DETERMINAÇÃO DE TRAJETÓRIA ÓTIMA EM NAVEGAÇÃO ROBÓTICA MÓVEL, UTILIZANDO ALGORITMO GENÉTICO.

Alexandre de Vasconcelos Siciliano

DISSERTAÇÃO SUBMETIDA AO CORPO DE DOCENTES DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA

Aprovada por:

_____

Prof. Antônio Carneiro de Mesquita Filho, Dr. d´État

_____

Prof. Jorge Lopes de Souza Leão, Dr. Ing.

_____

Prof. José Vicente Calvano, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 2006

SICILIANO, ALEXANDRE DE VASCONCELOS

Determinação de Trajetória Ótima em Navegação Robótica Móvel, Utilizando Algoritmo Genético [Rio de Janeiro] 2006

XIV, 95p. 29,7 cm (COPPE/UFRJ, M.Sc, Engenharia Elétrica, 2006)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Navegação Robótica Móvel
2. Algoritmo Genético
3. Otimização

I. COPPE/UFRJ II. Título (Série)

Dedicatória

Dedico este trabalho aos meus pais, Hugo e Marisa, por terem sido sempre meus maiores incentivadores em tudo o que fiz na minha vida.

Agradecimentos

Seria impensável concluir um trabalho desse porte sem a ajuda inestimável de diversas pessoas. Também seria impensável agradecer a todas sem cometer a injustiça de, por descuido, esquecer de mencionar o nome de algumas delas. Desde já me desculpo com todos que colaboraram comigo na execução deste trabalho, mas que porventura não tiveram seus nomes citados neste agradecimento. Tenham a certeza de que todos foram muito importantes para mim.

Mas, não poderia deixar de lembrar e de agradecer o inestimável apoio e paciência de minha família, sobretudo o de minha mulher, Adriana e minha filha, Gabriella, pois não foram uma ou duas vezes em que deixei de dar a devida atenção a elas para estar debruçado sobre livros e artigos. Saibam que sem o incentivo de vocês, não teria conseguido chegar até aqui.

Agradeço ao Professor, Doutor, Antonio Carneiro de Mesquita Filho, que me concedeu a honra de sua orientação e que, com seu grande conhecimento, pôde me ajudar a concluir esta dissertação. Quantas foram as vezes em que cheguei à COPPE com "mar revolto" e voltei para casa com a segurança de que poderia superar as adversidades e atingir meu objetivo. Mestre, muito obrigado pela imensa paciência e pela forma séria com que sempre me tratou me ajudando a contornar os problemas que surgiram durante a elaboração do trabalho.

Da mesma forma, essas mal traçadas linhas não estariam sendo escritas, não fosse a concessão feita pelo Sr Capitão-de-Mar-e-Guerra (EN) Francisco Roberto Portella Deiana, ex-Diretor do Centro de Mísseis e Armas Submarinas da Marinha (CMASM), quem primeiro me concedeu a permissão para freqüentar as aulas e ao Sr.

Capitão-de-mar-e-Guerra (EN) Ricardo Luiz Gomes Braga, atual Diretor do Centro, que a manteve, permitindo-me concluir o trabalho.

Agradeço também ao Sr Capitão-de-Fragata Arthur Luis de Amorim Moura, ex-Vice-Diretor do Centro e que durante todo o período em que serviu no CMASM sempre me incentivou, colaborando com artigos e, até mesmo, propondo assuntos para pesquisa.

Também não poderia deixar de agradecer ao amigo de turma e Doutor em engenharia, Capitão-de-Corveta (EN) José Vicente Calvano, que em um momento crucial para mim, me incentivou a prosseguir com o estudo.

Por fim agradeço a Deus por me abençoar e me dar saúde e tranqüilidade nos momentos difíceis da vida.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DETERMINAÇÃO DE TRAJETÓRIA ÓTIMA EM NAVEGAÇÃO ROBÓTICA MÓVEL, UTILIZANDO ALGORITMO GENÉTICO.

Alexandre de Vasconcelos Siciliano

Julho/2006

Orientador: Antônio Carneiro de Mesquita Filho

Programa: Engenharia Elétrica

Um método para determinar a trajetória ótima para um robô móvel usando algoritmo genético é apresentado. A idéia principal é determinar a trajetória que um robô, que inicialmente se encontra na posição (***x,y***) do sistema de coordenadas associado ao ambiente de deslocamento, deve seguir até a posição de destino (x_d, y_d), sem colidir com qualquer um dos obstáculos existentes. O algoritmo genético é iniciado com a geração aleatória dos indivíduos que constituem a população inicial. Cada indivíduo representa um possível caminho. Para cada uma das trajetórias, o algoritmo verifica se, houve proximidade ou impacto com os obstáculos e se o caminho gerado é o menor possível.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DETERMINATION OF THE OPTIMAL PATH IN MOBILE ROBOT NAVIGATION USING GENETIC ALGORITHM

Alexandre de Vasconcelos Siciliano

July/2006

Advisor: Antônio Carneiro de Mesquita Filho

Department: Eletrical Engeneering

A method to determine the optimal path of a mobile robot using genetic algorithm is presented. The central idea is to determine the trajectory that the robot, which is initially in a position (x,y) of the displacement environment, shall follow until reach the final position (x_d, y_d), without colliding with any obstacles existing in the environment. The initial genetic algorithm population is randomically generated, each individual represents a possible path. For each trajectory the algorithm checks the proximity to the obstacle and the shortest possible path.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Convenções tipográficas

As seguintes convenções tipográficas serão utilizadas nesta dissertação para identificar certos tipos de informação.

CONVENÇÃO	DESCRIÇÃO
CAIXA ALTA	Termos ou expressões sob definição
MAÍUSCULA	Siglas ou acrônimos.
itálico	Palavras ou expressões em outro idioma.
Itálico negrito	Variáveis

Nomenclatura

AG – Algoritmo Genético

CCD - *charge-coupled device* (Dispositivo de Carga Acoplado)

EKF – *Extended Kalman Filtre* (Filtro de Kalman estendido)

GPS – *Global Position System* (Sistema de posicionamento Global)

SLAM – *Simultaneous Localization And Mapping* (Localização e mapeamento simultâneo)

SSR – *Stochastic Sampling with Replacement* (Amostragem estocástica com reposição)

SUS – *Stochastic Universal Sampling* (Amostragem estocástica universal)

uc – unidade de comprimento

1 - Introdução

A palavra ROBÔ tem origem nas línguas Européias orientais, primeiramente no Tcheco *robota,* que significa trabalho forçado, escravidão, mas também no esloveno *rabota* cujo significado é servidão e *rabi* significando escravo [1]. Assim, robô é uma palavra que vem sendo empregada para designar qualquer coisa (ou até mesmo pessoas) que cumprem ordens mecanicamente.

Já a palavra NAVEGAR vem do latim *navigare: navis* – navio, + *igare* – dirigir, ou seja, no entendimento mais direto da palavra, navegar é percorrer com um navio o mar. Porém, analisado em sentido mais amplo, o conceito de navegar se confunde com o de guiar e pode ser entendido como conduzir um veículo de um ponto a outro, usando recursos próprios.

Nesse contexto surge a NAVEGAÇÃO ROBÓTICA MÓVEL, uma área da engenharia elétrica dedicada a pesquisar e desenvolver métodos que permitam aos robôs se locomoverem de forma autônoma um ponto de origem a um ponto de destino, cumprindo uma tarefa específica.

A essência da robótica móvel é a navegação, habilidade comum à maioria das espécies animais, que por meio de seus sentidos, principalmente o da visão e o da audição, viabilizam um deslocamento seguro na busca de um objetivo, que pode ser comida ou abrigo.

Reproduzindo o que ocorre na natureza, o robô também usa as informações sensoriais disponíveis, para realizar sua navegação, permitindo o planejamento, a avaliação e efetivamente a guiagem, evitando impacto com obstáculos, minimizando o percurso, o consumo de energia e etc...

Historicamente, os primeiros veículos controlados por *softwares* surgiram no final dos anos 60. O robô Shakey do Instituto de Pesquisa de Stanford é considerado o primeiro. O sucesso das pesquisas realizadas nos anos 60 e 70 conduziu a um maior investimento na área da robótica móvel. Nas pesquisas se percebeu que é fácil fazer um computador com inteligência comparável a de um adulto. Mas é difícil dar-lhe habilidade de uma criança de um ano de idade, quando o assunto é a mobilidade ou a percepção [1].

As aplicações dos robôs autônomos são as mais diversas, atualmente os robôs se deslocam no mar, na terra, no ar e até no espaço, sendo, portanto, figura importante do ponto de vista econômico, militar e científico. Os robôs móveis colaboram com a prospecção de petróleo em águas profundas, realizando tarefas em pontos que oferecem grande risco aos mergulhadores. São também empregados em minas, extraindo minério em locais de pouca ventilação, insalubres e, muitas vezes, explosivos. A habilidade de caminhar desviando de obstáculos introduziu o robô móvel no meio militar, onde é empregado para localizar e mapear campos minados em terra e no mar. Por fim, o desejo dos cientistas em explorar o terreno de planetas e satélites distantes, onde as condições ambientais adversas representam potencial perigo a vida humana, seria impensável sem o apoio dos robôs móveis.

