Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnológicas
Programa de Pós-Graduação em Ciência da Computação

RAPHAEL GOMES SANTOS

METAHEURÍSTICAS PARA GERAÇÃO DE ALVOS PARA ROBÔS EXPLORATÓRIOS AUTÔNOMOS

São Luís
2016

RAPHAEL GOMES SANTOS

METAHEURÍSTICAS PARA GERAÇÃO DE ALVOS PARA ROBÔS EXPLORATÓRIOS AUTÔNOMOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, **como parte dos requisitos necessários** para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Profº Alexandre César Muniz de Oliveira

São Luís

2016

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Gomes Santos, Raphael.
Metaheurísticas para geração de alvos para robôs exploratórios autônomos / Raphael Gomes Santos. - 2016.
73 p.

Orientador(a): Alexandre César Muniz de Oliveira.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís, 2016.

1. Exploração Autônoma. 2. Metaheurística. 3. ROS. I. Oliveira, Alexandre César Muniz de. II. Título.

Aos meus pais.

Agradecimentos

Agradeço a Deus, pela vida e pelas oportunidades que nela recebi.

Aos meus pais, por todo amor, cuidado e conselhos ao longo de toda minha vida.

A minha noiva Cassia Betânia, por todo apoio, carinho e compreessão fundamentais para que este trabalho fosse feito.

Ao meu orientador Alexandre César Muniz de Oliveira, que desde a graduação tem me dado oportunidades e me axiliado no meu desenvolvimento como pesquisador.

Aos amigos feitos na Graduação, Pós-Graduação e no laboratório LACMOR.

"O êxito da vida não se mede pelo que você conquistou, mas sim pelas dificuldades que superou no caminho."
(Abraham Lincoln)

Resumo

Exploração autônoma, em robótica, pode ser definida como o ato de mover-se em um ambiente, a princípio desconhecido, enquanto constrói-se um mapa deste ambiente. Uma grande parte da literatura relata vários problemas que se relacionam com a estratégia de exploração: percepção, localização, trajetória, controle e mapeamento. Este trabalho visa apresentar um algoritmo de exploração autonoma baseado em metaheurísticas. Para tanto, o problema de exploração autônoma de robôs móveis é formulado como um problema de otimização, fornecendo dados para que metaheurísticas sejam capazes de buscar pontos no espaço de soluções que representam posições no mapa em construção que melhor satisfaçam os objetivos da exploração. Metaheuristicas são metodos aproximados que garantem soluções suficientemente boas para problemas de otimização. A proposta foi implementada e incorporada como um módulo de otimização em um sistema de localização e mapeamento simultâneos que foi executado em ambiente *Robot Operating System* e mostrou-se capaz de guiar um robô simulado sem intervenção humana. As metaheurísticas usadas foram o Algoritmo Genético e o Algoritmo de Vagalumes. Ambos os algoritmos obtiveram bons resultados, no entanto o Algoritmo de Vagalumes guiou o robô por trajetórias melhores.

Palavras-chaves: Exploração autônoma. Metaheurística. ROS.

Abstract

Autonomous exploration, in robotics, can be defined as the act of moving into an unknown environment, at *priori*, while building up a map of the environment. A great deal of literature describes several problems that are relate to the strategy exploration: perception, location, trajectory control and mapping. This work aims to present an autonomous exploration algorithm based on metaheuristics. Therefore, the problem of autonomous exploration of mobile robots is formulated as an optimization problem, providing data for metaheuristics that are able to search points in the space of solutions that represent positions on the map under construction that best meet the objectives of the exploration. Metaheuristics are approximate methods that guarantee sufficiently good solutions to optimization problems. The proposal was implemented and incorporated as an optimization module in a simultaneous location and mapping system that was run on the Robot Operating System environment and proved to be able to guide a simulated robot without human intervention. Two optimization metaheuristics were implemented to guide target to simulated robot: Genetic Algorithm and Firefly Algorithm. Both algorithms have achieved good results, however the second one was able to guide robot by best trajectories.

Keywords: Mobile Robotics. Autonomous exploration. Optimization metaheuristics.

Lista de ilustrações

Lista de tabelas

Lista de Siglas

AE Algoritmo Evolutivo

AG Algoritmo Genético

AV Algoritmo de Vagalumes

FBA *Feature-based Algorithm*

NBV *Next Best View*

ROS *Robot Operating System*

SLAM *Simultaneos Localization and Maping*

V-REP *Virtual Robot Experimentation Platform*

Sumário

1 Introdução

Um importante objetivo na área de robótica é a criação de robôs autônomos capazes de receber tarefas para que as executem, sem que haja descrições estritas de como realizar tais tarefas. O desenvolvimento da tecnologia necessária para construir robôs autônomos engloba algumas ramificações como: percepção, raciocínio automatizado e controle, surgindo delas vários outros problemas importantes.

Navegação autônoma de robôs, por sua vez, envolve percepção do ambiente, localização e mapeamento, planejamento de trajetória e controle de movimento. Enquanto percepção se refere ao entendimento de seus dados sensoriais, encontrar sua pose (combinação de posição e orientação) ou configuração nos arredores constitui a localização, esta geralmente integrada à construção de mapa. O planejamento do caminho, por meio de tomadas de decisão, em geral, requer integração com o controle de movimento (RAJA; PUGAZHENTHI, 2012).

As funções da navegação podem ser expressadas pelas quatro questões apresentadas a seguir (MURPHY, 2000):

- Onde estou? - capacidade de auto-localização usando informações sensoriais integradas ao mapeamento parcial ou total do ambiente;
- Para onde vou? - capacidade de raciocínio com base no conhecimento do ambiente e estratégia de exploração;
- Qual é o melhor caminho? - planejamento de trajetória, com ou sem interferência humana;
- Onde eu estive? - capacidade de memória ou manutenção do mapa.

A capacidade de raciocínio com base no conhecimento atual do ambiente e estratégia de exploração formam o foco principal deste trabalho. Pretende-se contribuir com o desenvolvimento de sistemas capazes de atribuir significados aos dados sensoriais e inferir, com autonomia, ações que melhor cumpram a estratégia pré-definida de exploração.

1.1 Exploração autônoma

De um modo geral, a navegação robótica pode ser simplificada pelo conhecimento prévio do ambiente a ser explorado. Nesse caso, a navegação seria utilizada para tarefas como vigilância ou inspeção. A intervenção humana, normalmente, se caracteriza pela prévia construção do mapa, indicando os locais exatos dos obstáculos ou pontos de interesse

(para mapas métricos) ou um grafo, representando a conectividade entre regiões abertas (para mapas topológicos).

Exploração pode ser definida como o ato de mover um robô em um ambiente desconhecido, enquanto constrói-se um mapa que pode ser usado para etapas subsequentes da navegação. Uma boa estratégia de exploração é aquela que gera um mapa completo ou parcial em uma quantidade de tempo satisfatória (YAMAUCHI, 1997). A questão central em exploração é: *dado o que você sabe do mundo, para onde você deve mover-se para ganhar o máximo de informação nova possível?* (YAMAUCHI, 1997). Ou seja, a preocupação central na exploração é como cobrir uma área desconhecida de forma eficiente, respeitando-se limites na autonomia do robô.

Neste contexto, surge um dos mais relevantes problemas de robótica móvel: o *problema de localização e mapeamento simultâneos* (*Simultaneos Localization and Maping - SLAM*). O problema do SLAM relaciona-se com a possibilidade de um robô móvel colocado em uma localização desconhecida, em um ambiente desconhecido e incrementalmente construir um mapa consistente deste ambiente enquanto simultaneamente determina sua localização dentro desse mapa (DURRANT-WHYTE; BAILEY, 2006). Portanto a solução deste problema elimina a necessidade de infraestruturas artificiais ou conhecimento topológico *a priori* do ambiente. Esta solução tem um valor inestimável para uma gama de aplicações em que uma posição absoluta ou informação precisa do mapa é inalcançável, incluindo entre outras, exploração planetária autônoma, veículos autônomos submarinos, veículos aéreos autônomos, e veículos autônomos para todos os tipos de terrenos em tarefas como mineração e construção (DISSANAYAKE et al., 2001).

Os algoritmos SLAM procuram o melhor modo de integrar dados dos sensores adquiridos pelo robô durante a navegação. Eles, entretanto não respondem a seguinte questão: Dado o mapa conhecido até agora, para onde o robô deveria mover-se para observar as regiões inexploradas? Do ponto de vista do planejamento de movimento, esta é a questão mais interessante na pesquisa de construção automática de mapas. Ela envolve a computação de posições de sensoriamento, iterativamente resolvendo o problema *next best view*. NBV é complementar ao SLAM. Um algoritmo SLAM constrói um mapa fazendo o melhor uso dos dados dos sensores disponiveis, enquanto que um algoritmo NBV guia o robô para locais que provêem os melhores dados de sensores disponiveis (GONZÁLEZ-BAñOS; HSU; LATOMBE, 2005).

1.2 Metaheurísticas

As estratégias presentes na literatura, em geral, definem regras e operações que, aos olhos do especialista, conduzem a escolhas de alvos que melhor atendem aos objetivos da estratégia de exploração, definindo-se uma função de utilidade que avalia as poses do

robô no ambiente, segundo critérios como o ganho de informação, distância para outros robô, etc. Do ponto de vista da tomada de decisão, essas estratégias são puramente locais e *gulosas*, i.e, estratégias que consideram características observáveis do ambiente pela ótica do robô, o que conduz a escolhas de boa qualidade, mas não ótimas (ou sub-ótimas) (OLIVEIRA, 2004).

Mapas de exploração reúnem muitas informações, sujeitas a ruídos e incertezas que tornam complexo o processo de localização e mapeamento. Modelos de representação de ambiente menos rígidos, ou seja, não baseados em regras, podem levar ao desenvolvimento de soluções mais flexíveis e de fácil adaptação.

Metaheurísticas de otimização são algoritmos que buscam escolhas com qualidade (não necessariamente globalmente ótimas) em espaços de possibilidades formados por variáveis de decisão discretas ou contínuas utilizando, em geral, apenas a avaliação sistemática de uma função objetivo (OLIVEIRA, 2004). Metaheurística são providas de uma ou mais estratégias para torná-las mais robustas e assim terem maiores chances de *escapar* de ótimos locais. O objetivo, portanto, é explorar eficientemente o espaço de busca a fim de encontrar soluções ótimas ou quase-ótimas.

1.3 Objetivos e Contribuições

O objetivo do trabalho é propor o uso de metaheurísticas para fornecer alvos de qualidade para guiar robôs móveis na tarefa de exploração autônoma, segundo estratégias pré-definidas. Para tanto, o ambiente em exploração é modelado como uma função de variáveis contínuas, no espaço euclidiano, na qual os pontos de mínima energia são potencialmente os melhores alvos para o robô. Assim, a geração de objetivos é definida com um problema de otimização contínua representado por uma função objetivo multimodal.

Uma função multimodal possui várias inflexões de sua superfície o que eleva a dificuldade para algoritmos de otimização baseados em gradiente ou outro qualquer com características de otimizador local (PRESS et al., 2007). Para esses casos, métodos baseados em metaheurísticas têm notável desempenho (OLIVEIRA, 2004).

Nesta abordagem, são providos alvos ao robô que melhor se adequam aos objetivos da exploração. Além disso, com esta abordagem pode-se alcançar uma maior flexibilidade do sistema diante de peculiaridades do ambiente não previstas. Outro fator importante que leva a escolha de metaheurísticas é que estas são rápidas em relação aos métodos exatos e enumerativos, ou seja, têm complexidade computacional menor.

Este trabalho investiga diferentes associações de parâmetros na composição da função-objetivo, bem como compara o desempenho de dois algoritmos de otimização: Algoritmo Genético e Algoritmo de Vagalumes.

A abordagem é implementada por meio do Sistema Operacional Robótico (*Robotic Operating System* - ROS) e da Plataforma de Simulação Robótica (*Virtal Robot Experimentation Plataform* - V-REP). O ROS é um sistema meta-operacional robótico de código aberto, que provê os serviços padrões de um sistema operacional tais como abstração de *hardware*, controle de baixo nível de dispositivos, implementação de funcionalidades mais usadas, passagem de mensagens entre processos, e gerenciador de pacotes (ROS/INTRODUCTION, 2014). Ele é baseado em uma arquitetura de grafo onde o processamento localiza-se em nós que podem receber e publicar mensagens através de tópicos. O simulador V-REP é um simulador robótico com um ambiente de desenvolvimento integrado, baseado em uma arquitetura de controle distribuída (V-REP, 2013).

As contribuições deste trabalho pontualmente são:

- Definição do problema da *Exploração Autônoma* como um problema de otimização contínua;
- Uso de metaheurísticas para solucionar o problema definido;
- Implementação de um sistema de exploração autônoma sobre *ROS*.

1.4 Organização do trabalho

Este documento está organizado em 6 capítulos, sendo este o primeiro, apresentando uma abordagem introdutória, a motivação e os objetivos do trabalho.

No Capitulo 2, são descritos os conceitos fundamentais relacionados ao problema de exploração autônoma e às varias estratégias de exploração presentes na literatura. Também são definidos conceitos relativos a problemas de otimização, algoritmos de otimização e metaheurísticas. No Capítulo 3, a exploração autônoma é descrita como um problema de otimização e uma modelagem metaheurística é proposta como parte do sistema de localização e mapeamento. No Capítulo 4, são detalhados a arquitetura e ambiente computacionais necessários para a implementação e validação da proposta. No Capítulo 5, são descritos os experimentos realizados. No Capítulo 6, são destacados os principais achados e trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, são descritos os conceitos fundamentais relacionados ao problema da Exploração Autônoma e abordagens para solucioná-lo. Bem como descreve algoritmos de otimização e conceitos relacionados, dando foco às metaheurísticas.

2.1 Exploração Autônoma

Na literatura, o problema da localização e mapeamento tem um papel de destaque, uma vez que a qualidade do mapa resultante depende fortemente da acurácia das estimativas de pose durante o processo de mapeamento. Entretanto, a acurácia do mapa também depende das escolhas de ponto de vista durante a exploração. Especialmente se sensores ruidosos são usados, o mapa será bastante impreciso em áreas que foram sensoriadas poucas vezes ou talvez mesmo de pontos de vistas desvantajosos. Exploração é a tarefa de guiar o veículo de tal modo que ele irá cobrir o ambiente com seus sensores. Estratégias de exploração também são relevantes para inspeção de superfície, varredura de minas, ou vigilância (STACHNISS; BURGARD, 2003).

2.2 Estratégias de Exploração

Uma *boa* estratégia de exploração para guiar o processo de construção do mapa deve ser: eficiente, para cobrir o ambiente tão rápido quanto possível; acurado, para construir um mapa preciso e confiável; e adaptável, para funcionar em diferentes tipos de ambientes. Os mapas assim produzidos podem ter diferentes representações: geométrica (pontos, segmentos) ou topológica (AMIGONI; GALLO, 2005).

Em (AMIGONI; GALLO, 2005), sistemas baseados em estratégias de exploração são classificados em duas abordagens principais. Na primeira classe, tem-se sistemas que usam trajetórias predefinidas ao longo das quais o robô move-se e coleta observações, o que exige um conhecimento *a priori* do ambiente. Os sistemas da segunda classe, chamados de sistemas *next-best-view* (próxima melhor vista), são mais flexíveis na adaptação a diferentes ambientes. Na exploração *next-best-view*, as posições de observação, e portanto as trajetórias ao longo das quais o robô move-se, são computadas durante a exploração. Usualmente, o robô seleciona, a cada passo, a próxima melhor posição de observação a partir da qual coletará informações do ambiente. A primeira operação do processo de exploração *next-best-view* é esta seleção. Essa escolha é usualmente feita pela avaliação de um conjunto de posições candidatas através de uma função de utilidade: esta função captura em um valor o quão interessante é uma posição de observação.

2.2.1 Estratégia de Exploração baseada em Fronteira

A exploração baseada em fronteira é uma estratégia simples para explorar um ambiente desconhecido, introduzida por Brian Yamauchi em 1997, com a ideia central de: "*Para ganhar a mais nova informação sobre o mundo, deve-se mover para o limite entre o espaço aberto e o território desconhecido*" (YAMAUCHI, 1997).

A estratégia é baseada em um mapa *occupancy grid* corrente m_t, onde todas as células são classificadas como abertas, ocupadas ou inexploradas (YAMAUCHI, 1997), (YAMAUCHI; SCHULTZ; ADAMS, 1998). A ideia básica do *occupancy grid* é representar um mapa do ambiente como um campo uniformemente espaçado de variáveis aleatórias binárias, cada uma representando a presença de um obstáculo no local no ambiente, onde para cada célula (i, j) do mapa, tem um valor de ocupação associado, representado a probabilidade da haver um obstáculo sobre aquela célula (THRUN; BURGARD; FOX, 2005). Usando as células classificadas, todas as células da fronteira podem ser determinadas. Uma célula de fronteira é uma célula que é classificada como aberta e é adjacente a uma célula classificada como inexplorada. Depois disso, a estratégia agrupa todas células de fronteira em regiões de fronteira (YAMAUCHI, 1997), (YAMAUCHI; SCHULTZ; ADAMS, 1998). A região de fronteira é uma região com células de fronteira adjacentes. A busca de diferentes regiões de fronteira pode ser feita por diferentes estratégias. Uma possibilidade é escolher uma celular de fronteira, marcar ela como parte de uma região de fronteira e checar todas as células vizinhas. Todas as vizinhas que são células de fronteiras irão ser marcadas como parte da região de fronteira e suas vizinhas também. Se nenhuma vizinha é uma célula de fronteira, a região completa é determinada. Depois disto, uma nova célula de fronteira é alcançada e o processo é repetido até que todas as células de fronteiras sejam marcadas. No final todas as regiões de fronteira serão extraídas (GANGL, 2014).

Depois de extraídas, todas as regiões de fronteira são avaliadas segundo o seu tamanho. Se uma região de fronteira é menor que o tamanho do robô ou um tamanho mínimo, esta é ignorada. Das regiões que sobram são calculados os centróides $c = (c_x, c_y)$, via:

$$c_x = \frac{1}{N_{região}} \cdot \sum_{i=1}^{N_{região}} x_{i,região} \tag{2.1}$$

$$c_y = \frac{1}{N_{região}} \cdot \sum_{i=1}^{N_{região}} y_{i,região} \tag{2.2}$$

Onde $N_{região}$ é a quantidade de pontos da região de fronteira. A região escolhida como próxima melhor para a exploração, é a região com distancia mínima do seu centroide para a pose do robô. Por isso esta estratégia também é chamada de detecção da fronteira mais próxima. Um bom critério de parada para esta estratégia é, se não há mais regiões de fronteira no mapa corrente, pare (GANGL, 2014).

Uma vantagem desta abordagem é sua capacidade de explorar tanto grandes espaços abertos, quanto espaços estreitos e desordenados, com paredes e obstáculos em orientações arbitrárias (GANGL, 2014).

2.2.2 Estratégia de Exploração Gulosa baseada no Ganho de Informação

Uma estratégia de exploração gulosa para um modelo de aprendizagem do ambiente pode ser determinada pelo *trade-off*[1] entre o ganho de informação e os custos de tomar aquela ação. Um ponto importante desta estratégia é que o ganho de informação é tratado de forma independente entre as diferentes células do *grid* do mapa. Então a estratégia de exploração escolhe a pose de exploração que maximiza o ganho de informação. Em (THRUN; BURGARD; FOX, 2005 apud GANGL, 2014), três diferentes metodologias para calcular o ganho por célula do grid, são mostradas:

- **Entropia:** A entropia de uma célula pode ser definida como:

$$H_p(m_i) = -p(m_i) \cdot log(p(m_i)) - (1 - p(m_i)) \cdot log(1 - p(m_i)) \tag{2.3}$$

  onde $p(m_i)$ é o probabilidade da célula no *occupancy grid* . Usando está entropia, células com maiores valores indica que a célula é inexplorada.

