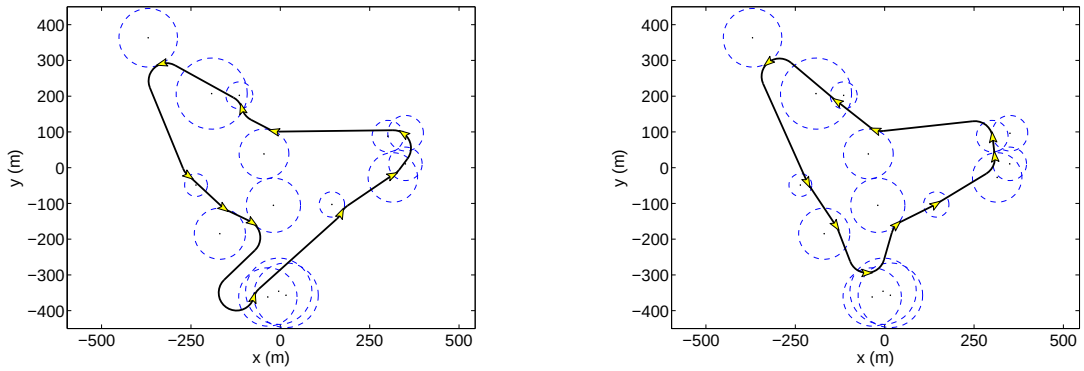
A Figura 4.23(a) apresenta o tempo médio mensurado para cada quantidade de regiões. Entretanto, conforme dito, foi avaliado o tempo para se executar todas as etapas do algoritmo. A primeira etapa consiste em justamente determinar pontos de visita nas interseções, o que influencia no comportamento. A Figura 4.23(b) apresenta o número médio de pontos de visita de acordo com a quantidade de regiões utilizada. Conforme pode ser observado, é importante tirar vantagem das regiões de interseção, uma vez que o número de pontos de visita foi inferior à metade do número de regiões ao final do experimento (200 regiões).

Como pode ser observado, os valores encontrados são consideravelmente inferiores aos obtidos pelo Algoritmo Evolutivo (mesmo para uma quantidade similar de pontos de visita). Esses valores servem apenas para referência, uma vez que uma otimização da implementação poderia produzir resultados ainda melhores.

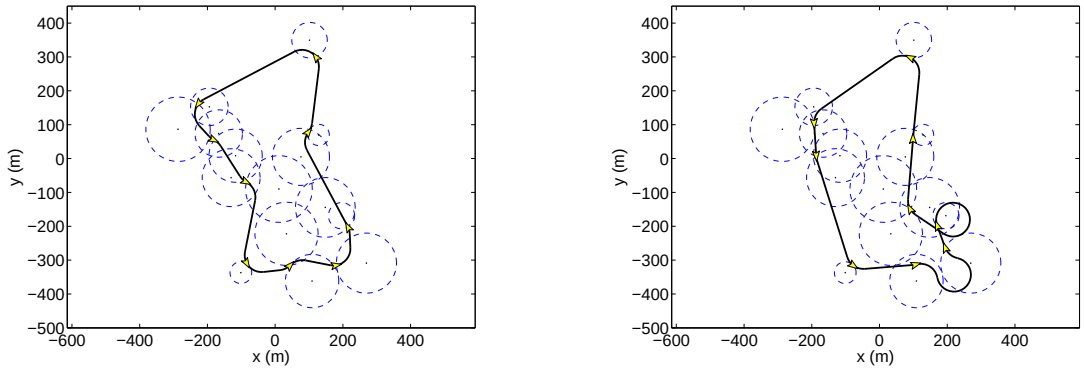
Para esse experimento, também foi avaliado o desempenho do algoritmo em relação ao comprimento do caminho, de acordo com a quantidade de regiões presentes no ambiente. Os resultados são apresentados na Figura 4.24.



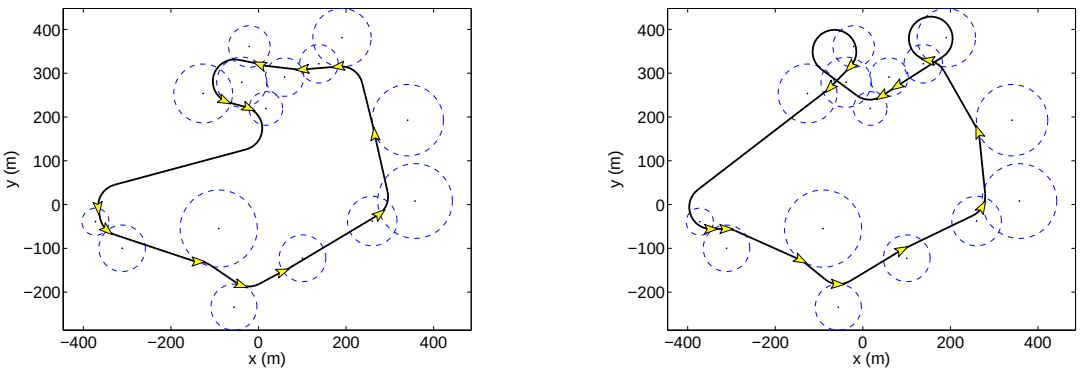
(a) Amostragem (2.245,45 m) vs. Prioritário (1.861,42 m): Redução de 17, 10%.



(b) Amostragem (2.377,52 m) vs. Prioritário (1.976,20 m): Redução de 16, 88%.

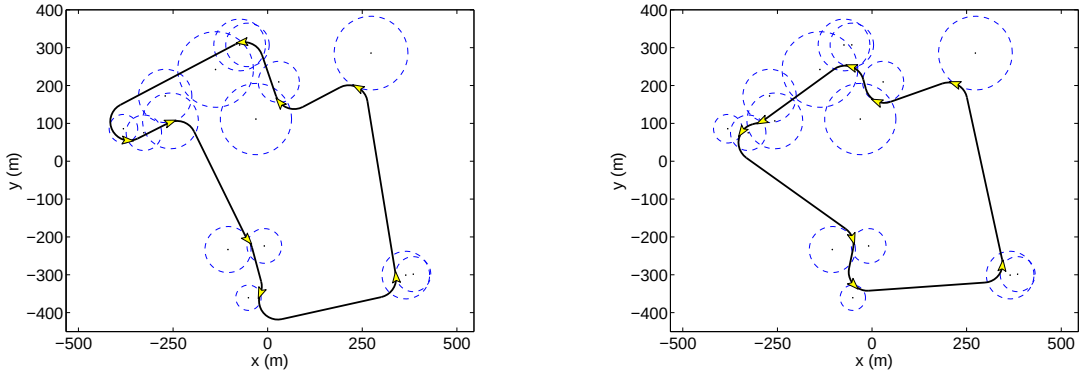


(c) Amostragem (1.898,46 m) vs. Prioritário (2.275,96 m): Aumento de 19, 88%.

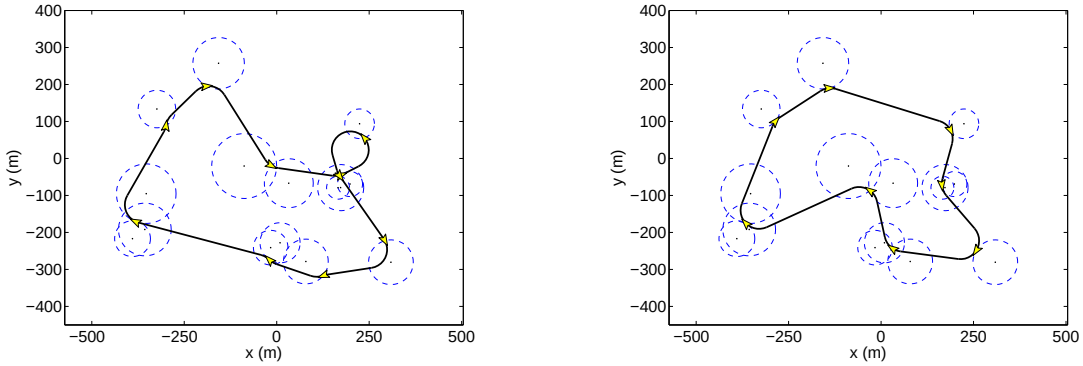


(d) Amostragem (2.071,40 m) vs. Prioritário (2.497,82 m): Aumento de 20, 58%.

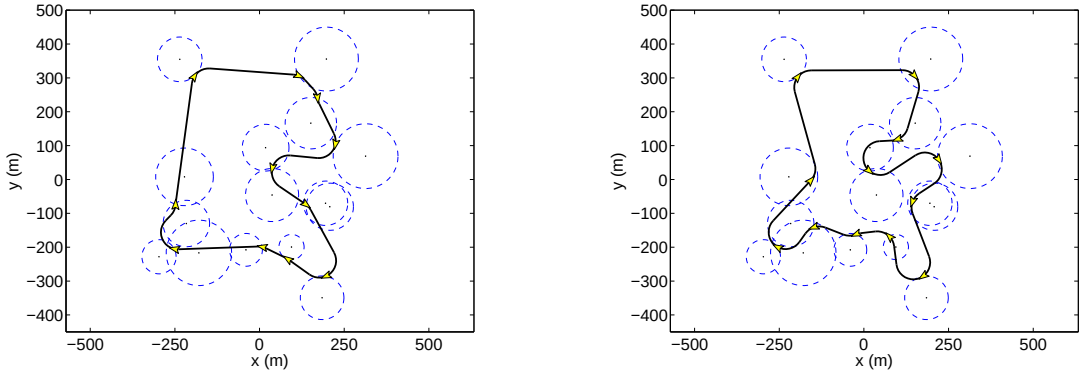
Figura 4.21. Melhores (4.21(a) e 4.21(b)) e piores (4.21(c) e 4.21(d)) resultados obtidos pelo algoritmo Prioritário em comparação aos obtidos pelo Amostragem [Isaacs et al., 2011].



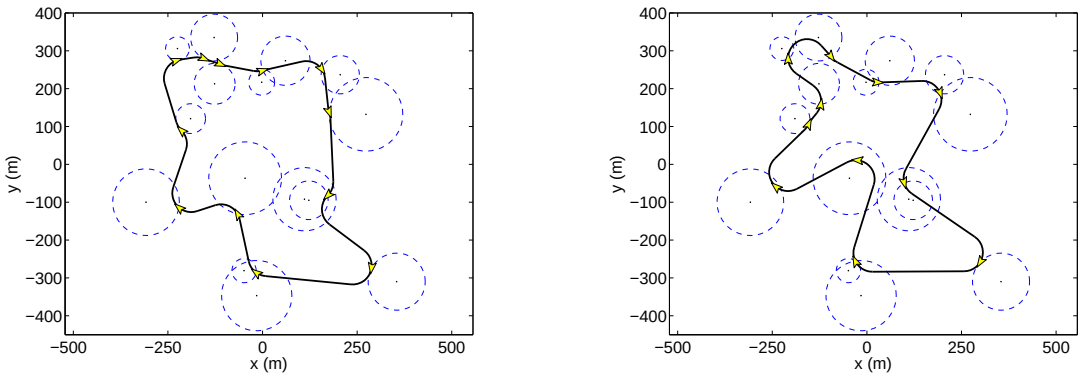
(a) Amostragem (2.551,67 m) vs. Evolutivo (2.166,30 m): Redução de 15, 10%.



(b) Amostragem (2.249,12 m) vs. Evolutivo (1.963,14 m): Redução de 12, 71%.

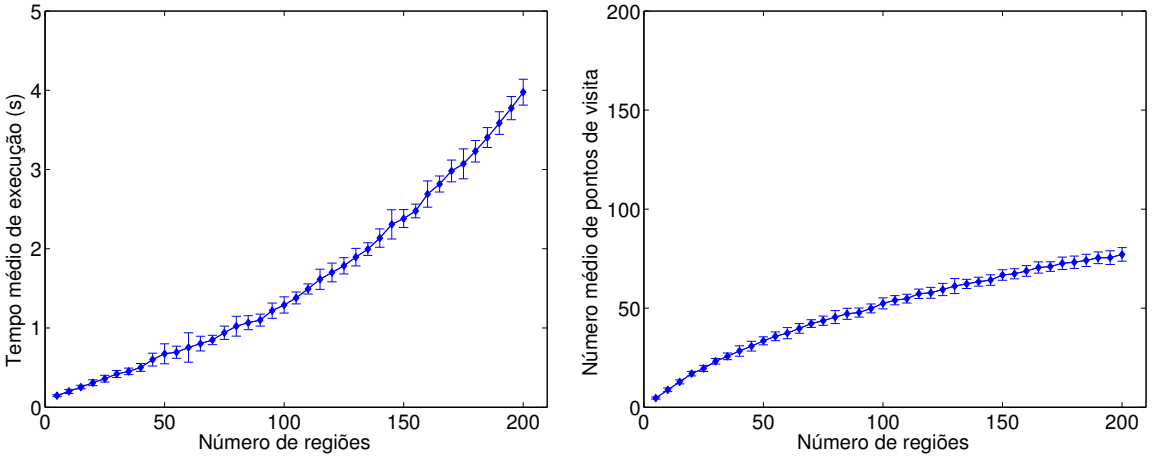


(c) Amostragem (2.203,52 m) vs. Evolutivo (2.505,29 m): Aumento de 13, 69%.



(d) Amostragem (2.087,63 m) vs. Evolutivo (2.376,37 m): Aumento de 13, 83%.

Figura 4.22. Melhores (4.22(a) e 4.22(b)) e piores (4.22(c) e 4.22(d)) resultados obtidos pelo algoritmo Evolutivo em comparação aos obtidos pelo Amostragem [Isaacs et al., 2011].



(a) (b)

Figura 4.23. Avaliação do desempenho computacional da heurística determinística proposta para o DTSPN. (a) Tempo médio de execução do algoritmo de acordo com o número de regiões no ambiente. (a) Número médio de pontos de visita utilizados de acordo com o número de regiões.

Inicialmente, a Figura 4.24(a) apresenta os resultados considerando-se apenas as partes do algoritmo que determinam os pontos nas interseções (PI) e as orientações (Prioritário) utilizadas no cálculo das curvas de Dubins. É realizada uma comparação com um algoritmo que utiliza os pontos de interseção previamente calculados, e partir de orientações aleatórias (OA) encontra uma solução para o ATSP. É possível observar que com o aumento na densidade de regiões no ambiente o método aleatório passa a obter resultados melhores que o método proposto. Com a maior quantidade de pontos de visita próximos uns dos outros, as orientações obtidas pelo algoritmo não possuirão mais tanto impacto no comprimento final.

A Figura 4.24(b) apresenta o resultado final do método, agora também considerando a etapa de otimização da posição dos pontos de visita. A comparação é realizada com um algoritmo totalmente aleatório (posição e orientação), e em seguida encontra-se uma solução para o ATSP. É possível observar que nesse caso foram obtidos resultados bem superiores ao caso totalmente aleatório. Além disso, a otimização da posição também permitiu resultados melhores que a etapa anterior.

Em ambos os casos também foi apresentado o valor do limite assintótico inferior esperado para o circuito Dubins. Entretanto, o limite assintótico foi apresentado considerado-se um ponto de visita para cada uma das regiões ($P = N$), o que não é o caso para nosso algoritmo. Dessa forma, também será realizado um experimento em que o número de pontos de visita é igual ao de regiões.

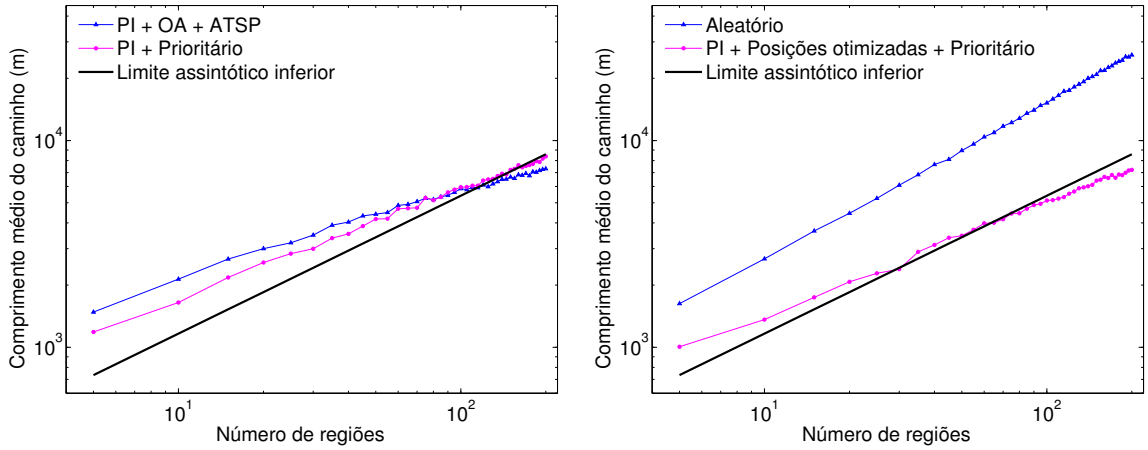


Figura 4.24. Comportamento assintótico do comprimento médio do caminho de acordo com o número de regiões, considerando-se pontos nas interseções (PI). (a) Não considerando a etapa de otimização da posição dos pontos de visita; (b) Considerando a etapa de otimização (metodologia completa).

Logo, o objetivo principal desse experimento é avaliar especificamente o desempenho do Algoritmo 5, em relação ao comprimento do caminho, de acordo com a quantidade de regiões presentes no ambiente. Foram realizadas simulações com a quantidade de regiões variando no conjunto $\{5, 10, 15, ..., 200\}$, e para cada valor 30 instâncias aleatórias foram utilizadas. Foi utilizado um ambiente com dimensões $500\,\text{m} \times 500\,\text{m}$ e as demais variáveis permaneceram com os mesmos valores do experimento anterior. Os pontos de visita foram todos determinados nos centros das regiões. O método foi comparado com o comportamento do Algoritmo Alternante (AA) e do Algoritmo Alternante Ótimo Generalizado (AAOG).

A Figura 4.25 apresenta os resultados comparativos do comportamento efetivo encontrado para os três algoritmos sendo avaliados nessa seção. O limite assintótico inferior apresentado corresponde ao mesmo valor utilizado na seção anterior, uma vez que foram utilizados os mesmos valores para as dimensões do ambiente e raio mínimo de curvatura do veículo.

O algoritmo Prioritário apresentou resultados melhores que o AA e comparáveis ao AAOG nos experimentos onde um número moderado de regiões foi utilizado ($\approx$ 30). Além disso, é possível observar que com o aumento do número de regiões, o ganho proporcionado pelo AAOG em relação ao AA apresenta uma redução, com uma tendência de se tornar desprezível. Acredita-se que o principal fator que influencia na qualidade dos algoritmos não é o número de regiões, mas a densidade de regiões existentes no ambiente e consequentemente a maior proximi-

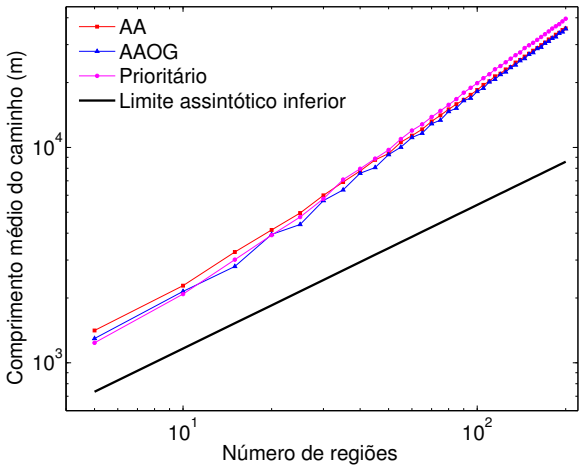


Figura 4.25. Comportamento assintótico do comprimento médio do caminho de acordo com o número de regiões para os algoritmos AA, AAOG e Prioritário, exibido em uma escala log-log. Os pontos de visita utilizados foram posicionados no centro das regiões.

dade dos pontos de visita (conforme anteriormente observado na comparação com os resultados obtidos pelo Amostragem).

A Tabela 4.3 apresenta a taxa de crescimento obtida empiricamente para os diferentes algoritmos utilizando-se uma regressão linear com base nos dados apresentados na Figura 4.25. Apesar do método proposto (Prioritário) apresentar um comportamento assintótico pior que os demais métodos, a sua utilização ainda é interessante nos casos onde as regiões se encontram distribuídas de forma mais esparsa no ambiente. Com o maior número de experimentos, também é possível obter uma melhor representação do comportamento do AA. Além disso, é importante observar que o valor obtido pelo AAOG, apesar de maior que o do AA, não é realista, uma vez que o caminho encontrado pelo AAOG sempre será menor ou igual ao AA. O valor encontado para o AAOG se compara ao valor encontrado para o AA em [Le Ny et al., 2012].

Tabela 4.3. Taxas de crescimento do comprimento médio dos caminhos obtidos pelos métodos AA, AAOG e Prioritário. Valor calculado experimentalmente para regiões aleatórias distribuídas uniformemente em um ambiente com dimensões 500 m × 500 m.

Algoritmo	**Comprimento médio do caminho**
AA	$5,6 \times n^{0,91}$
AAOG	$5,5 \times n^{0,94}$
Prioritário	$5,4 \times n^{0,97}$
Limite inferior	$5,5 \times n^{0,67}$

É interessante observar que apesar do baixo desempenho do algoritmo Prioritário com o aumento do número de regiões, na prática isso não será um problema. Quanto maior o número de regiões em um ambiente, maior a chance de se ter regiões de interseção, e com isso, o número de pontos de visita deve, na maior parte dos casos, ser consideravelmente menor que o número de regiões (Figura 4.23(b)). Além disso, a partir da análise separada de cada uma das etapas do algoritmo foi possível observar também que cada uma exerce influência direta no resultado final (Figura 4.24(b)).

4.2 Caso Dinâmico

Nessa seção é apresentado o problema ao qual denominamos de TSP Dubins com Vizinhanças Dinâmico (*Dynamic Dubins TSP with Neighborhoods*, ou DDTSPN). O DDTSPN consiste em uma generalização para o DTSPN, apresentado na seção anterior. A heurística apresentada na Seção 4.1.2 é estendida de forma a permitir que o caminho inicialmente planejado seja modificado, adicionando as novas regiões inseridas no ambiente.

O algoritmo proposto baseia-se na Heurística de Inserção Adiante (*Push-forward Insertion Heuristic*, ou PFIH) apresentada em [Solomon, 1987]. A PFIH utiliza uma estratégia construtiva gulosa para calcular o menor custo de inserção da nova demanda no caminho atual. Essa técnica não pode ser diretamente aplicada no contexto do DDTSPN, uma vez que ela considerada basicamente a parte combinatória do problema (sequência de visita). No caso do problema abordado aqui, a determinação da posição e orientação do ponto de visita (problema contínuo) tem grande influência na determinação do melhor lugar para inserir a nova região no caminho.

Portanto, o algoritmo para o caso dinâmico faz uso do método proposto anteriormente para o caso estático para eficientemente determinar a posição e orientação dos pontos de visita de acordo com as partes caminho que podem ser modificadas. Em seguida, tendo em vista o novo ponto de visita, o método determina de maneira gulosa a parte do caminho que será alterada, a fim de minimizar o impacto sobre o comprimento do percurso final.