Naturalmente, o problema de elaborar técnicas e métodos que permitam realizar o deslocamento em ambientes diversos deve ser encarado como um processo de otimização. O robô deve ser capaz de buscar, dentre todos os possíveis caminhos, o menor possível e livre de obstáculos.

Por PROCESSO DE OTIMIZAÇÃO, entende-se aquele do ajuste das entradas ou características de um sistema visando a obter um resultado considerado ótimo, em geral

um valor máximo ou mínimo. As entradas do processo, em geral, são os parâmetros do problema, o processo em si é conhecido como função custo ou função objetivo e o resultado final é o custo ou aptidão. A otimização trata, portanto, de achar a melhor solução, deixando implícito que, se existe uma melhor solução, é porque ela não é única [2].

Um dos métodos consagrados de busca da solução ótima é o ALGORITMO GENÉTICO. Trata-se de uma técnica evolutiva, inspirada na teoria proposta por Charles Darwin no século XIX em sua clássica obra "*On the Origin of the Species* (1859)", e ainda na genética molecular.

Os conceitos da teoria da evolução e da genética foram então usados por John Holland [3] para criar programas que simulam o processo de evolução de sistemas naturais.

A formulação do algoritmo genético produz vantagens quando comparada às demais técnicas de otimização [4].

- AGs não requerem formulações matemáticas complexas para solucionar um problema;
- Eficiência para encontrar o ótimo global, uma vez que buscam em todo o espaço; e
- Flexibilidade para introduzir heurísticas na implementação do algoritmo, que visem a melhorar o desempenho do AG para um problema específico.

Desta forma, no presente trabalho, o algoritmo genético será empregado como alternativa às técnicas tradicionais utilizadas para realizar o planejamento da navegação robótica, otimizando a trajetória a ser desenvolvida pelo robô.

Esta dissertação propõe um algoritmo genético para determinar a trajetória ótima de um robô, que deve deslocar-se em uma área com obstáculos, visando a atingir um ponto de destino.

A idéia do algoritmo é determinar coordenadas no ambiente de navegação que sirvam como pontos de guinada (*waypoints*) para o robô mudar de direção, realizando a trajetória mínima e livre de impactos até o ponto de destino.

Assim, para lidar com a natureza do problema, propõe-se um algoritmo genético como alternativa às técnicas que serão apresentadas no capítulo 3 (Revisão Bibliográfica), onde se apresentam algumas técnicas empregadas no planejamento do caminho e determinação de trajetória mínima.

A modelagem proposta é apresentada no capítulo 4, e consiste basicamente em fazer evoluir uma população inicial de indivíduos, gerada aleatoriamente, a qual representa possíveis caminhos para o robô. Cada indivíduo da população terá a sua aptidão avaliada, com o intuito de verificar se o caminho em análise não se aproxima, impacta ou passa por cima de um obstáculo qualquer.

O capítulo 5 apresentará os resultados obtidos. O capítulo 6 apresenta a conclusão.

Por fim, o capítulo 7 apresentará a referência bibliográfica, utilizada para elaboração do presente trabalho.

2 - Fundamentação Teórica e Definições

Como a proposta desta dissertação é apresentar um algoritmo para determinar TRAJETÓRIA ÓTIMA em NAVEGAÇÃO ROBÓTICA usando AG, neste capítulo são apresentados alguns conceitos e definições a respeito desses assuntos.

Entende-se por trajetória ótima em navegação robótica móvel, o percurso que permite a um robô móvel, navegando em um ambiente com obstáculos, atingir o ponto de destino, realizando o menor caminho possível livre de colisões.

2.1 - Percepção do ambiente

Quando um robô se desloca, a interação dele com o ambiente a sua volta, necessária para planejar sua trajetória, se da por meio de SENSORES PROPRIOCEPTIVOS, os quais informam dados internos ao robô, tais como temperatura dos motores, velocidade, nível de bateria e EXTEROCEPTIVOS, que informam dados externos ao robô [5]. Os dois tipos de sensores são necessários, uma vez que nos robôs móveis não é seguro basear a navegação apenas no sensoriamento interno, pois existe erro acumulado associado à medição da posição do robô. A capacidade de sensoriamento é imprescindível para um robô operar em ambiente completamente desconhecido, visto que não estão disponíveis informações a respeito dos objetos presentes no ambiente. E mesmo que houvesse, possíveis mudanças na posição dos objetos obrigaria a fazer atualizações contínuas das informações, novamente dependendo das informações dos sensores.

Exemplos de sensores internos são os de temperatura (para informar se algum item do robô está superaquecido), nível de bateria, codificadores óticos (associados às rodas indicam a velocidade de deslocamento).

Os sensores externos mais populares, devido ao custo benefício, são os de infravermelho, os de ultra-som, as câmeras CCD e os dispositivos de laser. Existem ainda os baseados no sistema global de posicionamento (GPS).

Os robôs móveis são empregados nos mais diversos ambientes, podendo ser o solo, o ar, a água ou ainda o espaço, e são dotados de diversos sistemas cujas propriedades possibilitam a execução de tarefas sem a necessidade de intervenção externa. Uma das principais características do comportamento autônomo decorre da necessidade do sistema em se adaptar e, eventualmente, se auto-organizar, para contornar situações imprevistas.

Para ser bem sucedido na tarefa de se deslocar de forma autônoma, o robô deve ser capaz de realizar:

- O controle do movimento - o robô deve se movimentar rápido, desviando de obstáculos estáticos ou dinâmicos;

- A modelagem do ambiente - enquanto se movimenta, o robô deve ser capaz de obter conhecimento do ambiente, dispondo de alguma forma de modelo que possa representar o ambiente;

- A localização – saber sua posição no ambiente é fundamental para viabilizar a navegação correta e serve como métrica do desempenho no alcance de uma meta preestabelecida; e

- O planejamento do caminho – decidir qual o caminho a seguir e se movimentar através dele, segundo algum critério de otimização, como a escolha do caminho mais curto ou o caminho que favoreça o menor consumo de energia.

2.2 - Modelagem do ambiente

De forma abrangente, pode-se dizer que existem dois modelos que permitem estudar o deslocamento do robô. O primeiro é o MODELO DO MUNDO, onde podem ser utilizados mapas topológicos ou métricos. Nestes modelos, os mapas são previamente construídos e armazenados para consulta durante o deslocamento do robô. Definido o caminho, a próxima etapa é a guiagem do robô por esse caminho, que depende da localização do robô e da eventual atualização do mapa com base nas informações sensoriais. A grande desvantagem desse modelo está no fato de que medições são, por definição, imprecisas. Portanto, dentre os inúmeros problemas da abordagem estão a imprecisão da representação do mundo, a imprecisão do próprio robô, a inconsistências entre o modelo armazenado e o ambiente, o qual é inerentemente dinâmico.

A partir dessas dificuldades surgiu o segundo modelo, que é o BASEADA NO COMPORTAMENTO (*based-behavior model*) [6] onde se assume que a melhor representação do ambiente de navegação é o próprio ambiente de navegação. Assim, o robô infere, por meio dos seus sensores e em tempo real as informações necessárias para se mover.

Conhecendo as características físicas dos sensores utilizados, a incerteza associada às medições pode ser isolada e modelada. Isto permite o uso das teorias probabilísticas para representar as incertezas associadas ao ambiente e também ao robô, surgindo os algoritmos estocásticos para o planejamento do caminho.

2.3 - Representação do ambiente por meio de mapas

O MAPA é a representação gráfica das regiões contidas em um ambiente. Ele deve incluir informações sobre as propriedades das regiões, objetos contidos nela e etc...

A percepção do ambiente pelo robô é fundamental na navegação robótica móvel. Várias são as atividades em que o robô está navegando em ambiente ESTRUTURADO, aqueles onde existem marcos confiáveis para a navegação, ou SEMI-ESTRUTURADO, onde geralmente é necessário inseri-los. Esquemas de navegação métrica, topológica ou híbrida podem fazer uso de diferentes tipos de marcos no ambiente.

MAPAS MÉTRICOS ou baseados em grades (*grid base)* representam o ambiente por meio de grades igualmente espaçadas, formando células. Cada célula indica a presença de um obstáculo na região correspondente [1, 7, 8]

Geralmente são mapas fáceis de construir e manter mesmo em ambientes amplos e uma vez que as grades correspondem diretamente ao ambiente (mundo real), a posição do robô dentro do modelo pode ser determinada pela posição e orientação do mundo real, as quais podem ser obtidas de forma suficientemente precisa mesmo com o uso de sensores ultra-sônicos. Assim, mapas métricos permitem ao robô estimar suas coordenadas. Como conseqüência, posições diferentes para as quais os sensores produzem mesma medição, ou seja, posições "parecidas" (*look alike*) não são confundidas. Um exemplo de mapa, construído a partir da informação sensorial do robô é mostrado na figura 2.1.