- **Ganho de informação esperado:** A entropia descrita no item anterior não leva em consideração a informação dos sensores. A entropia esperada depois do sensoriamento é dada por:

$$E[H_{p'}(m_i)] = p^+ \cdot H^+_{p'}(m_i) + p^- \cdot H^-_{p'}(m_i) \tag{2.4}$$

  onde $H^+_{p'}(m_i)$ é a entropia da probabilidade posterior de que a célula seja medida como ocupada. E $H^-_{p'}(m_i)$ é a entropia da probabilidade posterior de que a célula seja medida como livre. A probabilidade de que a célula seja medida como ocupada é dada por:

$$p^{[+]} = p_{true} \cdot p(m_i) + (1 - p_{true}) \cdot (1 - p(m_i)) \tag{2.5}$$

  onde p_{true} é a probabilidade de que a ocupação é corretamente medida e a atualização do occupancy grid:

$$p_i^{'} = \frac{p_{true} \cdot p_i}{p_{true} \cdot p_i + (1 - p_{true}) \cdot (1 - p(m_i))} \tag{2.6}$$

  A entropia do posterior $H^+_{p'}(m_i)$ é gerada pela equação 2.1, o mesmo vale para $H^-_{p'}(m_i)$. Analogamente a probabilidade de que a célula seja medida como ocupada, a probabilidade de ser medida como livre $p_{[-]}$ pode ser calculada como o a probabilidade complementar de $p_{[+]}$. No fim, o ganho de informação esperado é:

$$I(m_i) = H_p(m_i) - E[H_{p'}(m_i)] \tag{2.7}$$

[1] *trade-off* é uma expressão que define uma situação em que há conflito de escolha.

- **Ganho Binário:** Esta metodologia assume que a célula é explorada ou inexplorada e portanto as regiões com maior numero de células inexploradas é a que tem maior ganho de informação . Isto é a base da estratégia de exploração baseada em fronteiras descrita anteriormente.

O critério de parada desta estratégia é, pare se novas informações não podem ser obtidas pelo movimento do robô, isto é se o ganho esperado de cada célula é zero (GANGL, 2014).

2.2.3 Estratégia de Exploração baseada em Características

Recentemente, muito tem sido escrito sobre algoritmos de Localização e Mapeamento simultâneos (SLAM) baseados em características. Tais algoritmos tentam responder a pergunta "onde eu estou e o que está ao meu redor?". Isto leva a uma óbvia próxima pergunta, que é "para onde eu deveria ir?". A estratégia proposta por Newman e colaboradores em (NEWMAN; BOSSE; LEONARD, 2003) visa fornecer uma resposta plausível para esta questão, tentando resolver o problema não de uma perspectiva teórica idealizada, mas a partir de um contexto de navegação em tempo real com sensores ruidosos em um ambiente desconhecido e incerto.

Foi adotada a ideia de que as características observadas devem inspirar por elas mesmas o processo de exploração (NEWMAN; BOSSE; LEONARD, 2003). Intuitivamente, examinar um mapa existente deveria inspirar um plano para expandir sua cobertura e refinar seus detalhes. Os seguintes benefícios são elencados para exploração (NEWMAN; BOSSE; LEONARD, 2003):

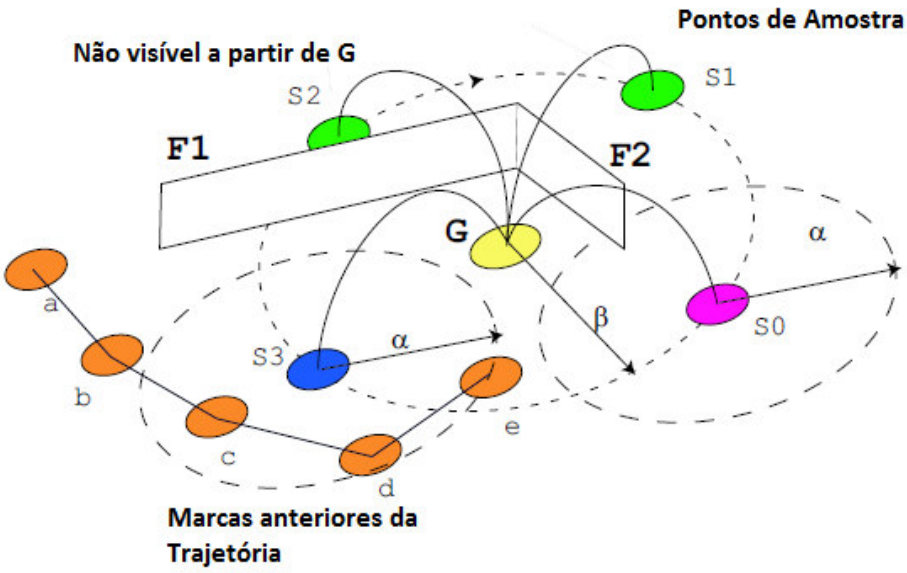
- o objetivo da exploração é descobrir e habilitar o mapeamento de uma nova área. A navegação deve se basear em observações confiáveis do ambiente;
- o foco da exploração é derivado da localização e geometria das características que podem servir para guiar o robô eficientemente para lugares de interesse fora de sua imediata vizinhança;
- as características locais, quando consideradas, determinam que a complexidade computacional e de memória do algoritmo são constantes e independentes do tamanho do mapa;
- a exploração baseada em características é independente do tipo de sensoriamento empregado.

A estratégia foca no problema de seleção de ações e na tarefa de exploração da perspectiva da construção de mapa na presença de incerteza. As incertezas no mapa poderiam ser manipuladas com técnicas de filtro de partículas e Filtro de Kalman Estendido (THRUN;

BURGARD; FOX, 2005). O filtro de partículas é uma distribuição probabilística de possíveis poses do robô. Cada partícula, corresponde a uma possível pose, dada as observações dos sensores $\phi_{1:t}$ e do comando de controle $u_{1:t}$, que vem da odometria. Para cada instância de partícula, é associado um mapa local, que corresponde ao mapeamento dos marcos, dada a provável pose (BARCELOS; VIDAL; ROSA, 2014). Newman e seus colaboradores definem propriedades como sendo as constituintes de uma exploração que visam levar o robô a visitar áreas abertas e esparsas do ambiente, a mantê-lo longe de lugares já visitados, a explorar localmente e somente explorar áreas distantes quando a vizinhança já for totalmente explorada, e explorar com aumento de cobertura gradual. A seleção da ação é feita pela avaliação da utilidade de visitar qualquer conjunto de localizações (NEWMAN; BOSSE; LEONARD, 2003).

Cada característica gera um conjunto de objetivos, para os quais um conjunto de pontos de amostra S são regularmente distribuídos ao redor dele, com um determinado raio. Espera-se então atribuir a cada objetivo uma pontuação que é a indicação de sua utilidade. Para isso retiram-se elementos de S, segundo os critérios de *esparsidade* e *novidade*. O critério de *esparsidade* avalia se há um caminho livre entre o objetivo e o ponto de amostra. Já o critério de *novidade* verifica se o campo de visão a partir daquele ponto de amostra já não foi coberto. Se um dos pontos de amostra não cumprir um dos critérios, é removido de S, e ao final do processo é atribuído ao objetivo um valor de utilidade que é calculado como a razão do tamanho final de S pelo seu tamanho inicial. A Figura ?? apresenta uma visualização deste algoritmo.

Figura 1 – A figura abaixo mostra a avaliação de alvo G. Onde pontos de amostra $(S1, S2, S3, S4)$ são colocados ao redor com raio β. Verfica-se se não há marcas prévias da trajetória (a, b, c, d, e) dentro de um raio α, se um ponto de amostra é visível a partir de G.



Fonte: (NEWMAN; BOSSE; LEONARD, 2003).

O processo de seleção de ação requer a determinação da existência de um caminho livre e claro entre diferentes entidades. Os autores então definem uma função de visibilidade que avalia para um valor não nulo v_{ij} se há um caminho claro entre duas entidades e_i e e_j (NEWMAN; BOSSE; LEONARD, 2003).

Um algoritmo de exploração precisa operar tanto em contexto local quanto global. Quando operando em contexto local, as decisões são feitas em relação ao que é visível e as características próximas. O modo global de operação é requerido para determinar regiões interessantes que estão distantes da atual posição do robô.

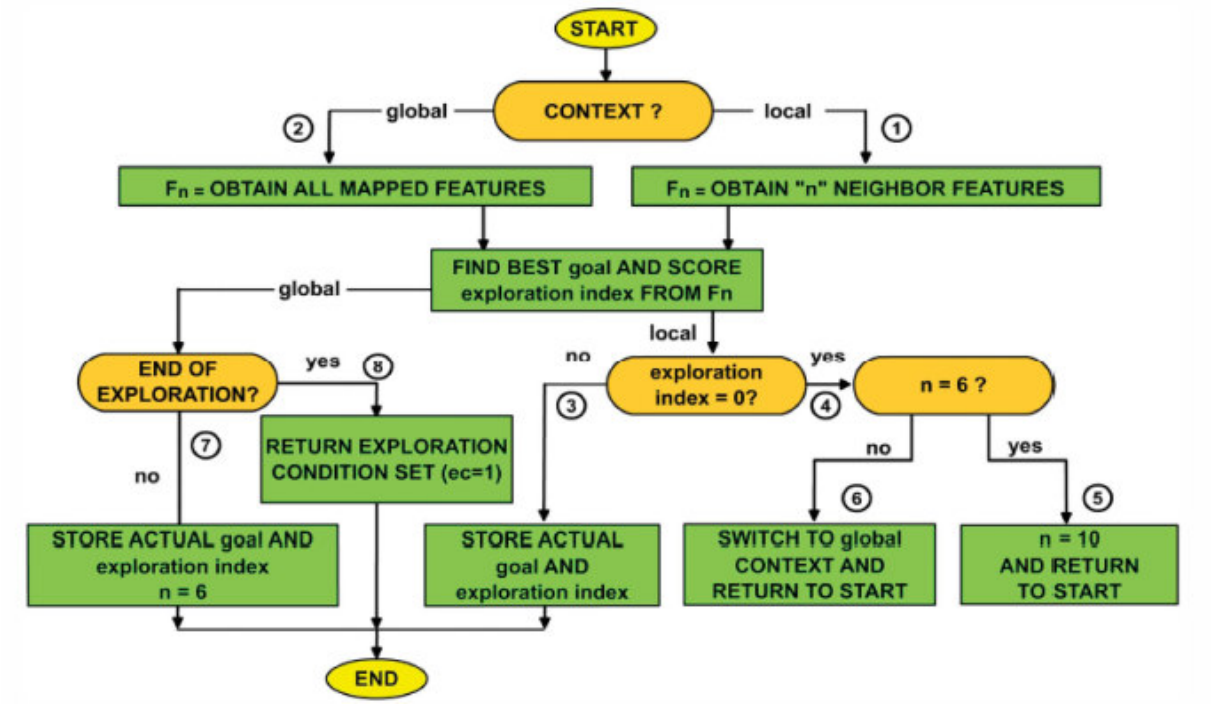
Buonocore e colaboradores apresentam um algoritmo de exploração autônoma aplicado a robô munido de sensores de baixo custo, o que significa que os dados brutos adquiridos são ruidosos e difíceis de se trabalhar com eles em tempo real (BUONOCORE; ALMEIDA; NASCIMENTO, 2014). O algoritmo proposto, denotado como *Feature-based Algorithm - FBA*, opera nos contextos local e global, tendo dois critérios básicos de seleção de ação, como proposto em (NEWMAN; BOSSE; LEONARD, 2003): geração de objetivos, e seleção de objetivos.

O *FBA* é chamado por um filtro de partículas, *FastSLAM*, sempre quando o robô atinge uma pose e tem adquirido medidas do ambiente. Os dados são processados por um algoritmo de fusão, gerando um conjunto de segmentos de retas (características). Estas características são incorporadas ao mapa em construção. A próxima pose do robô a ser escolhida é definida como objetivo. Se o algoritmo está no contexto local, seis ou dez características próximas são usadas. No contexto global, todas as características são processadas (Ver Figura 2) (BUONOCORE; ALMEIDA; NASCIMENTO, 2014).

O *FBA* sempre se inicia no contexto local. Caso a busca por um objetivo falhe, o número de características usadas sobe para dez. Se um objetivo é selecionado, este é passado para o filtro de partículas. No contexto global há três possibilidades. A primeira regra é procurar objetivos que são parcialmente visíveis a partir da posição atual do robô. A segunda regra, para o caso de falha da primeira, é procurar por objetivos que foram avaliados na pose inicial da tarefa de exploração. A terceira regra procura aberturas no ambiente que não são visíveis a partir da pose atual do robô (BUONOCORE; ALMEIDA; NASCIMENTO, 2014).

Cada característica envolvida na fase de geração de objetivos produz dois ou quatro objetivos, localizados próximos aos pontos finais do segmento de reta. Nesta fase, cada objetivo é testado para ser validado como apto para a próxima fase, buscando-se evitar a seleção de objetivos muito próximos das características para evitar colisões. Na segunda fase, o *FBA* gera seis pontos de amostra radialmente distribuídos ao redor de cada objetivo. Estes pontos de amostra permitem atribuir um valor a um objetivo para revelar sua 'potencialidade'. As três fases intermediárias são regras aplicadas a fim de reduzir-se a pontuação original de cada objetivo. Estas regras aplicam o conceito de visibilidade total

Figura 2 – Fluxograma da operação do FBA.



Fonte: (BUONOCORE; ALMEIDA; NASCIMENTO, 2014).

entre duas entidades (BUONOCORE; ALMEIDA; NASCIMENTO, 2014).

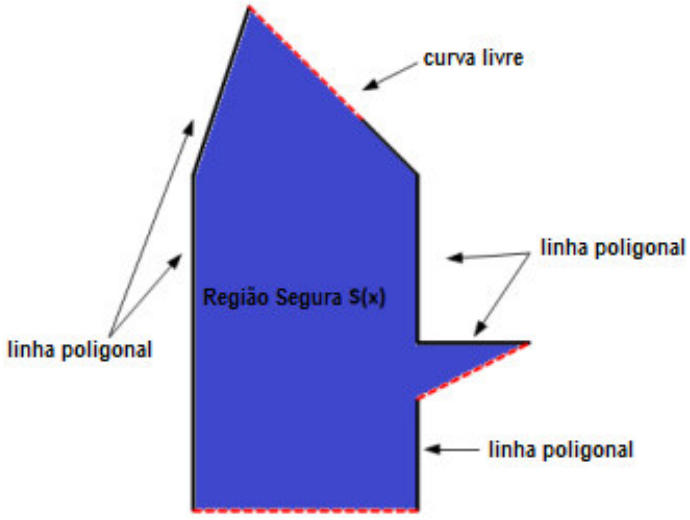
A primeira regra, chamada de 'critério de Avaliação de Esparsidade', tem o objetivo de manter a pontuação de áreas pouco povoadas por características, tornando-as mais atrativas. A segunda regra, 'critério de Avaliação de Novidade', tenta eliminar pontos de amostra muito próximos, o que leva o robô a buscar áreas inexploradas. A terceira regra, 'critério de Aberturas no ambiente' (*Environment Openings criterion*), remove os objetivos que não são visíveis a partir da pose atual do robô. Então, na última fase um dos objetivos restantes é escolhido (BUONOCORE; ALMEIDA; NASCIMENTO, 2014).

2.2.4 Estratégia de Exploração baseada em Pontos de Vista

Proposta por González-Banos e Latombe, a estratégia de exploração baseada em pontos de vista busca construir um mapa através da seleção da melhor localização para a exploração, levando em consideração a qualidade e a quantidade de novas informações acessíveis a partir da localização escolhida (GONZÁLEZ-BAÑOS; LATOMBE, 2002).

A ideia central desta estratégia é o uso de regiões seguras. Uma região segura é a maior região do mapa garantidamente livre de obstáculos, considerando todos os dados sensoriados (GONZÁLEZ-BAÑOS; LATOMBE, 2002). O robô, a partir da posição x_k, realiza varredura do seu ambiente. A partir dos dados escaneados as regiões seguras locais são determinadas. Estas podem ser extraídas agrupando todos os pontos de escaneamento e colocando linhas poligonais ao redor dos grupos de pontos (GONZÁLEZ-BAÑOS;

Figura 3 – Região segura.



Fonte: (GANGL, 2014).

LATOMBE, 2002). Do conjunto de linhas poligonais extraídas, a região segura S_{x_k} pode ser determinada e delimitada pelas linhas poligonais e curvas livres, conectando os finais das linhas poligonais (GANGL, 2014). A Figura 3 exemplifica as regiões seguras.

Após isso, várias poses de exploração candidatas serão geradas. Estas são uniformemente distribuídas sobre a região segura computada S_{x_k} antes e são denotadas por $L_{candidatas}$. Depois de geradas, as poses candidatas são avaliadas. A pontuação para uma pose candidata $q \in L_{candidatas}$ pode ser calculada como seguir (GONZÁLEZ-BAÑOS; LATOMBE, 2002):

$$g(q) = A(q) \cdot e^{-\lambda \cdot l(x_k, q)} \tag{2.8}$$

onde $l(x_k, q)$ é o tamanho do caminho livre de colisão da pose corrente para a candidata e λ é uma constante, que pondera a influência dos custos do movimento sobre a função. A medida $A(q)$ é o ganho de informação na pose candidata q. A pose candidata com pontuação máxima é então escolhida como próximo alvo para exploração.

O critério de parada é dado pela seguinte regra: se os limites não contêm mais as curvas livres então o mapeamento está completo (GANGL, 2014). Na prática, as curvas livres são checadas pelo seu tamanho. Caso o tamanho seja menor que um limiar, o mapeamento é considerado completo (GONZÁLEZ-BAÑOS; LATOMBE, 2002).

2.2.5 Estratégia de Exploração baseada em Comportamentos

Considerando o nível de planejamento, Juliá e colaboradores classificam as técnicas de exploração em dois grupos (JULIÁ et al., 2011):

- **Deliberativo:** A exploração é dirigida por uma camada de auto nível que avalia

um planejamento de movimento a longo prazo com conhecimento total do mapa estimado global e das posições do robô;

- **Reativo:** A exploração é conduzida por comportamentos de baixo nível que trabalham em tempo real com conhecimento parcial do mapa e avaliam somente movimentos a curto prazo.

No primeiro grupo, o modo de operação usual consiste em planejar um caminho para uma célula de fronteira. Estas técnicas diferem no modo em que elas coordenam os robôs, atribuindo a eles diferentes fronteiras. No segundo grupo, a exploração é realizada com comportamentos reativos básicos que são usualmente modelados como campos potenciais (ARKIN, 1998 apud JULIÁ et al., 2011). Em (JULIÁ et al., 2011), propõe-se o uso de vários comportamentos para compor a exploração, sendo que cada um deles considera um aspecto da exploração e avalia um padrão de comportamento definido por um campo potencial. Campos potenciais são gerados por funções gaussianas simples que são facilmente ajustadas por meio da largura e amplitude desejadas. A largura da função gaussiana para cada comportamento está relacionada ao raio de influência desejado. É necessário prestar atenção no sinal da gaussiana. Um sinal positivo significa um comportamento repulsivo (potencial alto), enquanto que um sinal negativo significa um comportamento atrativo (potencial baixo). Alguns dos comportamentos usados por (JULIÁ et al., 2011) são listados a seguir:

- *Vá para a Fronteira:* As células da fronteira atraem os robôs uma vez que estas células conduzem o robô a zonas novas de exploração:

$$P_1(i,j) = -\sum_{k \in c_f} exp\left(-\frac{(i-k_i)^2 + (j-k_j)^2}{2\sigma_1^2}\right), \tag{2.9}$$

  sendo $P_1(i,j)$ o potencial da célula (i,j) associado a este comportamento, σ_1 é a largura para este comportamento, c_f é o subconjunto das células de fronteira, e (k_i, k_j) são as coordenadas da célula k.

- *Vá para as Células Inexploradas:* Este comportamento considera as células inexploradas como pontos de atração:

$$P_2(i,j) = -\sum_{k \in c_i} exp\left(-\frac{(i-k_i)^2 + (j-k_j)^2}{2\sigma_2^2}\right), \tag{2.10}$$

  onde $P_2(i,j)$ é o potencial da célula (i,j) associado a este comportamento, σ_2 é a largura para este comportamento, c_i é o subconjunto das células inexploradas.