Conforme discutido na parte final do Capítulo 2, uma das principais diferenças do problema abordado neste trabalho (visita a regiões) para os casos normalmente encontrados na literatura (visita a posições exatas) está no fato de a inserção de uma nova demanda no ambiente não necessariamente implicar em uma alteração no caminho atual.

Considerando essa característica, essa etapa será composta por dois passos básicos: (i) inicialmente verifica-se se é possível visitar a nova região inserida sem a necessidade de se alterar explicitamente o caminho; caso a primeira parte não seja possível, (ii) é avaliado qual parte do caminho atual deve ser alterada incorporando a nova região com o menor impacto possível para o comprimento do caminho.

Decidiu-se por não efetuar nenhuma alteração na posição ou orientação dos pontos de visita já inseridos no caminho. Existem duas razões principais para essa decisão: (i) a posição não é alterada pois, conforme visto anteriormente, a escolha de pontos de visita em interseções não é trivial; (ii) a orientação não é modificada porque uma uma pequena alteração em um ponto de visita poderia resultar em uma sequência de alterações em cadeia, aumentando consideravelmente o custo desta etapa.

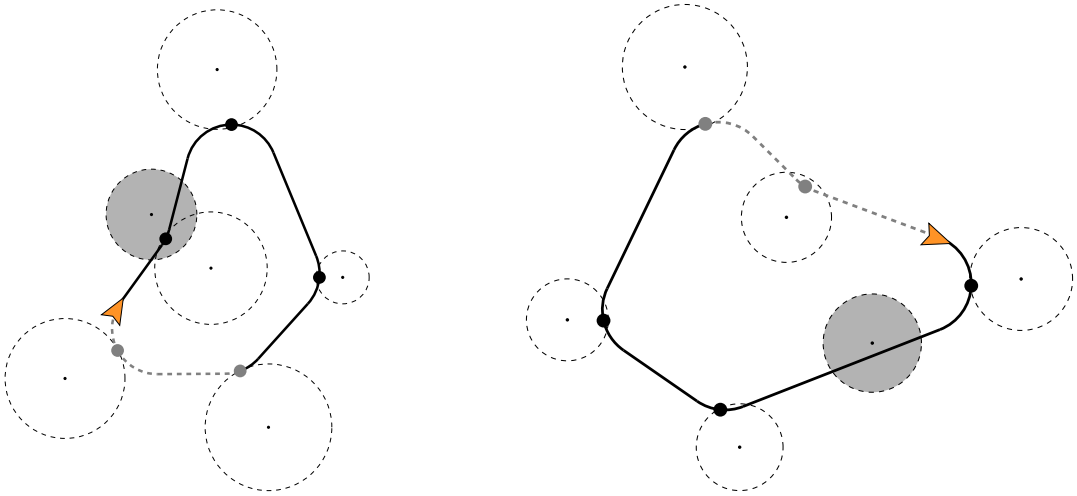
Assumindo um caminho relativamente eficiente obtido para a situação inicial do ambiente, o principal objetivo é realizar apenas as alterações necessárias e que irão resultar no menor impacto possível no caminho atual. O tratamento da metodologia proposta a cada nova demanda de acordo com essa classificação será melhor detalhado nas próximas seções.

4.2.1 Caminho Permanece Inalterado

Esta etapa da metodologia é responsável por tratar as novas demandas inseridas ao ambiente, mas que não demandam nenhuma alteração no atual caminho planejado e sendo executado pelos veículos.

São consideradas basicamente duas situações onde novas demandas não modificam o caminho: (i) um ponto de visita ativo (ou seja, ainda não visitado) no caminho se encontra ao alcance da área delimitada pelo raio da nova região inserida e (ii) a nova região possui interseção com uma determinada parte do caminho planejado atual que ainda não foi executada pelo veículo.

A Figura 4.26 apresenta duas instâncias distintas exemplificando as duas situações que não ocasionarão alteração no caminho. Os pontos de visita inativos (já visitados) e a parte do caminho já percorrida pelo veículo estão demarcados pela cor cinza e por uma linha tracejada. A Figura 4.26(a) apresenta o caso onde uma nova demanda foi adicionada ao ambiente e o ponto de visita ativo (preto) de uma região ainda não visitada encontra-se dentro da nova região. A Figura 4.26(b) exemplifica o caso em que a nova região possui interseção com uma parte do caminho que ainda não foi executado pelo veículo.



(a) Ponto de Visita ao Alcance (b) Interseção com Parte do Caminho

Figura 4.26. Exemplos de instâncias em que a inserção de uma nova região não necessariamente ocasionaria na alteração do caminho atual. (a) Situação em que um ponto de visita ativo se encontra dentro dos limites da região. (b) Caso em que a nova região possui interseção com alguma parte do caminho ainda não executada pelo veículo.

É interessante destacar, também, que o caso trivial, ou seja, a situação em que uma nova demanda é inserida no ambiente, e a posição atual do veículo já se encontra dentro da nova região, é inicialmente verificado antes da execução dos demais passos da metodologia. Nesse caso, um novo ponto de visita com a posição e orientação atuais do veículo é criado e imediatamente já marcado como visitado. Uma das razões para a criação desse ponto de visita está na possibilidade de se ter um histórico mais completo e próximo do que efetivamente foi executado.

4.2.1.1 Ponto de Visita Ativo ao Alcance

Esta seção apresenta o passo da metodologia responsável por verificar se algum ponto de visita atualmente ativo se encontra inserido na nova região inserida no ambiente.

Conforme pode-se inferir, trata-se de uma etapa bem simples e é solucionada pelo Algoritmo 6. O algoritmo deve percorrer toda a lista de pontos de visita ($\mathcal{P}$) verificando se algum ponto ainda ativo se encontra dentro da nova região ($\mathcal{H}_{\text{new}}$). Caso um ponto que atende à essas condições seja encontrado, a nova região é então relacionada como uma das regiões atendidas por aquele ponto de visita. O ponto de visita selecionado é utilizado como valor de retorno do método, sendo retornado o valor nulo caso nenhum ponto de visita seja escolhido.

Algoritmo 6 Relacionar Nova Região a Ponto de Visita ao Alcance($\mathcal{Q}$, $\mathcal{H}_{\text{new}}$)

for $i = 1$ **to** P **do**
  if ($\mathbf{p}_i$ está Ativo) **and** ($\|\mathbf{p}_i - \mathbf{n}_{\text{new}}\| \leq r_{\text{new}}$) **then**
    Relacionar a região $\mathcal{H}_{\text{new}}$ ao ponto de visita $\mathbf{p}_i$
    return $\mathbf{p}_i$
  end if
end for
return nulo

Trata-se de um método guloso em sua essência, onde o primeiro ponto de visita encontrado já é selecionado (mesmo que existam outros pontos que atendem às condições).

4.2.1.2 Interseção com Parte do Caminho Ativo

Esta seção apresenta o passo da metodologia responsável por verificar se a nova região inserida possui interseção com alguma parte do caminho atualmente ativa, ou seja, ainda não realizada pelo veículo.

A determinação da possível interseção da nova região com o caminho será realizada a partir de um processo de amostragem. É proposta uma solução numérica iterativa que funciona a partir da discretização do possível deslocamento do veículo ao longo do caminho, descrita pelo Algoritmo 7. Optou-se por essa solução por se tratar de uma alternativa mais simples em comparação à resolução dos diversos sistemas envolvidos em uma abordagem analítica. Além disso, é suficientemente genérica para lidar com qualquer tipo de região convexa (não apenas regiões circulares, conforme simplificado aqui).

Algoritmo 7 Criar Novo Ponto de Visita no Caminho Atual($\mathcal{Q}$, $\mathcal{H}_{\text{new}}$)

$l \leftarrow \mathcal{L}_\rho^{\text{veic}}$
while $l < \mathcal{L}_\rho$ **do**
  $\mathbf{q}_{\text{aux}} \leftarrow$ obterCoordenadasNaDistancia(l)
  if $\|\mathbf{p}_{\text{aux}} - \mathbf{n}_{\text{new}}\| \leq r_{\text{new}}$ **then**
    $\mathbf{q}_{\text{new}} \leftarrow \mathbf{q}_{\text{aux}}$
    $\mathcal{Q} \leftarrow \mathcal{Q} \bigcup$ Novo ponto de visita $\mathbf{q}_{\text{new}}$ relativo à $\mathcal{H}_{\text{new}}$
  end if
  $l \leftarrow l + \Delta l$
end while
return nulo

O parâmetro $\mathcal{L}_{\rho}^{\text{veic}}$ representa a distância atual já percorrida pelo veículo considerando o caminho inicial. Enquanto o final do caminho não é alcançado, é calculada a posição ($\mathbf{q}_{\text{aux}}$) que o veículo ocuparia no caminho caso ocorresse um deslocamento de valor l a partir de sua posição atual. Caso essa posição virtual esteja dentro dos limites da nova região inserida, um novo ponto de visita ($\mathbf{q}_{\text{new}}$) é inserido no caminho nessa exata posição. É importante mencionar que, os pontos virtuais além da informação de posição também possuem o valor de orientação que o veículo ocuparia naquela posição.

Define-se o novo ponto de visita na primeira amostra no interior da nova região por diferentes razões, tais como: (i) melhorar a eficiência do método, sem a necessidade de se pesquisar através de toda a região (ii) um ponto de visita quase na fronteira aumenta a probabilidade dele fazer parte de uma nova região inserida futuramente e (iii) permite uma maior liberdade para o resto do caminho ser modificado caso necessário.

É interessante ressaltar a importância de se criar um novo ponto de visita, mesmo a região já sendo atendida pelo caminho. Com a inserção de uma nova região, havendo a necessidade de se alterar o caminho, a região pode deixar de possuir essa interseção. Dessa forma, a adição do ponto de visita fixa a curva naquele ponto, ou seja, garantindo a visita da região.

A Figura 4.27 apresenta todas as variáveis utilizadas no algoritmo, provendo uma visão geral de seu funcionamento. A partir da posição atual do veículo, os círculos pretos menores representam as possíveis posições virtuais ($\mathbf{q}_{\text{aux}}$). A primeira posição virtual dentro da nova região é então transformada em um novo ponto de visita ($\mathbf{q}_{\text{new}}$), representado pelo círculo preto maior dentro da nova região.

Experimentos realizados mostraram que os resultados são consideravelmente precisos se for selecionado um tamanho de passo (Δl) significativamente menor que o menor raio r_{min} dentre todas as regiões (por exemplo, $r_{\text{min}}/100$). Apesar da escolha do tamanho do passo ser importante, em um caso extremo, caso nenhum $\mathbf{q}_{\text{aux}}$ seja determinado dentro da região, as etapas seguintes devem evitar que a região não seja visitada.

De maneira semelhante à etapa anterior, o método apresentado aqui também possui uma característica gulosa. Ou seja, mesmo que a nova região possua interseção com mais de uma parte do caminho, o novo ponto de visita será adicionado na parte do caminho mais próxima da posição atual do veículo.

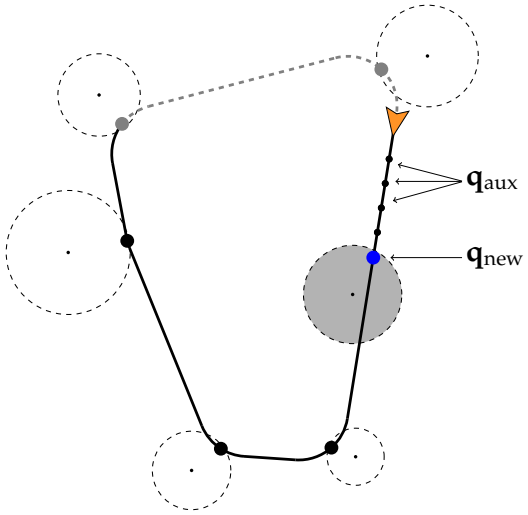


Figura 4.27. Exemplo de uma instância com a inserção de uma nova região e a possível execução do Algoritmo 7. São destacadas as posições virtuais ($\mathbf{q}_{\text{aux}}$) e a posição do novo ponto de visita ($\mathbf{q}_{\text{new}}$).

4.2.2 Caminho Sofre Alteração

Esta etapa da metodologia é responsável por alterar o atual caminho planejado e sendo executado pelo veículo de maneira a incorporar as novas demandas inseridas ao ambiente.

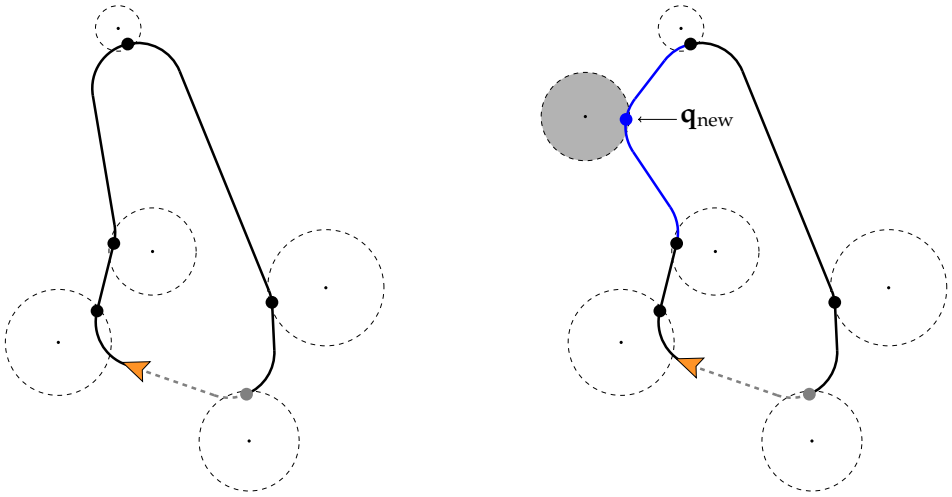
Após a inserção de uma nova demanda, serão considerados dois tipos diferentes de modificação no caminho: (i) a nova região é adicionada de forma que o movimento atual do veículo não sofre alteração, e (ii) o veículo altera o deslocamento atual em execução para visitar imediatamente a nova demanda. O valor do comprimento da deflexão (desvio) necessária para se visitar a nova região de acordo com cada tipo de caso é calculado, e o caso a ser utilizado é determinado priorizando sempre o menor comprimento.

O Algoritmo 8 apresenta os passos para se obter o menor custo de se adicionar a nova região ao caminho, sem alterar o movimento atual do veículo. A partir do próximo ponto de visita ativo (ainda não visitado), é calculado o custo de se inserir a nova demanda entre o ponto ativo e o ponto seguinte a esse. A posição do novo ponto de visita é calculado de maneira semelhante ao método proposto na Seção 4.1.2.2, e a orientação é determinada como o valor médio das orientações dos pontos vizinhos. Todos os desvios possíveis são calculados e o menor valor é retornado.

Algoritmo 8 Modificar Caminho com Novo Ponto de Visita($\mathcal{Q}$, $\mathcal{H}_{\text{new}}$)

1: $\text{Custo}_{\text{min}} = \infty$
2: **for** $i =$ Ordem do próximo ponto de visita ativo **to** P **do**
3: Criar ponto de visita $\mathbf{p}_{\text{aux}}$ no centro da região $\mathcal{H}_{\text{new}}$
4: Determinar a posição de $\mathbf{p}_{\text{aux}}$ de acordo com o Algoritmo 4
5: Atribuir orientação a $\mathbf{p}_{\text{aux}}$ de acordo com a Equação 4.11
6: $\text{Custo}_{\text{aux}} \leftarrow \mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{\text{aux}}) + \mathcal{D}_\rho(\mathbf{q}_{\text{aux}}, \mathbf{q}_{i+1})$
7: **if** $\text{Custo}_{\text{aux}} < \text{Custo}_{\text{min}}$ **then**
8: $\mathbf{q}_{\text{new}} \leftarrow \mathbf{q}_{\text{aux}}$
9: $\text{Custo}_{\text{min}} \leftarrow \text{Custo}_{\text{aux}}$
10: **end if**
11: **end for**
12: **return** $\mathbf{q}_{\text{new}}$, $\text{Custo}_{\text{min}}$

A Figura 4.28 exemplifica o caso abordado pelo Algoritmo 8. A Figura 4.28(a) representa a situação inicial do ambiente, com o veículo já se deslocando. A Figura 4.28(b) apresenta o caminho resultante após a inserção de uma nova região. A parte do caminho e o ponto, ambos em azul, representam as modificações realizadas no caminho. Como pode ser observado, foi determinado a partir da execução do algoritmo qual parte do caminho deveria ser modificada de forma a provocar o menor impacto possível no comprimento atual do caminho.



(a) Situação Inicial (b) Caminho Modificado

Figura 4.28. Exemplo de uma instância com a inserção de uma nova região e o resultado da execução do Algoritmo 8. Destacado em azul as modificações realizadas no caminho e o novo ponto de visita inserido ($\mathbf{q}_{\text{new}}$).

O Algoritmo 9 apresenta os passos utilizados para se obter o custo de se modificar o movimento atual do veículo, visitando a nova região imediatamente. Esse algoritmo é bem semelhante ao algoritmo previamente descrito, possuindo como principal diferença o ponto inicial utilizado no cálculo. É criado um ponto de visita virtual auxiliar na posição atual e com a orientação do veículo. Em seguida, calcula-se o custo de ir desse ponto até o ponto de visita otimizado relativo à nova região e finalmente o custo do caminho da nova região até o ponto de visita previamente definido como o próximo ponto que deveria ser visitado.

Algoritmo 9 Modificar Movimento Atual com Novo Ponto de Visita($\mathcal{Q}$, $\mathcal{H}_{\text{new}}$)

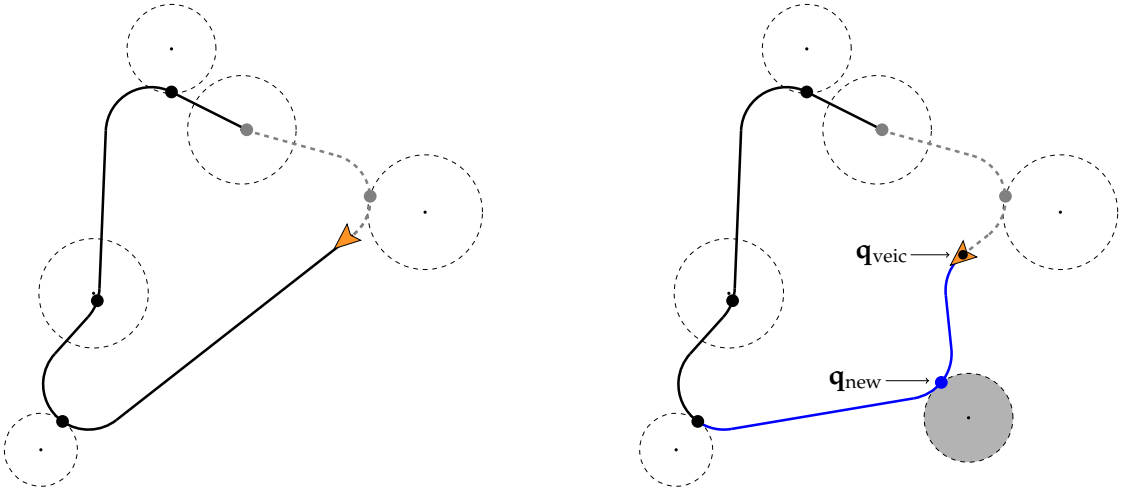
1: $prox \leftarrow$ Ordem do próximo ponto de visita ativo
2: Criar ponto de visita $\mathbf{q}_{\text{veic}}$ na posição atual do veículo e com mesma orientação
3: Criar ponto de visita $\mathbf{p}_{\text{new}}$ no centro da região $\mathcal{H}_{\text{new}}$
4: Determinar a posição de $\mathbf{p}_{\text{new}}$ de acordo com o Algoritmo 4
5: Atribuir orientação a $\mathbf{p}_{\text{new}}$ de acordo com a Equação 4.11
6: $\text{Cost}_{veic} \leftarrow \mathcal{D}_\rho(\mathbf{q}_{\text{veic}}, \mathbf{q}_{\text{new}}) + \mathcal{D}_\rho(\mathbf{q}_{\text{new}}, \mathbf{q}_{prox})$
7: **return** $\mathbf{q}_{\text{veic}}$, $\mathbf{q}_{\text{new}}$, Custo_{veic}

A Figura 4.29 exemplifica o caso abordado pelo Algoritmo 9. A Figura 4.29(a) representa a situação inicial do ambiente, com o veículo já se deslocando. A Figura 4.29(b) apresenta o caminho resultante após a inserção de uma nova região. Nesse caso, após a execução de ambos os algoritmos, foi determinado que o menor impacto no comprimento do caminho seria obtido se o veículo alterasse a direção atual de movimento e visitasse primeiramente a nova região inserida.

Novamente, a principal razão para a criação do ponto de visita na posição atual do veículo ($\mathbf{q}_{\text{veic}}$) está na possibilidade de se ter um histórico mais completo de toda a missão, e além disso, permitir recalcular o caminho efetivamente executado durante a navegação. Lembrando que a curva entre o último ponto visitado e o próximo ponto a ser visitado está associado à existência ou não de um ponto intermediário (*e.g.* $\mathbf{q}_{\text{new}}$) que também deve ser visitado.

4.2.3 Análise de Complexidade

Nesta seção é realizada a análise da complexidade computacional da abordagem proposta considerando o caso onde ocorre a inserção de novas regiões ao ambiente. Inicialmente é avaliado o comportamento assintótico relativo à execução do algoritmo, em seguida é discutido o limite superior para o comprimento do caminho.



(a) Situação Inicial (b) Movimento Atual Modificado

Figura 4.29. Exemplo de uma instância com a inserção de uma nova região e o resultado da execução do Algoritmo 9. Destacado em azul as modificações realizadas no caminho e o novo ponto de visita inserido ($\mathbf{q}_{\text{new}}$). Também é apresentado o novo ponto de visita inserido relativo à posição do veículo ($\mathbf{q}_{\text{veic}}$).

- **Caminho Permanece Inalterado**: A avaliação do caso trivial possui custo constante $O(1)$, uma vez que apenas é verificado se a distância da posição atual veículo para o centro da nova região é menor que o raio delimitador. O Algoritmo 6, no pior caso, deve percorrer todos os P pontos de visita, possuindo, então, custo linear $O(n)$, onde n representa o número de pontos de visita. Como o valor de passo utilizado no Algoritmo 7 é constante durante a execução, pode-se concluir que o seu custo será linear proporcional ao comprimento do caminho. Dessa forma, o custo final dessa etapa será $O(n_{\text{aux}})$, com n_{aux} representando o número de pontos de visita auxiliares ($\mathbf{q}_{\text{aux}}$) necessários para se alcançar o final do caminho a partir da posição atual do veículo. No pior caso, essa quantidade é obtida dividindo-se o comprimento do caminho pelo valor de passo.

- **Caminho Sofre Alteração**: Nessa etapa, O Algoritmo 8, no pior caso, deve percorrer todos os P pontos de visita ativos, possuindo então custo linear $O(n)$, onde n representa o número de pontos de visita ainda não visitados. O Algoritmo 9 possui custo $O(1)$, uma vez que todos os seus passos são compostos unicamente por operações aritméticas que são executadas em tempo constante.