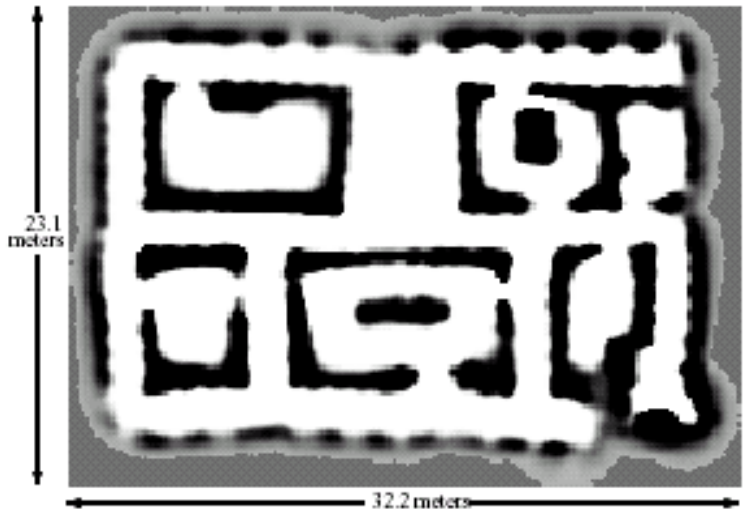


Figura 2.1 - Exemplo de mapa métrico

MAPAS TOPOLÓGICOS [1, 8]representam o ambiente em grafos, como o do exemplo mostrado na figura 2.2. Os vértices desses grafos correspondem a marcos dentro do ambiente. Eles são conectados por ramos, se existir caminho direto entre eles. A compacidade desses mapas dá a eles sua maior vantagem, pois permitem rápido planejamento e programação simplificada. Se comparado com os mapas métricos, os mapas topológicos sofrem menos os efeitos do "deslizamento" ou "deriva" do robô, que são fenômenos comuns em robótica móvel, tornando-se mais precisos. Contudo, as aproximações topológicas podem falhar no reconhecimento de lugares que são próximos geometricamente ou nos lugares "parecidos" (*look alike*).

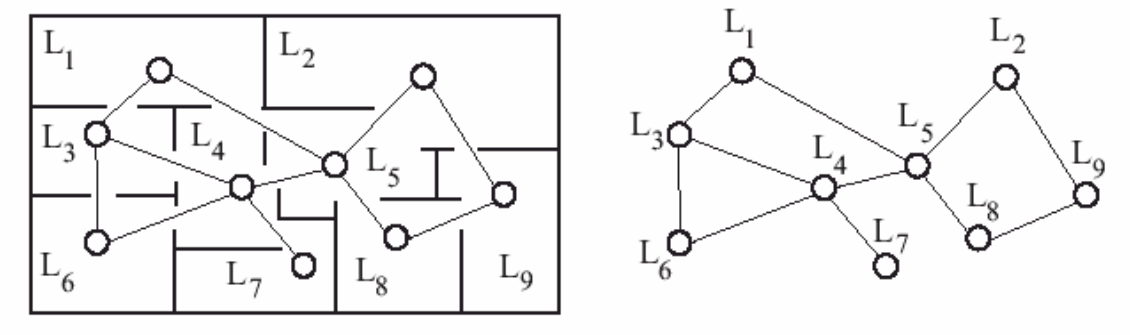


Figura 2.2 - Exemplo de mapa topológico. Os vértices representam as regiões e seus ramos o caminho entre elas.

Tabela 2.1 - Vantagens e desvantagens das aproximações métricas e topológicas

	Aproximação métrica	Aproximação topológica
Vantagens	Fácil de construir, representar e armazenar.	Permite um planejamento do caminho mais eficiente.
	Reconhecimento de lugares é não ambígua/ não sensível ao ponto de observação	Não necessita de determinação precisa da posição do robô.
	A programação de caminhos curtos é facilitada.	Facilidade na programação simbólica
Desvantagens	Planejamento do caminho ineficiente.	Difícil de construir e manter, sobretudo em ambientes amplos.
	Necessita de determinação precisa da posição do robô.	Reconhecimento de lugares é ambígua e sensível ao ponto de observação.
	Não é apropriada para programações simbólicas.	Pode levar a caminhos não otimizados.

OS MAPAS HÍBRIDOS [1]misturam as características positivas de ambos de forma a compensar a fraqueza individual de cada tipo de mapa.

2.4 - Planejamento do caminho

Em robótica móvel, PLANEJAR UM CAMINHO refere-se principalmente à competência que um robô possui de escolher, dentre várias possibilidades, a rota que pretende percorrer. Para tal, ele faz uso do mapa, baseado no qual, avalia as opções, a posição corrente por onde está se movendo e os obstáculos existentes.

O problema de planejamento de caminhos é bastante complexo. Complexidade essa que varia segundo fatores como a forma geométrica do robô, seus graus de liberdade, o conhecimento e tamanho do ambiente, coordenação, tipo e objetivo do movimento, entre outros. Existem duas vertentes principais [9]: o planejamento global

e o planejamento local, que devem ser utilizados simultaneamente para que se possa ter um melhor desempenho do móvel em um ambiente real.

O PLANEJAMENTO GLOBAL é responsável pelo mapeamento do ambiente onde está inserido o móvel em um modelo simplificado, estático, previamente armazenado, que lhe permite traçar um caminho mesmo por um local que não consiga perceber com seus sensores, seja por estar fora do alcance ou por estar obstruído por algum obstáculo.

O PLANEJAMENTO LOCAL é responsável pela navegação em curta distância, baseada nos valores coletados pelos seus sensores. Uma vez modelado o ambiente, o planejamento local será o responsável pela localização do robô neste mundo simplificado e dinâmico, devido a constante atualização dos dados não fornecidos pelo modelo global.

Para o planejamento do caminho é muito empregado o métodos *DEAD RECKONING* que é um procedimento matemático simples para a determinação da localização atual do robô, usando informações prévias de sua posição em função da velocidade, do deslocamento e do curso, durante um período de tempo [5, 10].

A forma mais simples é geralmente conhecida como HODOMETRIA, na qual o deslocamento do veículo é obtido diretamente por sensores hodométricos *on-board* [10]. Uma forma comum de realizar essas medidas é por meio de um *encoder* óptico diretamente acoplado no eixo do robô.

Como a idéia fundamental da hodometria é a integração do deslocamento no tempo, o acumulo de erro é inevitável. Os erros presentes podem ser inerentes aos sensores; erros devido ao deslize das rodas; erros devido a deformações no terreno; erros devido ao diâmetro diferente das rodas e etc...

A NAVEGAÇÃO INERCIAL [11]surge como uma alternativa para melhorar a performance do *dead reckoning* .O princípio de operação envolve medições contínuas da aceleração em cada um dos três eixos e a integração no tempo para obter a velocidade e a posição. Uma plataforma com giroscópios é usada para implementar um referencial inercial. Embora o método seja simples no conceito, a implementação é complexa. A necessária precisão dos giroscópios eleva muito o custo de produção e manutenção.

Outro importante assunto em robótica móvel é o conceito de AUTOLOCALIZAÇÃO do robô. Ela pode ser feita de forma estática ou de forma dinâmica. Na forma estática, o robô assume uma posição qualquer dentro do ambiente e determina a configuração desse ambiente. Na configuração dinâmica, o robô se move continuamente e vai atualizando a configuração do ambiente. É comum na autolocalização a ocorrência da construção ou atualização de um mapa. Tanto a localização quanto o mapeamento podem ser efetuados simultaneamente e em tempo real. Esse procedimento é conhecido como SLAM – *Simultaneous Localization And Mapping*. A abordagem dominante para a resolução do problema de SLAM foi introduzida em artigo escrito por Smith, Self et al. [12], no qual é proposto o uso de filtros de Kalman extendido (EKF) para progressivamente estimar a distribuição *a posteriori* do robô em função de marcos no mapa.

2.5 – Métodos para navegação robótica

O planejamento do caminho é muito importante para a realização da navegação robótica. Ele traz implícita a idéia de que o robô deve, de alguma forma, evitar impactos com os obstáculos existentes, por isso, esta importante etapa só é obtida com os MÉTODOS PARA NAVEGAÇÃO ROBÓTICA.

Uma das chaves para o sucesso da navegação robótica móvel é a capacidade de desviar de obstáculos. Todos os robôs autônomos possuem sensores e algum tipo de algoritmo que permitem, a partir das informações disponíveis, desviar dos obstáculos encontrados pelo caminho. Os métodos são os mais variados, existem aqueles nos quais o robô deve parar, processar as informações e então atuar de forma a desviar dos obstáculos. E existem aqueles, mais elaborados, em que o robô contorna os obstáculos a medida em que se desloca, realizando o processamento das informações em tempo real.