- *Evite Obstáculos:* celulas ocupadas repelem os robôs a fim de evitar colisões:

$$P_3(i,j) = \sum_{k \in c_o} exp\left(-\frac{(i-k_i)^2 + (j-k_j)^2}{2\sigma_3^2}\right), \tag{2.11}$$

sendo $P_3(i,j)$ o potencial da célula (i,j) associado a este comportamento, σ_3 é a largura para este comportamento, c_o é o subconjunto das células ocupadas.

- *Evite outros robôs*: múltiplos robôs requerem sinalizadores específicos:

$$P_4(i,j) = \sum_{k \in c_r} exp\left(-\frac{(i-k_i)^2 + (j-k_j)^2}{2\sigma_4^2}\right), \tag{2.12}$$

onde $P_4(i,j)$ trata-se do potencial da célula (i,j) associado a este comportamento, σ_4 é a largura para este comportamento, c_r é o subconjunto das células onde os outros robôs estão situados.

Várias técnicas podem ser aplicadas para fazer a fusão de comportamentos como, por exemplo, subsunção, votação, soma ponderada ou lógica *fuzzy* (JULIÁ et al., 2011). Uma simples coordenação consistindo de uma soma ponderada do potencial gerado por cada comportamento é suficiente quando é escolhida a largura apropriada para cada comportamento. O peso de cada comportamento representa sua relativa importância, dentro do potencial global, dado por (JULIÁ et al., 2011):

$$P = \sum_b e_b \lambda_b P_b \tag{2.13}$$

onde P_b é o campo potencial para o comportamento b, λ_b é seu peso, e e_b assume os valores 0 ou 1, designando se o comportamento está habilitado ou não.

Usar este tipo de comportamentos pode conduzir ao aparecimento de mínimos locais, que ocorrem quando os comportamento anulam-se mutuamente de tal maneira que um ponto de potencial mínimo aparece. Uma vez que os robôs se movem seguindo as zonas de menor potencial eles podem ficar presos nestes pontos.

2.3 Problemas de Otimização

Matematicamente, os problemas de otimização podem ser escritos na seguinte forma genérica (YANG, 2008):

$$\begin{array}{lll} minimizar & f_i(\mathbf{x}) & onde\ i = 1, ..., M \\ sujeito\ a & h_j(\mathbf{x}) = 0, & onde\ j = 1, ..., J \\ & g_k(\mathbf{x}) \leq 0, & onde\ k = 1, ..., K \end{array} \tag{2.14}$$

onde $f_i(\mathbf{x})$, $h_j(\mathbf{x})$ e $g_k(\mathbf{x})$ são funções do vetor

$$\mathbf{x} = (x_1, x_2, ..., x_n)^T. \tag{2.15}$$

Os componentes x_i de $\mathbf{x}$ são chamados de variáveis de decisão, e elas podem ter domínio continuo, discreto ou ambos (OLIVEIRA, 2004). A função $f_i(\mathbf{x})$ onde $i = 1, 2, ..., M$

define um objetivo que pode ser tanto minimizado, quanto maximizado. O espaço de combinação de valores, aceitáveis ou não, para as variáveis de decisão é chamado de espaço de busca. As igualdades h_j e desigualdades g_k definem regras matematicamente modeladas e são chamadas de restrições. .

Uma solução $s^* \in S$ é um ótimo global se ele tem o valor de função objetivo melhor que todas as soluções do espaço de soluções, isto é, $\forall s \in S$, $f(s^*) \leq f(s)$. Por isso, o principal objetivo na solução de um problema de otimização é a solução ótima global s^*. É importante dizer que muitas soluções ótimas podem existir para um dado problema (TALBI, 2009).

2.4 Algoritmos de Otimização

Algoritmos exatos têm a capacidade de obter as melhores soluções e garantir sua optimalidade. Para problemas NP-Difíceis, algoritmos exatos são algoritmos de tempo não-polinomiais ou intratáveis (exponencial, fatorial). Métodos aproximados (ou heurística) geram soluções de alta qualidade em um tempo razoável para o uso prático, sem a garantia de encontrar uma solução ótima global (BLUM; ROLI, 2003).

Na classe de métodos exatos, pode-se encontrar os seguintes algoritmos clássicos: programação dinâmica; *branch-and-bound* e derivados, desenvolvidos pela comunidade de pesquisa operacional, programação por restrições (*constraint programming*); a família de algoritmos de busca A^*, desenvolvidos pela comunidade de inteligência artificial (TALBI, 2009); e a família de algoritmos de otimização baseados em gradiente (PRESS et al., 2007).

Algoritmos heurísticos ou aproximados são desenvolvidos objetivando a resolução de problemas para os quais métodos exatos tendem a executar em tempo computacional impraticável. Todavia, há um claro compromisso entre o esforço computacional da heurística e a qualidade da solução que ela consegue obter (OLIVEIRA, 2004).

É possível categorizar algoritmos de otimização dependendo de vários fatores, qual sejam (TALBI, 2009):

- estratégia construtiva ou vizinhança;
- ponto único ou populacional;
- baseadas em memória ou *multi-start*;
- evolutivas ou *black-board*;

Heurísticas **construtivas** são aquelas que, a partir de uma solução vazia, adicionam-se novos elementos (variáveis, vértices, arestas) iterativamente, como parte do

processo decisório que vai levar a uma solução completa viável. As heurísticas gulosas se enquadram nesse comportamento, tentando a cada iteração, maximizar a melhoria local (OLIVEIRA, 2004).

Uma heurística de melhoria (ou de **vizinhança**) é aquela em que, de acordo com regras pré-estabelecidas, modifica uma dada solução num instante qualquer, gerando novas soluções a cada iteração até que um determinado critério de parada seja alcançado. Tipicamente, o critério de parada é atingido quando a solução corrente não puder ser melhorada. Neste caso, diz-se que a solução corrente é um *ótimo local*, segundo a heurística (OLIVEIRA, 2004).

Tanto as heurísticas construtivas, quanto as de melhoria, em geral, são procedimentos específicos para determinada classe de problemas e determinísticos. Para uma mesma solução inicial e os mesmos parâmetros de ajuste do método, sempre se chega a uma mesma solução final, geralmente ótima dentro de uma certa *localidade* (BLUM; ROLI, 2003).

Pode-se dizer que um algoritmo heurístico qualquer, em uma dada execução, *está preso* a um ótimo local, quando ele assumir um conjunto de pontos do espaço de busca a partir do qual ele não consegue mais alcançar nenhum outro ponto melhor, usando os valores correntes de seus parâmetros de desempenho.

O termo metaheurística primeiramente introduzido em (GLOVER, 1986), deriva da composição de duas palavras gregas. *Heurística* deriva do verbo *heuriskein* que significa "encontrar", enquanto o sufixo *meta* significa "além, em um auto nível".

"Uma metaheurística pode ser definida como processo de geração iterativa que guia uma heurística subordinada ao combinar de forma inteligente diferentes conceitos para a exploração e intesificação no espaço de busca. Estratégias de aprendizagem são usadas para estruturar a informação a fim de que se ache eficientemente soluções quase-ótimas." (OSMAN; LAPORTE, 1996 apud BLUM; ROLI, 2003)

"Uma metaheurística é um conjunto de conceitos que podem ser usados para definir métodos heurísticos que podem ser aplicados para um amplo conjunto de diferentes problemas. Em outras palavras, uma metaheurística pode ser vista como um framework algorítmico que pode ser aplicado para diferentes problemas de otimização com relativamente poucas modificações para fazê-lo aplicável para um problema especifico." (METAHEURISTICS..., 2000 apud BLUM; ROLI, 2003)

Em níveis de hierarquia, uma metaheurística é uma heurística que emprega outras heurísticas, com certo grau de flexibilidade para possibilitar sua fácil adaptação a uma grande variedade de problemas, providas de uma ou mais estratégias para torná-las mais robustas e assim terem maiores chances de *escapar* de ótimos locais (BIANCHI et al., 2009).

Algoritmos Evolutivos (AE), por exemplo, são populacionais, o que permite explorar mais de uma região do espaço de busca ao mesmo tempo (LINDEN, 2006), mantendo-se uma **memória** representativa das soluções encotradas ao longo da evolução. O *recozimento simulado* opera com somente uma solução, mas aceita soluções piores do que a corrente, dependendo de certa probabilidade (KIRKPATRICK; GELATT; VECCHI, 1983). A busca tabu mantém mecanismos de memória que servem para evitar movimentos repetidos (GLOVER, 1996).

Mecanismos de *reinicialização* (ou ***multi-start***), quando soluções melhores não podem mais ser encontradas, também são comuns em algumas metaheurísticas de **ponto único**, como o *GRASP* (FEO; RESENDE, 1989). Quanto mais efetivos forem tais mecanismos, mais a heurística passa a ter uma característica de otimizador global, ou seja, algoritmo com capacidade de encontrar o ótimo global, independente da solução inicial ou conjunto delas (OLIVEIRA, 2004).

Segundo (BLUM; ROLI, 2003), as propriedades fundamentais que caracterizam as metaheuristicas são:

- Metaheuristicas são estratégias que guiam o processo de busca;
- O objetivo é explorar eficientemente o espaço de busca a fim de encontrar soluções ótimas ou quase-ótimas;
- Técnicas que constituem as metaheurísticas variam dos procedimentos de busca local para processos complexos de aprendizagem;
- Algoritmos metaheurísticos são aproximados e usualmente não-determinísticos;
- Incorporarem usualmente mecanismos para escaparem de áreas confinadas do espaço de busca;
- Os conceitos básicos de metaheurísticas permitem um nível abstrato de descrição;
- Metaheurísticas não são especializadas em um determinado problema
- Conhecimento de domínio específico para controle de heurísticas de baixo nível pode ser empregado como estratégia de nível maior;
- Metaheurísticas podem usar experiência ou mecanismo de memória para guiar a busca.

2.4.1 Algoritmo Genético

Algoritmos Genéticos (AG) foram propostos por John Holland (**??**), em sua tese, na década de 70, como uma estratégia adaptativa que pode ser usada como otimizador

global e inspirado pela genética e evolução populacional, incluindo ideias como hereditariedade, frequência gênica, bem como pelo entendimento Mendeliano de estrutura, tal como cromossomos, genes e alelos, e dos mecanismos, tais como recombinação e mutação (BROWNLEE, 2011).

O AG segue a metáfora: indivíduos de uma população contribuem com seus materiais genéticos, na forma de prole, chamado de genótipo, proporcionalmente à adequação do seu genoma expresso, chamado de fenótipo, ao ambiente. A próxima geração é criada através de um processo de acasalamento que envolve recombinação de dois genomas individuais na população com a introdução de erros de cópia aleatórios, chamada de mutação. Este processo iterativo pode resultar em uma melhor adaptação de ajuste entre os fenótipos dos indivíduos de uma população e o meio ambiente.

O objetivo do algoritmo genético é maximizar a recompensa das soluções candidatas (indivíduos) na população e minimizar uma função de custo do domínio do problema. Para isso o algoritmo repetidamente aplica processos (operadores genéticos) que imitam os mecanismos da recombinação (crossover) e mutação genética na população de soluções candidatas, onde a função de custo aplicada a uma representação decodificada de uma candidata, governa as contribuições probabilísticas que um individuo candidato pode fazer para a próxima geração de soluções candidatas. Neste trabalho usamos um algoritmo genético de estado fixo (steady-state) com codificação real, empregando como operadores genéticos a seleção por roleta, o blend crossover (BLX 0.25) e a mutação não uniforme, usados em (OLIVEIRA; LORENA, 2004). O Algoritmo 1 mostra a estrutura de um Algoritmo Genético:

Algoritmo 1 Algoritmo Genético

Entrada: O tamanho da população P_{size}, $P_{crossover}$, $P_{mutacao}$
Saída: o melhor global $\mathbf{s}_{best}$.

1: Função objetivo $f(x)$, $\mathbf{x} = (x_1, ..., x_d)^T$
2: Gere a população inicial $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
3: **enquanto** ¬ CriterioDeParada() **faça**
4: Pais ← SelecionaPais($Populacao, P_{size}$)
5: Filhos ← ∅
6: **para cada** $Pai_1, Pai_2 \in Pais$ **faça**
7: $Filho_1, Filho_2$ ← Crossover($Pai_1, Pai_2, P_{crossover}$)
8: Filhos ← Mutacao($Filho_1, Filho_2, P_{mutacao}$)
9: **fim para**
10: AvaliaPopulacao(Filhos)
11: $\mathbf{s}_{best}$ ← MelhorSolucao(Filhos)
12: Populacao ← Substitui(Populacao, Filhos)
13: **fim enquanto**
14: **retorne** $\mathbf{s}_{best}$

2.4.2 Algoritmo de Vagalumes

O Algoritmo de Vagalumes (AV) (*Firefly Algorithm - FA*) foi desenvolvido por Xin-She Yang entre 2007 e 2008 na Universidade de Cambridge (YANG, 2009) e baseia-se nos padrões da luz emitidos por vagalumes, supostamente associados ao comportamento desses insetos. Em essência, o AV usa três regras idealizadas (YANG; HE, 2013):

- Vagalumes são unissex de forma que são atraídos para outros vagalumes independente do seu sexo;
- A atratividade é proporcional ao brilho, e este decresce a medida em que a distância cresce. Assim para quaisquer dois vagalumes brilhando, aquele de menor intensidade irá mover-se em direção ao de maior. Se não houver diferença de brilho o movimento ocorre ao acaso;
- O brilho de um vagalume é determinado pela imagem da função objetivo;

Como a atratividade do vagalume é proporcional à intensidade da luz vista por seus vizinhos, pode-se definir a variação da atratividade β com a distancia r por:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.16}$$

onde β_0 é a atratividade em $r = 0$. Esta equação deriva da ideia de que intensidade da luz I varia com a distancia r seguindo a lei do quadrado inverso:

$$I(r) = \frac{I_s}{r^2} \tag{2.17}$$

onde é I_s é a intensidade da luz na fonte. Além disso, para um dado meio com coeficiente de absorção de luz γ segundo a equação:

$$I(r) = I_0 e^{-\gamma r} \tag{2.18}$$

onde I_0 é a intensidade de luz original. A fim de evitar a singularidade no $r = 0$, o efeito combinado da lei do quadrado inverso com a atração pode ser aproximada com seguinte forma Gaussiana:

$$I(r) = I_0 e^{-\gamma r^2} \tag{2.19}$$

Substituindo $I(r)$ por β e I_0 por β_0, uma vez que estes são proporcionais, obtém-se a equação 2.16.

O movimento do vagalume i que é atraído para outro vagalume j mais atrativo, é determinado por (YANG; HE, 2013):

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \beta_0 e^{-\gamma r^2}(\mathbf{x}_j^t - \mathbf{x}_i^t) + \alpha_t \epsilon_i^t \tag{2.20}$$

onde o segundo termo diz respeito a atração do vagalume i sobre o vagalume j. O terceiro termo introduz aleatoriedade no movimento do vagalume i, com α_t sendo o parâmetro de aleatoriedade, e ϵ_i^t é um vetor de numeros aleatórios distribuídos segundo uma distribuição Gaussiana ou uniforme no tempo t. Se $\beta_0 = 0$, o movimento tende a um simples andar aleatório (*random walk*). Em contrapartida, se $\gamma = 0$, o algoritmo é reduzido a uma variante da Otimização por Exame de Particulas (*Particle Swarm Optimization - PSO*) (YANG; HE, 2013). O pseudocódigo 2 mostra o Algoritmo de Vagalumes.

Algoritmo 2 Algoritmo de Vagalumes

Entrada: O tamanho da população P_{size}, coeficiente de absorção de luz γ
Saída: o melhor global $\mathbf{g}_*$.

```
Função objetivo f(x), x = (x_1, ..., x_d)^T
Gere a população inicial de vagalumes x_i (i = 1, 2, ..., n)
A intensidade do brilho I_i em x_i é determinada por f(x_i)
enquanto ¬ CriterioDeParada() faça
    para i ← 1, ..., n faça
        para j ← 1, ..., n faça
            se I_i < I_j então
                Mova o vagalume i em direção a j
            fim se
            Varie a atratividade com distancia r via exp(−γr)
            Avalie as novas soluções e atualize a intensidade do brilho
        fim para
    fim para
    RanqueiaPopulacao(Populacao)
    g_* ← MelhorSolucao(Populacao)
fim enquanto
retorne g_*
```

2.4.3 Paisagem de Aptidão

Um problema de busca é caracterizado por um *espaço de busca codificado*, que pode ser entendido como um conjunto de pontos S sobre o qual a busca é conduzida, e uma função de avaliação f responsável pela atribuição de um valor numérico para cada elemento $s \in \mathsf{S}$. Cada ponto significa uma solução, i.e., uma combinação diferente de valores que cada variável do problema pode assumir.

O espaço de busca codificado[2] não necessariamente é o mesmo espaço do domínio real do problema. Durante o processo de modelagem de um AE, o conjunto de soluções que os indivíduos podem representar passa por um processo de codificação que o torna computacionalmente implementável. Para a avaliação de aptidão é necessário antes a decodificação da solução.

[2] O termo *espaço de busca* é usado, neste trabalho, como uma simplificação para *espaço de busca codificado.*, mas têm o mesmo significado.

O desempenho das metaheurísticas é sensível à forma como é feita a codificação. A população de indivíduos (cromossomos ou vagalumes), em uma dada iteração, é uma amostra relativamente pequena dos pontos de S e são necessárias sucessivas *amostragens* para que soluções melhores sejam obtidas. Uma *amostragem* significa geração de indivíduos e a avaliação do ponto que ele assumiu. Mesmo o espaço de busca contínuo é discretizado através de amostras.

A função de avaliação f mede a qualidade da solução candidata e está associada à função objetivo do problema. No caso do AG, f é conhecida como aptidão (*fitness*) do indivíduo, enquanto para o AV f é a intensidade do brilho do indivíduo. Em ambos os algoritmos, f quantifica o indivíduo em termos da função objetivo:

$$f : \mathsf{S} \rightarrow \Re^{\tau} \tag{2.21}$$

onde $\tau > 1$ significa f relacionando múltiplos objetivos (COELLO, 2000).

A função de avaliação pode ser a própria função objetivo do problema ou pode ser utilizado algum tipo de normalização para reduzir a diferença de avaliação entre indivíduos e assim evitar que um indivíduo super-avaliado domine excessivamente o processo de seleção (LINDEN, 2006).

Uma métrica de distância está relacionada ao esforço necessário para transformar uma solução em outra, ou seja, transitar de um ponto para outro no espaço de busca. Existem várias formas de defini-la. A distância de *hamming* e a distância *euclidiana* são típicas métricas usadas em espaços de busca discretos e contínuos, respectivamente.

No processo de geração de novas soluções a partir de uma solução inicial s e usando-se uma dada heurística H, define-se vizinhança $\varphi^H(s)$ como uma relação de *proximidade* entre soluções no grafo $\mathcal{G}(V, A)$. A vizinhança de uma solução é um conjunto de soluções que podem ser geradas (ou alcançadas), a partir dela, executando-se movimentos permitidos pela heurística em uso.

Em um espaço de busca contínuo, por exemplo, a vizinhança pode ser definida como um conjunto $\varphi^r(s)$ de pontos dentro da hiperesfera de centro s e um raio r qualquer que pode estar relacionado à métrica euclidiana de distância.

2.4.4 Características dos algoritmos

O AG e o AV têm em comum o fato de serem baseados em população, ou seja, trabalham com um conjunto de pontos ao mesmo tempo, utilizando o conceito de vizinhança e memória (TALBI, 2009). Portanto, estes tipos de algoritmos descrevem o refinamento de um conjunto de soluções no espaço de busca (PANTRIGO et al., 2005).