Dessa forma, é possível concluir que no pior caso, o limite assintótico do método dinâmico considerando apenas um veículo será de $O(n) + O(m)$, onde n representa o número de pontos de visita auxiliares e m é o número de pontos de visita ativos.

A primeira parte dessa etapa (abordando o caso trivial, ponto de visita ao alcance e interseção com a curva) não altera o caminho, e consequentemente não possui nenhum impacto no comprimento final obtido. Na segunda parte, após a determinação do local do caminho que será modificado para atender à nova região, o segmento atual é substituído por duas novas curvas. Supondo, então, o caso em que o segmento representado por $\mathbf{q}_i \rightarrow \mathbf{q}_{i+1}$ foi substituído por $\mathbf{q}_i \rightarrow \mathbf{q}_{\text{new}}$ e $\mathbf{q}_{\text{new}} \rightarrow \mathbf{q}_{i+1}$. Relembrando por clareza, de acordo com o Teorema 3.4 de [Savla et al., 2008], o comprimento de uma curva Dubins ligando duas configurações possui um comprimento máximo dado por:

$$\mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_j) \leq \|\mathbf{p}_i - \mathbf{p}_j\| + \mathsf{k}\rho\pi, \tag{4.21}$$

onde $\mathsf{k} \in [2,657\ ,\ 2,658]$. Logo, a cada inserção de uma nova região no caminho $\mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{i+1})$ será substituído por $\mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{\text{new}})$ e $\mathcal{D}_\rho(\mathbf{q}_{\text{new}}, \mathbf{q}_{i+1})$, onde

$$\mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{i+1}) \leq \mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_{\text{new}}) + \mathcal{D}_\rho(\mathbf{q}_{\text{new}}, \mathbf{q}_{i+1}) \tag{4.22}$$

uma vez que a métrica utilizando-se curvas de Dubins satisfaz a desigualdade triangular [Yadlapalli et al., 2007].

Dessa forma, pode-se assumir que o comprimento médio aproximado do caminho em um instante de tempo t é obtido por

$$\mathcal{L}_\rho^t \lesssim \mathcal{L}_\rho^0 + (\lambda t)(2D_{\max}\mathsf{k}\rho\pi), \tag{4.23}$$

onde $\mathcal{L}_\rho^0$ representa o comprimento inicial do caminho para o caso estático e λt é o número médio esperado de novas regiões no intervalo de tempo $(0, t]$ de acordo com a taxa λ do processo de Poisson utilizado. O valor $D_{\max}$ é definido como

$$D_{\max} = \max_{\forall i,j \in [1,P]} \|\mathbf{p}_i - \mathbf{p}_j\| \tag{4.24}$$

e corresponde à distância euclidiana máxima entre dois pontos de visita presentes no caminho.

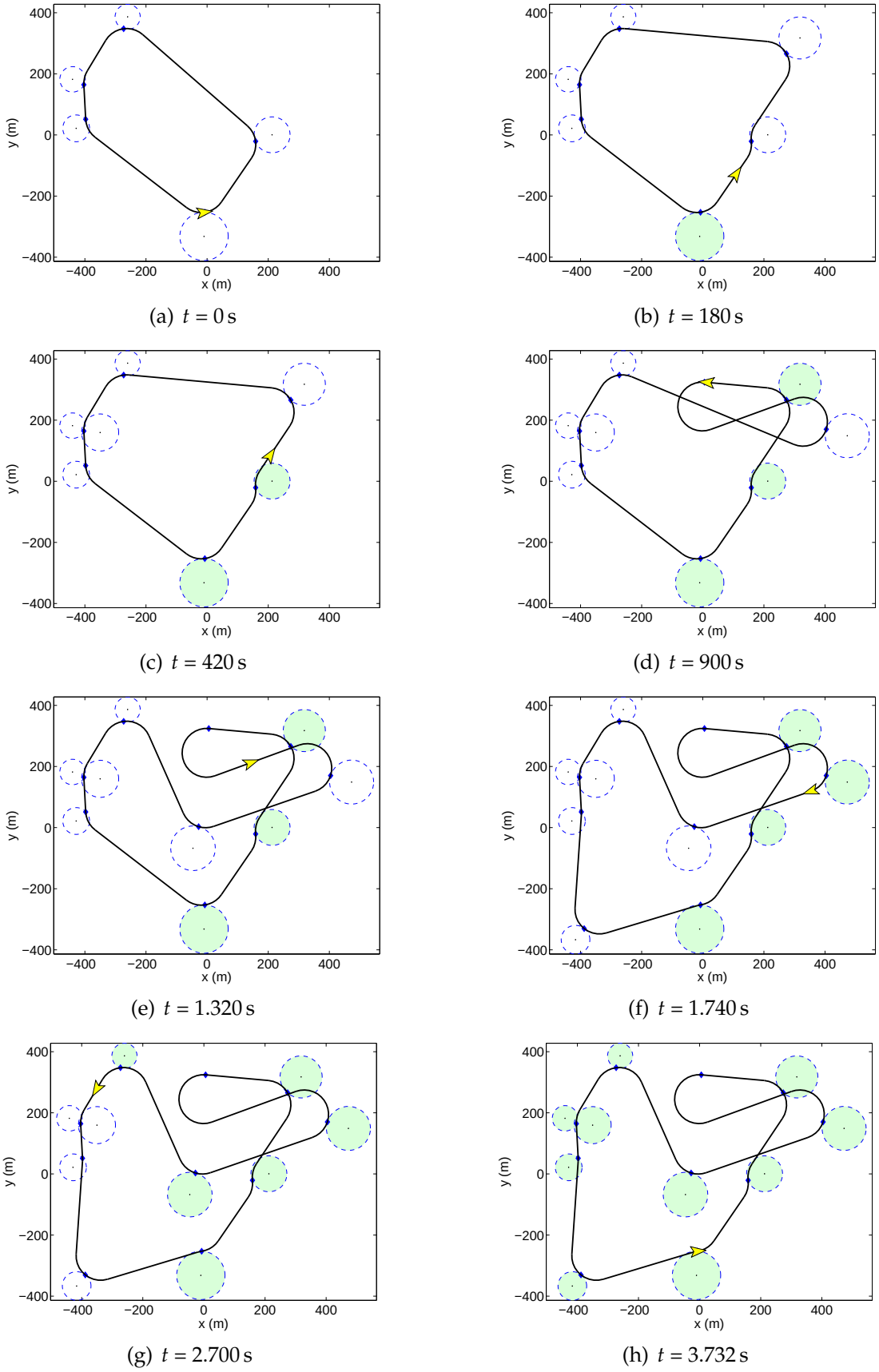
4.2.4 Experimentos

Nesta seção são apresentados os experimentos e suas respectivas análises estatísticas, verificando-se a eficácia do método de roteamento dinâmico proposto. É abordado o caso onde novas demandas não previamente conhecidas são determinadas no ambiente (caso dinâmico). Os resultados aqui obtidos representam uma visão geral de toda a metodologia considerando apenas um veículo, uma vez que a solução inicial é obtida a partir da metodologia desenvolvida na Seção 4.1.2.

Para execução das simulações foi desenvolvido um simulador em Matlab, e todos os experimentos foram executados em um PC com um processador Intel Core i7-2720QM 2.20 GHz, 8 Gb de RAM e o sistema operacional Ubuntu 11.10 64-bit. Todas as soluções para as instâncias relativas ao ETSP e ATSP foram calculadas à otimalidade utilizando-se a biblioteca `Concorde` [Concorde, 2011].

As imagens exibidas na Figura 4.30 apresentam um exemplo da execução da metodologia ao longo do tempo para uma determinada instância. Nesse exemplo, é utilizado um ambiente com dimensões $1.000\,\mathrm{m} \times 1.000\,\mathrm{m}$ contendo, inicialmente, 5 regiões aleatoriamente distribuídas de forma uniforme, o veículo simulado utilizado possui raio mínimo de curvatura $\rho = 80\,\mathrm{m}$ e se desloca com velocidade constante $v = 1\,\mathrm{m/s}$. O raio de cada região circular também foi determinado de forma aleatória dentro no intervalo $[40\,\mathrm{m}, 80\,\mathrm{m}]$. Foi utilizada uma taxa de $\lambda = \frac{1}{300}$, que significa que pelo menos uma nova região deve ser adicionada ao ambiente a cada 300 segundos. Também foi determinado que novas demandas seriam adicionadas no intervalo $(0\,\mathrm{s}, 2.000\,\mathrm{s}]$ (esse valor é utilizado pelo processo de Poisson para criar a agenda com os tempos de inserção).

A Figura 4.30(a) apresenta os alvos inicialmente já presentes no ambiente e o caminho originalmente planejado. No instante de tempo representado pela Figura 4.30(b) é possível observar o início do deslocamento do veículo em relação à posição inicial e a inserção de uma nova região, sendo necessário alterar o caminho. As regiões em destaque (verde) representam as demandas que já foram atendidas (visitadas). A Figura 4.30(c) apresenta a inserção de uma nova região que já possui um ponto de visita ao alcance, logo, o caminho não necessita ser modificado. A Figura 4.30(d) apresenta o caso onde a inserção de uma nova região resulta na alteração do movimento atual sendo realizado pelo veículo, dando prioridade de visita à região recém inserida. As Figuras 4.30(e) e 4.30(f) apresentam a inserção de novas regiões, novamente havendo a necessidade de se alterar o caminho. Finalmente, as Figuras 4.30(g) e 4.30(h) apresentam o restante da execução até que todas as regiões tenham sido visitadas e o veículo retorne à posição inicial.



(a) $t = 0\,\text{s}$ (b) $t = 180\,\text{s}$

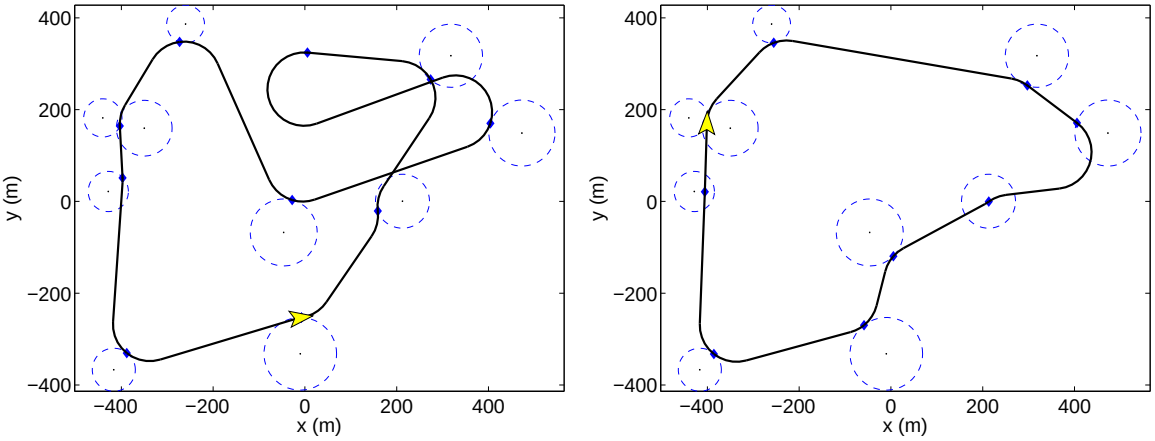
(c) $t = 420\,\text{s}$ (d) $t = 900\,\text{s}$

(e) $t = 1.320\,\text{s}$ (f) $t = 1.740\,\text{s}$

(g) $t = 2.700\,\text{s}$ (h) $t = 3.732\,\text{s}$

Figura 4.30. Exemplo inicial apresentando a execução da metodologia dinâmica ao longo do tempo para uma determinada instância. As regiões em destaque (verde) representam as demandas que já foram atendidas (regiões visitadas).

A Figura 4.31 apresenta uma comparação entre o caminho final obtido dinamicamente de forma incremental (Figura 4.31(a)) e o caminho que seria obtido pelo método estático caso todas as regiões já fossem conhecidas previamente (Figura 4.31(b)). Conforme pode ser observado, o não conhecimento de todas as informações inicialmente pode exercer um impacto considerável no resultado final, nesse caso específico um aumento de aproximadamente 46 % no comprimento do caminho.



(a) Dinâmico (3.728,25 m) (b) Estático (2.543,66 m)

Figura 4.31. Imagens evidenciando a perda de eficácia do algoritmo devido à falta de informações completas no início da execução. (a) Caminho obtido pelo algoritmo *online* de forma incremental. (b) Caminho obtido pelo método estático *offline*, caso em que todas as regiões são conhecidas previamente.

Uma das maneiras para se avaliar o desempenho de algoritmos denominados *online* (ou seja, aqueles que lidam com instâncias parcialmente conhecidas ao início e que são incrementadas gradualmente) consiste em realizar uma comparação com o resultado obtido por um algoritmo *offline*, onde todos os dados da instância são conhecidos *a priori*. A abordagem mais aceita e utilizada pelos trabalhos encontrados na literatura para se medir o desempenho de algoritmos *online* é denominada análise competitiva [Larsen et al., 2008]. Essa abordagem foi introduzida em [Sleator & Tarjan, 1985], e apresenta uma forma de se avaliar o desempenho dos algoritmos baseando-se no critério da razão de competitividade.

A razão de competitividade de um algoritmo é definido como a razão entre o pior resultado obtido pelo método *online* e o resultado ótimo obtido por um método *offline*. De maneira mais formal esse valor é definido como:

$$\mathbf{cr}_A = \sup_I \frac{z(A, I)}{z^*(I)}, \tag{4.25}$$

onde $z(A, I)$ é o custo do resultado encontrado por um algoritmo *online* A para a instância I, e z^* é o resultado ótimo encontrado por um algoritmo *offline* com conhecimento prévio de toda a instância I.

Define-se, então, que um algoritmo possui competitividade **c** (**c**-*competitive*) se a razão de competitividade é no máximo **c** [Jaillet & Wagner, 2006], ou seja

$$\text{Custo}_{\text{online}}(I) \leq \mathbf{c}\, \text{Custo}_{\text{ótimo offline}}(I), \qquad \forall \text{ instâncias } I. \tag{4.26}$$

Neste trabalho é realizada uma análise estatística para determinação da possível razão de competitividade da metodologia apresentada. Dessa forma, foram executados 300 experimentos em um ambiente com dimensões $1.000\,\text{m} \times 1.000\,\text{m}$ contendo inicialmente 5 regiões distribuídas uniformemente de maneira aleatória, onde o raio de cada região circular também foi escolhido aleatoriamente no intervalo $[40\,\text{m}, 100\,\text{m}]$. O veículo simulado possui um raio mínimo de curvatura de $\rho = 80\,\text{m}$, e se desloca com velocidade constante $\nu = 2\,\text{m/s}$. Foi utilizada uma taxa para inserção de novas regiões no valor de $\lambda = \frac{1}{120}$, e foi determinado que as novas inserções ocorreriam no intervalo $(0\,\text{s}, 3.000\,\text{s}]$. Uma vez que ainda não existe na literatura um algoritmo comprovadamente ótimo para o problema aqui abordado, a comparação será feita com a heurística determinística apresentada na Seção 4.1.2.

A análise competitiva avalia o desempenho do algoritmo sendo avaliado para o pior caso. Após a execução dos experimentos, o maior valor da razão de proporção entre o custo do caminho obtido a partir do algoritmo *online* e o caminho obtido pelo algoritmo *offline* foi 3,21. É importante ressaltar novamente que esse valor de competitividade não necessariamente representa o valor máximo possível, uma vez que esse valor foi determinado empiricamente para uma amostra da população.

Conforme mencionado, a análise competitiva é a abordagem avaliativa mais utilizada para verificação do desempenho de algoritmos *online*. Apesar disso, uma das principais críticas referentes a essa técnica está no fato de em grande parte dos casos a razão de competitividade ser um valor extremamente pessimista e não representativo do comportamento médio do algoritmo [Bullo et al., 2011]. Dessa forma, é apresentado na Figura 4.32 um histograma com os valores de razão encontrados,

permitindo uma melhor noção do comportamento geral do algoritmo.

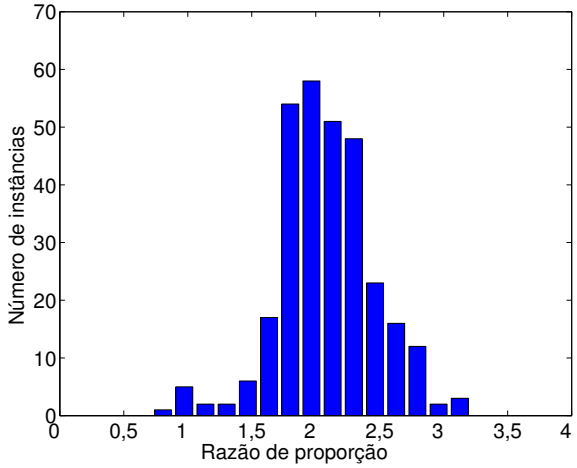


Figura 4.32. Histograma da razão de competitividade dos resultados obtido pelo algoritmo *online* em comparação aos do método *offline* (heurística).

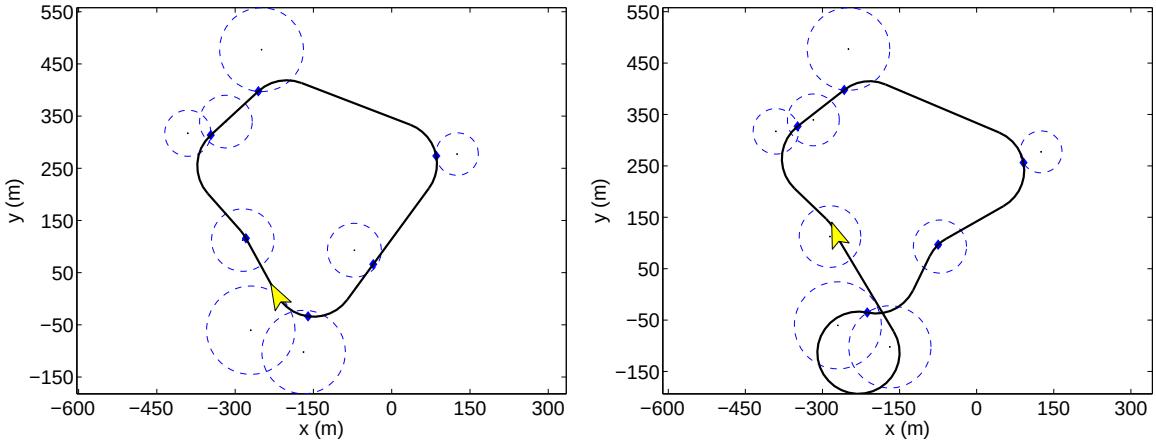
Como pode ser observado, o comportamento do algoritmo mostra-se melhor que o esperado a partir da análise competitiva, possuindo uma razão proporcional média consideravelmente menor que o valor proposto para a razão de competitividade. Na média, foi obtida uma razão com valor 2,08 (com desvio padrão de 0,38). Foi realizada uma análise utilizando-se o teste de hipótese *Teste t*, onde foi verificada a seguinte hipótese:

$$H_0 : \frac{\text{Custo}_{online}}{\text{Custo}_{offline}} = 1 \quad \text{vs.} \quad H_1 : \frac{\text{Custo}_{online}}{\text{Custo}_{offline}} > 1, \tag{4.27}$$

ou seja, a hipótese nula representa o caso em que ambos algoritmos possuiriam o mesmo desempenho. O resultado obtido informa com significância estatística que a hipótese nula foi rejeitada com $t(299) = 94.88$ e $p = 0$, e a razão de proporção encontra-se no intervalo (2,04, 2,12) com 95 % de confiança.

Um resultado interessante de se destacar foi o fato de que para um caso específico o caminho encontrado pelo algoritmo dinâmico foi menor que o obtido pelo algoritmo estático. Conforme pode ser observado na Figura 4.33, o principal motivo para isso foi o fato do algoritmo estático ter utilizado um ponto de visita na interseção das regiões dispostas na parte mais baixa do ambiente, o que acabou produzindo um caminho maior que o caminho passando por dois pontos de visita.

Permitindo uma avaliação comparativa do valor encontrado (3,21), a Tabela 4.4 apresenta as razões de competitividade dos algoritmos ótimos para as versões *online* dos problemas clássicos TSP e ATSP. Os primeiros métodos (não zelosos), são



(a) Dinâmico (1.368,95 m) (b) Estático (1.890,28 m)

Figura 4.33. Exemplo de uma instância em que o algoritmo dinâmico produziu um resultado melhor que o algoritmo estático. (a) Caminho obtido pelo algoritmo *online* de forma incremental. (b) Caminho obtido pelo método estático *offline*, caso em que todas as regiões são conhecidas previamente.

os casos em que o veículo pode permanecer parado na origem por um determinado tempo até que seja decidido o melhor momento para se iniciar a navegação. Os métodos denominados "zelosos" se referem aos casos em que o veículo não pode esperar, ou seja, deve permanecer em constante movimento caso ainda existam demandas não atendidas no ambiente [Ausiello et al., 2008] (caso mais próximo ao problema abordado nesse trabalho).

Tabela 4.4. Razões de competitividade dos algoritmos ótimos para as versões *online* dos problemas clássicos TSP e ATSP [Ausiello et al., 2008]. Os métodos zelosos se referem aos casos em que o veículo deve permanecer em movimento enquanto existirem demandas não atendidas no ambiente.

Problema	**Razão de competitividade**	
OL-TSP	2	[Ausiello et al., 2001]
OL-ATSP	$\frac{3+\sqrt{5}}{2}$	[Ausiello et al., 2008]
OL-TSP (zeloso)	2	[Ausiello et al., 2001]
OL-ATSP (zeloso)	3	[Ausiello et al., 2008]

É possível observar que a restrição do veículo permanecer em movimento não exerce influência no pior caso para o TSP. Entretanto, é importante considerar essa condição para o caso assimétrico, uma vez que nesse caso a razão de competitividade é dependente dessa restrição.

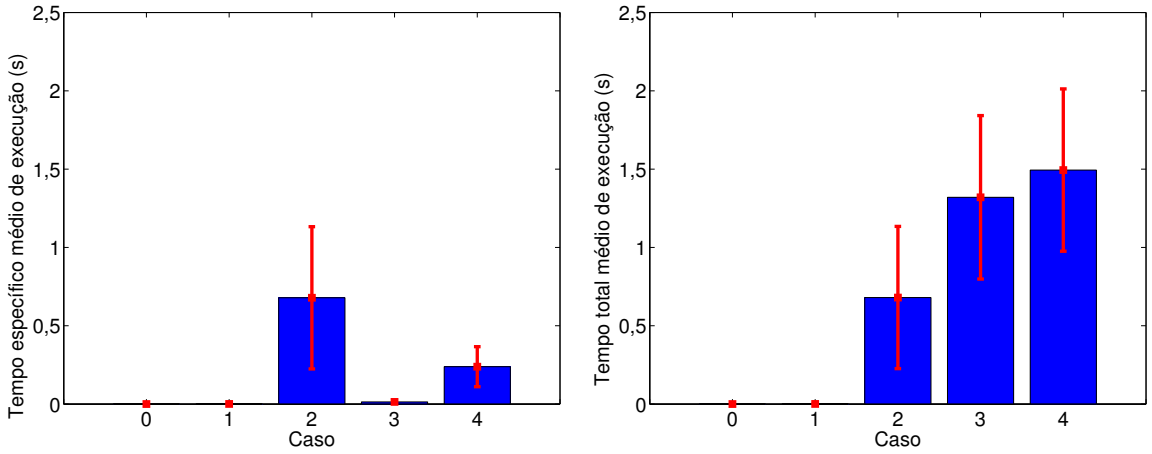
A razão de competitividade encontrada empiricamente para a metodologia proposta apresentou um valor ligeiramente superior ao valor ótimo (7 %). Apesar dessa comparação, é importante mencionar uma característica básica que difere a metodologia desse trabalho e o algoritmo para o OL-ATSP (zeloso). O algoritmo ótimo apresentado para o OL-ATSP (zeloso) calcula caminhos ótimos considerando as demandas atuais no ambiente, caso uma nova demanda seja inserida o veículo retorna à origem e repete a etapa anterior agora considerando as novas demandas inseridas. Nesse trabalho, decidiu-se que o veículo apenas deveria retornar à origem caso nenhuma nova demanda exista atualmente no ambiente. Essa decisão possui impacto na razão de competitividade, uma vez que nesse caso não são mais utilizadas subsoluções ótimas, o que certamente resultará em um aumento desse valor.