O desvio de obstáculos se refere, então, à metodologia de se adequar a trajetória do robô de forma a contornar os obstáculos, evitando colisões ou aproximações não desejadas. Os métodos dependem da posição atual do robô e das leituras dos sensores. Existem vários algoritmos [5, 7, 10, 13] para desvio de obstáculos e as técnicas propostas diferem principalmente no uso dos dados sensoriais e no controle de movimento do robô.

Os principais tipos de algoritmo para desvio de obstáculos são:

- Campos Potenciais
- Grades de ocupação (ou grade de certeza)
- Força de campo vetorial e
- Histograma de força vetorial

2.5.1 - Métodos de campos potenciais

Fazer um robô desviar de obstáculos usando campos potenciais está baseado em um simples e poderoso princípio [5, 13, 14] o de forças de atração e repulsão. Neste método, o robô é considerado como sendo uma partícula que se encontra dentro de um campo potencial artificial gerado pelo ponto de destino e pelos obstáculos presentes no

ambiente. O ponto de destino gera uma força de atração, enquanto que os obstáculos geram forças repulsivas.

O movimento do robô imerso em um campo potencial pode ser interpretado como sendo o de uma partícula sujeita a um campo vetorial gerado por cargas positivas e negativas. Nesta analogia, o robô é a carga positiva, o ponto de destino é a carga negativa e os obstáculos também são consideradas cargas positivas.

O campo potencial artificial gerado será representado pela equação 2.1 abaixo [13]:

$$U(q) = U_{atr}(q) + \sum_{i=1}^{n} U_{rep_i}(q) \tag{2.1}$$

Onde ***q*** é a posição do robô, representado pelo par ordenado (***x,y***), (no caso de deslocamento no plano) e $U_{rep_i}(q)$ representa o potencial repulsivo gerado por cada um dos n obstáculos.

Considerando que ***U(q)*** é diferenciável, em cada ponto ***q***, o gradiente do campo potencial, denotado por $\nabla U(q)$ é um vetor que aponta na direção que localmente aumenta ***U(q).***

A força que governará o robô pode ser escrita sob a forma da expressão 2.2:

$$\vec{F}(q) = \vec{F}_{atr}(q) + \vec{F}_{rep}(q) = -\nabla U_{atr}(q) - \nabla U_{rep}(q) \tag{2.2}$$

Onde ***F(q)*** pode ser considera como sendo o vetor velocidade do robô.

O potencial de atração usualmente empregado é uma parábola que cresce com a distância do destino [5, 13] (equação 2.3)

$$U_{atr} = k_{atr} d_{obj}^2(q) \tag{2.3}$$

onde $d_{obj}^2(q)$ é a distância euclidiana existente entre a posição atual do robô e o ponto de destino ou objetivo ($\boldsymbol{q_{obj}}$).

A força atrativa para este caso é obtida pela expressão:

$$\vec{F}_{atr}(q) = -\nabla U_{atr}(q) = -k_{atr}(q - q_{obj}) \tag{2.4}$$

Percebe-se da equação 2.4 que a força que governa o robô diminui a medida que ele se aproxima do destino, reduzindo sua velocidade. A figura 2.3 mostra um ponto sob a ação de um campo atrativo.

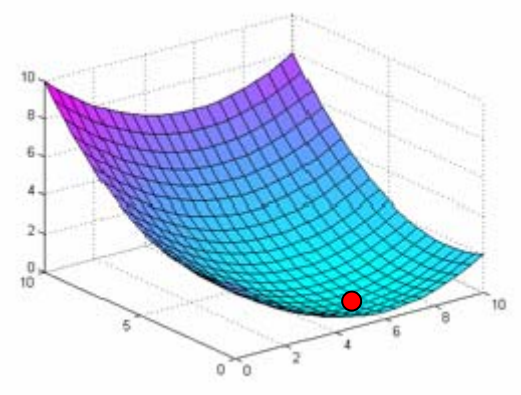


Figura 2.3 - Potencial de atração. O ponto vermelho representa o ponto de destino

O potencial repulsivo mantém o robô afastado dos obstáculos enquanto ruma para o ponto de destino. Ele é mais forte quando o robô está mais perto do obstáculo e menos intenso quando o robô está afastado de obstáculos.

$$U_{rep}(q) = \sum_{i=1}^{n} U_{rep_i}(q) \tag{2.5}$$

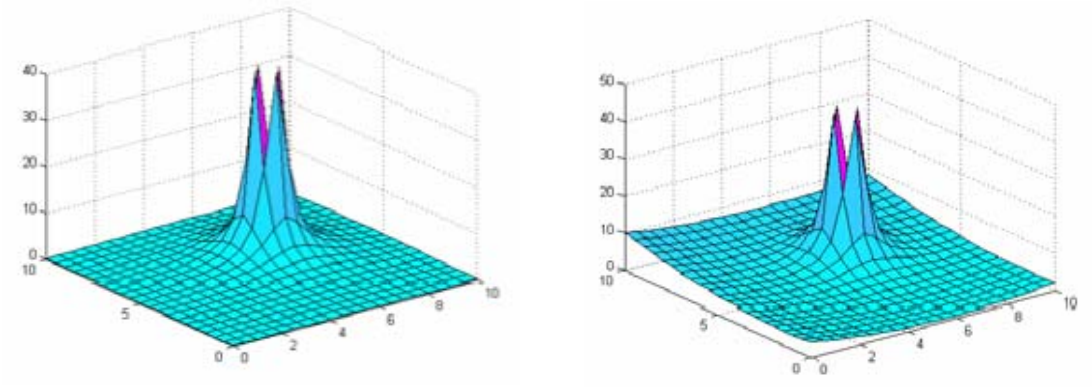
A influência repulsiva de um obstáculo está limitada à sua área circunvizinha e a magnitude de seu potencial deve crescer a medida que o robô se aproxima do obstáculo. Assim, a equação 2.6 representa o potencial de repulsão [13]:

$$U_{rep}(q) = \begin{cases} k_{obst_i}\left(\dfrac{1}{d_{obst_i}(q)} - \dfrac{1}{d_o}\right)^2 & \text{se } d_{obst_i}(q) < d_o \\ 0 & \text{se } d_{obst_i}(q) \geq d_o \end{cases} \qquad (2.6)$$

onde $d_{obst_i}(q)$ é a menor distância do ponto $\boldsymbol{q}$ para o obstáculo i, $\boldsymbol{k_{obst}}$ é uma constante e $\boldsymbol{d_o}$ é o limite de influência do obstáculo.

A força de repulsão será escrita da forma:

$$\vec{F}_{rep_i}(q) = -\nabla U_{rep_i} = \begin{cases} k_{obst_i}\left(\dfrac{1}{d_{obst_i}(q)} - \dfrac{1}{d_o}\right)\dfrac{1}{d^2_{obst_i}(q)}\dfrac{q - q_{obst}}{d_{obst_i}} & \text{se } d_{obst_i}(q) < d_o \\ 0 & \text{se } d_{obst_i}(q) \geq d_o \end{cases}$$



a) b)

Figura 2.4 - Potencial de repulsão (a) e potencial resultante (b).

A figura acima apresenta o potencial repulsivo para 3 obstáculos (2.4a) e o campo potencial resultante (2.4b), obtido pela soma do potencial de atração, indicado na figura 2.3. Este exemplo deixa bastante claro que o deslocamento do robô para o ponto de destino é similar ao de uma bola que é largada em um ponto qualquer do campo representado na figura 2.4b.

A grande desvantagem deste método está na existência de mínimos locais, que geralmente aparecem devido à simetria do ambiente e ao comportamento oscilatório do

robô, quando atravessando ambientes estreitos, como o caso exemplificado na figura 2.5.

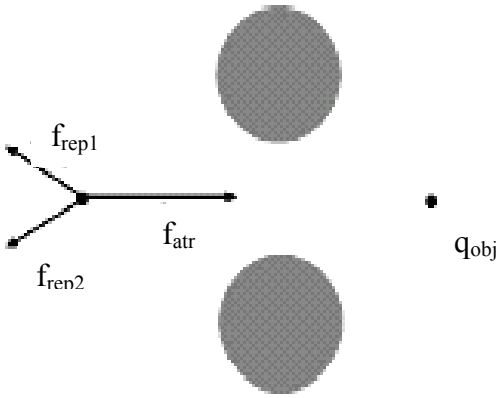


Figura 2.5 - Mínimo local devido a simetria dos obstáculos

2.5.2 - Grades de Ocupação ou de certeza

É um método para representação probabilística de obstáculos em um ambiente dividido em grades ou células. Este modelo é especialmente apropriado para mapas métricos [5, 7].