Existem muitas vantagens do AG sobre os algoritmos tradicionais de otimização e duas das mais notáveis vantagens são: a habilidade de lidar com problemas complexos e paralelismo; capacidade de lidar com vários tipos de otimização independente se a função objetivo é estacionária ou não (ou seja, muda com o tempo), linear ou não linear, contínua ou descontínua, ou com ruídos aleatórios (YANG, 2008). Por isso, o AG é um algoritmo muito popular na literatura, tendo sido aplicado com sucesso em uma ampla variedade de problemas (**??**).

Em comparação com outros algoritmos evolucionários, o AV tem sido aplicado a problemas complexos de otimização não linear devido a características como: codificação em números reais, fácil implementação, mecanismo de alta estabilidade, dependente apenas da comunicação global entre as partículas e, por isso, baixo custo computacional (RAMESH; MOHAN; REDDY, 2013).

Além disso, o AV tem sido eficiente para encontrar o ótimo global com altas taxas de sucesso, alcançando o ótimo, com menos chamadas função objetivo que outros algoritmos como a Otimização por Exame de Partículas e o Algoritmo Genético (YANG, 2009). Esta característica torna o algoritmo de Vagalumes aplicável ao problema exploração autônoma, uma vez que este é contínuo e exige respostas rápidas, coerente com a agilidade do robô em questão. O cálculo da função objetivo pode ser custoso, dependendo da complexidade do algoritmo que a representa ou da quantidade de dados usados para seu cálculo.

3 Exploração Autônoma baseada em Metaheurísticas

Características visíveis do mapa e campos potenciais têm sido usados para guiar robôs na tarefa de exploração autônoma. Campos potenciais, por exemplo, são usados para calcular o vetor direção correspondente à associação de robô à célula, indicando a direção a ser seguida, caso a trajetória de um dado robô utilize aquela célula. Essa estratégia reativa, do ponto de vista da otimização, é puramente gulosa e local, sem considerar anseios globais da exploração.

Uma estratégia global de exploração deve ter maior amplitude, limitada evidentemente ao fato de que toda a informação necessária para a exploração ótima não está inicialmente disponível e precisa ser adquirida progressivamente como parte do processo de exploração.

Neste Capítulo, a proposta de realizar exploração autônoma com suporte metaheurístico é descrita em detalhes. Inicialmente, define-se o ambiente a ser explorado como a superfície de uma função objetivo na qual os pontos de mínima energia são potencialmente os melhores alvos para o robô. Dessa forma, a geração de objetivos para o robô é definida com um problema de otimização contínua.

3.1 *Framework* conceitual

A tarefa de exploração pode ser desmembrada nas seguintes diretivas (ou objetivos):

1. Buscar a fronteira entre o conhecido e o desconhecido (maximizar o ganho de informação);
2. Realizar percursos usando as informações providas pelas características observadas do mapa (minimizar o risco de colisão);
3. Afastar-se de áreas já exploradas do mapa (minimizar esforço repetitivo);
4. Cumprir os objetivos da exploração por meio de comportamento simples, ponderando as diretivas anteriores (abordagem mono-objetivos).

O ambiente em exploração é modelado por uma função que associa a cada ponto do espaço euclidiano em $\mathbf{R}^2$ um valor $\in \mathbf{R}$ que corresponde ao grau de utilidade das células do *grid* do mapa. O grau de utilidade é computado em relação ao estado atual

do mapa, baseando-se em: a) ganho de informação; b) obstáculos mapeados; c) histórico de exploração do robô. Essas informações são mapeadas para a função que expressa o objetivo da exploração, segundo um comportamento desejado.

O domínio da chamada *função objetivo* forma o espaço de busca em que cada ponto assume seu respectivo grau de utilidade que varia à medida em que o mapa sofre atualizações. Considerando que a melhor solução para um dado mapa (ou solução ótima) corresponde à melhor pose para o robô, sucessivas otimizações tendem a buscar uma sequencia de poses que representam um comportamento globalmente ótimo dados os diferentes estados do mapa assumidos durante a exploração.

3.1.1 Comportamentos desejados

A função objetivo é formada por funções componentes que representam as possíveis reações do robô à informação coletada até o momento. Portanto, os componentes representam comportamentos capazes de guiar a exploração, quais sejam:

- *Vá para a fronteira:* as células da fronteira atraem o robô, conforme (JULIÁ et al., 2011), definindo-se este comportamento como:

$$C_f(p) = -\sum_{k \in c_f} exp\left(-\frac{d(p,k)}{2\sigma_f^2}\right), \tag{3.1}$$

  onde C_f é o potencial do ponto $p = (x, y)$ associado a este comportamento, σ_f é a largura do subespaço considerado para este comportamento, c_f é o subconjunto das células de fronteira e $d(p, k)$ é função distância entre os pontos p e k. Em termos de espaço de busca, as posições com menor distância para mais pontos de fronteira acumulada minimizam este componente da função. A ideia subjacente é que este componente conduza o robô a regiões que maximizem o ganho de informação futura;

- *Evite obstáculos:* as células ocupadas com obstáculos devem repelir o robô, similarmente a (JULIÁ et al., 2011), definindo-se este componente como:

$$C_o(p) = \sum_{k \in c_o} exp\left(-\frac{d(p,k)}{2\sigma_o^2}\right), \tag{3.2}$$

  onde $C_o(p)$ é o potencial de p associado ao comportamento, σ_o é a largura do subespaço considerado para o comportamento, c_o é o subconjunto de pontos marcados como ocupados. Em termos de espaço de busca, as posições com maior distância acumulada para mais pontos ocupados minimizam este componente da função, gerando uma expectativa que o robô evite posições próximas a obstáculos, e priorize regiões livres;

- *Evite posições visitadas:* células do mapa já visitadas devem repelir o robô, definindo-se o referido componente como:

$$C_v(p) = \sum_{k \in c_v} exp\left(-\frac{d(p,k)}{2\sigma_v^2}\right), \tag{3.3}$$

onde C_v é o potencial do ponto p associado a este comportamento, σ_v é a largura do subespaço considerado para este comportamento, c_v é o subconjunto dos pontos já visitados pelo robô, o que implica na busca por posições com maior distância acumulada para mais pontos visitados. Este comportamento busca evitar que o robô volte a regiões do ambiente já visitadadas.

3.2 Função Objetivo

A forma básica da função objetivo é a soma ponderada das funções componentes, baseada na proposição de (JULIÁ et al., 2011):

$$f(p) = \alpha \cdot C_f(p) + \beta \cdot C_o(p) + \gamma \cdot C_v(p) \tag{3.4}$$

onde α, β e γ correspondem às importâncias relativas dadas aos comportamentos. Dessa forma, o problema é notadamente multi-objetivo, com abordagem ponderada. Minimizar 3.4 implica em minimizar a distância do ponto p aos pontos de fronteira e maximizar a distância para os pontos ocupados e visitados. Com esta função busca-se um comportamento global que conduza o robô a explorar o ambiente de forma segura buscando sempre regiões novas.

Pode-se adicionar novos componentes à Eq. 3.4, sobretudo tentando incorporar comportamento que privilegie ganho de informação localmente. O conceito de localidade permite organizar melhor a exploração, buscando-se ordenar os pontos de interesse do mais próximo aos mais distantes. Em outras palavras, reduz-se assim o ganho procedente de grandes agrupamentos de pontos de fronteira distantes do robô.

As seguintes formulações buscam incorporar o conceito de localidade, relacionando a Eq. 3.4 com a distância δ entre o ponto de interesse p e a pose atual do robô r:

- *Perda Condicional:* células dentro de um raio de distância do robô são priorizadas chaveando-se ganhos diferentes:

$$f(p) = \begin{cases} \alpha_1 \cdot C_f(p) + \beta \cdot C_o(p) + \gamma \cdot C_v(p), & \text{se } \delta(p,r) \leq k \\ \alpha_2 \cdot C_f(p) + \beta \cdot C_o(p) + \gamma \cdot C_v(p), & \text{caso contrário} \end{cases} \tag{3.5}$$

onde k é o limiar de localidade para a distância euclidiana $\delta(p,r)$ e as constantes $\alpha_1 >> \alpha_2$ são positivas;

- *Perda linear:* a distância da célula para o robô equivale a um fator de penalidade linear :

$$f(p) = \alpha \cdot C_f(p) + \beta \cdot C_o(p) + \gamma \cdot C_v(p) + \rho\delta(p, r) \tag{3.6}$$

  onde ρ é a relevância do componente distância para o robô;

- *Perda proporcional:* a distância da célula para o robô equivale a fator de penalidade global proporcional a $\frac{1}{\rho\delta(p,r)}$:

$$f(p) = \frac{\alpha \cdot C_f(p) + \beta \cdot C_o(p) + \gamma \cdot C_v(p)}{\rho d(p, r)} \tag{3.7}$$

3.3 Decodificação

A função-objetivo modela o ambiente em exploração num dado instante, atribuindo graus de utilidade para as coordenadas $(x, y) \in \mathbb{R}^2$. A decodificação de (x, y) para o grid do mapa (i, j) é dado por:

$$\begin{aligned} i &= \left\lfloor \frac{x}{W} \right\rfloor \\ j &= \left\lfloor \frac{y}{H} \right\rfloor \end{aligned} \tag{3.8}$$

onde W e H são a largura e altura do mapa, respectivamente. Para o caso do ponto de interesse estar em um subespaço do grid para o qual não se tem informação alguma, ou seja não tenha sido *sensoriado* pelo robô, o processo de decodificação atribui a p um valor M grande de forma que $M >> \max\{C_*(p)\}$, sendo C_* a maior avaliação possível dentre todas as funções componentes da função objetivo, como mostra o pseudocódigo 3:

Algoritmo 3 Codifica a restrição referente à geração de soluções inviáveis

Entrada: Um ponto $p = (x, y)$.
Saída: Fitness do ponto p.

```
1: i = ⌊x/w⌋
2: j = ⌊y/h⌋
3: se mapa[i][j] é conhecido então
4:     retorne f(p)
5: else
6:     retorne M
7: fim se
```

3.4 Problema de Otimização

Tendo em vista o que foi dito nas sessões anteriores pode-se definir de forma geral o problema de exploração autônoma como o seguinte problema de otimização:

$$\begin{array}{ll} minimizar & \sum \alpha_i C_i(\mathbf{x}), \qquad onde\ k = 1, ..., N \\ sujeito\ a & h(k) = 1. \end{array} \tag{3.9}$$

onde

$$\mathbf{x} = (x, y)^T. \tag{3.10}$$

Sendo $C_k(\mathbf{x})$ a função componente referente ao k-ésimo comportamento que compõe a exploração e α_k o peso dado esse comportamento. A função $h(\mathbf{x})$ é uma função binária que assume o valor 1 nos pontos (x, y) que correspondem às células (i, j) do mapa conhecidas e 0 caso a célula (i, j) seja desconhecida.

4 Arquitetura do sistema e Ambiente

Neste capítulo, descreve-se a aplicação desenvolvida para ambiente simulado, usada para validação da proposta, incluindo também as tecnologias necessárias para o sua implementação, tais como o Sistema Operacional Robótico, que incorpora componentes para sensoriamento, localização e mapeamento, bem como controle robótico.

São apresentados os experimentos computacionais em simulador *V-REP*, considerando um cenário onde um robô youBot KUKA (YOUBOT, 2015), munido de scanner a laser, deve explorar um ambiente desconhecido *indoor*. O youBot KUKA é uma plataforma móvel omnidirecional, sobre a qual está montado um braço robótico de cinco eixos com garra dupla. Contudo, o braço e a plataforma móvel também podem ser utilizados separadamente. O grupo-alvo do youBot KUKA são instalações de pesquisa e escolas superiores, que podem realizar de forma muito simples os seus próprios comandos e ideias de aplicação com o robô (YOUBOT, 2015).

4.1 Sistema Operacional Robótico

O *Robot Operating System* (*ROS*) é um *framework* para escrever software robótico que inclui uma coleção de ferramentas, bibliotecas e convenções que objetivam simplificar a tarefa de criar comportamentos robóticos complexos e robustos através de uma grande variedade de plataformas robóticas (ROS, 2016).

No mais baixo nível, o *ROS* oferece uma interface de passagem de mensagens que provê comunicação interprocessos e é comumente chamada de *middleware*, oferecendo as seguintes facilidades (ROS, 2016):

- passagem de mensagem anônima via esquema *publish/subscribe*;
- gravação e reprodução de mensagens;
- chamada de procedimentos remotos do tipo *request/response*;
- sistema de parâmetros distribuído.

O ROS foi desenhado para ser neutro em relação à linguagem. Atualmente o ROS suporta quatro diferentes linguagens: C++, Python, Octave, and LISP, com o suporte a várias outras linguagens sendo desenvolvido (QUIGLEY et al., 2009).

4.1.1 Sistema de Tópicos

Mensagens são roteadas através de um sistema de transporte com semântica *publish/subscribe*, no qual cada nó envia uma mensagem pela fila de *publicação* em um dado tópico. O tópico, por sua vez, é usado para identificar o conteúdo da mensagem, cabendo ao nó interessado em determinado tipo de dado *subscrever* o tópico apropriado. Podem existir múltiplos publicadores e subscritores para um único tópico, e um único nó pode publicar e/ou subscrever para múltiplos tópicos. Em geral, publicadores e subscritores não sabem da existência uns dos outros. A ideia é desacoplar a produção de informação de seu consumo (ROS, 2016).

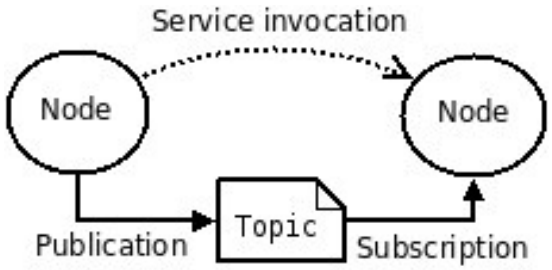
Pode-se pensar em um tópico como um barramento de mensagens fortemente tipado. Cada barramento tem um nome, e qualquer um pode se conectar a este barramento para enviar ou receber mensagens desde que elas sejam do tipo certo (ROS, 2016).

4.1.2 Serviços

O modelo *publish/subscribe* é um paradigma de comunicação bastante flexível, mas seu modo de transporte muitos-para-muitos não é apropriado para interações do tipo requisição/reposta, que são requeridos em sistemas distribuídos. O modelo *requisição/resposta* é feito via serviços, que são definidos por um par de estrutura de mensagens: uma para a requisição e outra para a resposta (ROS, 2016).

Um nó fornecedor oferece um serviço com um nome e um cliente pode usá-lo pelo envio de uma mensagem de requisição e espera da resposta. A biblioteca de cliente do ROS geralmente apresenta esta interação para o programador como se ela fosse uma chamada de procedimento remota (ROS, 2016). A Figura 4 ilustra os sistemas de comunicação dos ROS por tópicos e serviços.

Figura 4 – Comunicação publish/subscribe do RoS.



Fonte: http://wiki.ros.org/ROS/Concepts.

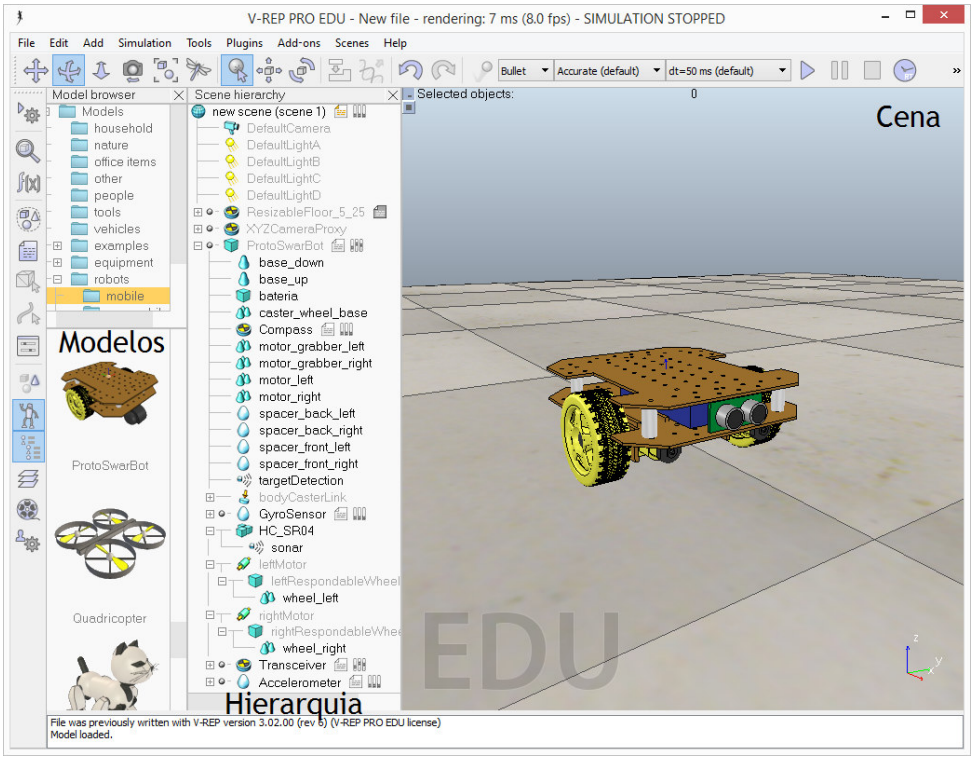
4.2 Plataforma de Experimentação Robótica Virtual

A plataforma conhecida como V-REP (*Virtual Robot Experimentation Platform*) foi desenvolvida para a criação e simulação de robôs para diversas aplicações, tanto móveis

quanto estáticos, de solo, aéreos ou aquáticos. Diversos modelos de robôs amplamente utilizados estão disponíveis na V-REP, podendo serem incluídos também novos robôs simulados e cenários para a simulação (TEIXEIRA, 2016).

Na Figura 5, é apresentada a interface da V-REP, onde se observa sua composição: a Cena, onde os elementos 3D são inseridos; a Hierarquia, onde se pode verificar todos os elementos que fazem parte da Cena; a área de Modelos, onde é possível navegar e selecionar todos os elementos disponíveis pelo simulador e adicioná-los à Cena. Além dessas três áreas principais, existem barra de ferramentas principal, que fica na parte superior logo abaixo do menu, e a barra de ferramentas secundária, que fica na lateral esquerda da V-REP (TEIXEIRA, 2016).

Figura 5 – Simulador V-REP, versão educacional.



Fonte: (TEIXEIRA, 2016).

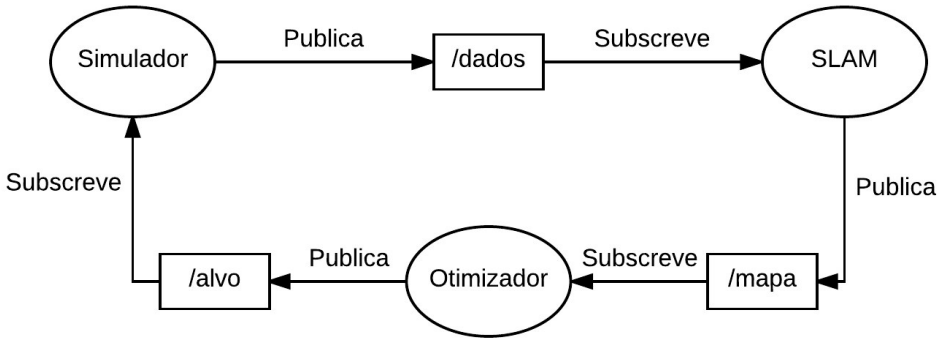
A V-REP contém três tipos de motores de simulação de física para a execução de cálculos e simulação de elementos de maneira realística. Estes motores de simulação proporcionam os mais complexos comportamentos no ambiente de simulação que vão desde a criação de robôs seguidores de linha até robôs estacionários capazes de simular o processo de soldagem e corte de materiais. Além disso, a V-REP permite a definição de características dos elementos como atrito, massa, momento de inércia etc. Essas características proporcionam a criação de ambientes de simulação realistas capazes de apresentar problemas semelhantes, aos que ocorrem no funcionamento do robô real (TEIXEIRA, 2016). A linguagem de controle da V-REP é Lua, uma linguagem de *script* simples (TEIXEIRA,

2016). A biblioteca da V-REP é vasta e proporciona o controle de todos os elementos via programação.