Finalmente, foi realizado um experimento com o objetivo de avaliar empiricamente o tempo de execução do algoritmo. Foram executados 200 experimentos em um ambiente com dimensões $2.000\,\mathrm{m} \times 2.000\,\mathrm{m}$ contendo inicialmente 5 regiões distribuídas uniformemente de maneira aleatória, onde o raio de cada região circular também foi escolhido aleatoriamente no intervalo $[40\,\mathrm{m}, 100\,\mathrm{m}]$. O veículo simulado possui um raio mínimo de curvatura de $\rho = 80\,\mathrm{m}$, e se desloca com velocidade constante $\nu = 2\,\mathrm{m/s}$. Foi utilizada uma taxa para inserção de novas regiões no valor de $\lambda = \frac{1}{120}$, e foi determinado que as novas inserções ocorreriam no intervalo $(0\,\mathrm{s}, 6.000\,\mathrm{s}]$. Ou seja, devem ser inseridas na média 50 novas regiões.

O tempo foi avaliado de acordo com os diferentes tipos de casos que podem ocorrer com a inserção de uma nova região no ambiente. Relembrando que a metodologia proposta considera os seguintes casos:

- **Caso 0:** A nova região já abrange a posição atual do veículo (caso trivial);
- **Caso 1:** Um ponto de visita no caminho se encontra ao alcance da região;
- **Caso 2:** A região inserida possui interseção com uma parte ativa do caminho;
- **Caso 3:** O veículo deve alterar o movimento atual de forma a já visitar a região;
- **Caso 4:** Uma parte ativa do caminho deve ser modificada para chegar à região.

A Figura 4.34 apresenta os valores médios mensurados para cada caso. A Figura 4.34(a) apresenta o tempo específico necessário para se calcular uma solução para um determinado caso. A Figura 4.34(b) considera o tempo total (acumulativo) para se determinar qual caso será utilizado após a inserção de uma nova região.



(a) Tempo específico (b) Tempo total

Figura 4.34. Tempo médio de execução do algoritmo dinâmico. (a) Valor considerando apenas o tempo específico de cada caso. (b) Valor total (acumulativo) para se determinar qual caso será utilizado.

É possível observar na Figura 4.34(a) que o Caso 2 e Caso 4 são os que demandam a maior quantidade de tempo para serem executados. O experimento serviu para confirmar o que já era esperado, uma vez que o Caso 2 deve avaliar diversas posições auxiliares ao longo do caminho e no Caso 4 deve-se calcular o custo de alterar o caminho para todos os pares de pontos ainda na parte ativa.

Os resultados apresentados na Figura 4.34(b) mostram que a metodologia é eficiente para ser efetivamente utilizada em um sistema real. No pior caso são necessários em média $\approx 1{,}5\,\text{s}$ para se determinar qual decisão tomar. Além de existirem partes na implementação passíveis de otimização, é importante mais uma vez relembrar que um código em Matlab não possui a mesma eficiência de outras linguagens (*e.g.* C++).

Ainda no contexto desse experimento, foi realizado um levantamento estatístico do número de alterações realizadas no caminho de acordo com cada caso (Figura 4.35). Como o número de regiões iniciais (cinco) é pequeno em relação ao tamanho do ambiente ($2.000\,\text{m} \times 2.000\,\text{m}$) e a quantidade média de novas regiões a serem inseridas (50), temos que em aproximadamente 65 % das novas inserções é necessário se alterar o caminho. É evidente que os parâmetros acima mencionados possuem impacto direto nessa estática, por exemplo, em um ambiente menor, maior é a chance de uma nova região já possuir contato com o caminho.

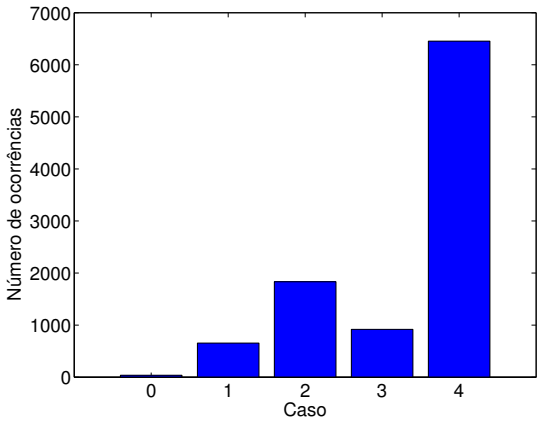


Figura 4.35. Histograma do número de ocorrências de cada caso.

Capítulo 5

Planejamento de Caminhos para Múltiplos Veículos

Nesta seção é apresentada e discutida a metodologia proposta para o problema de planejamento de caminhos de visita a regiões para o caso onde são utilizados múltiplos veículos não-holonômicos modelados como veículos Dubins.

Primeiramente é abordado o caso onde todas as regiões são conhecidas e permanecem inalteradas durante a execução (caso estático). Em seguida, o problema é estendido para o caso onde após o início da execução pelo veículo do caminho inicial, novas regiões são inseridas no ambiente (caso dinâmico).

5.1 Caso Estático

Nessa seção é apresentado o problema ao qual denominamos de k-TSP Dubins com Vizinhanças (*k-Dubins TSP with Neighborhoods*, ou k-DTSPN). O k-DTSPN consiste então em uma generalização para o TSP para o cenário onde se deve gerar rotas de visita a regiões no ambiente para um conjunto formado por k veículos Dubins que devem iniciar/finalizar a navegação em uma base em comum. A função de custo será o comprimento da maior rota, visando assim a redução do tempo de visita.

Inicialmente, é apresentado um algoritmo que calcula uma solução para o problema subdividindo-o em subproblemas, e solucionando cada um desses subproblemas utilizando-se técnicas clássicas da literatura. Em seguida, é utilizado um algoritmo memético onde a sequência de visita dos pontos que fazem parte de cada rota são modificados a cada iteração. Além disso, ocorre uma etapa de busca local com o objetivo de se encontrar a melhor posição e orientação dos pontos de visita internamente a uma rota.

5.1.1 *k*-DSPLITOUR

A primeira técnica que apresentamos com o objetivo de encontrar uma solução para o *k*-DTSPN é composta por dois passos básicos, baseados em técnicas encontradas na literatura. Inicialmente, uma solução para o TSP clássico (considerando o custo euclidiano entre os pontos) é calculada e em seguida separada em k diferentes rotas (passando a ser uma solução para o *k*-TSP). O segundo passo é responsável por calcular as curvas de Dubins entre as posições existentes em cada rota, tornando o caminho realizável por veículos com restrições de curvatura.

Para o cálculo da solução do *k*-TSP é utilizada uma técnica clássica da literatura denominada *k*-SPLITOUR [Frederickson et al., 1978]. Esse algoritmo foi o primeiro a obter soluções com um fator de aproximação constante da solução ótima, com um valor de $\varepsilon + 1 - \frac{1}{k}$, onde ε é a razão de aproximação do algoritmo utilizado para encontrar-se o circuito inicial (TSP). O método calcula as k diferentes rotas a partir da separação de um circuito inicial obtido como solução para o TSP. Os passos exatos da técnica são apresentados no Algoritmo 10.

Algoritmo 10 *k*-SPLITOUR($\boldsymbol{\mathcal{P}}$) [Frederickson et al., 1978]

1: $\boldsymbol{\mathcal{T}} \leftarrow \emptyset$
2: $\Sigma \leftarrow \text{ETSP}(\boldsymbol{\mathcal{P}})$
3: $\tau_0 = \boldsymbol{\mathcal{P}}_\Sigma$
4: **for** $j = 1$ **to** $k - 1$ **do**
5: Encontrar o último vértice $\mathbf{p}_{i(j)}$ tal que o custo da distância euclidiana de $\mathbf{p}_0$ a $\mathbf{p}_{i(j)}$ ao longo de τ_0 não é maior que $(j/k)(L - 2c_{\max}) + c_{\max}$ onde

$$c_{\max} = \max_{\forall\, n \,\in\, [1,P]} \|\mathbf{p}_0 - \mathbf{p}_n\|$$

e L é o comprimento do circuito euclidiano obtido a partir de τ_0

6: **end for**
7: Determinar e inserir as k rotas euclidianas em $\boldsymbol{\mathcal{T}}$ de acordo com

$$\begin{aligned}
\tau_1 &= (\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{i(1)}, \mathbf{p}_0), \\
\tau_2 &= (\mathbf{p}_0, \mathbf{p}_{i(1)+1}, \ldots, \mathbf{p}_{i(2)}, \mathbf{p}_0), \\
&\vdots \\
\tau_k &= (\mathbf{p}_0, \mathbf{p}_{i(k-1)}, \ldots, \mathbf{p}_P, \mathbf{q}_0)
\end{aligned}$$

8: **return** $\boldsymbol{\mathcal{T}}$

O comprimento de uma rota qualquer τ_j, $1 \leq j \leq k$ obtida utilizando-se o algoritmo k-SPLITOUR não excederá

$$\frac{1}{k}(L - 2c_{\max}) + 2c_{\max}, \tag{5.1}$$

onde k é o número de rotas em que o circuito inicial será separado, L representa o comprimento do circuito inicial, e $c_{\max}$ é a maior distância existente entre o ponto escolhido como base ($\mathbf{p}_0$) e todos os demais pontos. Esse valor também irá representar o custo médio de todas as rotas [Frederickson et al., 1978].

O segundo passo consiste em adaptar as rotas previamente obtidas, tornando-as realizáveis por veículos com restrições de curvatura. Para isso, iremos utilizar o Algoritmo Alternante [Savla et al., 2005b] para a atribuição das orientações em cada posição, permitindo assim em seguida o cálculo das curvas de Dubins. Esse método já foi apresentado anteriormente na Seção 4.1.1, entretanto, será brevemente relembrado aqui para uma maior clareza. Cada par de pontos da sequência é ligado utilizando-se segmentos de reta ou curvas de Dubins. A orientação ψ_p de um determinado ponto de visita p, é então determinada por:

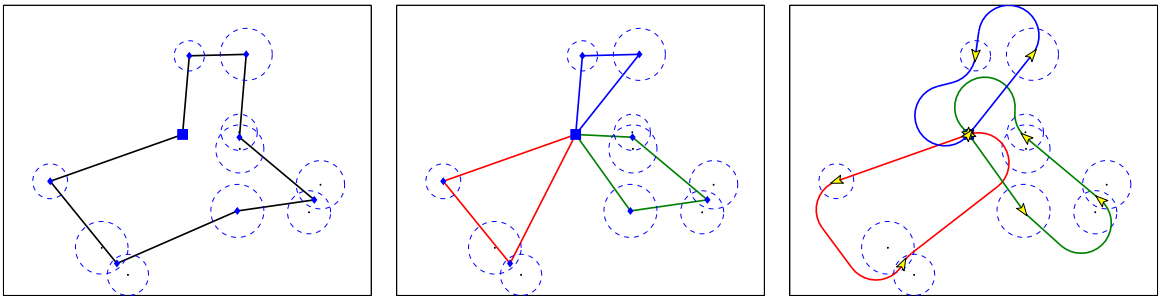
$$\psi_p = \begin{cases} \text{dir}(\mathbf{p}_p, \mathbf{p}_{p+1}) & \text{, se } p \text{ é ímpar} \\ \text{dir}(\mathbf{p}_{p-1}, \mathbf{p}_p) & \text{, se } p \text{ é par} \end{cases} \quad 1 \leq p \leq P. \tag{5.2}$$

Logo, propomos o Algoritmo 11 para encontrar uma solução para o k-DTSPN. Esse algoritmo é baseado nos passos previamente descritos, e além disso também possui uma etapa inicial responsável pela determinação dos pontos de visita a partir das regiões no ambiente. O método recebe como entrada um conjunto $\mathcal{H}$ de regiões distribuídas no ambiente, e retorna um conjunto de rotas $\mathcal{T}$ formadas por curvas de Dubins.

Algoritmo 11 k-DSPLITOUR($\mathcal{H}$)

1: Determinar o conjunto de pontos de visita $\mathcal{P}$ de acordo com o Algoritmo 3
2: Determinar o conjunto de rotas euclidianas $\mathcal{T}$ de acordo com o Algoritmo 10
3: Atribuir orientações ao pontos de visita em $\mathcal{T}$ de acordo com o Algoritmo 2
4: **return** $\mathcal{T}$

A Figura 5.1 apresenta um exemplo de uma possível execução do método para uma determinada instância. Cada uma das imagens apresenta as soluções obtidas após a execução de cada uma das etapas.



(a) (b) (c)

Figura 5.1. Exemplo dos caminhos resultantes em cada passo do Algoritmo 11. (a) Circuito inicial utilizando a métrica euclidiana; (b) rotas obtidas após a execução da etapa referente ao k-SPLITOUR; (c) caminhos Dubins finais obtidos após a atribuição das orientações pelo AA.

Nesse exemplo, 10 regiões foram aleatoriamente posicionadas no ambiente e no máximo 3 veículos poderiam ser utilizados na tarefa. A Figura 5.1(a) apresenta o circuito inicial utilizando a métrica euclidiana passando pelos pontos de visita determinados no primeiro passo (devido às interseções o número de pontos de visita é menor que o número de regiões). O circuito inicial é então separado em rotas utilizando-se o algoritmo k-SPLITOUR(Figura 5.1(b)). Finalmente, cada rota é transformada em um circuito Dubins, onde as orientações nos pontos de visita são determinadas utilizando-se o Algoritmo Alternante (Figura 5.1(c)).

5.1.1.1 Análise de Complexidade

Nesta seção é apresentada a análise da complexidade computacional do Algoritmo 11 (k-DSPLITOUR), utilizado para obter-se uma solução para o k-DTSPN. Inicialmente é abordado o comportamento assintótico relativo ao tempo de execução do algoritmo, em seguida, é discutido o limite superior para o comprimento do caminho.

O método é composto por três etapas fundamentais: (i) determinação dos pontos de visita; (ii) cálculo das k diferentes rotas euclidianas e (iii) atribuição de orientações e cálculo das curvas de Dubins.

- **Determinação dos pontos de visita:** Nessa etapa foi utilizado o Algoritmo 3, apresentando na Seção 4.1.2. Dessa forma, no pior caso (nenhuma região com interseção) esse algoritmo irá percorrer todo o conjunto de regiões, possuindo um comportamento assintótico de $O(n)$, onde n corresponde à quantidade N de regiões.

- **Cálculo das rotas euclidianas:** O primeiro passo do k-SPLITOUR consiste em encontrar uma solução para uma instância do ETSP, que é NP-difícil ($O(n!)$). A segunda parte do k-SPLITOUR possui complexidade linear, uma vez que todos os pontos de visita devem ser percorridos verificando-se o limite dado pela Equação 5.1 para a criação de mais uma rota. Dessa forma, apesar de existirem algoritmos exatos eficientes para se encontrar uma solução para uma instância do ETSP, o custo completo desse passo será exponencial.

- **Atribuição de orientações:** Nessa etapa é utilizado o Algoritmo Alternante. Esse algoritmo possui complexidade linear, uma vez que todos os pontos de visita são percorridos e a orientação é dada conforme a posição desse (par ou ímpar) na sequência de visita. Conforme visto anteriormente, as curvas de Dubins podem ser calculadas em tempo constante $O(1)$. Logo, essa etapa possui um comportamento assintótico de $O(n)$, onde n corresponde à quantidade P de pontos de visita.

Dessa forma, é possível concluir que no pior caso, o limite assintótico do k-DSPLITOUR será exponencial, dada a necessidade de se encontrar uma solução para uma instância do ETSP na etapa de cálculo das rotas euclidianas. Caso a sequência de visita já seja conhecida, o custo será $O(n)$, onde n representa o número de regiões, uma vez que $P \leq N$.

O limite superior para o comprimento da maior rota obtida utilizando-se o k-DSPLITOUR pode ser obtido baseando-se nos limites de ambas as técnicas utilizadas. Inicialmente, considera-se o comprimento de uma rota qualquer obtida pelo k-SPLITOUR é dado pela Equação 5.1. Conforme visto anteriormente, o comprimento do caminho obtido utilizando-se o Algoritmo Alternante é dado por

$$\mathcal{L}_{\rho}^{\mathrm{AA}} \leq \mathrm{ETSP}(\boldsymbol{\mathcal{P}}) + \left\lceil \frac{P}{2} \right\rceil \mathsf{k}\rho\pi. \tag{5.3}$$

Logo, é possível concluir que a maior rota possui um limite superior dado por

$$\mathcal{X} \leq \frac{1}{k}(L - 2c_{\max}) + 2c_{\max} + \left\lceil \frac{N}{2} \right\rceil \mathsf{k}\rho\pi. \tag{5.4}$$

onde L representa o comprimento do circuito inicial, ou seja, $L = \mathrm{ETSP}(\boldsymbol{\mathcal{P}}_{\mathrm{cent}})$, e $c_{\max}$ é a maior distância existente entre o ponto escolhido como base ($\mathbf{p}_0$) e todos os demais pontos.

5.1.2 Algoritmo Memético

O método proposto na seção anterior (*k*-DSPLITOUR) é de fácil implementação e obtém soluções razoáveis para o problema (considerando os limites de ambas as técnicas utilizadas). Entretanto, conforme pode ser observado, todas as rotas passam pelo centro das regiões, dessa forma não tirando vantagem de toda a vizinhança.

É evidente que este problema é intratável (o *k*-TSP é NP-difícil [Bektas, 2006]), e encontrar metodologias eficazes e eficientes é uma tarefa bastante desafiadora. Entre as soluções existentes para problemas semelhantes, e como visto para o caso de geração de caminhos para um veículo (Seção 4.1.1), os Algoritmos Evolutivos têm mostrado ser uma boa escolha.

Um grupo de algoritmos dentro do conjunto de EAs que tem se destacado mais recentemente são os chamados MAs (algumas vezes também referidos por Algoritmos Genéticos Híbridos). A principal característica dos MAs está no fato deles utilizarem uma otimização global baseada em uma população de indivíduos em conjunto com um método de busca local para auxiliar na exploração do espaço de buscas [Moscato, 1989; Moscato & Cotta, 2010].

A Figura 5.2 apresenta um fluxograma com uma visão geral dos passos de um Algoritmo Memético.

A busca local (parte híbrida do MA) possui como foco a parte contínua do problema, ou seja, a posição e orientação das configurações que serão utilizadas para o cálculo das curvas de Dubins. Os passos seguintes correspondem às etapas clássicas de um GA (*e.g.* Seleção, Cruzamento e Mutação), e são principalmente responsáveis por lidarem com a parte combinatória do problema, ou seja, a sequência de visita das configurações.

Uma etapa fundamental para se obter bons resultados com a utilização de GAs é a escolha de uma boa representação (cromossomo) para os indivíduos. Geralmente os indivíduos representam uma solução candidata válida para o problema sendo abordado. Neste trabalho, o cromossomo (Figura 5.3) foi definido como uma sequência (permutação) de configurações, que representam as posições que devem ser visitadas e as orientações a elas atribuídas. Dessa forma, um único cromossomo é capaz de representar o conjunto $\mathcal{T}$, formado por diferentes rotas. É importante observar que a base é representada como uma configuração diferente para cada uma das rotas, uma vez que decidiu-se por deixar a orientação livre nesse ponto (podendo assumir diferentes valores entre as rotas).

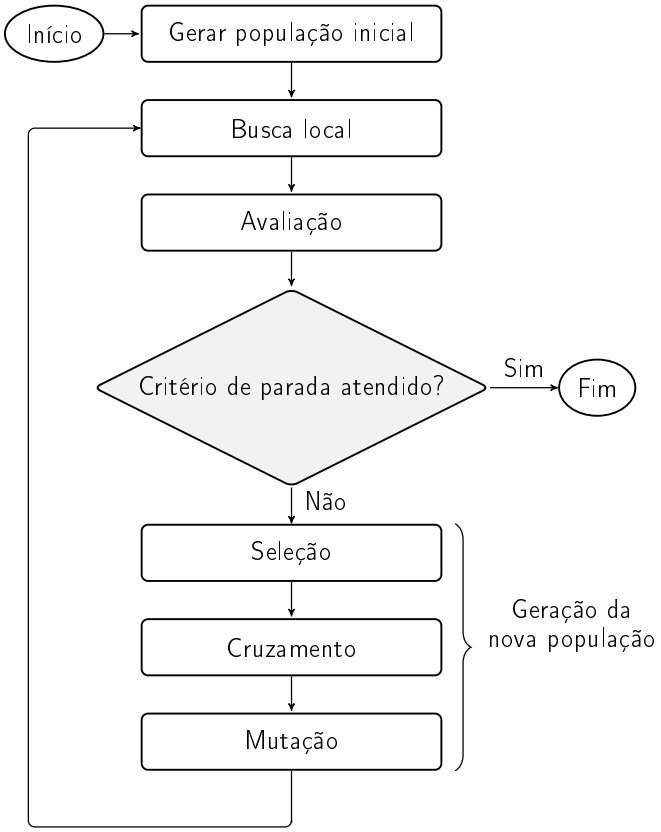


Figura 5.2. Fluxograma referente às etapas de um Algoritmo Memético.

$$\boldsymbol{\Omega} = \begin{array}{|c|c|c|c|c|c|c|} \hline \mathbf{q}_0^1 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_0^k & \cdots & \mathbf{q}_P \\ \hline \end{array}$$

Figura 5.3. Estrutura de um cromossomo.

Após a escolha da representação a ser utilizada para o cromossomo, é importante também a determinação de um método capaz de interpretar (decodificar) essas informações, obtendo assim uma solução final viável para o problema. É proposto então o Algoritmo 12, responsável por realizar a separação em rotas de um determinado cromossomo.