A área de trabalho do robô é representada por uma matriz de elementos quadrados, denotados de células. Para cada célula será atribuído um valor de certeza, o qual indica a medida de confiança de que um obstáculo existe dentro da área da célula. Essa medida é calculada e atualizada, levando-se em conta as características do sensor, por meio de uma função de probabilidade que o descreve.

O exemplo típico deste método emprega sensores ultra-sônicos. Eles retornam a distância radial para o objeto mais próximo dentro de um cone, sem especificar a localização angular dele. Se um objeto for detectado pelo sensor ultra-sônico, é mais provável que ele esteja mais próximo do eixo acústico do sensor, (e, assim, nestas células o valor de certeza é maior) do que nas células da periferia. A figura 2.6

apresenta um robô navegando nas proximidades de um obstáculo. O seu sensor "percebe" um objeto dentro de cone de medição e atribui valores maiores de probabilidade nas células dentre dessa área.

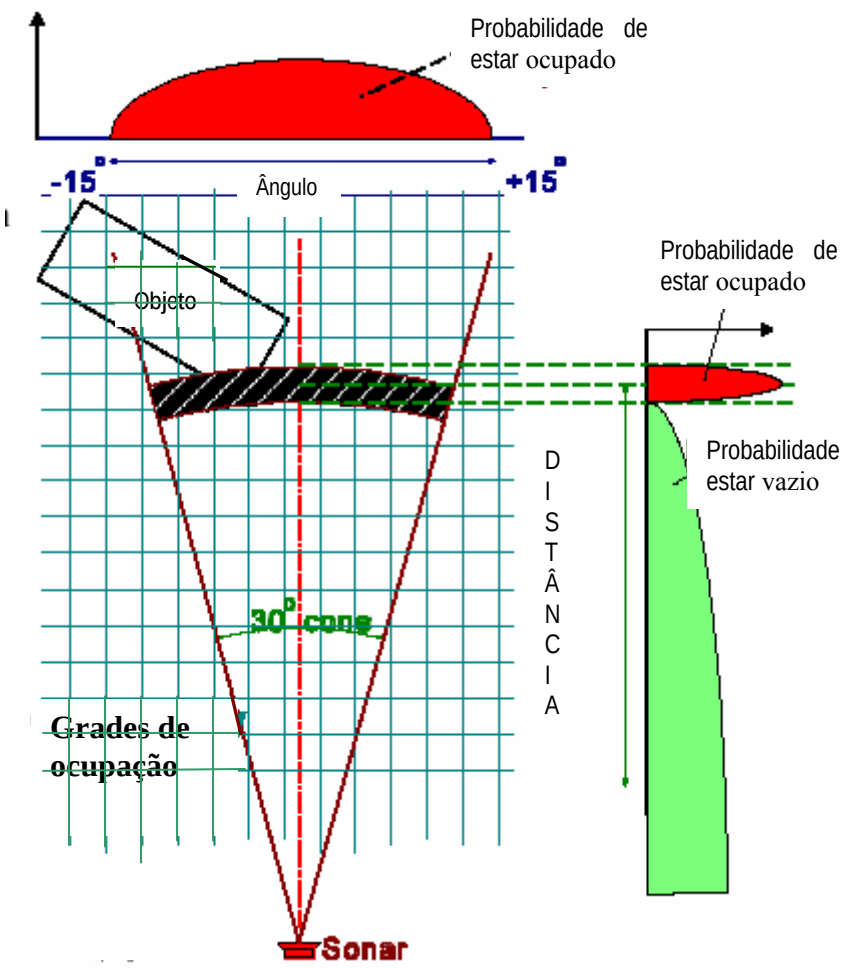


Figura 2.6 - Projeção bidimensional do campo de visada de um sensor ultra-sônico. A área escurecida indica a existência de um obstáculo.

2.5.3 - Campo de força virtual (CFV)

O campo de força virtual foi um método desenvolvido visando à diminuição dos problemas de mínimos locais observados no método de campos potenciais. Ele permite navegação contínua com velocidade maior do que a observada naquele tipo de

navegação. Para evitar os mínimos locais, o método mistura os conceitos de grades de ocupação com os de campos potenciais [5, 7].

Um histograma cartesiano de duas dimensões representa a distribuição dos obstáculos. Como no método de grades de certeza, cada célula (***i,j***) do histograma possui um valor de certeza ($C_{i,j}$) que representa a confiança de existência de obstáculo naquela posição. A diferença entre os métodos está na forma de como ele é construído e como é atualizado.

No método da grade de certeza, a probabilidade obtida é calculada para todas as células que são afetadas diretamente pela abrangência dos sensores. Isto é computacionalmente caro. O método de campo de força vetorial atualiza somente uma célula do histograma para cada leitura de distância, criando uma distribuição de probabilidade com pouco custo computacional.

Apesar de parecer uma simplificação grande do método de grades de certeza, a distribuição de probabilidade é verdadeiramente obtida pela continua e rápida amostragem de cada sensor enquanto o robô se move. Portanto, a mesma célula e sua vizinhança são repetidas vezes incrementadas.

Isto resulta em uma distribuição de probabilidade na qual altos valores de certeza são obtidos nas células próximas a localização do obstáculo. Após o levantamento da distância, o próximo passo é aplicar a idéia de campos potenciais, usando força repulsivas e atrativas.

A deficiência do método é que, devido a natureza do histograma, pequenas variações de uma célula para a outra pode levar a uma grande variação de força, dificultando a navegação. Outro problema que surge, é quando o robô atravessa um

corredor estreito, onde as forças atrativas e repulsivas podem levar a uma navegação oscilatória.

2.5.4 - Histograma de campo vetor (HCV)

É um método para navegação e desvio de obstáculos, proposto por Borenstein e Koren [7], também apresenta um histograma cartesiano bidimensional. Este modelo é continuamente atualizado com dados de distância a partir dos sensores a bordo do robô. O método emprega um processo de redução de dados de dois estágios de forma a obter os comandos de controle do robô.

No primeiro estágio, um subconjunto do histograma cartesiano considerado, em torno da posição atual do robô, é convertido em histograma polar. Cada setor do histograma polar contém um valor de densidade de obstáculos, na direção do setor.

No segundo estágio, o algoritmo escolhe o melhor setor dentre todos (com a menor densidade polar de obstáculos) e o governo do robô é alinhado naquela direção.

Os 3 estágios principais para a implementação do algoritmo, podem ser resumidos da seguinte forma:

1. Construir um histograma cartesiano de representação de obstáculos;
2. Do histograma obtido, considere uma janela em torno do robô e filtre o histograma cartesiano em um histograma polar unidimensional;
3. Calcule a guiagem e a velocidade de controle a partir do histograma unidimensional como resultado de um procedimento de otimização.

O método apresenta vantagens em relação aos anteriores. Por não trabalhar com forças atrativas e repulsivas, não fica preso em mínimos locais. A navegação oscilatória do método verificada no método de histograma de campo vetor também não é

verificada. Entretanto, não é um método para a localização do caminho ótimo e também pode levar o robô a um caminho sem saída.

2.6 - Algoritmos Genéticos

2.6.1 - A estrutura do AG

Os AG são algoritmos de busca global, inspirados na teoria da evolução, impondo o princípio de sobrevivência do mais apto a fim de produzir soluções melhores a cada geração. Ou seja, a cada geração, uma nova população de indivíduos é criada a partir das informações genéticas dos melhores indivíduos da geração anterior, selecionados a partir de um critério específico.

O método foi desenvolvido por Holland, J [3] e popularizado por um de seus alunos, David Goldberg.

A idéia do algoritmo é codificar soluções potenciais para um dado problema em uma estrutura simples chamadas de CROMOSSOMO e aplicar a elas operadores de recombinação, de forma a preservar a informação crítica. Cada cromossomo consiste em um número de GENE, sendo cada gene composto por bits.

A figura 2.7 apresenta as características de um AG e a figura 2.8 um fluxograma de sua execução, com o critério de parada adotado, o valor da aptidão máxima.