4.3 Arquitetura do sistema

Na Figura 6, a arquitetura da aplicação é apresentada em termos da estrutura de nós e tópicos do ROS. As elipses representam os nós ou programas executando no ROS, e os quadrados são os tópicos onde cada nó pode publicar e subscrever mensagens.

Figura 6 – Arquitetura da aplicação no ROS



A arquitetura da aplicação apresenta um nó *Simulador* que publica no tópico */dados* os dados que são captados pelos sensores do robô simulado. O nó *SLAM*, executando um algoritmo de mapeamento e localização simultâneos, recebe as mensagens publicadas em */dados* e transforma-os em *occupancy grid* correspondente ao mapa do ambiente para ser publicado no tópico */mapa*. O mapa é recebido pelo nó *Otimizador* que retorna um alvo (próxima pose do robô), segundo os critérios da exploração. O alvo é então publicado no tópico */alvo* para que o nó *Simulador* possa receber o ponto do grid para onde o robô deve ir. A exploração, por conseguinte, é devidamente simulada através do pacote V-REP.

4.3.1 Localização e Mapeamento

O pacote *GMapping* é o responsável pelo mapeamento e localização simultâneos, por meio da implementação de um filtro de partículas *Rao-Blackwellized* (RBPF) que gera um grid a partir dos dados adquiridos por laser de distância (GRISETTI; STACHNISS; BURGARD, 2007). O filtro de partículas *Rao-Blackwellized* para o SLAM consiste em estimar a pose posterior $p(x_{1:t}|\phi_{1:t}, u_{0:t})$ num mapa α através de uma trajetória dada por $x_{1:t} = x_1, ..., x_t$ (BARCELOS; VIDAL; ROSA, 2014). A estimativa da pose, é feita a partir das observações feitas pelos sensores e por comandos de controle obtidos a partir da odometria do robô, onde $\phi_{1:t} = \phi_1, ..., \phi_t$ e $u_{1:t-1} = u_1, ..., u_{t-1}$ são observações feitas e

comandos de controle, respectivamente. O modelo matemático do RBPF é dado por:

$$p(x_{1:t}, \alpha|\phi_{1:t}, u_{1:t-1}) = p(\alpha|x_{1:t}, \phi_{1:t}) \cdot p(x_{1:t}|\phi_{1:t}, u_{1:t-1}) \tag{4.1}$$

Com esta fatoração pode-se estimar a trajetória do robô, e a partir desta construir o mapa. A construção do mapa depende da pose do robô num dado momento. Dada esta transformação, é possível estimar um mapa posterior $p(\alpha|x_{1:t}, \phi_{1:t})$ utilizando-se a fatoração da Eq. 4.1. Temos como varáveis conhecidas, as poses da trajetória do robô $x_{1:t}$ e as observações $\phi_{1:t}$. Para se estimar a pose posterior $p(x_{1:t}|\phi_{1:t}, u_{1:t-1})$ do robô a através de uma trajtória, se faz necessário o uso de um filtro de partículas. Uma característica importante num filtro de partículas, é a Importância de Reamostragem Sequencial (SIR), ou sejam como devem se comportar a distribuição probabilística para uma reamostragem de partículas, num segundo passo. A SIR do RBPF, é baseada nas observações e/ou no. comando de controle, quando disponíveis (BARCELOS; VIDAL; ROSA, 2014). Nesta técnica cada instância de partícula tem um peso associado.Este peso, significa o quanto aquela partícula é probalisticamente fiel à pose real do robô. O peso de cada partícula pode ser calculado por

$$\omega_t^{(i)} = \frac{p(x_{1:t}^i|\phi_{1:t}, u_{1:t-1})}{\gamma(x_{1:t}^{(i)}|\phi_{1:t}, u_{1:t-1})} \tag{4.2}$$

Em que a função γ, que contém o histórico de observações do robô, é dada por:

$$\gamma(x_{1:t}|\phi_{1:t}, u_{1:t-1}) = \gamma(x_t|x_{1:t}, \phi_{1:t}, u_{1:t-1}) \cdot \gamma(x_{1:t-1}|\phi_{1:t-1}, u_{1:t-2}) \tag{4.3}$$

O *GMapping* é baseado em técnicas adaptativas para reduzir o número de partículas e computa uma distribuição de probabilidade, levando em consideração não somente o movimento do robô mas também os dados observados mais recentes. Isto reduz drasticamente a incerteza sobre a pose do robô no etapa de predição do filtro (GRISETTI; STACHNISS; BURGARD, 2007). O GMapping é implementado pelo pacote do ROS `gmapping` que provê um algoritmo *SLAM* baseado em scanner a laser que roda como um nó do ROS chamado de *slam_gmapping*. Usando o *slam_gmapping*, pode-se criar um occupancy grid 2-D a partir do dados do scanner a laser e da pose do robô (GMAPPING, 2015).

4.3.2 Otimizador

O nó `otimizador` implementa a geração de alvos por meio de um conjunto de técnicas centradas em heurísticas de otimização: codificador de ambiente, metaheurística de otimização, heurística de menor caminho, decodificador de caminho.

No processo de codificação o *occupancy grid* corrente do ambiente gera três listas: FRONTEIRA, OCUPADOS e VISITADOS. A lista FRONTEIRA guarda todos os pontos adjacentes a pelo menos um ponto com valor $v = -1$ no *occupancy grid*, estes pontos

com $v = -1$ são considerados desconhecidos. A lista OCUPADOS guarda todos os pontos com $v > 0$, os demais pontos, como $v = 0$, são considerados livres. A lista VISITADOS guarda todas as pontos do mapa visitados pelo robô, este são obtidos através da posição atual do robô em coordenadas (i, j) do *occupancy grid*. A construção dessas listas é necessária para o cálculo da função objetivo, que é computada com base nas distâncias dos pontos do mapa para os pontos presentes nessas três listas. Além disso, com a lista FRONTEIRA podemos verificar se ainda há pontos de fronteira no mapa, medindo o tamanho da lista |FRONTEIRA|. Assim se |FRONTEIRA| $< c$, a exploração termina, onde c é uma constante que determina a quantidade máxima de pontos de fronteira que podem ser ignorados. Pontos de fronteira são ignorados devido ao fato de que o sensor do robô não é perfeito, e por isso pode ignorar pontos isolados do mapa .

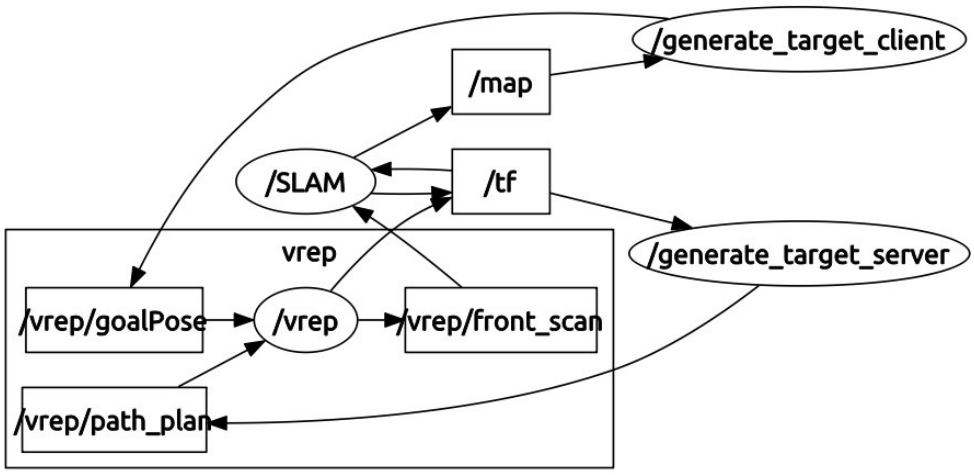
A metaheurística de otimização gera um conjunto de pontos espalhados no ambiente aleatoriamente segundo uma distribuição de probabilidade uniforme. Então estes pontos são avaliados segundo a função objetivo escolhida e ranqueados. Após isso seguindo as estratégia da metaheurística usada (algoritmo genético ou algoritmo de vagalumes) novos pontos são gerados. No Algoritmo genético os melhores pontos são combinados gerando pontos filhos, e estes substituem os pontos da população que forem piores que estes, este processo continua até que o número máximo de gerações é alcançado então o melhor ponto da população é escolhido como alvo do robô. No Algoritmo de Vagalumes, os pontos de menores valores na função objetivo são movidos em direção aos pontos melhores, e então são reavaliados. Semelhantemente ao algoritmo genético este processo é refeito até que o número máximo de iterações é alcançado, e então o melhor ponto é escolhido como alvo para o robô.

Então uma heurística computa o menor caminho entre o robô e o alvo gerado. O algoritmo usado neste trabalho foi o Algoritmo A*, este é descrito na sessão 4.3.3 a seguir. Este caminho é então publicado no tópico */path_plan*.

O otimizador foi implementado no esquema de serviços do ROS, onde há dois nós generate_target_server e generate_target_client. O nó generate_target_client ao receber um novo mapa atraves do tópico */map*, requisita um alvo para o nó generate_target_server este responde com uma mensagem que leva as coordenadas do alvo, e estas são publicadas no tópico */goalPose*. O otimizador foi implementado na liguagem C++.

Usando uma ferramenta gráfica do ROS, o `rqt_graph` pode-se visualizar o grafo de computação da aplicação com seus nós e tópicos ativos durante a simulação em execução, Figura 7.

Figura 7 – Nós e tópicos ativos durante a simulação



4.3.3 Algoritmo A*

Para conduzir o robô ao alvo é necessário gerar um caminho entre ele e o alvo sugerido pelo otimizador. Para isso, foi implementado um algoritmo A* (HART; RAPHAEL, 1968).

O A* gera uma cadeia de pontos conectados de um estado inicial $s_{inicio} \in S$ para um estado objetivo $s_{objetivo} \in S$, onde S é o conjunto de estados em um dado espaço de estados finito. Para fazer isto, ele guarda uma estimativa $g(s)$ do custo do caminho do estado inicial para cada estado s. Inicialmente, $g(s) = \infty$ para todos os estados $s \in S$ (FERGUSON; LIKHACHEV; STENTZ, 2005)

O algoritmo inicia atribuindo o custo zero ao estado inicial, então coloca este estado em fila de prioridade conhecida como *lista aberta*. Cada elemento s nesta fila é ordenado de acordo com a soma do custo corrente do caminho do estado inicial até ele, g(s), e uma estimativa heurística do custo do caminho de s até o objetivo $s_{objetivo}$, $h(s, s_{objetivo})$. O estado para o qual esta soma é mínima estará na frente da fila de prioridade. A heurística $h(s, s_{objetivo})$ tipicamente subestima o custo do caminho ótimo de s a s_{alvo} e ele é usado para focar a busca (FERGUSON; LIKHACHEV; STENTZ, 2005)

O algoritmo então tira da fila o estado s que está na frente desta e atualiza o custo de todos os estados alcançáveis a partir de s: se o custo $g(s)$, somado ao custo do movimento de s para seu estado vizinho s', $c(s, s')$, é menor que o custo corrente do estado s', então o estado de s' é alterado para este valor menor. Se o custo de s' mudar, então ele é colocado na *lista aberta*. O A* continua tirando estados da fila até que ele tire o estado alvo. Neste estágio, se a heuristica é admisível, isto é, garantidamente não superestima o custo do caminho de qualquer estado para o alvo (FERGUSON; LIKHACHEV; STENTZ, 2005). o Algoritmo 4 mostra o A*.

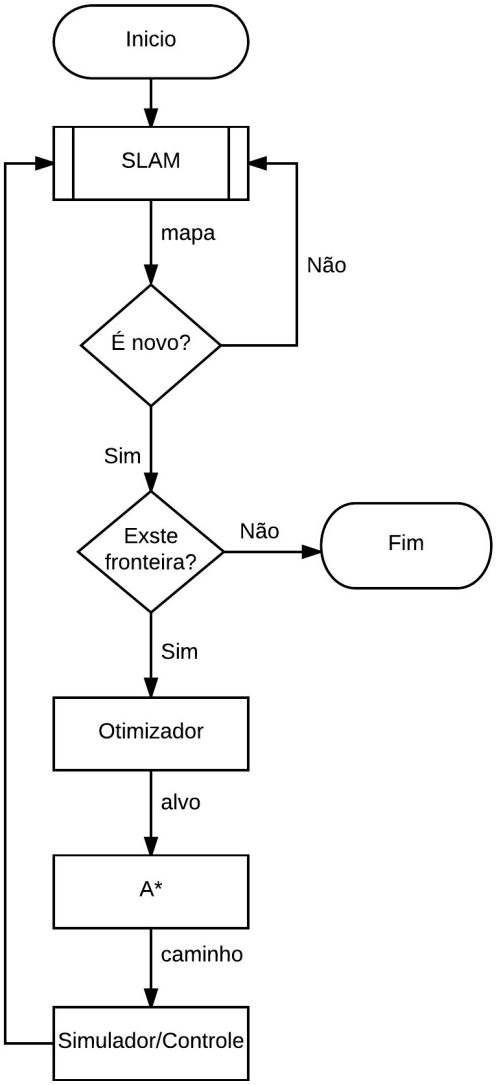
Algoritmo 4 Algoritmo A*

1: **para cada** $s \in S$ **faça**
2: $g(s) = \infty$
3: **fim para**
4: $g(s_{inicio}) = 0$
5: $OPEN = \emptyset$
6: insira s_{inicio} em $OPEN$ com valor $(g(s_{inicio}) + h(s_{inicio}, s_{objetivo}))$
7: **enquanto** $argmin_{s \in OPEN}(g(s) + h(s, s_{objetivo})) \neq s_{objetivo}$ **faça**
8: **para cada** $s' \in Sucessores(s)$ **faça**
9: **se** g(s') > g(s) + c(s,s') **então**
10: g(s') = g(s) + c(s,s')
11: insira s' em $OPEN$ com valor $(g(s') + h(s', s_{objetivo}))$
12: **fim se**
13: **fim para**
14: **fim enquanto**
15: **retorne** $\mathbf{s}_{best}$

Neste trabalho, utilizam-se as chamadas *zonas de perigo*, formada por pontos do *grid* próximos aos obstáculos, atribuindo-lhes custos de acessos $c(s, s')$ elevados, de forma a aumentar o custo associado a caminhos que intersectam essas zonas. O Algoritmo A*, portanto, considera esses custos adicionais no cálculo do menor caminho. Este algoritmo foi encapsulado no nó generate_target_server e o caminho criado é publicado no tópico */path_plan*.

Um algoritmo de controle provido pelo simulador por meio do *script* da cena é o responsável pelo movimento do robô pelo caminho gerado pelo A*. O fluxograma apresentado na Figura 8 mostra sequência completa de processos seguida pela aplicação.

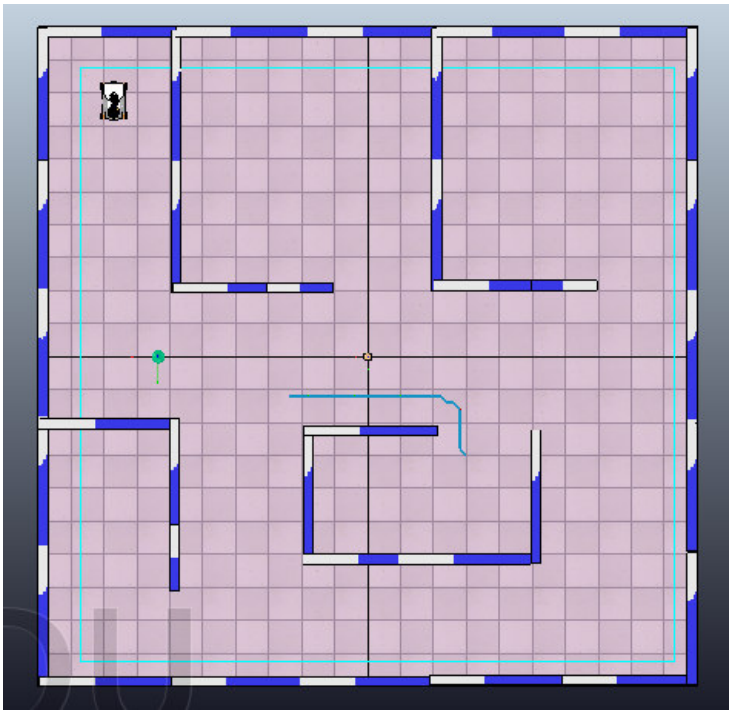
Figura 8 – Fluxograma do sistema otimizador



5 Experimentos Computacionais

Neste Capítulo, são apresentados e discutidos os experimentos realizados com intuito de validar a modelagem do ambiente como um problema de otimização, bem como as metaheurísticas implementadas para resolvê-lo. Para tanto, foram construídos em ambiente de simulação *V-REP*, dois cenários representando espaços fechados, semelhantes ao mostrado na Figura 9, onde um robô youBot KUKA é colocado para explorar.

Figura 9 – Ambiente de Simulação



Os tipos da função objetivo (equações de 3.4 a 3.7), bem como seus ganhos (ou relevância) foram avaliados em diferentes aspectos de interesse ao processo exploratório. Destaca-se que a configuração de parâmetros, aqui apresentada, foi obtida baseada na literatura com pequenas flutuações em decorrência da experiência adquirida nas diversas simulações realizadas. Partindo da configuração inicial mostrada na tabela 1, semelhante à usada em (JULIÁ et al., 2011), e após vários experimentos usando a metodologia de *tentantiva e erro* chegou-se às configurações de parâmetros que são mostradas nas próximas seções.

Tabela 1 – Configuração inicial de parâmetros

pesos		amplitude	
α	3	σ_f	4,5
β	100	σ_o	0,15
cdc γ	2	σ_v	3,45

5.1 Funções objetivos e seus parâmetros

Para avaliar as formulações da função objetivo propostas, foi observada inicialmente a superfície gerada de cada função em dado momento da exploração. Verificou-se que a superficie gerada reflete os comportamentos desejados, dado o estado do mapa (*occupancy grid*) atual.

A seguir, têm-se exemplos de surpefícies geradas a partir de cada formulação da função, com uma determinada configuração de parâmetros, para um determinado estado do mapa. Nestas figuras, foi feito um corte a altura 0 nas posições que são desconhecidas, ou seja, foi atribuído o valor 0 para células desconhecidas do mapa, facilitando a visualização da função objetivo, em face dos profundos *vales* causados por regiões ainda desconhecidas para o robô, conforme a Seção 3.3.

5.1.1 *Forma básica*

Definida pela Equação 3.4, a formulação básica da função objetivo foi configurada com os parâmetros:

Tabela 2 – Parâmetros da forma básica da função objetivo

pesos		amplitude	
α	50	σ_f	1,5
β	100	σ_o	4
γ	3	σ_v	3,45

A Figura 10 mostra o estado corrente do mapa (lado esquerdo da figura), onde a região cinza é o espaço conhecido e livre, e a pose do robô em um dado momento da exploração (lado direito da figura). A Figura 11 mostra a superfície da função gerada para esta configuração. Cada unidade nos eixos x e y do gráfico é equivalente a 5 cm.