Dessa forma, considerando o seguinte cromossomo como exemplo

$$\begin{array}{|c|c|c|c|c|c|c|c|c|} \hline \mathbf{q}_0^1 & \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \mathbf{q}_0^2 & \mathbf{q}_4 & \mathbf{q}_0^3 & \mathbf{q}_5 & \mathbf{q}_6 \\ \hline \end{array},$$

Algoritmo 12 Decodificar Cromossomo($\mathbf{\Omega}$)

```
1: r ← 1
2: b ← 0
3: 𝒯 ← Criar e inserir rota vazia τ_r
4: for i = 1 to |Ω| do
5:     if Ω^[i] é um ponto de base then
6:         if b == 1 then
7:             Replicar a configuração τ_r^[1] no final da rota
8:             r ← r + 1
9:             𝒯 ← Criar e inserir rota vazia τ_r
10:        else
11:            b ← 1
12:        end if
13:    end if
14:    Adicionar configuração Ω^[i] ao final da rota τ_r
15: end for
16: return 𝒯
```

após a decodificação obtemos $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$, onde cada rota é formada por

$$\tau_1 = \langle \mathbf{q}_0^1,\ \mathbf{q}_1,\ \mathbf{q}_2,\ \mathbf{q}_3,\ \mathbf{q}_0^1 \rangle,$$
$$\tau_2 = \langle \mathbf{q}_0^2,\ \mathbf{q}_4,\ \mathbf{q}_0^2 \rangle,$$
$$\tau_3 = \langle \mathbf{q}_0^3,\ \mathbf{q}_5,\ \mathbf{q}_6,\ \mathbf{q}_0^3 \rangle.$$

É importante destacar que, de acordo com o algoritmo, não necessariamente uma configuração da base será o ponto inicial e final de uma rota. Esse fato não possui nenhum impacto para nosso método como um todo, uma vez que o caminho em si permanecerá o mesmo. Bastaria que, no momento da execução do caminho pelo veículo, a permutação fosse modificada (sem alterar a ordem de visita) colocando o ponto da base nas extremidades da rota.

Além disso, dependendo da configuração do cromossomo, uma rota pode não possuir nenhum outro ponto de visita alocado à ela. Nesses casos, o veículo poderia decidir por apenas permanecer parado na base. O próprio método se encarregará de eliminar essas soluções durante a execução, uma vez que o objetivo é minimizar o comprimento da maior rota, e assim fazendo com que haja uma repartição mais igualitária dos pontos de visita.

Logo, considerando essas situações e o seguinte cromossomo como exemplo

$\mathbf{q}_1$	$\mathbf{q}_2$	$\mathbf{q}_0^1$	$\mathbf{q}_3$	$\mathbf{q}_0^2$	$\mathbf{q}_4$	$\mathbf{q}_5$	$\mathbf{q}_6$	$\mathbf{q}_0^3$

,

após a decodificação obtemos $\boldsymbol{\mathcal{T}} = \{\tau_1, \tau_2, \tau_3\}$, onde cada rota é formada por

$$\begin{aligned}\tau_1 &= \langle \mathbf{q}_1,\ \mathbf{q}_2,\ \mathbf{q}_0^1,\ \mathbf{q}_3,\ \mathbf{q}_1 \rangle,\\ \tau_2 &= \langle \mathbf{q}_0^2,\ \mathbf{q}_4,\ \mathbf{q}_5,\ \mathbf{q}_6,\ \mathbf{q}_0^2 \rangle,\\ \tau_3 &= \langle \mathbf{q}_0^3,\ \mathbf{q}_0^3 \rangle.\end{aligned}$$

A população inicial será obtida a partir da geração aleatória de indivíduos, ou seja, escolhendo-se diferentes sequências de visita (consequentemente diferentes rotas). Em seguida, cada um desses indivíduos é decodificado e é utilizado o Algoritmo Alternante para atribuir as orientações iniciais a cada uma das configurações, uma vez que essa técnica fornece uma boa estimativa inicial. Apenas a sequência é escolhida aleatoriamente, as posições iniciais dos pontos de visita em si são determinadas nas regiões de interseção.

Após a geração de uma nova população (inicial ou intermediária) estar completa, um procedimento de busca local é então executado com o objetivo de melhorar o valor da *fitness* (aptidão) de cada indivíduo. Lembrando que em nosso caso a *fitness* de um indivíduo corresponde ao comprimento da maior rota obtida a partir desse cromossomo, ou seja, a função de *fitness* $\mathcal{F}(\cdot)$ é calculada como

$$\mathcal{F}(\boldsymbol{\Omega}) = \max_{1 \leq i \leq |\boldsymbol{\mathcal{T}}|} \mathcal{L}_\rho(\boldsymbol{\mathcal{T}}^{[i]}), \tag{5.5}$$

onde $\boldsymbol{\mathcal{T}}$ é o conjunto de rotas obtidas após a decodificação do cromossomo.

A busca local é composta por dois passos básicos: (i) otimização da posição da configuração dentro da região a qual pertence, minimizando o comprimento do circuito euclidiano formado por todas as configurações de uma rota; (ii) otimização das orientações das configurações a fim de se encontrar o menor circuito, formado por curvas de Dubins, ligando as posições. Cada um dos passos da busca local será efetuado utilizando-se o Algoritmo 4 e Algoritmo 5, respectivamente.

Finalmente, o Algoritmo 13 apresenta os passos gerais do método proposto para encontrar-se uma solução para o k-DTSPN utilizando um MA.

O método recebe como entrada um conjunto $\boldsymbol{\mathcal{H}}$ de regiões distribuídas no ambiente, e retorna um conjunto de rotas $\boldsymbol{\mathcal{T}}$ formadas por curvas de Dubins. Durante a execução, a população $\boldsymbol{\mathcal{G}} = \{\boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \ldots, \boldsymbol{\Omega}_I\}$ é constantemente modificada com o objetivo de sempre melhorar os indivíduos atuais. É importante destacar que durante a etapa de geração de uma nova população sempre é avaliado se o novo indivíduo sendo criado é válido, ou seja, corresponde a uma solução válida para o problema (*e.g.* não possui mais rotas que veículos disponíveis).

Algoritmo 13 Algoritmo Memético para o k-DTSPN($\mathcal{H}$)

1: $\mathcal{P} \leftarrow$ Determinar pontos de visita de acordo com o Algoritmo 3
2: $\mathcal{G}_{\text{pos}} \leftarrow$ Gerar população inicial sem orientações baseando-se em $\mathcal{P}$
3: $\mathcal{G} \leftarrow$ Decodificar rotas de $\mathcal{G}_{\text{pos}}$ e atribuir orientações utilizando o Algoritmo 2
4: **while** critério de parada não atendido **do**
5: $\quad \widehat{\mathcal{G}} \leftarrow$ Executar busca local nos indivíduos de $\mathcal{G}$
6: $\quad \mathcal{G} \leftarrow$ Gerar nova população a partir de $\widehat{\mathcal{G}}$
7: **end while**
8: $\mathbf{\Omega}^* \leftarrow$ Determinar indivíduo em $\mathcal{G}$ com a melhor *fitness*
9: $\mathcal{T} \leftarrow$ Decodificar $\mathbf{\Omega}^*$ de acordo com o Algoritmo 12
10: **return** $\mathcal{T}$

5.1.2.1 Análise de Complexidade

Nesta seção é apresentada a análise da complexidade computacional do Algoritmo 13, que faz uso de um Algoritmo Memético para determinar uma solução para k-DTSPN. Inicialmente é abordado o comportamento assintótico relativo ao tempo de execução do algoritmo, em seguida, é discutido o limite superior para o comprimento do caminho (maior rota).

- **Determinação dos pontos de visita:** Nessa etapa foi utilizado o Algoritmo 3, apresentando na Seção 4.1.2. Conforme já visto anteriormente, o algoritmo possui um comportamento assintótico de $O(n)$, onde n corresponde à quantidade N de regiões.

- **Geração da população inicial:** Para cada um dos indivíduos que será inserido na população deve-se gerar uma permutação aleatória dos pontos de visita e determinar uma orientação aleatória. Logo, essa etapa possui complexidade linear $O(n)$, onde n corresponde à quantidade I de indivíduos que se deseja utilizar.

- **Busca local:** Nessa etapa são utilizados o Algoritmo 4 e Algoritmo 5. Conforme já visto anteriormente, ambos os algoritmos possuem custo linear $O(n)$, onde n corresponde ao número P de pontos de visita.

- **Geração de uma nova população:** Assumindo que os operadores (seleção, cruzamento e mutação) utilizados nessa etapa são os mais básicos encontrados na literatura, a geração de um novo indivíduo possui custo constante $O(1)$. Entretanto, é preciso substituir todos os indivíduos atuais da população, logo, essa etapa possuirá complexidade $O(n)$, onde n corresponde à quantidade I de indivíduos.

- **Avaliação de um população:** Essa etapa é responsável por percorrer todos os indivíduos da população e calcular o valor da *fitness* para cada um deles. A primeira parte já infere um custo pelo menos linear $O(n)$, onde n corresponde à quantidade I de indivíduos. A decodificação de um cromossomo possui complexidade $O(m + p)$, onde m corresponde ao número P de pontos de visita e p ao número k de veículos (cada veículo representa um novo ponto de base). Assumindo um número constante de veículos, esse valor pode ser reduzido então à $O(m)$. Logo, o custo final dessa etapa será $O(nm)$.

Dessa forma, é possível concluir que, no pior caso, o limite assintótico do Algoritmo 13 será de $O(nm)$, onde n representa o número de regiões ($P \leq N$) e m corresponde à quantidade de indivíduos na população. Assumindo uma quantidade constante de indivíduos é possível então obter-se um custo total linear, dependente apenas do número de regiões no ambiente.

Conforme visto anteriormente, não existem garantias de que $|\boldsymbol{\mathcal{T}}| = |\boldsymbol{\mathcal{V}}|$, ou seja, é possível que um determinado indivíduo, de acordo com seu cromossomo, possua apenas uma rota visitando todas regiões. Ao particionar essa rota inicial em subrotas, serão encontrados caminhos menores, uma vez que a métrica utilizando-se curvas de Dubins satisfaz a desigualdade triangular. Além disso, o melhor indivíduo será substituído apenas se um indivíduo mais apto for encontrado.

Dessa forma, pode-se afirmar que no pior caso, o comprimento da maior rota obtida pelo Algoritmo Memético possuirá um limite superior dado por

$$\mathcal{X} \leq \mathrm{ETSP}(\boldsymbol{\mathcal{P}}_{\mathrm{cent}}) + \left\lceil \frac{N}{2} \right\rceil \mathsf{k}\rho\pi, \tag{5.6}$$

onde $\boldsymbol{\mathcal{P}}_{\mathrm{cent}}$ representa a posição inicial dos pontos de visita (no centro das regiões), e considerando-se o caso em que $N > 2$ e $\rho > 0$. Ao utilizar o Algoritmo Alternante para a inicialização das orientações podemos afirmar que metade dos caminhos conectando dois pontos será um segmento de reta e a outra metade será formada por curvas de Dubins.

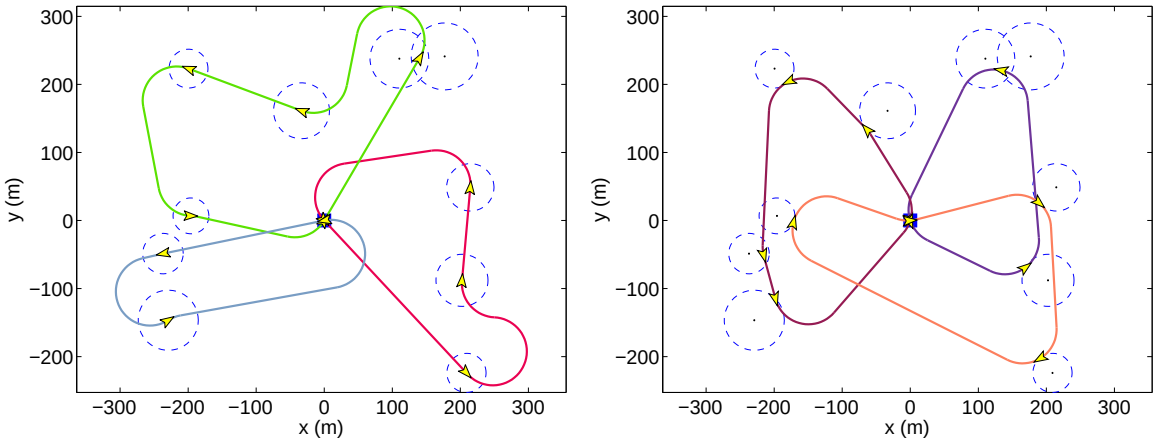
5.1.3 Experimentos

Nesta seção são apresentados os resultados obtidos a partir de simulações numéricas utilizando-se a metodologia proposta para o caso onde são utilizados múltiplos veículos e as demandas são previamente conhecidas, não ocorrendo a inserção de novas demandas ao ambiente (caso estático).

Para execução das simulações foi desenvolvido um simulador em Matlab, e todos os experimentos foram executados em um PC com um processador Intel Core i7-2720QM 2.20 GHz, 8 Gb de RAM e o sistema operacional Ubuntu 11.10 64-bit. Todas as soluções para as instâncias relativas ao ETSP foram calculadas à otimalidade utilizando-se a biblioteca `Concorde` [Concorde, 2011].

Inicialmente é apresentado um exemplo de possíveis resultados para o k-DTSPN obtidos pelas técnicas propostas para uma instância composta por poucas regiões (para uma melhor visualização). Foi utilizado um ambiente com dimensões $500\,\text{m} \times 500\,\text{m}$, onde 10 regiões foram distribuídas aleatoriamente de maneira uniforme. Cada região possui um raio que foi determinado no intervalo $[25\,\text{m}, 50\,\text{m}]$. Foi determinada a utilização de, no máximo, 3 veículos, cada um possuindo um raio mínimo de curvatura $\rho = 50\,\text{m}$.

A Figura 5.4(a) e Figura 5.4(b) apresentam os caminhos obtidos a partir do k-DSPLITOUR e do Algoritmo Memético, respectivamente. É possível observar claramente na Figura 5.4(b) a redução no comprimento das rotas obtidas como resultado da busca local, responsável por encontrar um melhor posicionamento e orientação dos pontos de visita.



(a) k-DSPLITOUR (1.337,16 m) (b) MA (1.075,15 m)

Figura 5.4. Exemplo inicial apresentando os possíveis caminhos resultantes obtidos pelos métodos apresentados. (a) k-DSPLITOUR e (b) Algoritmo Memético. O valor entre parênteses representa o comprimento da maior rota.

Como o MA é uma técnica probabilística, é importante a realização de uma análise estatística, verificado o comportamento geral do algoritmo para um número significativo de execuções. Dessa forma, foram executados 200 experimentos em um ambiente com dimensões $1.500\,\text{m} \times 1.500\,\text{m}$ contendo um total de 30 regiões distribuídas uniformemente de maneira aleatória, onde o raio de cada região foi escolhido aleatoriamente no intervalo $[50\,\text{m}, 100\,\text{m}]$. Também foi definido como 3 o número máximo de veículos que poderiam ser utilizados, cada veículo possuindo um raio mínimo de curvatura $\rho = 100\,\text{m}$. A Tabela 5.1 apresenta os demais parâmetros específicos relacionados à execução do algoritmo.

Tabela 5.1. Parâmetros utilizados na execução do Algoritmo Memético.

Parameter	Value
Tamanho da população	100
Número de gerações	50
Método de seleção de pais	Torneio de tamanho 2
Elitismo	Sim
Probabilidade de cruzamento	0.8
Operador de cruzamento	*Order-based* (OX2)
Probabilidade de mutação	0.5
Operador de mutação	Inversão simples

A Figura 5.5 apresenta o histograma relativo ao percentual de redução do comprimento da maior rota obtida pelo MA em comparação a comprimento da maior rota resultante do k-DSPLITOUR, para esse caso. É possível observar que o MA produziu na média caminhos 20% menores que o k-DSPLITOUR para 97% das instâncias, com um desvio padrão de aproximadamente 8%.

Além disso, foi utilizado o teste de hipótese *Teste t* pareado, onde foram verificadas as seguintes hipóteses:

$$H_0 : \mu_{\text{MA}} - \mu_{\text{KD}} = 0 \quad \text{vs.} \quad H_1 : \mu_{\text{MA}} - \mu_{\text{KD}} < 0, \tag{5.7}$$

onde μ_{MA} e μ_{KD} representam o comprimento médio do caminho obtido utilizando-se o MA e k-DSPLITOUR, respectivamente. Ou seja, deseja-se verificar se a redução média do caminho possui significância estatística. Foi verificado que o valor obtido possui significância estatística com $t(199) = 26.64$ e $p = 0$.

Em seguida, é apresentado na Figura 5.6 um exemplo contendo os resultados de ambos os métodos para uma instância contendo um número maior de regiões e veículos disponíveis. Nesse exemplo, podem ser utilizados no máximo 10 veículos

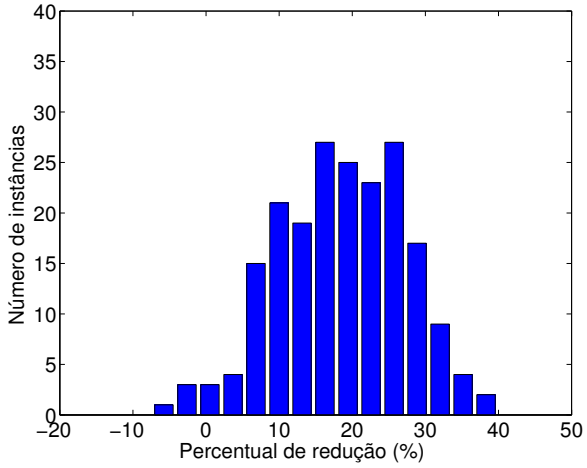
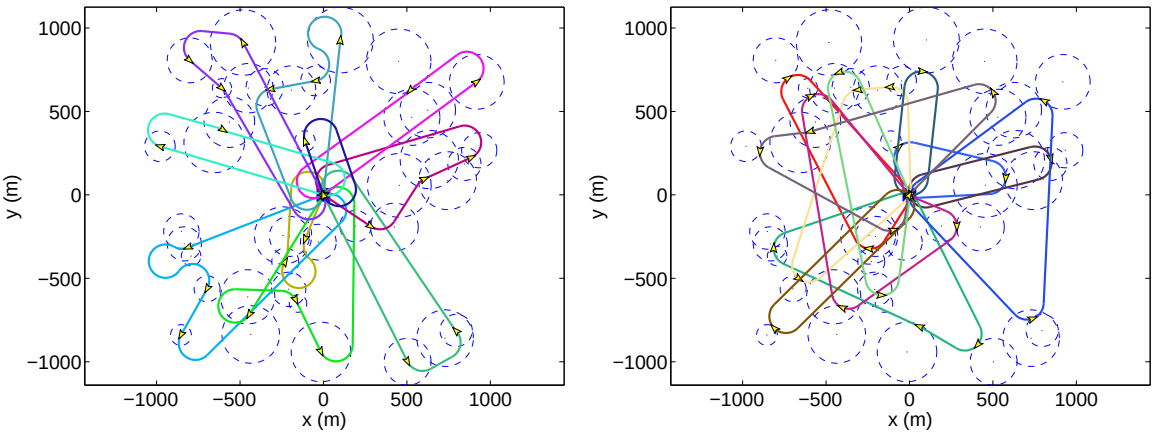


Figura 5.5. Histograma do percentual de redução do da maior rota obtida pelo Algoritmo Memético em comparação a rota resultante do *k*-DSPLITOUR.

para se visitar 40 regiões distribuídas uniformemente de maneira aleatória em um ambiente com dimensões 2.000 m $\times$ 2.000 m.



(a) *k*-DSPLITOUR (3.731,60 m) (b) MA (3.536,35 m)

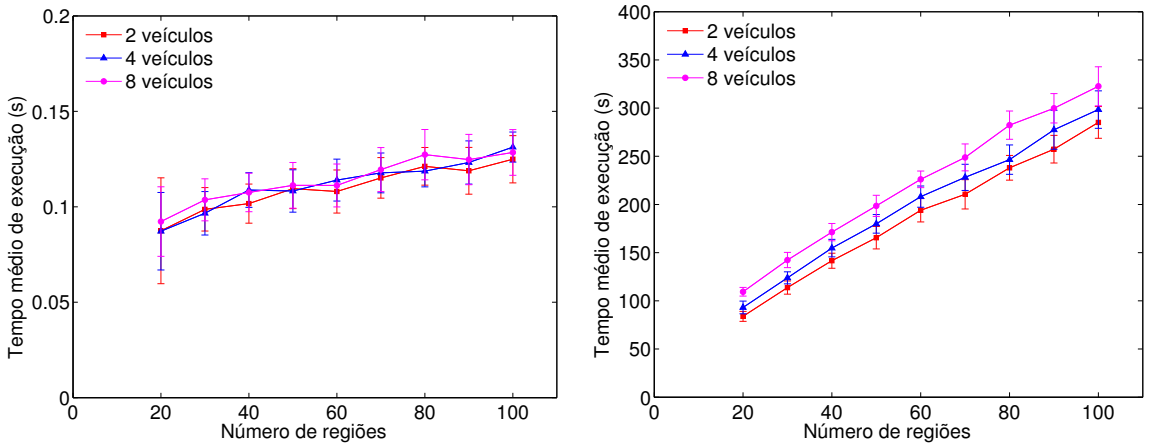
Figura 5.6. Exemplo dos possíveis caminhos resultantes obtidos utilizando-se um conjunto de 10 veículos. (a) *k*-DSPLITOUR e (b) Algoritmo Memético.

É importante destacar um resultado interessante, em que mesmo com um número relativamente pequeno de gerações (50) quando comparado com o número geralmente utilizado em algoritmos evolutivos (centenas ou milhares de gerações), o Algoritmo Memético foi capaz de obter soluções satisfatórias. Entretanto, é evidente que ao se executar o algoritmo durante um número maior de gerações existe a possibilidade de se encontrar soluções ainda melhores.

Em seguida foi realizado um experimento com o objetivo de avaliar empiricamente o comportamento assintótico do tempo de execução do algoritmo e a qua-

lidade dos resultados obtidos (comprimento da maior rota). Foram realizadas simulações com a quantidade de regiões variando no conjunto $\{20, 30, ..., 100\}$, e para cada valor 30 instâncias aleatórias foram geradas. Foi utilizado um ambiente com dimensões 2.500 m $\times$ 2.500 m e o raio de cada região foi escolhido aleatoriamente no intervalo $[50\,\text{m}, 100\,\text{m}]$. Para cada instância o número máximo de veículos a ser utilizado foi escolhido no conjunto $\{2, 4, 8\}$, todos possuindo um raio mínimo de curvatura de $\rho = 100\,\text{m}$. Os parâmetros específicos utilizados pelo MA foram os mesmos apresentados na Tabela 5.1, exceto o tamanho da população, agora composta por 50 indivíduos.

A Figura 5.7 apresenta o tempo médio de execução mensurado para cada quantidade de regiões. Conforme esperado, o MA apresentou um comportamento linear (Figura 5.7(b)), quando considerada a quantidade de regiões como o único parâmetro variável. Entretanto, é possível observar que o *k*-DSPLITOUR é executado em um tempo consideralvemente menor (Figura 5.7(a)). Além disso, o impacto do número de veículos no tempo de execução do *k*-DSPLITOUR é insignificante, o mesmo não podendo ser visto no MA.