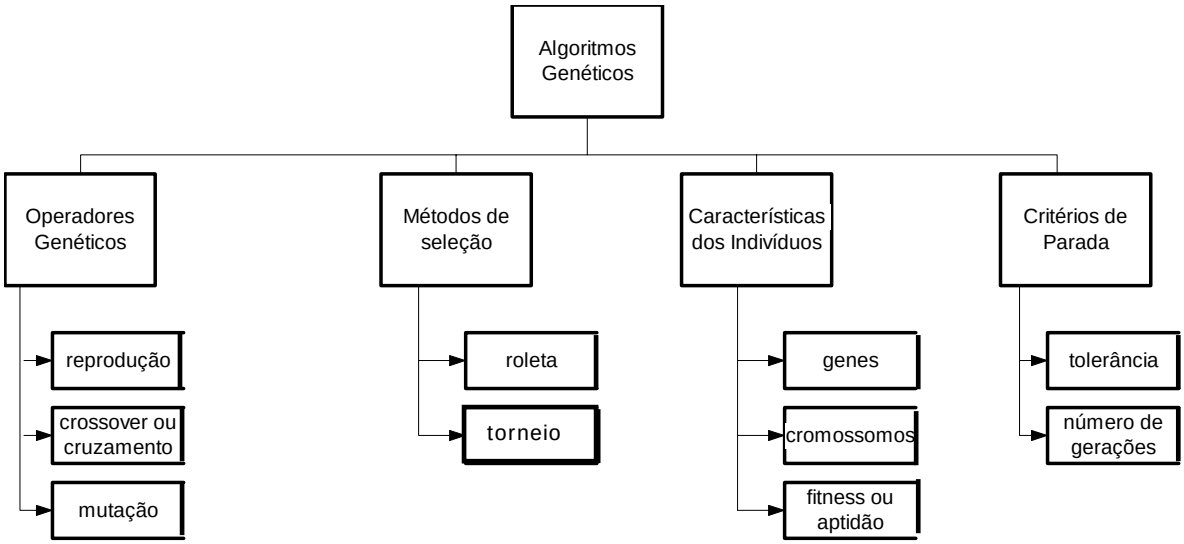


Figura 2.7 - Características dos algoritmos genéticos [15]

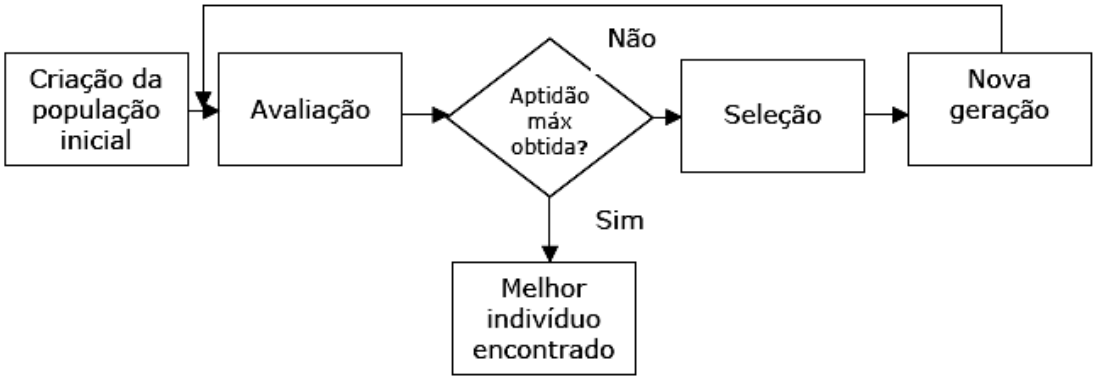


Figura 2.8 - Fluxograma de uma algoritmo genético [15]

2.6.2 - População inicial e codificação do cromossomo

Uma implementação típica de um AG começa pela geração de uma POPULAÇÃO INICIAL de cromossomos de forma aleatória. No AG básico o cromossomo consiste em uma *string* geralmente binária de comprimento L, como a mostrada na figura 2.9. Esta população é avaliada, de forma tal que, aqueles indivíduos que representam uma melhor solução para o problema, recebem mais chances de se reproduzir e de gerar

descendentes do que aqueles que oferecem soluções mais pobres. A APTIDÃO (*fitness)* indica a qualidade da solução. É, em geral, definida em relação à população atual.

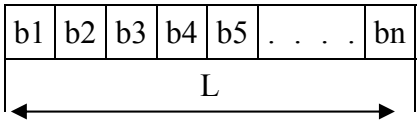


Figura 2.9 - *String* binária de comprimento L

2.6.3 - Função aptidão

As noções de avaliação e aptidão são, algumas vezes, consideradas equivalentes [16]. Contudo, é importante distinguir entre função de avaliação ou objetivo e função aptidão. A FUNÇÃO AVALIAÇÃO fornece uma forma adequada de qualificar o indivíduo em relação ao objetivo que se deseja alcançar. Ela provê uma medida de performance do indivíduo para um conjunto particular de parâmetros. A FUNÇÃO DE APTIDÃO transforma aquela medida de performance em oportunidade reprodutiva para os indivíduos. A avaliação de um indivíduo, representando um conjunto de parâmetros, é independente da avaliação de qualquer outro indivíduo, mas a aptidão do indivíduo é sempre definida em relação aos outros membros da população corrente.

A aptidão pode ser definida como sendo f_i/f, onde f_i é a avaliação do cromossomo i e f é a média das avaliações de todos os cromossomos da população. Nesse caso, ela é dita aptidão proporcional. Existe ainda a aptidão em escalonamento linear, onde o relacionamento entre a função avaliação ou objetivo e a função aptidão é linear e a aptidão por pontuação, onde são atribuídos graus de aptidão aos indivíduos, tipicamente no intervalo [0, 2], os quais são ordenados em ordem decrescente, do menos apto ao mais apto, antes de se aplicar a função aptidão. Esse processo gera um grave problema que é o da convergência prematura para um mínimo local.

2.6.4 - Processo de seleção

No processo evolutivo, na transição de uma geração para a outra, os indivíduos passam por um processo de SELEÇÃO NATURAL, que escolhe quais os indivíduos mais aptos a gerar os descendentes da geração seguinte.

Existem vários modos de se fazer a seleção [2, 16, 17]. Pode-se imaginar, por exemplo, a população disposta em um mapeamento sobre uma ROLETA, onde cada indivíduo é representado pelo espaço que proporcionalmente corresponde a sua aptidão.

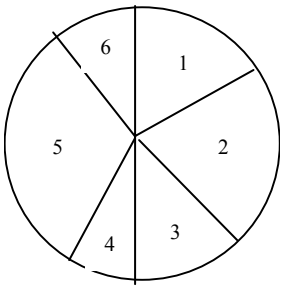


Figura 2.10 - Exemplo de roleta

Nota-se que na figura 2.10, o indivíduo cinco possui maior grau de aptidão, ocupando conseqüentemente maior intervalo. Por outro lado, o indivíduo 6, com menor aptidão, ocupa menos espaço na roleta.

Para selecionar um indivíduo, um número aleatório é gerado para simular um giro da roleta até esse número. O indivíduo correspondente ao intervalo onde a roleta pára é selecionado. O processo é repetido até que se atinja o número desejado de indivíduos da nova população.

O método utilizado na roleta é o do SSR (*Stochastic Sampling with Replacement*) [16]. Com o SSR qualquer indivíduo tem a possibilidade de formar, por completo, a próxima população.

Outro método similar é o método do SUS [16](*Stochastic Universal Sampling*) que consiste na amostragem dos elementos da população utilizando-se um único giro da roleta. A roleta, mostrada na figura 2.11, é construída tomando-se por base os valores de aptidão dos elementos da população, os quais definem proporcionalmente o tamanho de cada uma das seções.

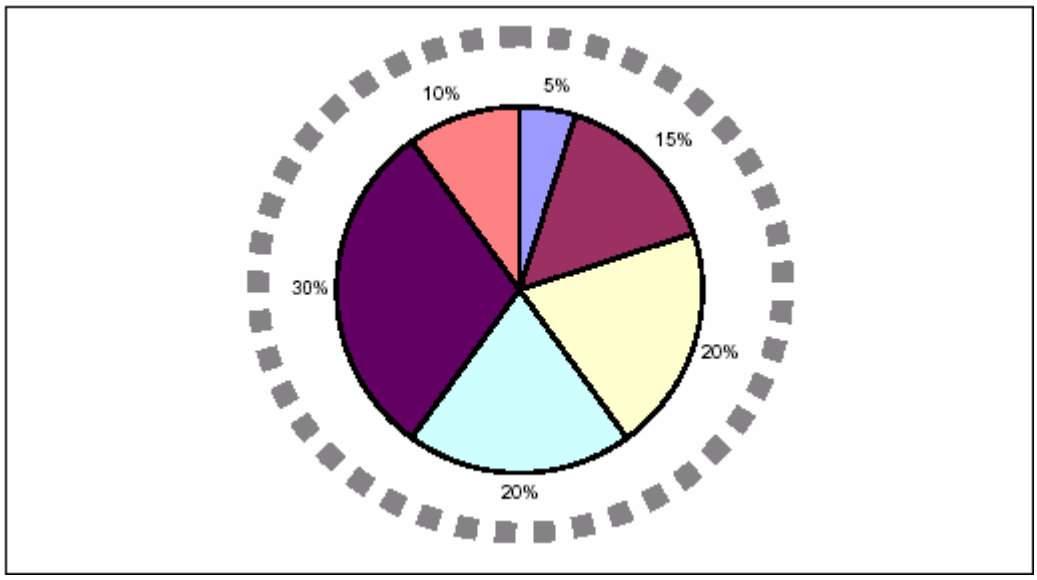


Figura 2.11 - Método de seleção SUS

No lugar de se usar um só ponteiro, são utilizados ***N*** ponteiros igualmente espaçados ao redor da roleta, onde ***N*** é o número de indivíduos a serem selecionados. Esse processo é realizado por meio da geração de um número randômico ***R*** dentro do intervalo de seleção. Os ***N*** indivíduos são então escolhidos gerando-se ***N*** ponteiros espaçados de 1 [***R, R+1, R+2.....R+N***] e selecionados aqueles cujos intervalos da roleta são interceptados por esses ponteiros. O custo computacional do SUS é menor do que o do SSR, uma vez que este último é da ordem de *NlogN*, e o SUS é de ordem N.