Figura 10 – Posição do robô e estado do mapa - forma básica da função

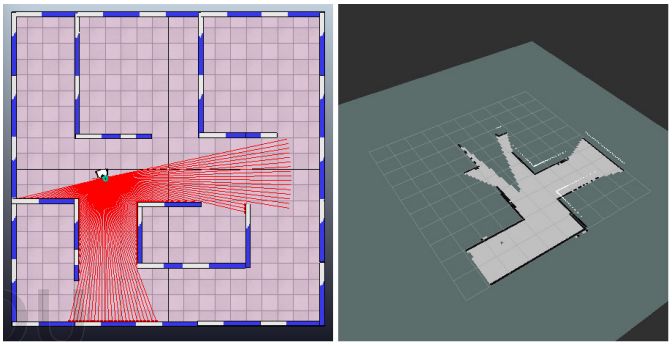
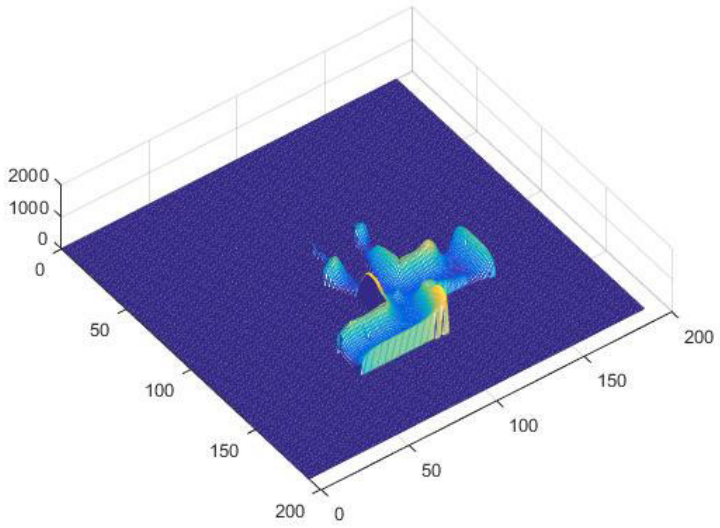


Figura 11 – Visualização 3D da forma básica da função



Observa-se que esta função apresenta um comportamento interessante, atribuindo valores altos à ns posições onde há obstáculos próximos e valores baixo àquelas regiões onde se concentram os pontos de fronteira. No entando, esta função tem valores míninos ou ótimos em regiões com maior aglomerado de pontos de fronteira, e estas regiões podem estar bem mais distantes do robô que outras regiões de fronteira. Isto pode acarretar em uma ordem indesejada de seleção de objetivos, priorizando-se regiões distantes em detrimento de regiões de fronteira próximas, o que leva a um comportamento indesejado.

5.1.2 *Perda condicional*

Definida pela Equação 3.5, esta função foi configurada com os parâmetros:

Tabela 3 – Parâmetros da função com *perda condicional*

pesos		amplitude	
α_1	$3 \cdot \alpha_2$	σ_f	1,5
β	10	σ_o	4
γ	30	σ_v	3,45
r	30	-	-
α_2	500	-	-

A Figura 12 mostra o estado corrente do mapa e a pose do robô em um dado momento da exploração. A Figura 13 mostra a superfície da função gerada para esta configuração.

Figura 12 – Posição do robô e estado do mapa - função com *perda condicional*

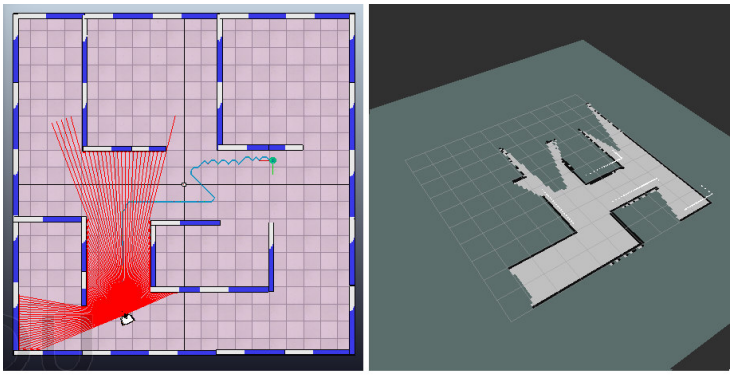
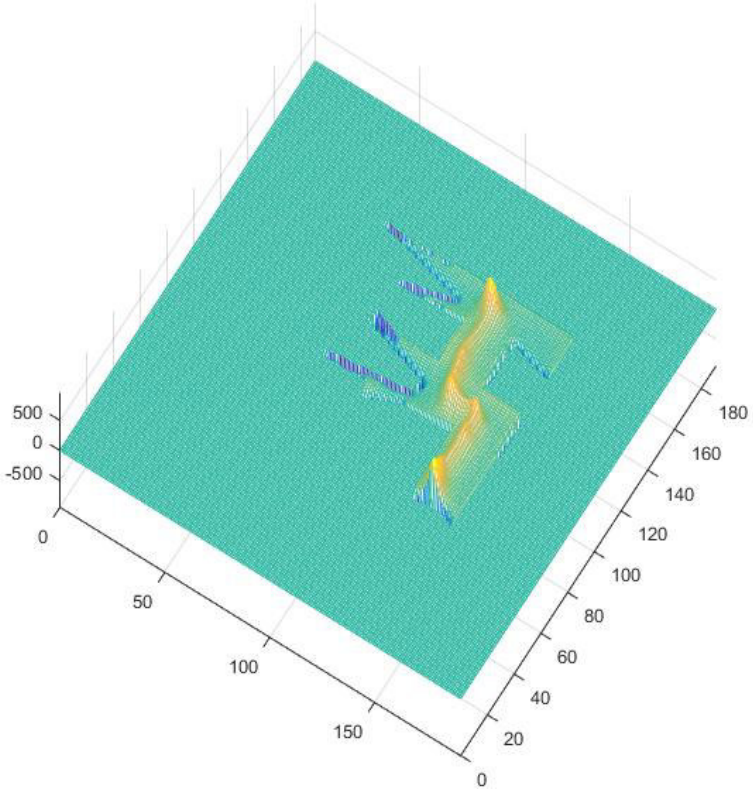


Figura 13 – Visualização 3D da função com *perda condicional*



Nota-se na superficie gerada, que os pontos visitados pelo robô, recebem os valores mais altos, o que conduz o robô para longe das regiões já exploradas. Os valores mínimos da função estão nas regiões de fronteira mais distantes dos pontos das regiões visitadas. No entanto, em relação aos pontos visitados, os pontos ocupados têm um valor baixo. Isto pode levar o algoritmo a considerar pontos ocupados como pontos bons e possivelmente, se não conseguir explorar bem o espaço de busca, gerar alvos em pontos ocupados.

5.1.3 *Perda linear*

Definida pela Equação 3.6, esta função foi configurada com os parâmetros:

Tabela 4 – Parâmetros da função com *perda linear*

pesos		amplitude	
α	5000	σ_f	1,5
β	1000	σ_o	4
γ	2000	σ_v	3,45
ρ	10	-	-

A Figura 14 mostra o estado corrente do mapa e a pose do robô em um dado momento da exploração. A Figura 15 mostra a superfície da função gerada para esta configuração.

Figura 14 – Posição do robô e estado do mapa - função com *perda linear*

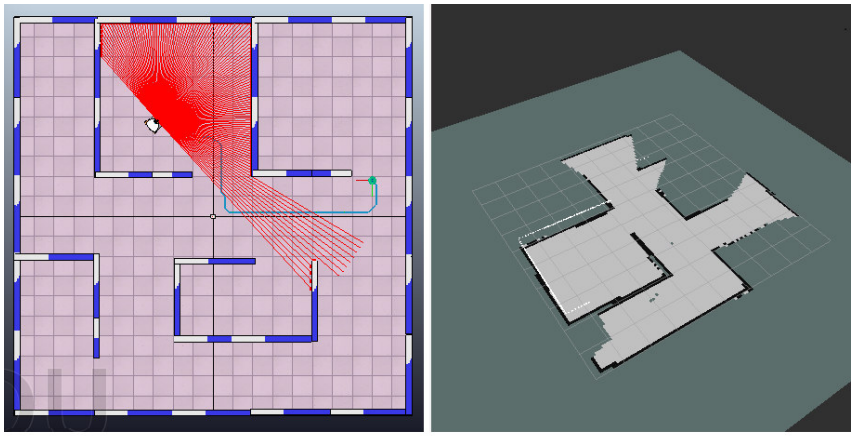
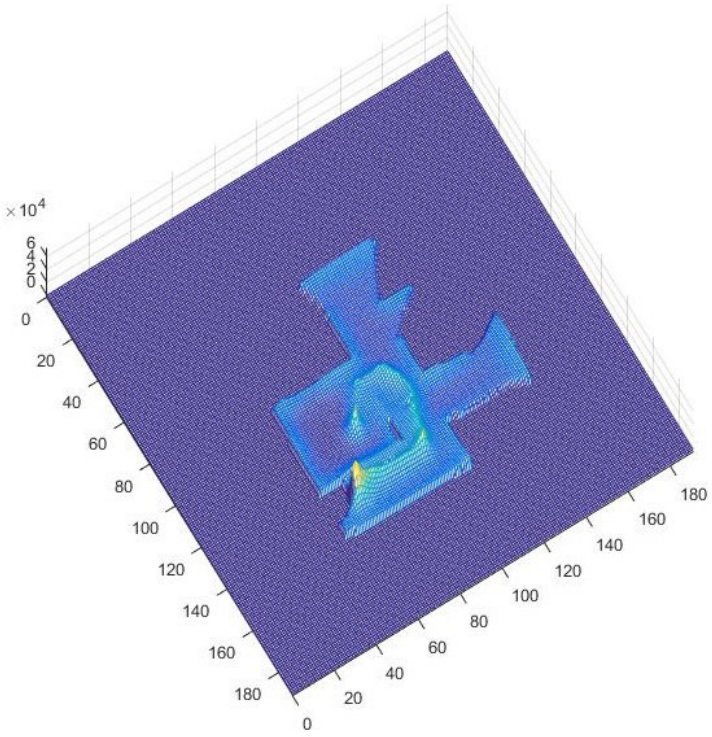


Figura 15 – Visualização 3D da função com *perda linear*



Observa-se um comportamento esperado nesta função, uma vez que tem-se valores altos atribuídos aos pontos ocupados e aos pontos visitados, e valores baixos associados a pontos de fronteira mais próximos ao robô. Levando o algoritmo a selecionar alvos distantes dos obstaculos e de regiões já explorados, e o mais próximo possível tanto das regiões de fronteira quanto do robô.

5.1.4 *Perda Proporcional*

Definida pela Equação 3.7, esta função foi configurada com os parâmetros:

Tabela 5 – Parâmetros da função com *perda proporcional*

pesos		amplitude	
α	500	σ_f	1,5
β	150	σ_o	4
γ	3	σ_v	3,45
ρ	0.1	-	-

A Figura 16 mostra o estado corrente do mapa e a pose do robô em um dado momento da exploração. A Figura 17 mostra a superfície da função gerada para esta configuração.

Figura 16 – Posição do robô e estado do mapa - função com *perda proporcional*

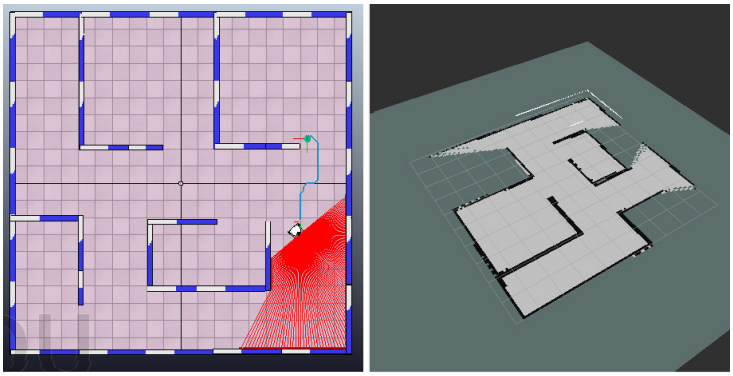
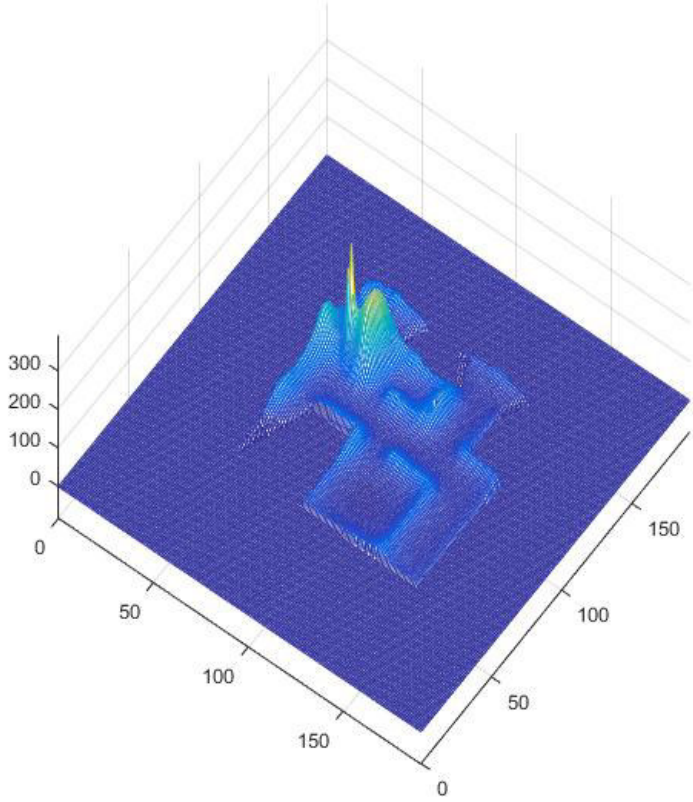


Figura 17 – Visualização 3D da função com *perda proporcional*



Outra função a apresentar um comportamento interessante foi a função com *perda proporcional*. Pontos ocupados recebem valores mais altos e pontos de fronteira recebem os valores mais baixos. Além disso, nesta função os pontos próximos recebem um valores mais altos que os distantes. Isto induz o robô a sair de onde ele está.

5.2 Metaheurísticas e seus parâmetros

Duas metaheurísticas de otimização diferentes foram implementadas e avaliadas durante os experimentos realizados: Algoritmo de Vagalumes (AV) e Algoritmo Genético

(AG). Os parâmetros de desempenho destes algoritmos foram ajustados conforme as tabelas 6 e 7, seguindo orientação encontrada na literatura (OLIVEIRA; LORENA, 2004), (YANG, 2009).

Tabela 6 – Parâmetros do Algoritmo Genético

paramêtro	valor
$P_{crossover}$	1
$P_{mutacao}$	0,1
P_{size}	1024

Tabela 7 – Parâmetros do Algoritmo de Vagalumes

paramêtro	valor
β_0	1
α_t	0,3
γ_{av}	0,4
P_{size}	20

Foram realizadas simulações para verificar a distância entre o alvo ótimo e os alvos gerados pelas metaheurísticas AG e AV. O alvo ótimo é obtido exaustivamente, avaliando-se o valor da função objetivo em todas as células do mapa. Comparando-se os dois algoritmos, observou-se que o AV apresenta alvos mais próximos da célula avaliada como ótima. As figuras de 18 a 25 exemplificam esta vantagem, tomando a função objetivo modelada com *perda proporcional*. Nessas figuras, o ponto de cor verde é alvo ótimo, o azul é o alvo gerado pelo AV e o amarelo é o gerado pelo AG. Vale ressaltar que os pontos gerados pelo AG não são pontos de baixa qualidade, sendo considerados ótimos locais, levando em conta a natureza multi-modal da função objetivo.

5.2.1 Caminhos gerados

As trajetorias geradas pelas metaheurística no tocante à função objetivo *perda proporcional* são analisadas nesta Seção. Esta função objetivo modelou melhor os ambientes a serem explorados, permitindo melhores resultados alcançados tanto pelo AV quanto AG.

Nas Figuras de 26 a 49, o ponto vermelho é a origem da trajetória e o ponto a azul é ponto final da trajetória. Pode-se perceber que as trajetórias geradas cobrem todas a regiões do ambiente e se regiões já exploradas são revisitadas. Pode ocorrer variações nas trajetórias do robô, indicando melhor qualidade aquela que gera o mapeamento completo do ambiente, com menor comprimento de percurso. Observa-se que a combinação da função com *perda proporcional* com o AV gerou melhores trajetórias, que conduziram o robô a fazer o mapeamento completo do ambiente, com poucas revisitas a regiões.

Figura 18 – Algoritmos e seus parâmetros - Teste 1

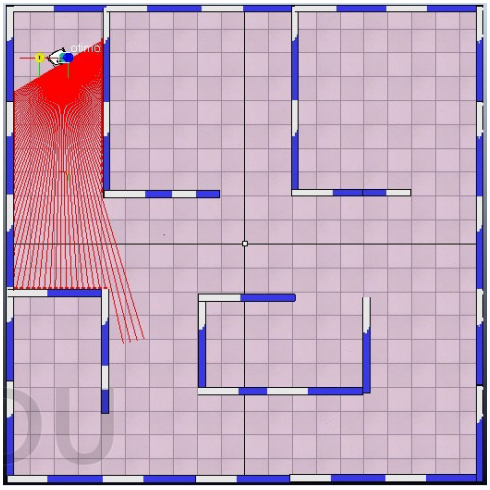


Figura 19 – Comparação dos algoritmos - Teste 2

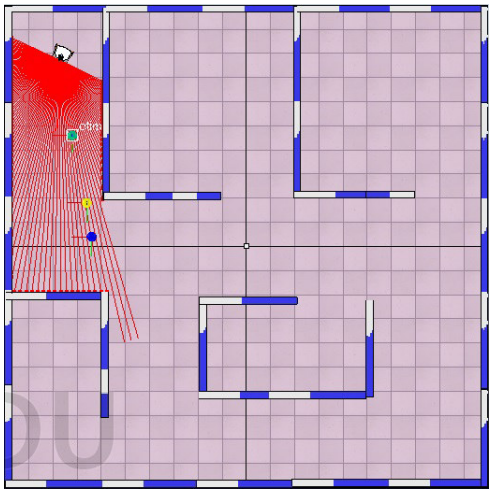


Figura 20 – Comparação dos algoritmos - Teste 3

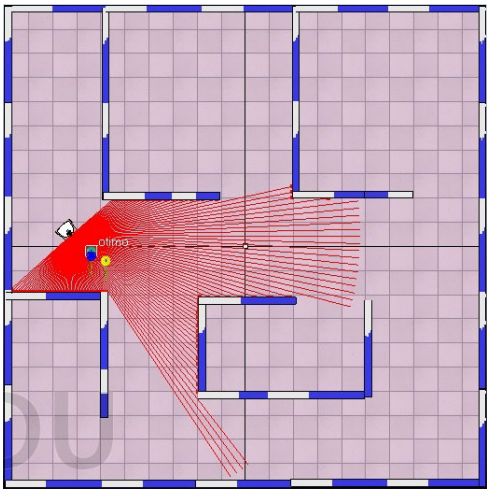


Figura 21 – Comparação dos algoritmos - Teste 4

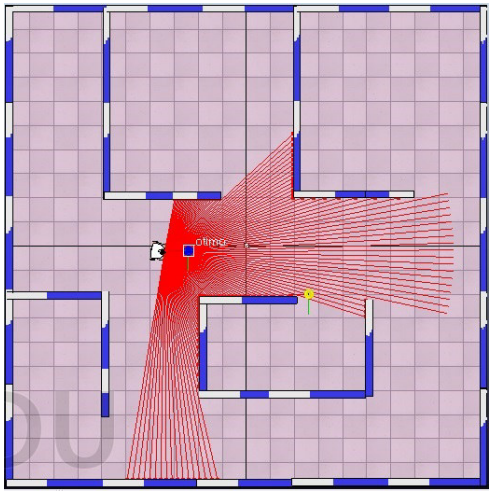


Figura 22 – Comparação dos algoritmos - Teste 5

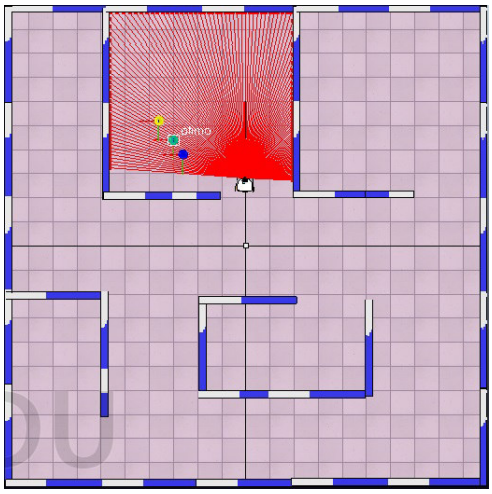


Figura 23 – Comparação dos algoritmos - Teste 6

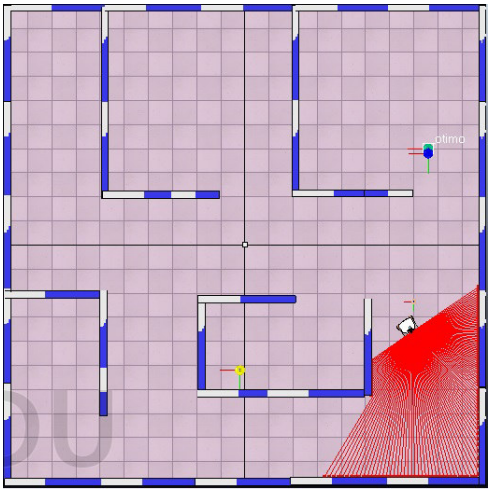


Figura 24 – Comparação dos algoritmos - Teste 7

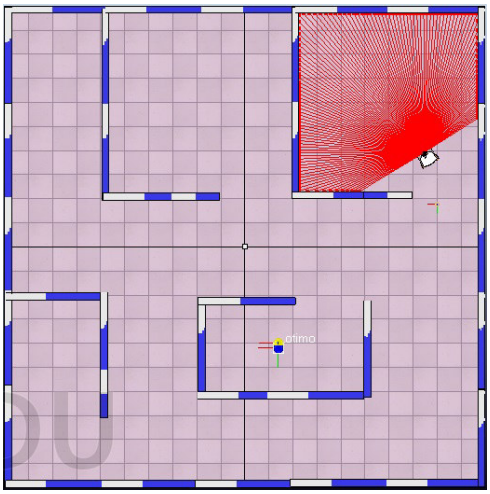
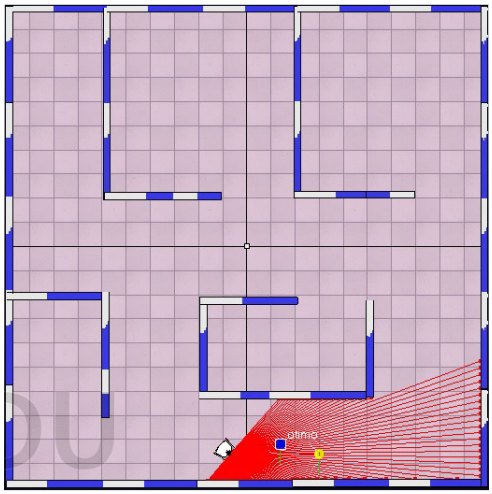