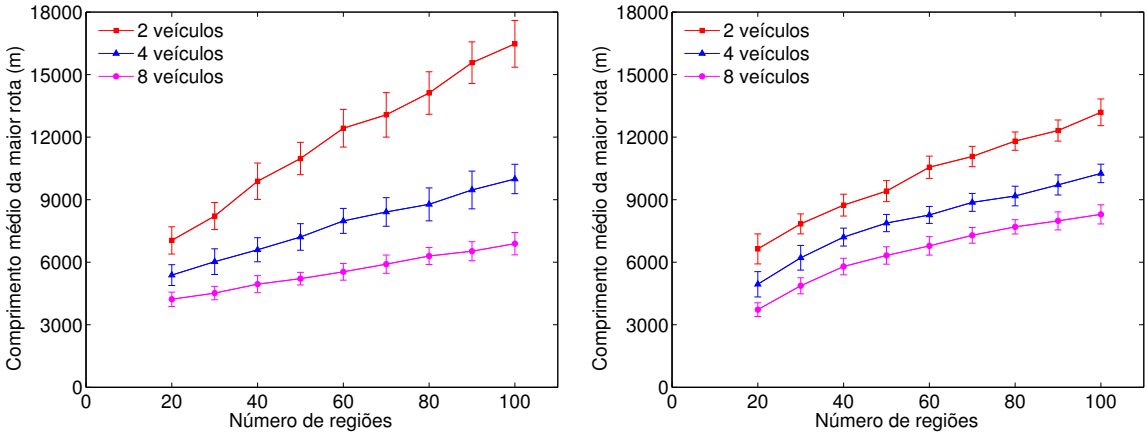


(a) *k*-DSPLITOUR (b) MA

Figura 5.7. Tempo médio de execução de acordo com a quantidade de regiões no ambiente e número de veículos. No caso do MA o valor corresponde ao tempo total necessário para se executar 50 gerações (critério de parada) com uma população formada por 100 indivíduos.

A Figura 5.8 apresenta o comprimento médio da maior rota considerando a quantidade de regiões. É possível observar que com uma quantidade menor de regiões e veículos o MA obtém resultados melhores. Entretanto, com o aumento nessa quantidade os resultados do *k*-DSPLITOUR passam a ser ligeiramente superiores. Uma das razões para isso está na utilização de um baixo número de gerações, uma

vez que com o maior número de veículos seria necessário mais iterações para se encontrar boas sequências (permutações) de visita para todas as rotas.



(a) *k*-DSPLITOUR (b) MA

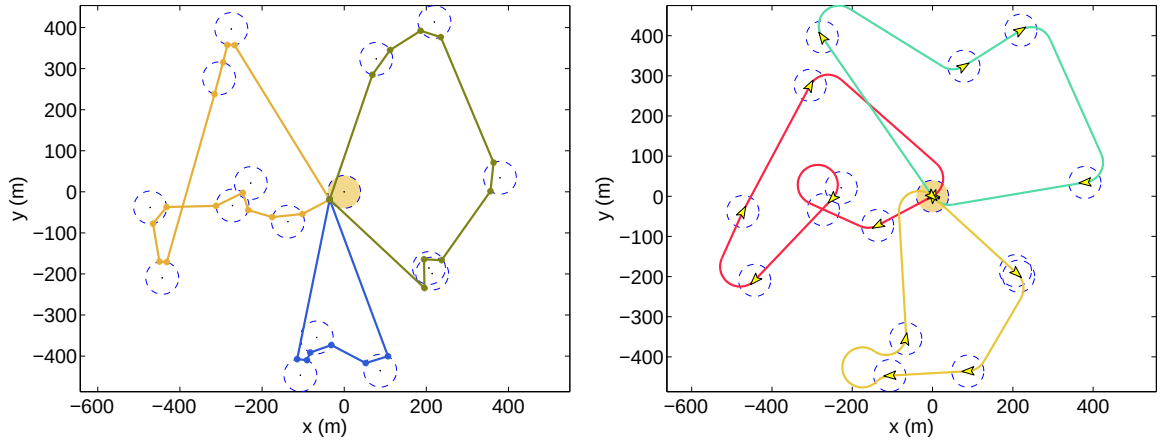
Figura 5.8. Comportamento assintótico do comprimento médio da maior rota de acordo com a quantidade de regiões no ambiente e número de veículos.

Dessa forma, cabe ao usuário decidir sobre optar entre o tempo de computação em detrimento de qualidade da solução, ajustando por exemplo o tamanho da população, número de gerações e demais parâmetros do MA. Uma solução interessante seria a inserção do resultado obtido a partir do *k*-DSPLITOUR como um dos indivíduos da população inicial do MA, garantindo-o como o resultado final caso não sejam encontrados resultados melhores (considerando a utilização de elitismo).

Conforme mencionado anteriormente, não foram encontrados trabalhos que abordem o exato problema aqui apresentado (*k*-DTSPN). Dessa forma, optou-se por realizar uma comparação com a técnica tratando o problema mais próximo, o *k*-TSPN. Em [Bhadauria et al., 2011] é apresentado o algoritmo denominado DGP-Tour, utilizado na coleta de dados em redes de sensores sem fio por robôs móveis. A técnica também considera, no custo final do caminho, o tempo utilizado para o *download* da informação de cada sensor, entretanto, durante os experimentos de comparação esse valor foi considerado como zero.

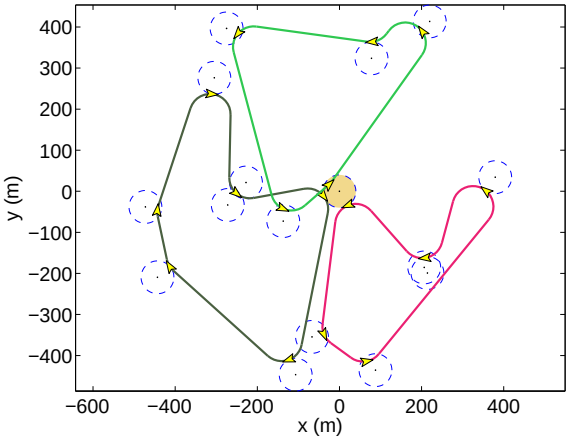
A Figura 5.9 apresenta os caminhos resultantes obtidos a partir das técnicas que serão utilizadas para comparação para uma instância exemplo, permitindo, assim, uma ideia do comportamento de cada uma. É importante destacar que a técnica DGPTour demanda que o raio de todas as regiões seja o mesmo, inclusive para a posição que representa a base (destacada em laranja na figura). Nesse exemplo, é utilizado um ambiente com dimensões 1.000 m $\times$ 1.000 m contendo 15 regiões aleatoriamente distribuídas de forma uniforme, todas possuindo um raio de 40 m. Foi

determinada a utilização de no máximo 3 veículos, cada um possuindo um raio mínimo de curvatura $\rho = 50\,\text{m}$.



(a) DGPTour (1.619,23 m)

(b) k-DSPLITOUR (2.002,40 m)



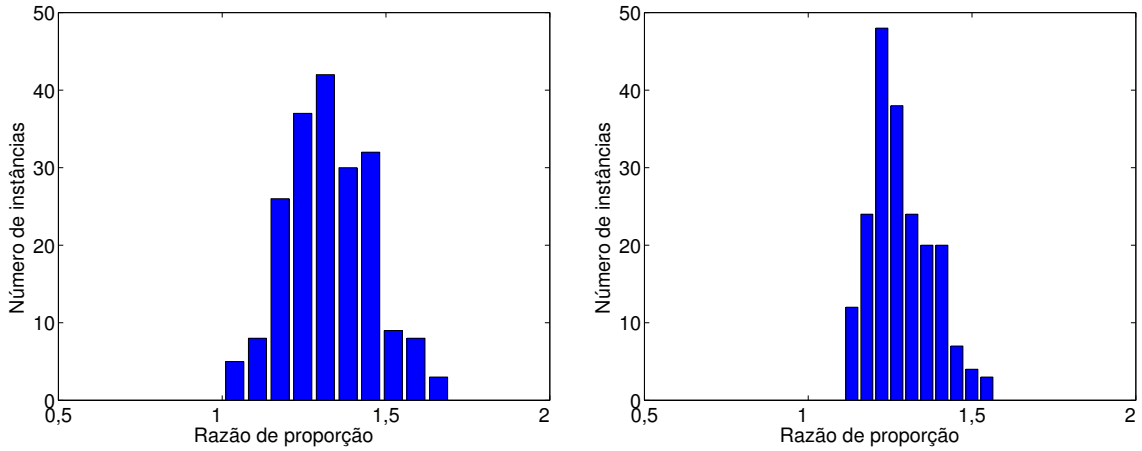
(c) MA (1.762,95 m)

Figura 5.9. Exemplo inicial apresentando possíveis caminhos resultantes obtidos pelas diferentes técnicas comparadas nessa etapa. (a) DGPTour [Bhadauria et al., 2011]; (b) k-DSPLITOUR e (c) MA. O círculo destacado em laranja representa a posição base.

Foram executados 200 experimentos comparativos, considerando um ambiente com dimensões 1.000 m $\times$ 1.000 m. Em cada experimento 40 regiões foram distribuídas uniformemente de maneira aleatória, todas possuindo um raio de 40 m. Foi determinada a utilização de no máximo 4 veículos, cada um possuindo um raio mínimo de curvatura $\rho = 50\,\text{m}$. Os parâmetros específicos utilizados pelo MA foram os mesmos apresentados na Tabela 5.1. Para cada instância o MA foi executado 10 vezes, e a média dos valores encontrados foi utilizada na comparação.

A Figura 5.10 apresenta os histogramas relativos à razão de proporção do comprimento da maior rota encontrada pelos métodos propostos (k-DSPLITOUR e MA)

em relação à técnica sendo comparada (DGPTour).



(a) k-DSPLITOUR vs. DGPTour (b) MA vs. DGPTour

Figura 5.10. Histogramas da razão de proporção do comprimento da maior rota. (a) k-DSPLITOUR vs. DGPTour e (b) MA vs. DGPTour.

A razão média encontrada pelo k-DSPLITOUR foi de $\approx$ 1,33, enquanto o MA apresentou, na média, valores $\approx$ 1,28 maiores que a maior rota encontrada pelo DGPTour. Entretanto, é interessante observar que apesar de o k-DSPLITOUR apresentar resultados médios piores que os do MA, em determinados casos ele conseguiu encontrar caminhos melhores, com razão de proporção pouco acima de 1.

Para se dimensionar melhor o que esses valores representam, é proposta a análise do impacto que haveria no comprimento do maior caminho obtido pelo DGPTour caso todas as arestas fossem substituídas por curvas de Dubins. Conforme visto anteriormente, de acordo com o Teorema 3.4 de [Savla et al., 2008], o comprimento de uma curva Dubins ligando duas configurações possui um comprimento máximo dado por:

$$\mathcal{D}_\rho(\mathbf{q}_i, \mathbf{q}_j) \leq \|\mathbf{p}_i - \mathbf{p}_j\| + \mathsf{k}\rho\pi, \tag{5.8}$$

onde $\mathsf{k} \in [2,657\ ,\ 2,658]$.

Dessa forma, considerando que a maior rota obtida pelo DGPTour é denominada $\text{DGPTour}_{\max}$ e possui P pontos, pode-se obter o limite superior máximo esperado para o impacto no comprimento resultante da utilização de curvas de Dubins conectando os pontos como

$$\text{Aumento máximo esperado} = \left(\frac{P\mathsf{k}\rho\pi}{\text{DGPTour}_{\max}} + 1\right) * 100. \tag{5.9}$$

Considerando que neste experimento os veículos simulados possuem $\rho = 50\,\text{m}$, a Figura 5.11 apresenta um histograma dos percentuais de aumento máximo esperado para todos os experimentos.

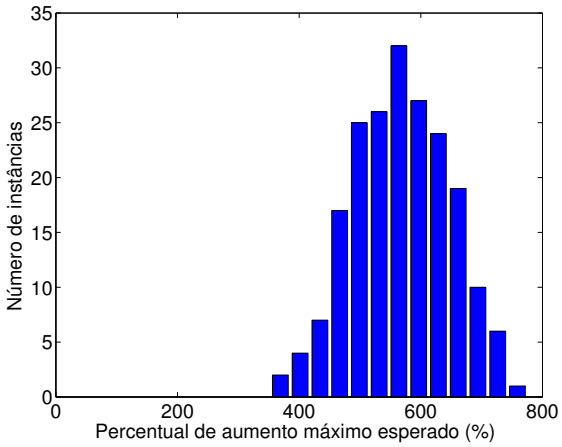


Figura 5.11. Histograma do percentual de aumento máximo esperado, considerando a substituição das arestas do maior caminho obtido pelo DGPTour por curvas de Dubins.

Como pode ser observado, no pior caso, é esperado um aumento médio de 6 vezes do comprimento do caminho. Esse valor pode ser interpretado como um limite superior do impacto ao substituir-se todas as arestas por curvas de Dubins. Logo, é possível concluir que os métodos propostos aqui apresentaram excelentes resultados, inferindo um impacto consideravelmente menor que o máximo esperado.

5.2 Caso Dinâmico

Nessa seção é apresentado o problema ao qual denominamos de k-TSP Dubins com Vizinhanças Dinâmico (*k-Dynamic Dubins TSP with Neighborhoods*, ou k-DDTSPN). O k-DDTSPN consiste em uma generalização para o k-DTSPN, apresentado na seção anterior. Neste cenário deve-se gerar rotas de visita a regiões, estáticas ou determinadas dinamicamente no ambiente, para um conjunto formado por k veículos Dubins. Os veículos devem iniciar e terminar a navegação em uma base central em comum, e o principal objetivo continua sendo a minimização da maior rota, que indiretamente representa o tempo necessário para se visitar as regiões.

É apresentado um algoritmo baseado no método de leilão para a alocação da nova região a uma rota já associada a um determinado veículo. O método apresentado é totalmente distribuído, não possuindo então a necessidade de se utilizar o papel de um leiloeiro central.

5.2.1 Leilão Descentralizado

Em um cenário estático (como o apresentado na seção anterior), é possível assumir que o cálculo da missão (no caso as rotas) é realizado inicialmente e então alocado para cada um dos veículos (e essa não será alterada durante a execução). Entretanto, em um cenário dinâmico, com a determinação de uma nova região de interesse no ambiente, é necessário tomar uma decisão durante a execução da missão, ou seja, qual veículo será o responsável por visitar a região. Dessa forma, ocorre a necessidade de haver algum tipo de coordenação entre os veículos.

Uma abordagem bastante utilizada na resolução do problema da coordenação de múltiplos agentes é baseada em modelos econômicos de mercado, mais especificamente o mecanismo de leilão [Dias et al., 2006]. Em um leilão, um conjunto de itens é oferecido por um leiloeiro, e os participantes podem efetuar lances de acordo com a pretensão de se obter tais itens. Em seguida, após todos os lances terem sido coletados, cabe ao leiloeiro determinar o vencedor de cada item.

Entretanto, conforme visto, um dos principais problemas do método de leilão clássico é a utilização de uma entidade centralizadora para a determinação dos vencedores, o leiloeiro. Dessa forma, caso algum problema ocorra com esse agente específico, o funcionamento de todo o sistema ficaria comprometido. Logo, é evidente a necessidade de aprimorar-se o método, tornando-o mais robusto a falhas individuais.

Considerando esse problema, diversos trabalhos apresentam extensões ao método de leilão modificando-o de forma a utilizá-lo de forma distribuída e descentralizada [Zavlanos et al., 2008; Michael et al., 2008; Choi et al., 2009], ou seja, removendo a necessidade de um leiloeiro central.

Neste trabalho são utilizados conceitos semelhantes aos trabalhos previamente mencionados. Ao determinar-se dinamicamente uma nova região de interesse, essa informação é enviada a todos os veículos. Cada veículo então calcula um valor de lance e o envia por *broadcast* para todos os demais veículos no ambiente. Após receber todos os lances, cada veículo executa um leilão internamente e determina o vencedor. O veículo vencedor então deve modificar sua rota de forma a visitar a nova região.

Suposição 4: *Assume-se uma rede de comunicação perfeita entre os veículos. Dessa forma, ao realizar-se um broadcast não haverá nenhuma perda de mensagens e a transmissão ocorre de forma instantânea (sem atrasos consideráveis).*

Para se garantir um método completamente descentralizado, uma etapa fundamental é a escolha das informações que irão compor um lance. Essas informações devem ser suficientes para que todos os veículos participantes do leilão possam sempre tomar a mesma decisão a respeito do lance vencedor. Além disso, deve-se dar prioridade a um número menor de campos, reduzindo o tamanho da mensagem e tornando-a mais fácil (e rápida) de ser enviada. Assim, define-se que um determinado lance **b** é uma tupla composta pelas seguintes informações:

$$\mathbf{b} = \langle \text{custo, tipo do lance, campo de desempate, id do veículo} \rangle,$$

onde

- **Custo**: Representa o custo para que o veículo visite a nova região. Caso o caminho não necessite ser modificado, de acordo com os casos discutidos na Seção 4.2.1, o lance possuirá custo zero. Caso o caminho previamente calculado necessite ser modificado, o custo associado ao lance será o valor do caminho ainda a ser percorrido considerando a alteração necessária, dado por

$$\hat{\mathcal{L}}_{\rho}^{\text{rem}} = \hat{\mathcal{L}}_{\rho} - \mathcal{L}_{\rho}^{\text{veic}}, \tag{5.10}$$

  onde $\hat{\mathcal{L}}_{\rho}$ representa o novo comprimento do caminho caso a alteração fosse realizada e $\mathcal{L}_{\rho}^{\text{veic}}$ é a distância já percorrida pelo veículo do caminho inicial. Ao utilizar o valor restante, o algoritmo conseguirá manter um melhor balanceamento dos caminhos entre os veículos, e consequentemente, otimizando o tempo total de visita.

- **Tipo do lance**: O tipo do lance é diretamente mapeado de acordo com o tipo de alteração que deve ser realizada para se visitar a nova região. Cada tipo recebe um identificador dentre os apresentados na Tabela 5.2, que será utilizado principalmente como um dos critérios de desempate.

- **Campo de desempate**: Parâmetro utilizado para o desempate caso os parâmetros anteriores não consigam garantir a unicidade do lance vencedor. Esse campo será definido de acordo com o tipo do lance, sendo:

  - **Tipo 0**: Determinado como o número total de pontos de visita (ativos e não ativos) no caminho atual do veículo, representado por $|\boldsymbol{\mathcal{Q}}_{\text{veic}}|$.
  - **Tipo 1**: Determinado como o número de regiões já associadas a um ponto de visita $\mathbf{q}_p$ determinado como ao alcance da nova região, representado por $|\boldsymbol{\mathcal{H}}_{\mathbf{q}_p}|$, onde

$$\boldsymbol{\mathcal{H}}_{\mathbf{q}_p} = \{\mathcal{H}_i \in \boldsymbol{\mathcal{H}}_{\mathbf{q}_p} : \mathbf{q}_p \in \mathcal{H}_i,\ \forall\ \mathcal{H}_i \in \boldsymbol{\mathcal{H}}\}. \tag{5.11}$$

  - **Tipo 2**: Determinado como a distância da posição atual veículo até a nova região, seguindo o caminho atual do veículo. Obtida considerando o valor da variável L, presente do Algoritmo 7, ao final da execução caso seja detectada uma interseção do caminho com a nova região.
  - **Tipo 3** e **4**: Determinado como o número total de pontos de visita ativos no caminho atual do veículo, representado por $|\boldsymbol{\mathcal{Q}}_{\text{veic}}^{\text{ativos}}|$.

  Em todos os casos a prioridade é dada ao lance que possuir o campo de desempate com o menor valor.

- **Id do veículo**: Identificador único de cada veículo definido como $\mathcal{V}_{\text{id}}$, utilizado para distingui-lo dos demais no conjunto de veículos. Será utilizado caso o empate nos lances persista. Nesse caso, pode-se considerar que qualquer um dos veículos que efetuou esses lances pode ser o vencedor, sem prejuízos na eficácia do método.

Tabela 5.2. Identificadores e descrição dos possíveis tipos de lance.

Identificador	Descrição
0	Caso trivial, nova região abrange a posição atual do veículo
1	Ponto de visita já existente ao alcance da nova região
2	Nova região possui interseção com parte ativa do caminho
3	Veículo deve alterar o movimento atual para visitar a região
4	Parte ativa do caminho deve ser modificada para alcançar região

O Algoritmo 14 apresenta os passos gerais do método proposto para se determinar o lance que será realizado por um veículo para a nova região inserida.

O Algoritmo 15 apresenta o método proposto para determinar o lance vencedor do leilão. O método recebe como entrada o conjunto $\boldsymbol{\mathcal{B}} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k\}$, composto por todos os lances de todos os veículos.

Algoritmo 14 Calcular Lance($\mathcal{H}_{\text{new}}$)

if É o caso trivial **then**
    return $\mathbf{b} = \langle 0,\ 0,\ |\mathcal{Q}_{\text{veic}}|,\ \mathcal{V}_{\text{id}} \rangle$
else if Pode executar Algoritmo 6 **then**
    return $\mathbf{b} = \langle 0,\ 1, |\mathcal{H}_{\mathbf{q}_p}|,\ \mathcal{V}_{\text{id}} \rangle$
else if Pode executar Algoritmo 7 **then**
    return $\mathbf{b} = \langle 0,\ 2, L,\ \mathcal{V}_{\text{id}} \rangle$
else if Algoritmo 9 produz um resultado melhor que o Algoritmo 8 **then**
    return $\mathbf{b} = \langle \hat{\mathcal{L}}_{\rho}^{\text{rem}},\ 3,\ |\mathcal{Q}_{\text{veic}}^{\text{ativos}}|,\ \mathcal{V}_{\text{id}} \rangle$
else
    return $\mathbf{b} = \langle \hat{\mathcal{L}}_{\rho}^{\text{rem}},\ 4,\ |\mathcal{Q}_{\text{veic}}^{\text{ativos}}|,\ \mathcal{V}_{\text{id}} \rangle$
end if

Algoritmo 15 Determinar Lance Vencedor($\mathcal{B}$)

$\mathcal{B}^* \leftarrow$ Recuperar lances em $\mathcal{B}$ com o menor custo
if $|\mathcal{B}^*| == 1$ **then**
    return Elemento único de $\mathcal{B}^*$
else
    $\mathcal{B}^* \leftarrow$ Recuperar lances de $\mathcal{B}^*$ com o menor identificador de tipo do lance
    if $|\mathcal{B}^*| == 1$ **then**
        return Elemento único de $\mathcal{B}^*$
    else
        $\mathcal{B}^* \leftarrow$ Recuperar lances de $\mathcal{B}^*$ de acordo com o campo de desempate
        if $|\mathcal{B}^*| == 1$ **then**
            return Elemento único de $\mathcal{B}^*$
        else
            return Lance de $\mathcal{B}^*$ com o menor identificador de veículo
        end if
    end if
end if

Dessa forma, é garantido que, de maneira descentralizada, todos os veículos devem determinar o mesmo lance vencedor para o leilão.

5.2.1.1 Análise de Complexidade

Nesta seção é apresentada a análise da complexidade computacional da metodologia proposta abordando o *k*-DDTSPN, e que faz uso da técnica de leilão distribuído. Inicialmente é abordado o comportamento assintótico relativo ao tempo de execução do algoritmo, em seguida, é discutido o limite superior para o comprimento do caminho (maior rota).