Um outro método de seleção muito empregado é o do TORNEIO [2, 17]. O processo se inicia com a ordenação da população por ordem decrescente de aptidão (*ranking*). Dentre aqueles indivíduos de maior aptidão será selecionado randomicamente um pequeno subconjunto de indivíduos que serão colocados para realizar o torneio. O individuo de maior aptidão ou aquele que superar um determinado limite previamente determinado será selecionado como sendo um dos genitores. Este processo ocorrerá tantas vezes quantas forem necessárias para se atingir a quantidade de genitores.

A cada torneio os melhores indivíduos são postos para se reproduzir, o que proporciona, a cada geração, um aumento do valor médio da aptidão da população.

Em todos os métodos de seleção existe, ainda, a possibilidade de se optar pelo ELITISMO, que consiste na repetição incondicional do indivíduo melhor adaptado de uma geração na geração seguinte, para que não haja risco de se perder a melhor solução alcançada até aquele momento.

Como mencionado, a seleção dos indivíduos da população é um processo muito importante e está relacionada com a escolha dos mais aptos. Mas dentro da formulação do AG, os indivíduos devem interagir entre si e esta interação é obtida por meio dos OPERADORES GENÉTICOS, os quais serão apresentados a seguir.

2.6.5 - Operadores genéticos

Os dois operadores genéticos mais empregados em um AG simples são o cruzamento ou *crossover* e a mutação.

A idéia por trás do CRUZAMENTO é promover a geração de novos indivíduos a partir do material genético de seus antecessores, aproveitando o que há de melhor em cada um deles. Com isso, a medida em que as gerações avançam, cresce o número de descendentes com maior chance de sobrevivência, por estarem mais aptos.

Os algoritmos genéticos tradicionais geralmente usam o cruzamento simples ou de um ponto, onde dois indivíduos, o "pai" e a "mãe", são cada um cortados em um ponto correspondente e seus segmentos, a partir do ponto de corte, são intercambiados. O cruzamento é a operação predominante em um AG e é realizada em grande parte da população. É regra prática a adoção da seguinte probabilidade de cruzamento [18]: 60% em populações grandes (da ordem de 100 ou mais indivíduos) e 90% em populações pequenas (da ordem de 30 indivíduos).

Apesar de ser o mais tradicional, o cruzamento simples não é o único tipo existente. Existem outras formas diferentes de se fazer a combinação dos pais, como por exemplo, cortá-los em dois ou mais pontos [19] ou fazer o chamado cruzamento uniforme, que é aquele onde cada gene do descendente é criado pela cópia do gene correspondente de um dos seus pais, escolhido de acordo com uma máscara randomicamente gerada (*crossover mask*). Quando na máscara existir um 1, o gene a ser copiado será o do "pai", quando for 0 será copiado o da "mãe".

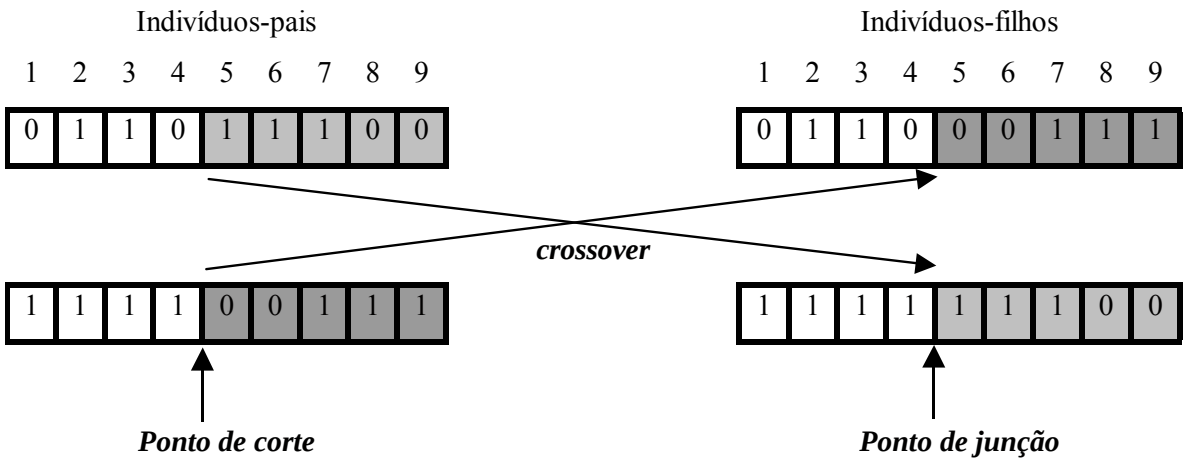


Figura 2.12 - Operação de *crossover* entre dois indivíduos

A operação de cruzamento tem uma característica desfavorável em longo prazo, na busca do ótimo global por ser um operador que torna população mais homogênea. Após algumas gerações a população terá indivíduos semelhantes, que habitam uma região do espaço de busca, onde existe um ótimo que pode não ser necessariamente um ótimo global, indicando uma convergência prematura.

Para contornar o problema, é necessário incluir no AG um operador que torne a população mais heterogênea. Esse operador é a MUTAÇÃO.

Os tipos de mutação mais empregados são os seguintes:

- Inversão: escolhe-se duas posições aleatórias dentro do cromossomo, e inverte-se o bloco compreendido entre essas posições;
- Inserção: escolhe-se uma posição aleatória do cromossomo a ser inserida em uma outra posição aleatória;
- Deslocamento: uma partição do cromossomo é aleatoriamente escolhida e inserida em outra posição do cromossomo; e
- Troca recíproca: duas posições do cromossomo são escolhidas aleatoriamente e trocadas.

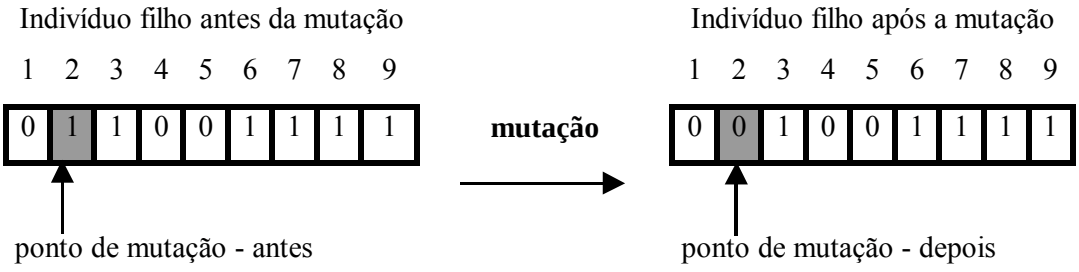


Figura 2.13 - operação de mutação em um indivíduo

2.6.6 - Critério de Parada.

Finalmente, nos AG podem ser estabelecidos dois critérios de parada, segundo os quais a evolução já teria completado seu objetivo: número de gerações ou tolerância. Na tolerância, a execução do algoritmo encerra-se após a constatação da existência de um indivíduo com as mesmas características do indivíduo-alvo, dentro de uma faixa de tolerância estabelecida. Trata-se, portanto, de um critério *a posteriori* No número de gerações, seja qual for a proximidade do alvo em que estiver o melhor indivíduo no fim da execução, consistindo em um critério *a priori.*

3 - Revisão Bibliográfica.

Neste capítulo encontram-se resumidos alguns artigos publicados sobre o planejamento de caminhos e navegação robótica. Foram selecionados alguns trabalhos que oferecem abordagens diferentes para o problema, apresentando um cenário do que tem sido produzido em termos de pesquisa sobre o assunto.

Uma classe de métodos utilizada com freqüência é baseada na teoria dos grafos. Alguns desses métodos fazem uso da geometria dos obstáculos para definir os vértices do grafo; os ramos sendo definidos como as ligações entre os vértices. Após a construção do grafo, podem ser usados vários métodos para a solução do problema do caminho mínimo entre os quais, o algoritmo de Dijkstra [20]e os baseados em grafos ponderados [21, 22], onde o custo de cada caminho está associado a uma probabilidade de se ter um caminho direto livre entre os dois vértices de cada ramo. Como principal inconveniente desta classe de métodos deve-se citar o aumento da complexidade e do custo com o aumento do número $\boldsymbol{n}$ de obstáculos ($O(\boldsymbol{n}^2)$) [23].