Figura 25 – Comparação dos algoritmos - Teste 8



5.2.1.1 Algoritmo Genético - ambiente 1

As figuras de 26 a 30 mostram as trajetórias geradas durante cinco experimentos realizados, nos quais o robô explora o ambiente 1 guiado pelo AG. No primeiro experimento (Figura 26), a trajetória não cobriu todo o ambiente, ou seja o robô não completou o mapeamento, ficando estagnado em um determinado ponto. No experimento 2 (Figura 27), o mapeamento foi completo, no entanto o robô revisitou regiões gerando uma trajetória maior que as outras. Nos experimentos 3 a 5 (figuras 28 a 30), o mapeamento foi completo e a trajetória revisita regiões poucas vezes. A Figura 31 exibe uma sobreposição de todas as trajetórias, mostrando a variabilidade destas.

Observam-se que as trajetórias, no geral, apresentam formas semelhantes.

5.2.1.2 Algoritmo Genético - ambiente 2

As figuras de 32 a 36 mostram as trajetórias geradas durante cinco experimentos realizados, nos quais o robô explora o ambiente 2 guiado pelo AG. No primeiro experimento (Figura 32), a trajetória cobriu todo o ambiente, no entanto tem-se a volta do robô para uma região já visitada. No experimento 2 (Figura 33), o mapeamento foi completo, com a trajétoria apresentando uma forma boa. Nos experimento 3 (Figura 34), o mapeamento foi incompleto e o robô revisitou várias vezes um mesmo ponto, e ao final ficou estagnado, ao lado de um obstáculo. No experimento 4 (Figura 35), o mapeamento foi incompleto e o robô revisitou regiões já mapeadas. No experimento 5 (Figura 36), o mapeamento foi completo e a trajetória apresenta um comportamento melhor. A Figura 37 exibe uma sobreposição de todas as trajetórias, mostrando a variabilidade destas.

Observam-se que nestes experimentos as trajetórias apresentaram muita variabiliadade, haja vista os pontos finais das trajetórias que estão muito distantes uns dos

Figura 26 – Algoritmo Genético - Teste 1 - Ambiente 1

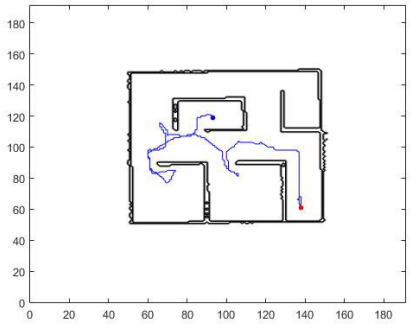


Figura 27 – Algoritmo Genético - Teste 2 - Ambiente 1

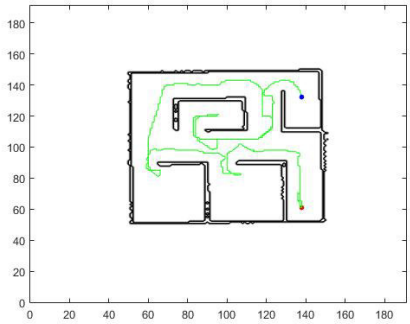


Figura 28 – Algoritmo Genético - Teste 3 - Ambiente 1

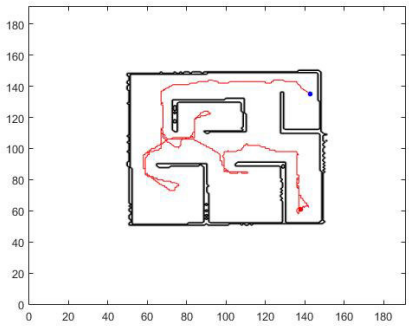


Figura 29 – Algoritmo Genético - Teste 4 - Ambiente 1

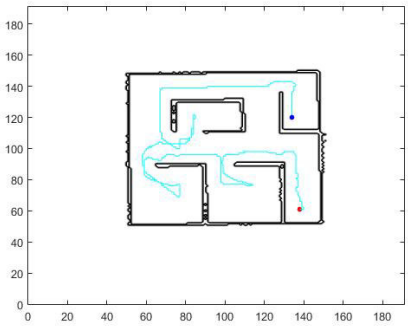


Figura 30 – Algoritmo Genético - Teste 5 - Ambiente 1

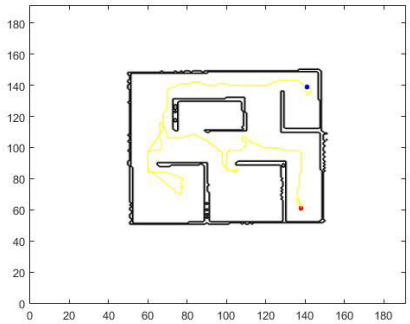
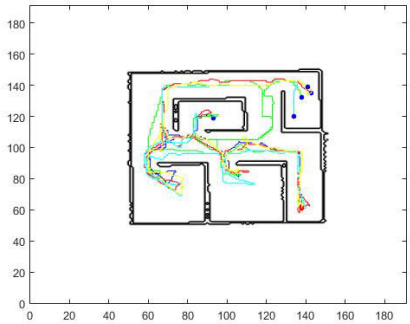


Figura 31 – Algoritmo Genético - Variabilidade das trajetórias - Ambiente 1



outros.

5.2.1.3 Algoritmo dos Vagalumes - ambiente 1

As figuras de 38 a 42 mostram as trajetórias geradas durante cinco experimentos realizados, nos quais o robô explora o ambiente 1 guiado pela metaheurística AV. Nos experimentos 1, 3 e 5 (figuras 38, 40 e 42), a trajetória cobriu todo o ambiente, de forma eficiente, i.e., sem revisitas. Nos experimento 2 e 3 (Figura 27), o mapeamento foi completo,

Figura 32 – Algoritmo Genético - Teste 1 - Ambiente 2

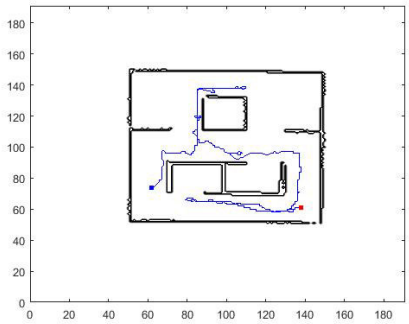


Figura 33 – Algoritmo Genético - Teste 2 - Ambiente 2

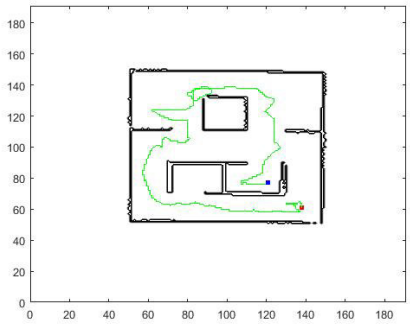


Figura 34 – Algoritmo Genético - Teste 3 - Ambiente 2

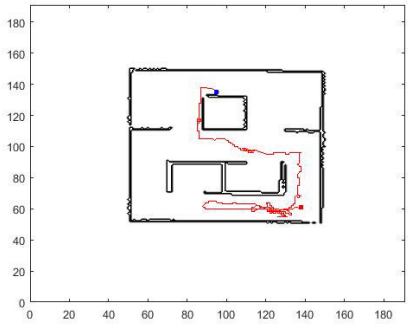


Figura 35 – Algoritmo Genético - Teste 4 - Ambiente 2

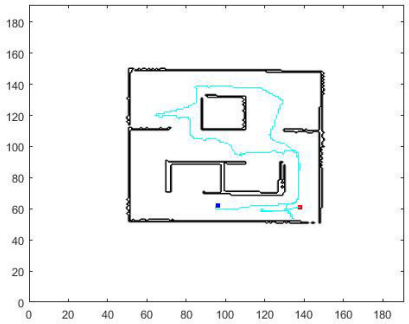


Figura 36 – Algoritmo Genético - Teste 5 - Ambiente 2

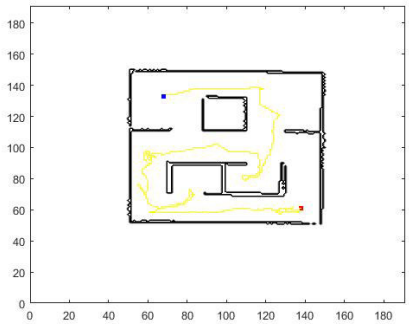
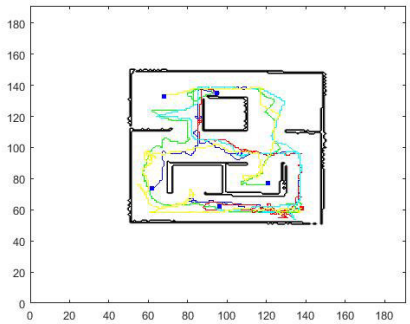


Figura 37 – Algoritmo Genético - Variabilidade das trajetórias - Ambiente 2



no entanto o robô revisitou regiões gerando um trajetória maior que as outras. A Figura 43 exibe uma sobreposição de todas as trajetórias, mostrando a variabilidade destas.

Observa-se que, no geral, as trajetórias apresentam formas semelhantes, e cobrem todas as regiões do ambiente.

Figura 38 – Algoritmos dos Vagalumes - Teste 1 - Ambiente 1

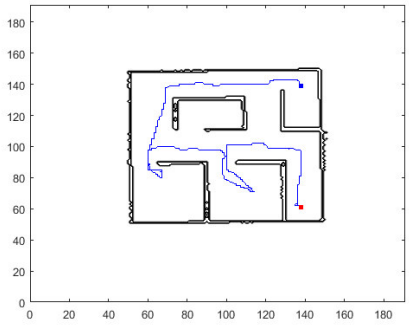


Figura 39 – Algoritmos dos Vagalumes - Teste 2 - Ambiente 1

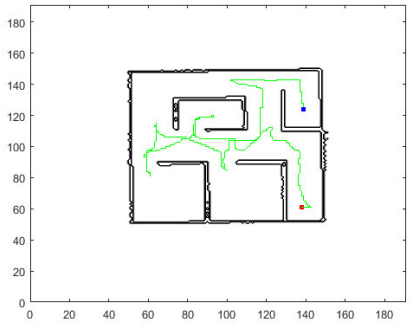


Figura 40 – Algoritmos dos Vagalumes - Teste 3 - Ambiente 1

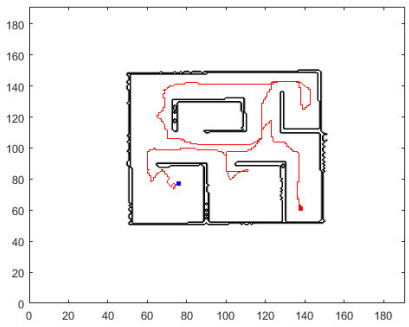


Figura 41 – Algoritmos dos Vagalumes - Teste 4 - Ambiente 1

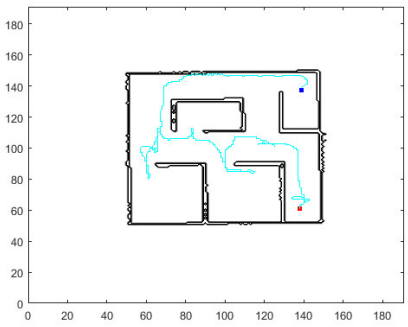


Figura 42 – Algoritmos dos Vagalumes - Teste 5 - Ambiente 1

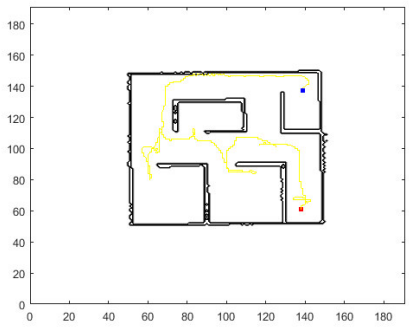
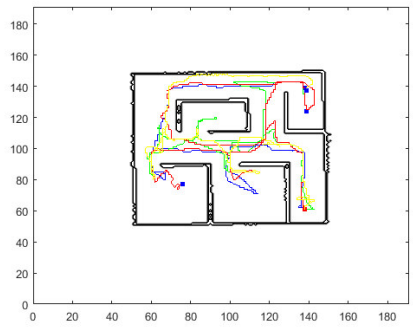


Figura 43 – Algoritmos dos Vagalumes - Variabilidade das trajetórias - Ambiente 1



5.2.1.4 Algoritmo dos Vagalumes - ambiente 2

As figuras de 44 a 48 mostram as trajetórias geradas durante cinco experimentos realizados, nos quais o robô explora o ambiente 2 com o AV gerando os objetivos. Em todos estes experimentos, as trajetórias cobriram todas as recintos do ambiente e, portanto, levaram a um mapeamento, ou quase completo deixando pequenas regiões não mapeadas.

Nos experimentos de 1 a 4 (figuras 44 a 48), as trajetórias apresetaram formas semelhantes, somente o no experimento 5 (Figura 48) tem-se uma trajetória bem diferente, inclusive com ponto final em uma região diferente das demais trajetórias.

Figura 44 – Algoritmos dos Vagalumes - Teste 1 - Ambiente 2

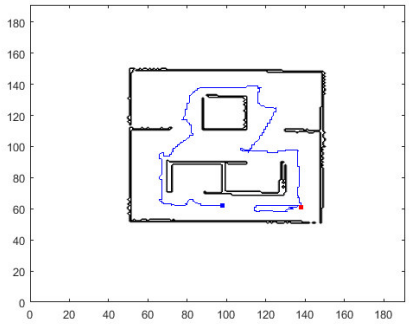


Figura 45 – Algoritmos dos Vagalumes - Teste 2 - Ambiente 2

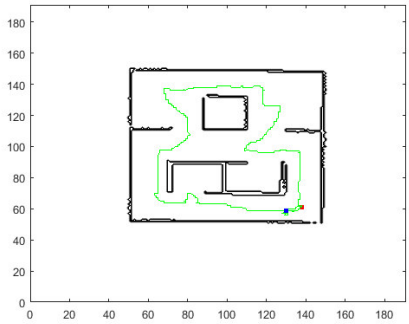


Figura 46 – Algoritmos dos Vagalumes - Teste 3 - Ambiente 2

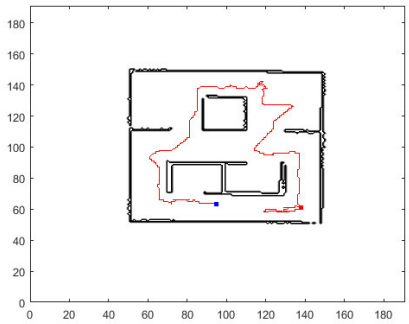


Figura 47 – Algoritmos dos Vagalumes - Teste 4 - Ambiente 2

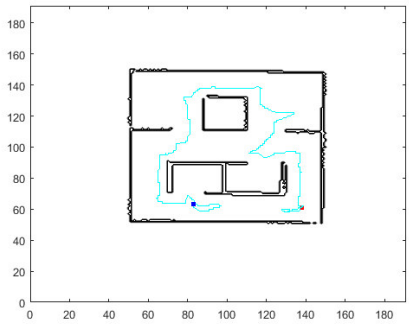


Figura 48 – Algoritmos dos Vagalumes - Teste 5 - Ambiente 2

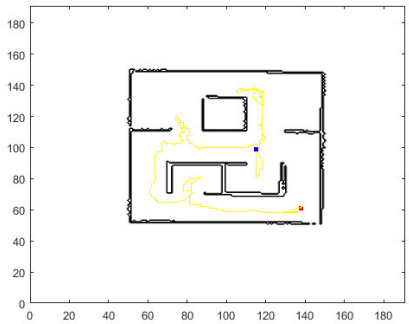
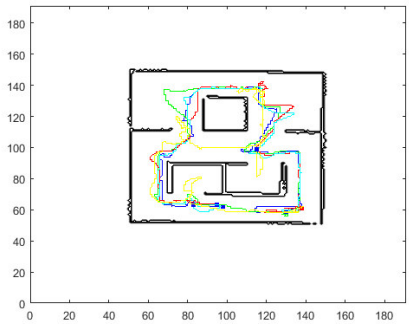


Figura 49 – Algoritmos dos Vagalumes - Variabilidade das trajetórias - Ambiente 2



A Figura 49 exibe uma sobreposição de todas as trajetórias, mostrando a variabilidade destas. Destaca-se que, como já foi mencionado, as trajetórias apresentam formas

semelhantes, e cobrem todas as regiões do ambiente.

5.3 Discussão dos resultados

A Tabela 8 apresenta a quantidade de células visitadas pelo robô ao final da exploração para cada algoritmo e para cada experimento realizado. Nota-se que, no geral, o AV conduziu o robô a visitar menos pontos para completar a exploração, em comparação com o AG.

Os resultados obtidos indicam que, embora os dois algoritmos tenham conduzido ao mapeamento completo dos ambientes na maioria das vezes, o AV obteve melhores resultados de acordo com os critérios de desempenho adotados neste trabalho, i.e., gerando trajetórias menores que cobrem todo o ambiente com poucas revisitas a recintos.

Tabela 8 – Quantidade de pontos visitados pelo robô.

Experimento	ambiente 1		ambiente 2	
	AV	GA	AV	GA
1	1433	928	1257	1960
2	1401	1730	1324	1709
3	1577	2094	1192	2112
4	1765	1675	1713	1608
5	1850	2207	1646	2227
média(dp)	**1605,2(±198)**	**1726,8(±502)**	**1426,4(±237)**	**1923,2(±262)**

Em uma análise estatística, usando-se o Teste t de duas amostras com variâncias diferentes (MONTGOMERY, 2001), pode-se evidenciar melhor esses achados. Para o ambiente 1, foi verificado que não há uma diferença estatisticamente significativa (*p-value* é igual a $0,635$) entre as médias das duas amostras. Isto é exemplicado no boxplot da Figura 50, onde o grupo A refere-se aos resultados do AV, e o grupo B aos resultados do AG. Para o ambiente 2, a diferença entre as médias é estatisticamente significativa (*p-value* igual a $0,0136$), portanto a hipótese nula (igualdade de desempenho) pode ser rejeitada com 95% de confiabilidade e com consequente conclusão de desempenho superior do AV, quando comparado ao AG. Na Figura 51, é apresentado o gráfico que ilustra bem essa diferença.