- **Cálculo do lance**: Nessa etapa são utilizados os mesmos algoritmos cujo comportamentos assintóticos foram discutidos na Seção 4.2.3. Dessa forma, tem-se o mesmo o limite assintótico, ou seja, $O(n) + O(m)$, onde n representa o número de pontos de visita auxiliares e m é o número de pontos de visita ativos. A alteração do caminho possuirá o mesmo limite assintótico, uma vez que após a determinação do lance vencedor os mesmos métodos serão novamente utilizados.

- **Determinar vencedor**: O primeiro passo dessa etapa sempre irá percorrer todo o conjunto de lances para se determinar o que possui o menor custo. Logo, tem-se um custo linear $O(n)$, onde n representa o número de lances efetuados, que é diretamente proporcional ao número de veículos sendo utilizados.

De maneira semelhante ao caso dinâmico para um veículo, lances vencedores que possuem tipos com identificadores entre 0 e 2, ou seja, caso trivial, ponto de visita ao alcance e interseção com parte ativa, não resultam em alteração no caminho, e consequentemente, não possuem nenhum impacto no comprimento final da maior rota. Lances com tipos 3 e 4, ou seja, deve alterar movimento atual ou parte ativa do caminho, o pior caso é se todas as novas regiões forem inseridas em uma mesma rota, fato que não deve ocorrer, devido à estratégia utilizada que prioriza uma melhor distribuição das regiões entre os veículos.

Dessa forma, pode-se assumir que o comprimento médio aproximado da maior rota em um instante de tempo t é obtido por

$$\mathcal{X}^t \lesssim \mathcal{X}^0 + (\lambda t)(2D^{\mathcal{X}}_{\max}\mathsf{k}\rho\pi), \tag{5.12}$$

onde $\mathcal{X}^0$ representa o comprimento inicial da maior rota para o caso estático e λt é o número médio esperado de novas regiões no intervalo de tempo $(0, t]$ de acordo com a taxa λ do processo de Poisson utilizado. O valor $D^{\mathcal{X}}_{\max}$ é definido como

$$D^{\mathcal{X}}_{\max} = \max_{\forall i,j \mid \mathbf{p}_i,\mathbf{p}_j \in \mathcal{X}} \|\mathbf{p}_i - \mathbf{p}_j\| \tag{5.13}$$

e corresponde à distância euclidiana máxima entre dois pontos de visita presentes na maior rota.

A quantidade de mensagens enviadas por cada veículo ao longo da execução é diretamente proporcional ao número de regiões inseridas, ou seja, (λt), e a quantidade de mensagens recebidas será $(k-1)(\lambda t)$. Logo, quantidade total de mensagens trafegadas na rede será $k(\lambda t)$.

5.2.2 Experimentos

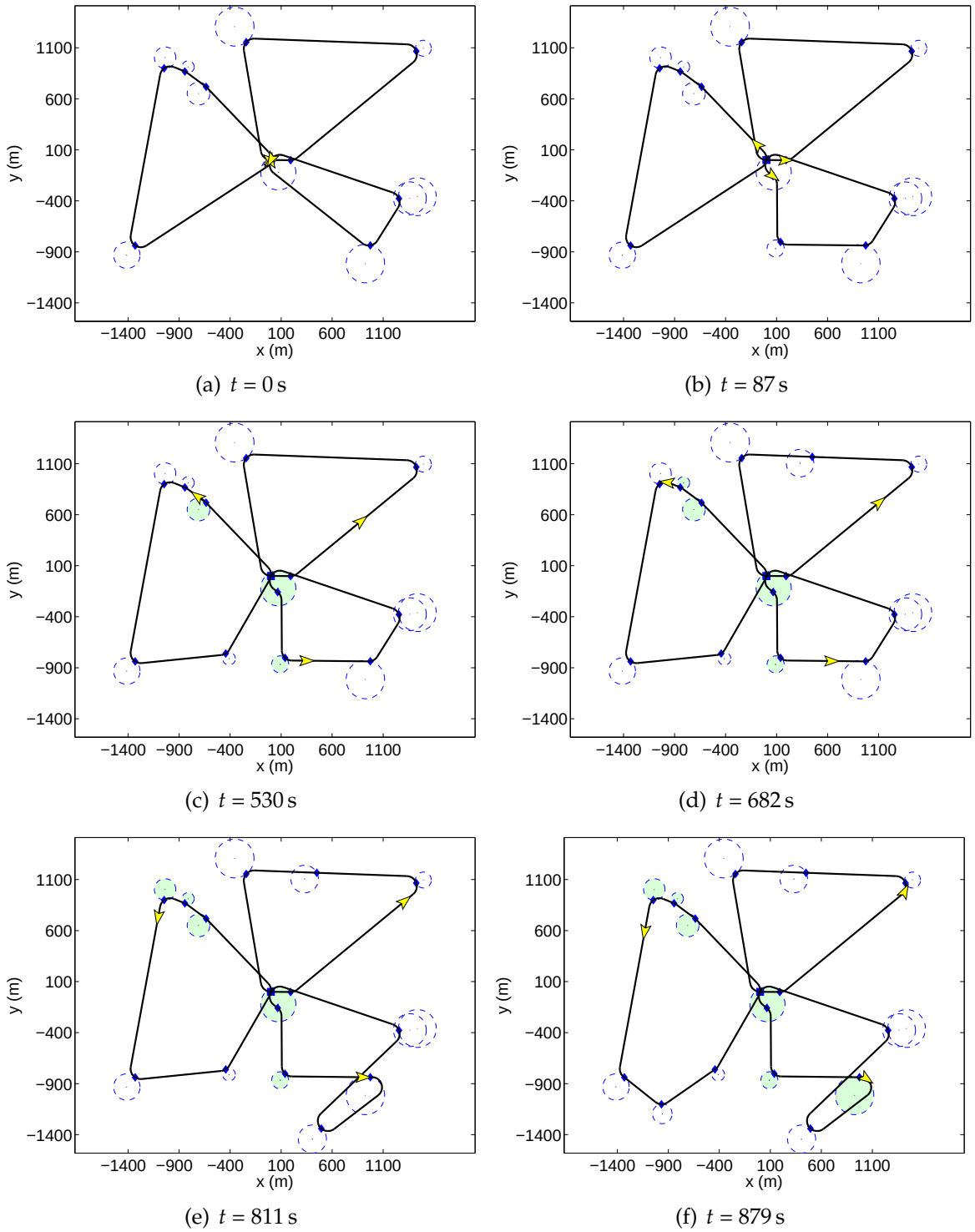
Nesta seção são apresentados e discutidos os resultados obtidos a partir de simulações numéricas utilizando-se a metodologia proposta para o caso onde são utilizados múltiplos veículos e novas demandas não previamente conhecidas podem ser determinadas no ambiente (caso dinâmico).

Para execução das simulações foi desenvolvido um simulador em Matlab, e todos os experimentos foram executados em um PC com um processador Intel Core i7-2720QM 2.20 GHz, 8 Gb de RAM e o sistema operacional Ubuntu 11.10 64-bit. Todas as soluções para as instâncias relativas ao ETSP foram calculadas à otimalidade utilizando-se a biblioteca `Concorde` [Concorde, 2011].

As imagens exibidas na Figura 5.12 apresentam um exemplo da execução da metodologia ao longo do tempo para uma determinada instância. Nesse exemplo, é utilizado um ambiente com dimensões $3.000\,\mathrm{m} \times 3.000\,\mathrm{m}$ contendo, inicialmente, 20 regiões aleatoriamente distribuídas de maneira uniforme. Foi determinada a utilização de, no máximo, 3 veículos, cada um possuindo um raio mínimo de curvatura $\rho = 100\,\mathrm{m}$ e se deslocando com velocidade constante $\nu = 2\,\mathrm{m/s}$. O raio de cada região circular também foi determinado de forma aleatória dentro no intervalo $[50\,\mathrm{m}, 200\,\mathrm{m}]$. Foi utilizada uma taxa de inserção de $\lambda = \frac{1}{180}$, e as novas demandas devem ser adicionadas no intervalo $(0\,\mathrm{s}, 1.600\,\mathrm{s}]$.

De maneira semelhante à análise realizada para o caso dinâmico utilizando apenas um veículo (Seção 4.2.4), será realizada uma análise estatística para determinação da possível razão de competitividade da metodologia apresentada.

Assim, foram executados 300 experimentos em um ambiente com dimensões $4.000\,\mathrm{m} \times 4.000\,\mathrm{m}$ contendo inicialmente 10 regiões distribuídas uniformemente de maneira aleatória, onde o raio de cada região circular também foi escolhido aleatoriamente no intervalo $[50\,\mathrm{m}, 250\,\mathrm{m}]$. Foi determinada a utilização de no máximo 4 veículos, cada um possuindo um raio mínimo de curvatura $\rho = 100\,\mathrm{m}$ e se deslocando com velocidade constante $\nu = 2\,\mathrm{m/s}$. A taxa para inserção de novas regiões possui o valor de $\lambda = \frac{1}{120}$, e foi determinado que as novas inserções ocorreriam no intervalo $(0\,\mathrm{s}, 3.600\,\mathrm{s}]$. Como não existe na literatura um algoritmo comprovadamente ótimo para o problema aqui abordado, a comparação será feita com o algoritmo memético apresentado na Seção 5.1.2. Esse método foi escolhido, pois, conforme visto na seção anterior ele apresentou resultados médios melhores que o k-DSPLITOUR.

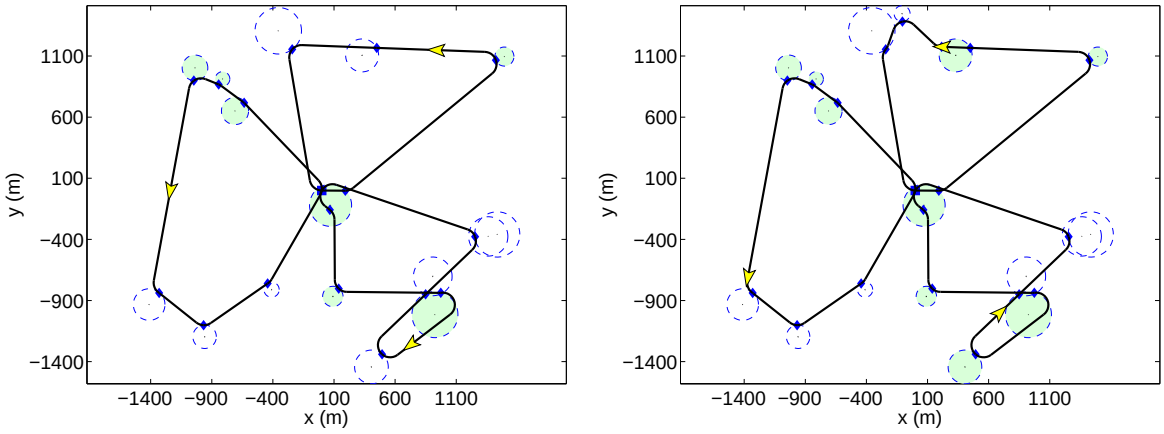


(a) $t = 0\,\mathrm{s}$ (b) $t = 87\,\mathrm{s}$

(c) $t = 530\,\mathrm{s}$ (d) $t = 682\,\mathrm{s}$

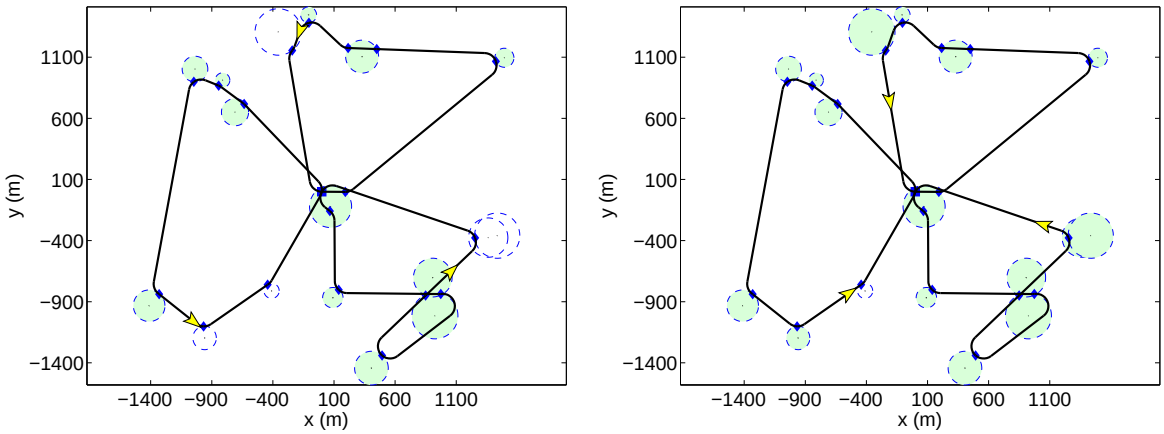
(e) $t = 811\,\mathrm{s}$ (f) $t = 879\,\mathrm{s}$

Figura 5.12. Exemplo inicial apresentando a execução da metodologia dinâmica para múltiplos veículos ao longo do tempo para uma determinada instância. As regiões em destaque (verde) representam as regiões que já foram visitadas.



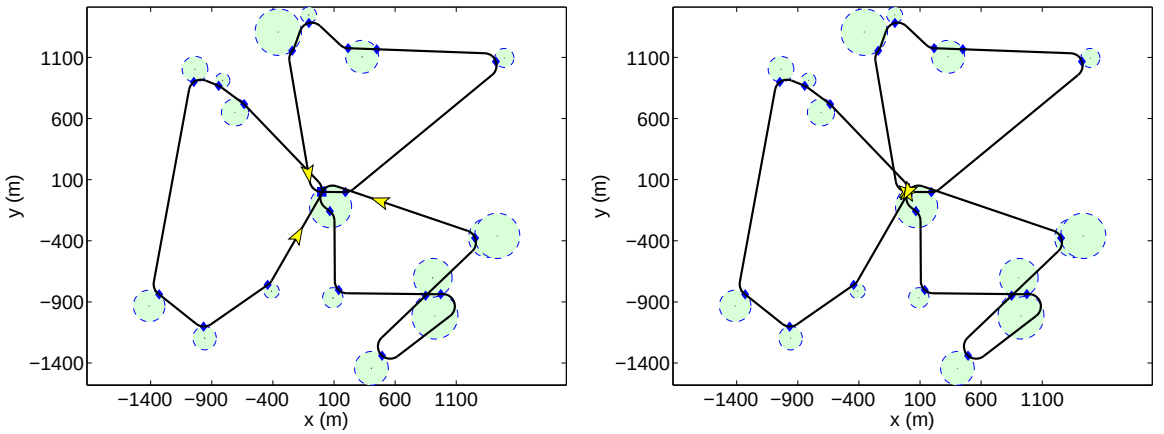
(g) $t = 1.185\,\mathrm{s}$

(h) $t = 1.546\,\mathrm{s}$



(i) $t = 1.800\,\mathrm{s}$

(j) $t = 2.100\,\mathrm{s}$



(k) $t = 2.400\,\mathrm{s}$

(l) $t = 2.652\,\mathrm{s}$

Figura 5.12. Exemplo inicial apresentando a execução da metodologia dinâmica para múltiplos veículos ao longo do tempo para uma determinada instância. As regiões em destaque (verde) representam as regiões que já foram visitadas (continuação).

Após a execução dos experimentos, o maior valor da razão de proporção entre o custo do caminho obtido a partir do algoritmo *online* e o caminho obtido pelo algoritmo *offline* foi 1,56. É importante relembrar que a análise competitiva avalia o desempenho do algoritmo para o pior caso. A Figura 5.13 apresenta o histograma com todos os valores de razão encontrados, permitindo uma melhor noção do comportamento geral do algoritmo.

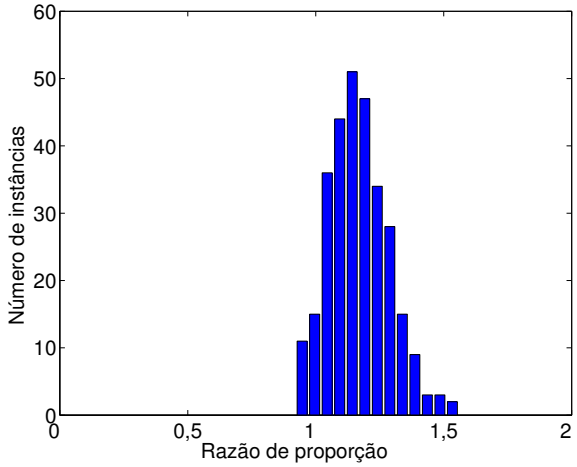


Figura 5.13. Histograma da razão de competitividade dos resultados obtido pelo algoritmo *online* em comparação aos do método *offline* (MA).

Como pode ser observado, o comportamento geral do algoritmo é melhor que o determinado a partir da análise competitiva, possuindo uma razão proporcional média consideravelmente menor que o valor proposto para a razão de competitividade. Na média, foi obtida uma razão com valor 1,17 (com desvio padrão de 0,12). Foi realizada uma análise utilizando-se o teste de hipótese *Teste t*, onde foi verificada a seguinte hipótese:

$$H_0 : \frac{\text{Custo}_{online}}{\text{Custo}_{offline}} = 1 \quad \text{vs.} \quad H_1 : \frac{\text{Custo}_{online}}{\text{Custo}_{offline}} > 1, \tag{5.14}$$

ou seja, a hipótese nula representa o caso em que ambos algoritmos possuiriam o mesmo desempenho. O resultado obtido informa com significância estatística que a hipótese nula foi rejeitada com $t(299) = 163.68$ e $p = 0$, e a razão de proporção encontra-se no intervalo (1,16, 1,19) com 95% de confiança.

É possível observar que o impacto no resultado final (comprimento da maior rota) ao se utilizar múltiplos veículos é consideravelmente menor que o impacto no resultado (comprimento total do caminho) quando apenas um veículo é utilizado.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Neste trabalho foi apresentado um conjunto de metodologias para os diferentes problemas relacionados ao roteamento dinâmico de visita a regiões para veículos não-holonômicos. Trata-se de um problema complexo, onde estão envolvidas etapas de otimização combinatória discreta (a sequência de visita) e contínua (orientação e posição dos pontos de visita). Inicialmente, foi abordado o caso onde apenas um veículo é utilizado (caso estático e dinâmico). Em seguida, foi proposta a extensão do método para sistemas compostos por múltiplos veículos (caso estático e dinâmico).

Foram apresentadas duas abordagens para o caso estático (denominado DTSPN). Primeiramente foi proposta a utilização de um algoritmo evolutivo, onde a posição e orientação dos pontos de visita, além da sequência de visita, são otimizados de uma forma combinada a cada iteração. Em seguida, as similaridades encontradas nos melhores caminhos "evoluídos" foram identificadas e analisadas, permitindo então o desenvolvimento de uma heurística determinística baseada nessas características.

A partir de uma extensiva análise experimental foi possível verificar que ambos os métodos apresentaram considerável redução no comprimento do caminho se comparados à solução trivial para o problema, comprovadamente possuindo significância estatística. Os métodos também foram comparados com a técnica mais recente encontrada na literatura [Isaacs et al., 2011], que consiste em uma abordagem baseada na amostragem de pontos de visita na borda das regiões. O algoritmo evolutivo apresentou um desempenho estatisticamente similar, enquanto a heurística obteve resultados melhores em aproximadamente 80% das instâncias, apresentando uma redução no comprimento do caminho de aproximadamente 7%.

A principal desvantagem da metodologia evolutiva foi a decisão de se movimentar os pontos de visita sempre em direção ao centroide das regiões, que como pôde ser verificado, não necessariamente reduzirá o comprimento do caminho. Um dos problemas referentes à heurística está no fato de a sequência de visita dos pontos ser definida inicialmente baseada na métrica euclidiana (a partir do TSP) e não ser mais alterada após a variação da posição do ponto de visita ou da atribuição da orientação. Uma das principais dificuldades desse problema está justamente na interdependência entre as variáveis sendo otimizadas, por exemplo, a alocação de uma boa orientação depende da sequência de visita, entretanto pode existir uma sequência de visita melhor para as orientações atribuídas.

Além da heurística apresentar resultados melhores que o método evolutivo, o tempo de execução também se mostrou significativamente menor. Entretanto, em ambos os casos, os valores se mostraram dentro dos limites aceitáveis ao se lidar com problemas de otimização dessa complexidade. Apesar disso, o EA apresentou uma taxa de crescimento do comprimento médio do caminho inferior à taxa da heurística. Uma das razões foi o fato da heurística não apresentar bons resultados em ambientes com alta densidade de regiões de interesse.

Em seguida, foi apresentada a estratégia utilizada para a modificação do caminho decorrente da seleção de novas regiões (caso dinâmico, denominado DDTSPN). A heurística previamente proposta é utilizada para a geração do caminho inicial passando pelas regiões já existentes. O próximo passo verifica a necessidade de se alterar o caminho para que a nova região seja efetivamente visitada. Não havendo necessidade, novos pontos de visita são criados sobre o caminho já existente. Caso o caminho necessite ser modificado, é avaliado qual parte do caminho pode ser alterada de forma a produzir o menor impacto no comprimento do caminho. Dependendo da posição da nova região, pode-se optar também por alterar o deslocamento atual do veículo, priorizando a visita imediata à nova região.

O desempenho do algoritmo *online* foi avaliado utilizando-se a técnica da análise competitiva, que verifica a razão entre o resultado encontrado pelo algoritmo *online* e o resultado ótimo obtido a partir de um algoritmo *offline*. Pode-se entender que essa razão, de certa forma, quantifica a perda de eficácia (qualidade de solução) de um algoritmo decorrente da falta de informações completas do problema para se tomar uma decisão. A razão encontrada, no pior caso, obteve o valor de 3,21 e a heurística proposta foi o método *offline* utilizado na comparação. Esse valor é ligeiramente superior ao valor ótimo ($\approx$ 7%) encontrado para o problema semelhante denominado OL-ATSP (zeloso) [Ausiello et al., 2008]. Apesar disso, o algoritmo apresentou um resultado consideravelmente melhor para o caso médio, 2,08.

Considerando o problema do planejamento de caminhos para múltiplos veículos no caso estático (denominado *k*-DTSPN), foram apresentados dois métodos. O primeiro método, denominado *k*-DSPLITOUR, consiste na utilização de duas técnicas já existentes na literatura. Primeiramente uma solução para o TSP é calculada, e em seguida é particionada em *k* rotas utilizando-se o algoritmo *k*-SPLITOUR [Frederickson et al., 1978]. Em seguida, as rotas são adaptadas utilizando o AA [Savla et al., 2005b], tornando-as então realizáveis por veículo com restrição de curvatura. O segundo método é baseado em um Algoritmo Memético. A etapa de otimização global do método é responsável pela parte discreta do problema (sequência de visita), enquanto a etapa de busca local é responsável pela parte contínua (posição e orientação dos pontos de visita).

Conforme mencionado anteriormente, não foram encontrados trabalhos que abordem exatamente o *k*-DTSPN. Dessa forma, optou-se por realizar uma comparação com a técnica tratando o problema mais próximo, o *k*-TSPN (as restrições do veículo não são consideras). Ambos os métodos apresentaram resultados (comprimento da maior rota) aproximadamente 30 % maiores que o método DGPTour [Bhadauria et al., 2011].