As figuras de 52 a 55 mostram exemplos de mapas (*occupancy grid*) gerados ao final da exploração. Os algoritmos de otimização tiveram mais dificuldade no segundo ambiente nem sempre mapeando-o completamente. As figuras 54 e 55 mostram pequenas regiões ainda não mapeadas. Isto pode ser explicado pela priorização da visita a regiões com maior ganho de informação e consequente dificuldade de retorno a regiões inicialmente não priorizadas devido a caminhos passando por regiões já visitadas. Apesar disso, pode-se

Figura 50 – Boxplot das amostras referentes aos pontos visitados no ambiente 1.

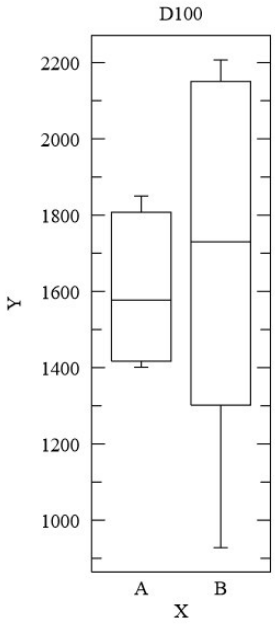


Figura 51 – Boxplot das amostras referentes aos pontos visitados no ambiente 2.

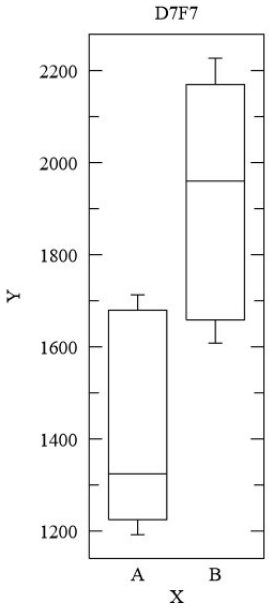


Figura 52 – Mapa do ambiente 1 gerado pela exploração realizada pelo Algoritmo de Vagalumes

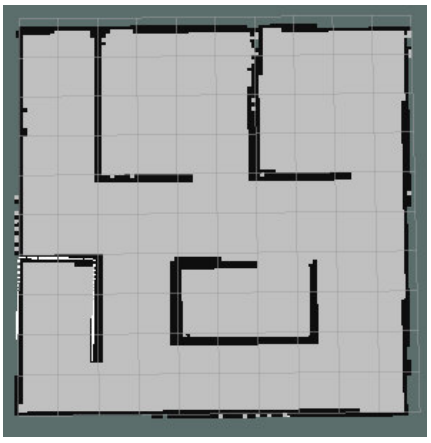
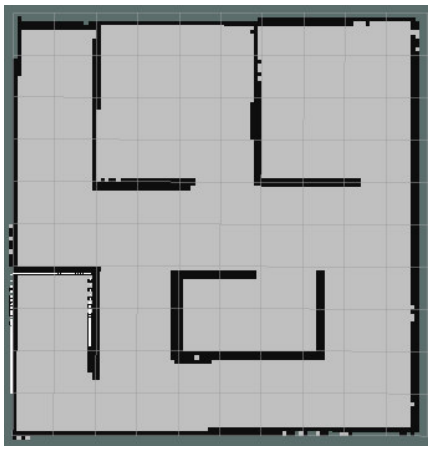


Figura 53 – Mapa do ambiente 1 gerado pela exploração realizada pelo Algoritmo Genético



dizer que, na maioria das vezes, as metaheurísticas conduziram o robô a explorar todas as regiões do ambiente, completando assim a tarefa de exploração autônoma.

Figura 54 – Mapa do ambiente 2 gerado pela exploração realizada pelo Algoritmo de Vagalumes

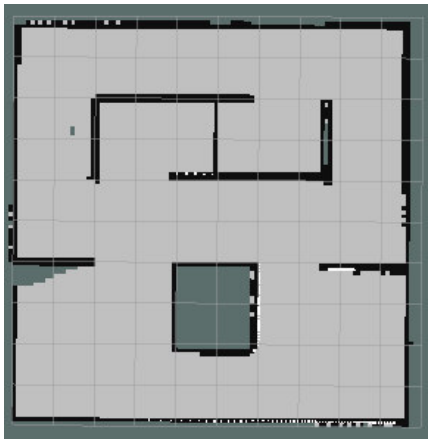
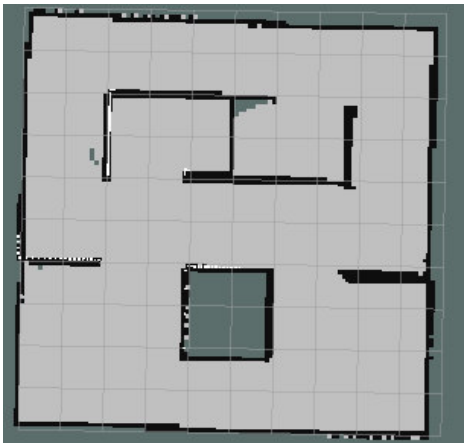


Figura 55 – Mapa do ambiente 2 gerado pela exploração realizada pelo Algoritmo Genético



6 Conclusão

Em robótica, navegação autônoma de robôs pode ser divida em etapas que vão desde a percepção do ambiente, localização e mapeamento, até planejamento e controle de trajetória. As funções da navegação podem ser expressas pela capacidade do artefato robótico de realizar a auto-localização, usando informações sensoriais; capacidade de raciocínio com base no conhecimento do ambiente; capacidade de planejamento de trajetória; e capacidade de memorização e manutenção do mapa.

As estratégias de exploração autônoma presentes na literatura, em geral, decidem novas poses com base em regras rígidas ou comportamentos locais, discretizando sobremaneira o ambiente em exploração. Portanto, podem ser vistas como heurísticas gulosas que conduzem a soluções localmente ótimas.

Este trabalho propõe o uso de Metaheurísticas para solucionar o problema da exploração autônoma. Dessa forma, o problema da exploração autônoma é modelado como uma sequencia de problemas de otimização que consideram dados obtidos do ambiente incrementalmente mapeado. A função objetivo é uma composição de funções componentes que refletem os comportamentos desejados para robô durante a exploração. A metaheurística de otimização tem o papel de buscar o ótimo global da função objetivo, i.e., a posição do mapa conhecido onde o ganho de informação é maximizado ao mesmo tempo em que são minimizados o risco de colisão e a possibilidade de retorno a áreas já exploradas.

Para validar a proposta foi contruído um sistema de exploração autônoma simulado com apoio de Sistema Operacional Robótico e Plataforma de Experimentação Robótica Virtual. Foram realizadas simulações empregando um modelo de robô youBot KUKA, equipado com escaner a laser, em dois cenários *indoor*. Foram avaliadas modificações em componentes da função objetivo de forma a melhor representar o ambiente e induzir o robô a cumprir uma estratégia de exploração eficiente. Igualmente, foram avaliadas duas metaheurísticas de otimização: Algoritmo Genético e Algoritmo de Vagalumes.

Observou-se que a função que melhor modelou o ambiente foi a função *tipo 4*, que agrega o princípio de localidade com penalidade proporcional à distância entre o robô e regiões de fronteira. O conceito de localidade permite organizar melhor a exploração, reduzindo-se o ganho procedente de grandes agrupamentos de pontos de fronteira distantes do robô que induzem uma exploração com necessidade de revisitações. Para a função com *perda proporcional* o *Algortimo de Vagalumes* mostrou-se o mais eficiente, levando o robô a fazer o mapeamento completo dos ambientes testados com percursos reduzidos em relação aos percursos guiados por *Algoritmo Genético*.

Como contribuições deste trabalho destacam-se:

- A modelagem do problema da exploração autônoma de robôs móveis como um problema de otimização;
- A geração alvos de exploração por meio de metaheurísticas, buscando alvos globalmente ótimos;
- Teste e validação de dois algoritmos metaheurísticos de otimização aplicados ao problema;
- Criação de um sistema de exploração autônoma sobre o Sistema Operacional Robótico.

Como trabahos futuros, pretende-se realizar uma avaliação mais detalhada dos resultados computacionais. Considerando a natureza estocástica das metaheurísticas, testes estatísticos são requeridos variando-se os tipos de ambiente, os parâmetros das funções objetivo, os locais de saída do robô, e os parâmetros de desempenho de cada algoritmo de otimização.

Um outro aspecto não investigado neste trabalho diz respeito à flexibilidade da abordagem para incorporação de novas estratégias ou incertezas de ambiente. Variações nos parâmetros de função objetivo, por exemplo, podem ser suficientes para ajustar a exploração autônoma a novos desafios, ou mesmo introduzir mais de um robô no cenário.

Da mesma forma, a introdução de ruído ou incertezas no ambiente pode requerer ajustes na função objetivo facilmente implementados. Não obstante, a abordagem pode ser suficientemente robusta para lidar com ruídos apenas como uma interferência momentânea no ambiente, sem efeitos estatisticamente significativos para os algoritmos de otimização. Por essa hipótese, células ruidosas podem ser evitadas pelo robô, modeladas como regiões não promissoras, até que o mapeamento permita maior segurança com relação aos seus valores de função objetivo.

Evidentemente, adicionar termos a função objetivo pode aumentar muito seu custo computacional, no entanto, como foi mencionado, a ideia do uso de metaheurísticas é que se tenha uma convergência rápida com poucas chamadas à função objetivo.

Numa segunda etapa, pretende-se ainda substituir o robô simulado Kuka/Youbot por um robô real. Essa tarefa deve ser facilitada pelo *framework* provido pelo Sistema Operacional Robótico.

Referências

AMIGONI, F.; GALLO, A. A multi-objective exploration strategy for mobile robots. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18-22, 2005, Barcelona, Spain*. [s.n.], 2005. p. 3850–3855. Disponível em: <http://dx.doi.org/10.1109/ROBOT.2005.1570708>. Citado na página 18.

ARKIN, R. C. *A Behavior-based Robotics*. 1st. ed. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262011654. Citado na página 26.

BARCELOS, A. d. O. P.; VIDAL, F. S.; ROSA, P. F. F. Construção de uma grade de ocupação utilizando câmera estereoscópica ativa para navegação autônoma. In: *Anais do XX Congresso Brasileiro de Automática*. Belo Horizonte, MG: [s.n.], 2014. Citado 3 vezes nas páginas 22, 44 e 45.

BIANCHI, L.; DORIGO, M.; GAMBARDELLA, L. M.; GUTJAHR, W. J. A survey on metaheuristics for stochastic combinatorial optimization. Kluwer Academic Publishers, Hingham, MA, USA, v. 8, n. 2, p. 239–287, jun. 2009. ISSN 1567-7818. Disponível em: <http://dx.doi.org/10.1007/s11047-008-9098-4>. Citado na página 29.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, set. 2003. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/937503.937505>. Citado 3 vezes nas páginas 28, 29 e 30.

BROWNLEE, J. *Clever Algorithms: Nature-Inspired Programming Recipes*. 1st. ed. [S.l.]: Lulu.com, 2011. ISBN 1446785068, 9781446785065. Citado na página 31.

BUONOCORE, L.; ALMEIDA, A. D.; NASCIMENTO, C. L. Autonomous feature-based exploration using a low-cost mobile robot. In: EDITOR, T. (Ed.). *Systems Conference (SysCon)*. [S.l.: s.n.], 2014. (5, v. 4), p. 452. Citado 2 vezes nas páginas 23 e 24.

COELLO, C. A. *An updated survey of GA-based multiobjective optimization techniques*. 2000. 109–143 p. Citado na página 34.

DISSANAYAKE, M. W. M. G.; NEWMAN, P.; CLARK, S.; DURRANT-WHYTE, H. F.; CSORBA, M. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, v. 17, p. 229–241, 2001. Citado na página 15.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, v. 2, p. 2006, 2006. Citado na página 15.

FEO, T.; RESENDE, M. *A probabilistic heuristic for a computationally difficult set covering problem*. [S.l.]: Elsevier, 1989. 67–71 p. Citado na página 30.

FERGUSON, D.; LIKHACHEV, M.; STENTZ, A. A guide to heuristicbased path planning. In: *in: Proceedings of the Workshop on Planning under Uncertainty for Autonomous Systems at The International Conference on Automated Planning and Scheduling (ICAPS*. [S.l.: s.n.], 2005. Citado na página 47.

GANGL, S. *Robotic exploration for mapping and change detection*. Dissertação (Mestrado) — Leibniz Universität Hannover, 2014. Citado 4 vezes nas páginas 19, 20, 21 e 25.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 13, n. 5, p. 533–549, maio 1986. ISSN 0305-0548. Disponível em: <http://dx.doi.org/10.1016/0305-0548(86)90048-1>. Citado na página 29.

GLOVER, F. *Tabu search and adaptive memory programing. Advances, applications and challenges*. Norwell: Kluwer Academic Publishers, 1996. 1–75 p. Citado na página 30.

GMAPPING. 2015. Disponível em: <http://wiki.ros.org/gmapping>. Citado na página 45.

GONZÁLEZ-BAÑOS, H. H.; LATOMBE, J. Navigation strategies for exploring indoor environments. *I. J. Robotic Res.*, v. 21, n. 10-11, p. 829–848, 2002. Disponível em: <http://dx.doi.org/10.1177/027836490202101083 4>. Citado 2 vezes nas páginas 24 e 25.

GONZÁLEZ-BAñOS, H. H.; HSU, D.; LATOMBE, J.-C. *Motion planning: Recent developments*. [S.l.], 2005. Citado na página 15.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *Trans. Rob.*, IEEE Press, Piscataway, NJ, USA, v. 23, n. 1, p. 34–46, fev. 2007. ISSN 1552-3098. Disponível em: <http://dx.doi.org/10.1109/TRO.2006.889486>. Citado 2 vezes nas páginas 44 e 45.

HART, N. J. N. P. E.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4, n. 2, p. 100–107, 1968. Citado na página 47.

JULIÁ, M.; REINOSO, O.; GIL, A.; BALLESTA, M.; PAYÁ, L. Behaviour based multi-robot integrated exploration. *International Journal of Innovative Computing, Information - Vol. 7, Issue 9, pp. 5225-5244*, 2011. Citado 6 vezes nas páginas 25, 26, 27, 37, 38 e 50.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. *Optimization by simulated annealing*. 1983. 671–680 p. Citado na página 30.

LINDEN, R. *Algoritmos Genéticos, uma importante ferramenta de Inteligência Computacional*. [S.l.: s.n.], 2006. Citado 2 vezes nas páginas 30 e 34.

METAHEURISTICS Network Website. [S.l.], 2000. Acessado em 17 de Julho de 2016. Disponível em: <http://www.metaheuristics.net/>. Citado na página 29.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. 5th ed. ed. [S.l.]: John Wiley, 2001. ISBN 0471316490,9780471316497. Citado na página 64.

MURPHY, R. R. *Introduction to AI Robotics*. 1st. ed. Cambridge, MA, USA: MIT Press, 2000. ISBN 0262133830. Citado na página 14.

NEWMAN, P. M.; BOSSE, M.; LEONARD, J. J. Autonomous feature-based exploration. In: *IEEE International Conference on Robotics and Automation*. Taiwan: [s.n.], 2003. Citado 3 vezes nas páginas 21, 22 e 23.

OLIVEIRA, A. C. M. *Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos*. Tese (Doutorado) — INPE - Instituto Nacional de Pesquisas Espaciais., 2004. Citado 5 vezes nas páginas 16, 27, 28, 29 e 30.

OLIVEIRA, A. C. M.; LORENA, L. A. N. Detecting promising areas by evolutionary clustering search. In: _____. *Advances in Artificial Intelligence – SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 385–394. ISBN 978-3-540-28645-5. Disponível em: <http://dx.doi.org/10.1007/978-3-540-28645-5_39>. Citado 2 vezes nas páginas 31 e 57.

OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. *Annals of Operations Research*, v. 63, p. 513–628, 1996. Citado na página 29.

PANTRIGO, J. J.; SANCHEZ, A.; GIANIKELLIS, K.; MONTEMAYOR, A. Combining particle filter and population-based metaheuristics for visual articulated motion tracking. *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, v. 5, n. 3, p. 68–83, 2005. ISSN 1577-5097. Disponível em: <http://elcvia.cvc.uab.es/article/view/107>. Citado na página 34.

PRESS, W. H.; TEUKOLSKY, S. A.; VETTERLING, W. T.; FLANNERY, B. P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3. ed. New York, NY, USA: Cambridge University Press, 2007. ISBN 0521880688, 9780521880688. Citado 2 vezes nas páginas 16 e 28.

QUIGLEY, M.; CONLEY, K.; GERKEY, B. P.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. Ros: an open-source robot operating system. In: *ICRA Workshop on Open Source Software*. [S.l.: s.n.], 2009. Citado na página 41.

RAJA, P.; PUGAZHENTHI, S. Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 2012. Citado na página 14.

RAMESH, B.; MOHAN, V. C. J.; REDDY, V. V. Application of bat algorithm for combined economic load and emission dispatch. *Int. J. of Electricl Engineering and Telecommunications*, v. 2, n. 1, p. 1–9, 2013. Citado na página 35.

ROS. 2016. Disponível em: <http://www.ros.org/>. Citado 2 vezes nas páginas 41 e 42.

ROS/INTRODUCTION. 2014. Disponível em: <http://wiki.ros.org/ROS/Introduction>. Citado na página 17.

STACHNISS, C.; BURGARD, W. Exploring unknown environments with mobile robots using coverage maps. In: *Proc. of the International Conference on Artificial Intelligence (IJCAI)*. [S.l.: s.n.], 2003. Citado na página 18.

TALBI, E.-G. *Metaheuristics: From Design to Implementation*. [S.l.]: Wiley Publishing, 2009. ISBN 0470278587, 9780470278581. Citado 2 vezes nas páginas 28 e 34.

TEIXEIRA, R. *ricardoteix.com: code, electronics, automation, research and more*. [S.l.], 2016. Acessado em 18 de Julho de 2016. Disponível em: <http://www.ricardoteix.com/v-rep-plataforma-virtual-para-experimentos-com-robos/>. Citado 2 vezes nas páginas 43 e 44.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. [S.l.]: The MIT Press, 2005. ISBN 0262201623. Citado 3 vezes nas páginas 19, 20 e 22.

V-REP. 2013. Disponível em: <http://www.coppeliarobotics.com/>. Citado na página 17.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: *In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation*. [S.l.: s.n.], 1997. p. 146–151. Citado 2 vezes nas páginas 15 e 19.

YAMAUCHI, B.; SCHULTZ, A.; ADAMS, W. Mobile robot exploration and map-building with continuous localization. In: *In Proceedings of the 1998 IEEE/RSJ International Conference on Robotics and Automation*. [S.l.: s.n.], 1998. p. 3715–3720. Citado na página 19.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms*. [S.l.]: Luniver Press, 2008. ISBN 1905986106, 9781905986101. Citado 2 vezes nas páginas 27 e 35.

YANG, X.-S. Firefly algorithms for multimodal optimization. In: *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*. Berlin, Heidelberg: Springer-Verlag, 2009. (SAGA'09), p. 169–178. ISBN 3-642-04943-5, 978-3-642-04943-9. Disponível em: <http://dl.acm.org/citation.cfm?id=1814087.1814105>. Citado 3 vezes nas páginas 32, 35 e 57.

YANG, X.-S.; HE, X. Firefly algorithm: Recent advances and applications. *CoRR*, abs/1308.3898, 2013. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1308.html#YangH13b>. Citado 2 vezes nas páginas 32 e 33.

YOUBOT. 2015. Disponível em: <http://www.kuka-robotics.com/br/products/education/youbot/>. Citado na página 41.