Finalmente, foi apresentado um algoritmo totalmente distribuído baseado no mecanismo de leilão para o caso dinâmico utilizando múltiplos veículos (denominado *k*-DDTSPN). Quando uma nova região de interesse é determinada no ambiente, todos os veículos devem efetuar um lance, e o veículo vencedor deverá incorporar a região à sua rota atual. O valor do lance efetuado por cada veículo é calculado a partir dos algoritmos apresentados para o caso dinâmico utilizando apenas um veículo (Seção 4.2).

Novamente, o desempenho do algoritmo *online* foi avaliado utilizando-se a técnica da análise competitiva, que verifica a razão entre o resultado encontrado pelo algoritmo *online* e o resultado ótimo obtido a partir de um algoritmo *offline*. A razão encontrada, no pior caso, obteve o valor de 1,56, sendo que o Algoritmo Memético apresentado foi o método *offline* utilizado na comparação. Entretanto, o algoritmo apresentou um resultado melhor para o caso médio, 1,17. É possível observar que o impacto no resultado final (comprimento da maior rota) ao se utilizar múltiplos veículos é consideravelmente menor que o impacto no resultado (comprimento total do caminho) quando apenas um veículo é utilizado.

6.2 Trabalhos Futuros

Por se tratar de um problema que apresenta características ainda pouco exploradas por outros trabalhos na literatura, este trabalho apresenta diversas possíveis direções futuras que podem ser seguidas.

A primeira extensão que poderia ser realizada está na generalização da técnica para considerar regiões não-convexas. Uma das possíveis soluções para esse problema é uso de técnicas baseadas em amostragem ([Obermeyer et al., 2010; Isaacs et al., 2011]), onde são consideradas apenas posições nas bordas das regiões. Poderiam ser utilizados também métodos de geometria computacional capazes de particionar uma região não-convexa em subregiões convexas ([de Berg et al., 2008]).

Outro ponto interessante, diretamente relacionado aos métodos *online* apresentados, seria a formalização da razão de competitividade máxima e a comparação do desempenho com o algoritmo ótimo proposto para o OL-ATSP (zeloso). Outras estratégias também poderiam ser desenvolvidas e avaliadas, por exemplo, uma questão em aberto é se tratar todas as novas demandas assim que são inseridas realmente é a melhor decisão, ou seria mais interessante esperar por uma determinada quantidade de novas regiões sejam determinadas até se tomar uma decisão.

Uma das principais dificuldades relativa à utilização de curvas Dubins está no fato de o caminho gerado possuir descontinuidade no perfil de curvatura, levando a acelerações laterais bruscas, resultando em caminhos que não são diretamente executáveis por um veículo real. Esse problema pode ser tratado diretamente no controlador utilizado para se seguir o caminho planejado. Outra solução alternativa seria o estudo de possíveis técnicas que já permitam a geração direta de caminhos com variações de aceleração mais suaves a partir da utilização de outros tipos de curvas, por exemplo clotoides [Shanmugavel et al., 2010].

Além disso, outras questões que estão fora do escopo do problema abordado neste trabalho ainda permanecerão em aberto, por exemplo o caso que algumas regiões possuem maior prioridade de visita que outras, ou o caso onde os veículos possuem um conhecimento parcial das novas demandas que são inseridas (limite de sensoriamento). A extensão para ambientes que possuam obstáculos também permitiria uma maior aplicabilidade da técnica em situações do mundo real.

Veículos aéreos ou subaquáticos não necessariamente precisam estar restritos a movimentos planares. Dessa forma, seria importante o cálculo de caminhos no espaço tridimensional. Além de aumentar a dificuldade no cálculo direto do caminho, outras restrições dos veículos passam a ser importantes nessa situação, por exemplo torção máxima e ângulo máximo de subida (no caso de UAVs).

A utilização de grupos de veículos heterogêneos também é um desafio que permanece em aberto. A extensão da metodologia atual para veículos com diferentes raios mínimos de curvatura pode ser feita de maneira direta. Entretanto, ao se utilizar veículos com diferentes velocidades, existe a necessidade de se passar a tratar os caminhos como trajetórias. Dessa forma, não apenas o comprimento deve ser considerado na função objetivo, mas sim o tempo efetivo que será utilizado na visita a todas as regiões.

Referências Bibliográficas

Aggarwal, A.; Coppersmith, D.; Khanna, S.; Motwani, R. & Schieber, B. (1997). The angular-metric traveling salesman problem. Em *Proceedings of the eighth annual ACM-SIAM Symposium on Discrete Algorithms (SODA'1997)*, pp. 221–229, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Alba, E. & Dorronsoro, B. (2004). Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms. Em Gottlieb, J. & Raidl, G., editores, *Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture Notes in Computer Science*, pp. 11–20. Springer Berlin / Heidelberg.

Alves Neto, A.; Macharet, D. G. & Campos, M. F. M. (2010). Feasible RRT-based path planning using seventh order Bézier curves. Em *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2010)*, pp. 1445–1450. ISSN 2153-0858.

Applegate, D. L.; Bixby, R. E.; Chvatal, V. & Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA. ISBN 0691129932, 9780691129938.

Arkin, E. M. & Hassin, R. (1994). Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55:197–218. ISSN 0166-218X.

Arsie, A.; Savla, K. & Frazzoli, E. (2009). Efficient Routing Algorithms for Multiple Vehicles With no Explicit Communications. *IEEE Transactions on Automatic Control*, 54(10):2302–2317. ISSN 0018-9286.

Arslan, G.; Marden, J. R. & Shamma, J. S. (2007). Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596.

Ausiello, G.; Bonifaci, V. & Laura, L. (2008). The on-line asymmetric traveling salesman problem. *Journal of Discrete Algorithms*, 6(2mo):290–298. ISSN 1570-8667. Se-

lected papers from CompBioNets 2004 - Algorithms and Computational Methods for Biochemical and Evolutionary Networks.

Ausiello, G.; Feuerstein, E.; Leonardi, S.; Stougie, L. & Talamo, M. (2001). Algorithms for the On-Line Travelling Salesman. *Algorithmica*, 29:560–581. ISSN 0178-4617.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219. ISSN 0305-0483.

Bertsimas, D. J. & van Ryzin, G. (1991). A Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plane. *Operations Research*, 39(4):601–615.

Bertsimas, D. J. & van Ryzin, G. (1993). Stochastic and Dynamic Vehicle Routing in the Euclidean Plane with Multiple Capacitated Vehicles. *Operations Research*, 41(1):60–76.

Bhadauria, D.; Tekdas, O. & Isler, V. (2011). Robotic data mules for collecting data over sparse sensor fields. *Journal of Field Robotics*, 28(3):388–404. ISSN 1556-4967.

Bopardikar, S.; Smith, S.; Bullo, F. & Hespanha, J. (2010). Dynamic Vehicle Routing for Translating Demands: Stability Analysis and Receding-Horizon Policies. *IEEE Transactions on Automatic Control*, 55(11):2554–2569. ISSN 0018-9286.

Bullo, F.; Frazzoli, E.; Pavone, M.; Savla, K. & Smith, S. (2011). Dynamic Vehicle Routing for Robotic Systems. *Proceedings of the IEEE*, 99(9):1482–1504. ISSN 0018-9219.

Choi, H.-L.; Brunet, L. & How, J. (2009). Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics*, 25(4):912–926. ISSN 1552-3098.

Choset, H.; Lynch, K. M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L. E. & Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA. ISBN 0262033275.

Christofides, N. (1972). Technical Note–Bounds for the Travelling-Salesman Problem. *Operations Research*, 20(5):1044–1056.

Clarkson, J. (2010). Powerrrrr!: Yara Viking Ship, Largest Man Made Moving Machine on the Planet! Jeremy Clarkson's Extreme Machines. BBC London.

Comarela, G.; Gonçalves, K.; Pappa, G. L.; Almeida, J. & Almeida, V. (2011). Robot routing in sparse wireless sensor networks with continuous ant colony optimization. Em *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO'2011)*, pp. 599–606, New York, NY, USA. ACM.

Concorde (2011). Concorde TSP Solver. Disponível em: `http://www.tsp.gatech.edu/concorde/index.html`. Acesso em: 17 de outubro de 2011.

Dantzig, G. B. & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1):80–91.

Darwin, C. (1859). *On the origin of species by means of natural selection, or, the preservation of favoured races in the struggle for life*. London: John Murray.

de Berg, M.; Cheong, O.; Kreveld, M. v. & Overmars, M. (2008). *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3ª edição.

de Berg, M.; Gudmundsson, J.; Katz, M. J.; Levcopoulos, C.; Overmars, M. H. & van der Stappen, A. F. (2005). TSP with neighborhoods of varying size. *Journal of Algorithms*, 57:22–36. ISSN 0196-6774.

Di Francesco, M.; Das, S. K. & Anastasi, G. (2011). Data Collection in Wireless Sensor Networks with Mobile Elements: A Survey. *ACM Transactions on Sensor Networks*, 8(1):7:1–7:31. ISSN 1550-4859.

Dias, M. B.; Zlot, R.; Kalra, N. & Stentz, A. (2006). Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257–1270. ISSN 0018-9219.

Dubins, L. E. (1957). On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497–516. ISSN 00029327.

Dudek, G. & Jenkin, M. (2010). *Computational Principles of Mobile Robotics*. Cambridge University Press, New York, NY, USA, 2ª edição. ISBN 0521871573-9780521871570.

Dumitrescu, A. & Mitchell, J. S. (2003). Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159. ISSN 0196-6774. Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms.

Eiben, A. & Smith, J. (2007). *Introduction to Evolutionary Computing*. Natural computing series. Springer, 2ª edição. ISBN 9783540401841.

Elbassioni, K.; Fishkin, A. & Sitters, R. (2006). On Approximating the TSP with Intersecting Neighborhoods. Em Asano, T., editor, *Algorithms and Computation*, volume 4288 of *Lecture Notes in Computer Science*, pp. 213–222. Springer Berlin / Heidelberg.

Elbassioni, K. M.; Fishkin, A. V.; Mustafa, N. H. & Sitters, R. (2005). Approximation Algorithms for Euclidean Group TSP. Em Caires, L.; Italiano, G.; Monteiro, L.; Palamidessi, C. & Yung, M., editores, *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pp. 1115–1126. Springer Berlin / Heidelberg.

Enright, J. & Frazzoli, E. (2006). Cooperative UAV Routing with Limited Sensor Range. Em *Proceedings of the AIAA Conference on Guidance, Navigation andControl*, Keystone, CO, USA.

Enright, J. J.; Savla, K.; Frazzoli, E. & Bullo, F. (2009). Stochastic and Dynamic Routing Problems for Multiple UAVs. *AIAA Journal of Guidance, Control, and Dynamics*, 32(4):1152–1166.

Farinelli, A.; Iocchi, L. & Nardi, D. (2004). Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5):2015–2028.

Frazzoli, E. & Bullo, F. (2004). Decentralized algorithms for vehicle routing in a stochastic time-varying environment. Em *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC'2004)*, volume 4, pp. 3357–3363. ISSN 0191-2216.

Frederickson, G. N.; Hecht, M. S. & Kim, C. E. (1978). Approximation Algorithms for Some Routing Problems. *SIAM Journal on Computing*, 7(2):178–193.

Frieze, A. M.; Galbiati, G. & Maffioli, F. (1982). On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39. ISSN 1097-0037.

Gelfand, A.; Diggle, P. & Guttorp, P. (2010). *Handbook of Spatial Statistics*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. Taylor & Francis Group. ISBN 9781420072877.

Gerkey, B. P. & Matarić, M. J. (2004). A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9):939–954.

Gudmundsson, J. & Levcopoulos, C. (1999). A fast approximation algorithm for TSP with neighborhoods. *Nordic Journal of Computing*, 6:469–488. ISSN 1236-6064.

Hahsler, M. & Hornik, K. (2007). TSP – Infrastructure for the Traveling Salesperson Problem. *Journal of Statistical Software*, 23(2):1–21. ISSN 1548-7660.

Isaacs, J. T.; Klein, D. J. & Hespanha, J. P. (2011). Algorithms for the traveling salesman problem with neighborhoods involving a dubins vehicle. Em *Proceedings of the IEE American Control Conference (ACC'2011)*, pp. 1704–1709. ISSN 0743-1619.

Jaillet, P. & Wagner, M. R. (2006). Online Routing Problems: Value of Advanced Information as Improved Competitive Ratios. *Transportation Science*, 40(2):200–210.

Jaillet, P. & Wagner, M. R. (2008). Online Vehicle Routing Problems: A Survey. Em Golden, B.; Raghavan, S.; Wasil, E.; Sharda, R. & Voß, S., editores, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pp. 221–237. Springer US.

Karaman, S. & Frazzoli, E. (2010). Optimal kinodynamic motion planning using incremental sampling-based methods. Em *Proceedings of the IEEE Conference on Decision and Control (CDC'2010)*, pp. 7681–7687. ISSN 0743-1546.

Khouadjia, M. R.; Sarasola, B.; Alba, E.; Jourdan, L. & Talbi, E.-G. (2012). A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests. *Applied Soft Computing*, 12(4):1426–1439. ISSN 1568-4946.

Knuth, D. E. (1997). *Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley Professional, 3ª edição. ISBN 9780201896848.

Kuwata, Y.; Fiore, G.; Teo, J.; Frazzoli, E. & How, J. (2008). Motion planning for urban driving using RRT. Em *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2008)*, pp. 1681–1686.

Larsen, A.; Madsen, O. B. & Solomon, M. M. (2008). Recent Developments in Dynamic Vehicle Routing Systems. Em Golden, B.; Raghavan, S.; Wasil, E.; Sharda, R. &

Voß, S., editores, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pp. 199–218. Springer US.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, New York, NY, USA. ISBN 0521862051.

Le Ny, J. & Feron, E. (2005). An Approximation Algorithm for the Curvature-Constrained Traveling Salesman Problem. Em *roceedings of the 43rd Annual Allerton Conference on Communications, Control and Computing*.

Le Ny, J.; Feron, E. & Frazzoli, E. (2012). On the dubins traveling salesman problem. *IEEE Transactions on Automatic Control*, 57(1):265–270. ISSN 0018-9286.

Le Ny, J.; Frazzoli, E. & Feron, E. (2007). The curvature-constrained traveling salesman problem for high point densities. Em *Proceedings of the 46th IEEE Conference on Decision and Control (CDC'2007)*, pp. 5985–5990. ISSN 0191-2216.

Lin, S. & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

Ma, X. & Castañón, D. A. (2006). Receding horizon planning for dubins traveling salesman problems. Em *Proceedings of the 45th IEEE Conference on Decision and Control (CDC'2006)*, pp. 5453–5458.

Macharet, D. G.; Alves Neto, A. & Campos, M. F. M. (2010). Feasible UAV path planning using genetic algorithms and Bézier curves. Em *Proceedings of the 20th Brazilian Conference on Advances in Artificial Intelligence (SBIA'2010)*, pp. 223–232, Berlin, Heidelberg. Springer-Verlag.

Mata, C. S. & Mitchell, J. S. B. (1995). Approximation algorithms for geometric tour and network design problems. Em *Proceedings of the eleventh annual Symposium on Computational Geometry (SCG'1995)*, pp. 360–369, New York, NY, USA. ACM.

Medeiros, A. & Urrutia, S. (2009). Otimizando trajetórias de veículos não-holomônicos. Em *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional (SBPO'2009)*.

Medeiros, A. & Urrutia, S. (2010). Discrete optimization methods to determine trajectories for Dubins' vehicles. *Electronic Notes in Discrete Mathematics*, 36:17–24. ISSN 1571-0653. ISCO 2010 - International Symposium on Combinatorial Optimization.

Michael, N.; Zavlanos, M.; Kumar, V. & Pappas, G. (2008). Distributed Multi-Robot Task Assignment and Formation Control. Em *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2008).*, pp. 128–133. ISSN 1050-4729.

Mitchell, J. S. B. (2007). A PTAS for TSP with neighborhoods among fat regions in the plane. Em *Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2007)*, pp. 11–18, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Moore, B. & Passino, K. (2007). Distributed Task Assignment for Mobile Agents. *IEEE Transactions on Automatic Control*, 52(4):749–753. ISSN 0018-9286.

Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms. Relatório técnico C3P Report 826, Caltech Concurrent Computation Program.

Moscato, P. & Cotta, C. (2010). A modern introduction to memetic algorithms. Em Gendreau, M. & Potvin, J.-Y., editores, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pp. 141–183. Springer US.

Obermeyer, K. J. (2009). Path planning for a UAV performing reconnaissance of static ground targets in terrain. Em *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Chicago, IL, USA.

Obermeyer, K. J.; Oberlin, P. & Darbha, S. (2010). Sampling-based roadmap methods for a visual reconnaissance UAV. Em *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Toronto, ON, Canada.

Papoulis, A. & Pillai, S. U. (2002). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Europe, 4^a edição. ISBN 0071226613.

Parker, L. E. (2008). Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems. *Journal of Physical Agents*, 2(1):5–14. ISSN 1888-0258.

Pavone, M.; Bisnik, N.; Frazzoli, E. & Isler, V. (2009). A Stochastic and Dynamic Vehicle Routing Problem with Time Windows and Customer Impatience. *Mobile Networks and Applications*, 14:350–364. ISSN 1383-469X.

Pavone, M. & Frazzoli, E. (2010). Dynamic vehicle routing with stochastic time constraints. Em *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'2010)*, pp. 1460–1467. ISSN 1050-4729.

Pavone, M.; Frazzoli, E. & Bullo, F. (2011). Adaptive and Distributed Algorithms for Vehicle Routing in a Stochastic and Dynamic Environment. *IEEE Transactions on Automatic Control*, 56(6):1259–1274. ISSN 0018-9286.

Psaraftis, H. N. (1988). Dynamic vehicle routing problems. Em Golden, B. L. & Assad, A. A., editores, *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, pp. 223--248. North-Holland, Amsterdam.

Rathinam, S.; Sengupta, R. & Darbha, S. (2007). A Resource Allocation Algorithm for Multivehicle Systems With Nonholonomic Constraints. *IEEE Transactions on Automation Science and Engineering*, 4(1):98–104. ISSN 1545-5955.

Rosen, K. H. (2012). *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, Boston, 7^a edição.

Safra, S. & Schwartz, O. (2006). On the complexity of approximating tsp with neighborhoods and related problems. *Computational Complexity*, 14:281–307. ISSN 1016-3328.

Savla, K.; Bullo, F. & Frazzoli, E. (2005a). On traveling salesperson problems for dubins' vehicle: stochastic and dynamic environments. Em *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'2005)*, pp. 4530–4535.

Savla, K.; Frazzoli, E. & Bullo, F. (2005b). On the point-to-point and traveling salesperson problems for dubins' vehicle. Em *Proceedings of the IEE American Control Conference (ACC'2005)*, volume 2, pp. 786–791. ISSN 0743-1619.

Savla, K.; Frazzoli, E. & Bullo, F. (2008). Traveling salesperson problems for the dubins vehicle. *IEEE Transactions on Automatic Control*, 53(6):1378–1391. ISSN 0018-9286.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Sixth-generation computer technology series. Wiley. ISBN 978-0-471-57148-3.

Shanmugavel, M.; Tsourdos, A.; White, B. & Żbikowski, R. (2010). Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs. *Control Engineering Practice*, 18(9):1084–1092. ISSN 0967-0661.

Shanmugavel, M.; Tsourdos, A.; White, B. A. & Żbikowski, R. (2005). Path planning of multiple UAVs using Dubins sets. Em *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, San Francisco, California, USA.

Shima, T.; Rasmussen, S. & Gross, D. (2007). Assigning Micro UAVs to Task Tours in an Urban Terrain. *IEEE Transactions on Control Systems Technology*, 15(4):601–612. ISSN 1063-6536.

Shkel, A. M. & Lumelsky, V. (2001). Classification of the Dubins set. *Robotics and Autonomous Systems*, 34(4):179–202. ISSN 0921-8890.

Siciliano, B. & Khatib, O., editores (2008). *Springer Handbook of Robotics*. Springer, Berlin, Heidelberg. ISBN 978-3-540-23957-4.

Siegwart, R.; Nourbakhsh, I. R. & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. MIT Press, Cambridge, MA, USA, 2^a edição. ISBN 0262015358.

Sleator, D. D. & Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208. ISSN 0001-0782.

Smith, S.; Bopardikar, S. & Bullo, F. (2009). A dynamic boundary guarding problem with translating targets. Em *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with the 28th Chinese Control Conference (CDC/CCC'2009)*, pp. 8543–8548. ISSN 0191-2216.

Smith, S. L.; Pavone, M.; Bullo, F. & Frazzoli, E. (2010). Dynamic Vehicle Routing with Priority Classes of Stochastic Demands. *SIAM Journal on Control and Optimization*, 48(5):3224–3245. ISSN 03630129.

Sofge, D.; Schultz, A. & DeJong, K. (2002). Evolutionary Computational Approaches to Solving the Multiple Traveling Salesman Problem Using a Neighborhood Attractor Schema. Em *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN*, pp. 153–162, London, UK, UK. Springer-Verlag.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254--265. ISSN 0030-364X.

Tang, Z. & Özgüner, Ü. (2005). Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21(5):898–908. ISSN 1552-3098.

Tekdas, O.; Isler, V.; Lim, J. & Terzis, A. (2009). Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22–28. ISSN 1536-1284.

Thrun, S.; Burgard, W. & Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press. ISBN 0262201623.

Toth, P. & Vigo, D., editores (2001). *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. ISBN 0-89871-498-2.

Valle, C. A.; da Cunha, A. S.; Aioffi, W. M. & Mateus, G. R. (2008). Algorithms for improving the quality of service in wireless sensor networks with multiple mobile sinks. Em *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'2008)*, pp. 239–243, New York, NY, USA. ACM.

Wu, F.-J.; Huang, C.-F. & Tseng, Y.-C. (2009). Data Gathering by Mobile Mules in a Spatially Separated Wireless Sensor Network. Em *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM'2009)*, pp. 293–298, Washington, DC, USA. IEEE Computer Society.

Yadlapalli, S.; Malik, W.; Rathinam, S. & Darbha, S. (2007). A lagrangian-based algorithm for a combinatorial motion planning problem. Em *Proceedings of the 46th IEEE Conference on Decision and Control (CDC'2007)*, pp. 5979–5984. ISSN 0191-2216.

Yuan, B.; Orlowska, M. & Sadiq, S. (2007a). Finding the Optimal Path in 3D Spaces Using EDAs — The Wireless Sensor Networks Scenario. Em *Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, Part I (ICANNGA'2007)*, pp. 536–545, Berlin, Heidelberg. Springer-Verlag.

Yuan, B.; Orlowska, M. & Sadiq, S. (2007b). On the Optimal Robot Routing Problem in Wireless Sensor Networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1252–1261. ISSN 1041-4347.

Zavlanos, M.; Spesivtsev, L. & Pappas, G. (2008). A Distributed Auction Algorithm for the Assignment Problem. Em *Proceedings of the 47th IEEE Conference on Decision and Control (CDC'2008).*, pp. 1212–1217. ISSN 0191-2216.