Em [21] os autores abordam o planejamento do caminho utilizando marcos visuais artificiais, inseridos no ambiente de navegação. Durante uma fase inicial o robô visita todos os marcos e constrói um grafo de visibilidade. Esse grafo será continuamente atualizado durante a navegação. Para cada ramo do grafo é atribuída uma probabilidade, tornando-o um grafo ponderado. Os autores resolvem o problema do menor caminho esperado usando dois algoritmos baseados no Processo de Decisão de Markov (MDP). O resultado obtido é comparado com aqueles encontrados usando o algoritmo de Dijkstra.

Em [22] os autores apresentam um algoritmo de planejamento de caminho em ambientes parcialmente desconhecidos baseados no método *Network Simplex* [24] Comparados com outros métodos de grafos, esse processo dá menos importância aos detalhes dos obstáculos, utilizando como vértices do grafo características do próprio ambiente. O método é similar ao algoritmo D* [25, 26], que permite mudar o custo do ramo durante a navegação do robô, mas difere nos seguintes pontos: a discretização não é feita por quadrículas e não existe a necessidade de se resolver o problema do menor caminho para cada possível ponto de partida. No método proposto, o problema de menor caminho é analisado como um fluxo de custo mínimo e é resolvido pelo método *Network Simplex*. Se não for levado em consideração, o método pode falhar no caso onde o terreno é bem plano e com poucos obstáculos, pois, podem ocorrer caminhos com o mesmo custo e o algoritmo escolherá um arbitrariamente. Isto pode causar problemas para o governo do robô. Outro problema está no fato de se trabalhar com o ambiente discretizado, que pode levar a duas situações: a primeira, na qual dentro do grafo gerado não existe um caminho livre para atingir o objetivo, mas no ambiente real ele existe e é realizável; e a segunda, na qual o menor caminho disponível não é um ramo do grafo, levando o robô a realizar um caminho mais longo.

Em outros artigos [8, 27] são usados o método do diagrama de Voronoi. Este tipo de diagrama fornece uma rede de caminhos bem clara e segura no ambiente de navegação. A rede é construída determinando-se caminhos que sejam eqüidistantes a dois obstáculos do plano. A dificuldade surge em ambientes amplos, onde os diagramas resultantes são extensos e difíceis de construir e armazenar.

O fato de que a melhor trajetória envolve a determinação do menor percurso sem colisão conduz naturalmente à formulação de um problema de otimização.

Na área da inteligência artificial os AGs vêm sendo usados como poderosa ferramenta de otimização, o que levou a alguns pesquisadores a propor soluções usando o AG , redes neurais e combinações desses métodos [14, 28, 29, 30, 31].

Em [14] os autores resolvem o problema do planejamento do caminho, introduzindo a figura dos pontos de guinada (*waypoints*), que servem como um objetivo local ao longo do caminho. A localização dos *waypoints* é obtida utilizando-se diagrama de Voronoi. Para a navegação do robô foi utilizado o método de campos potenciais, com otimização de seus parâmetros por meio de um AG. O algoritmo seleciona os melhores parâmetros para os campos potenciais devido aos obstáculos (repulsivo), ao destino e a cada um dos *waypoints* (atrativo).

Uma fragilidade do método, é que existem diversos *waypoints* que são obtidos no início do problema, mas que durante o desenrolar do algoritmo, são abandonados, não sendo usados efetivamente na navegação. Na formulação do problema, este "abandono" é necessário, pois a medida em que o robô avança, há o risco dele ser atraído por um *waypoints* pelo qual ele já passou. Com isto, dentre os parâmetros evoluídos no AG existe um que indica a distância mínima entre o robô e o *waypoints* para a qual ele poderá ser desconsiderado.

Em [28] os autores usam um AG para resolver o problema da cinemática inversa aplicam em um robô manipulador com seis graus de liberdade. Eles usam o mesmo algoritmo-base para planejar o caminho de um robô móvel. O espaço de busca é um subespaço de todas as trajetórias possíveis começando pela origem e os caminhos são diretamente codificados como sendo comandos de rotação e translação. A avaliação da aptidão do indivíduo usa o conceito de visibilidade para o destino.

Em [29], o problema de SLAM – *Simultaneous Localization and Mapping*- é apresentado como um problema de otimização global, cujo objetivo é a busca do espaço de todas as trajetórias possíveis. O autor emprega um algoritmo genético, no qual a população de possíveis soluções evolui de forma a atingir a solução ótima.

Os dados hodométricos do robô são usados como modelo para o qual uma população inicial de possíveis trajetórias é randomicamente gerado. Cada uma dessas trajetórias é então avaliada, construindo um mapa de ocupação global, usando os dados registrados pelo sensor de distância do robô ao longo da trajetória. A aptidão é calculada para cada candidato a solução, baseado na consistência e na compactação do mapa produzido.

A busca é conduzida no espaço de possíveis trajetórias, que pode ser definida como sendo um vetor $[\delta_1\alpha_1, \delta_2\alpha_2, \ldots\ldots\ldots\ldots \delta_n\alpha_n]$, onde δ e α são respectivamente a distância relativa e a rotação experimentada pelo robô em um *step* j. Os dados de hodometria são usados para gerar candidatos a solução, aplicando diferentes fatores de correção aos valores de δ e α medidos. A solução candidata é codificada como um vetor de fatores de correção. A trajetória é dividida em M segmentos. Uma possível solução consiste do vetor $[\Delta\delta_1\alpha_1, \Delta\delta_2\alpha_2, \ldots\ldots\ldots\ldots \Delta\delta_m\alpha_m]$, onde $\Delta\delta_k\alpha_k$ são os fatores de correção aplicados à distância e à rotação medidas em um segmento k.

Em [30], os autores descrevem o uso de uma rede neural recorrente para controlar um robô móvel. O planejamento resultante é baseado no uso de um mapa topográfico armazenado internamente, que permite ao robô escolher a trajetória apropriada como função de sua localização e da quantidade de energia disponível na bateria.

O procedimento evolucionário empregado consiste em aplicar o algoritmo genético aos valores de sinapses da rede neural que controla o robô. Os valores das sinapses foram individualmente codificados como sendo números em ponto flutuante. Cada cromossomo da população tem o mesmo tamanho, correspondente ao número de conexões de sinapses.

Em [31] os autores desenvolvem um AG para realizar o planejamento do caminho de um robô em um ambiente de navegação estático, cujo mapa é conhecido. O ambiente é divido em células de mesmas dimensões formando uma grade. Nas posições onde existem obstáculos, a célula e marcada como estando ocupada. As células livres são aquelas onde não existem obstáculos. As coordenadas dos pontos de partida e do ponto de chegada são conhecidas. O robô é assumido como sendo pontual e pode se mover, sobre a linha que une o centro das células, por todas as células livres. O robô segue pelo menor caminho possível até que se defronte com algum obstáculo. Quando isso ocorre, ele passa a se mover por meio de deslocamentos verticais (*row-wise*) e horizontais (*column-wise*). O robô é posto para se deslocar em ambientes gerados por computador e para cada ambiente proposto, o algoritmo é executado quinze vezes visando à determinação da taxa de sucesso, que vem a ser a relação entre quantidade de vezes em que o algoritmo gerou trajetórias válidas e a quantidade em que foi executado.

4 - Determinação de trajetória usando AG

4.1 - Definição do problema

Realizar o planejamento do caminho de um robô e, por conseguinte, sua trajetória, pode ser considerado um problema de otimização, pois visa a realização de um deslocamento seguro para o robô em um ambiente com obstáculos e, simultaneamente, fazer com que o percurso cumprido seja o mínimo possível, economizando tempo e energia.

Assim, o escopo do problema que se pretende solucionar é:

Fazer com que um robô móvel, dotado de sensores que permitam avaliar a sua distância a obstáculos e capaz de determinar sua posição no ambiente de navegação, seja capaz de poder atingir um ponto de destino dentro de um ambiente com obstáculos, sem colidir com qualquer um deles realizando o menor caminho possível, empregando para realizar o planejamento de caminho um AG.

A idéia do algoritmo proposto é determinar um conjunto de coordenadas no ambiente de deslocamento que serviriam de pontos de guinada na trajetória do veículo. Assim, a trajetória seria constituída por uma série de segmentos, definidos entre esses pontos de guinada.

Para a consecução deste objetivo, são assumidas as seguintes premissas:

- O ambiente de navegação é conhecido. Os obstáculos são estáticos e sua localização é aproximadamente conhecida;
- O ponto designado como sendo o de destino está dentro da área de alcance dos sensores;

- Os sensores do robô atuam radialmente;
- O robô é capaz de estimar sua própria posição.

A figura 4.1 mostra um possível ambiente de navegação, por onde o robô deve se deslocar. O triangulo azul no canto superior esquerdo é o ponto de partida. Os pontos pretos representam os centros dos obstáculos e as circunferências que os circundam indicam o limite máximo de aproximação entre o robô e o obstáculo. O ponto vermelho no canto inferior direito é o ponto de chegada.

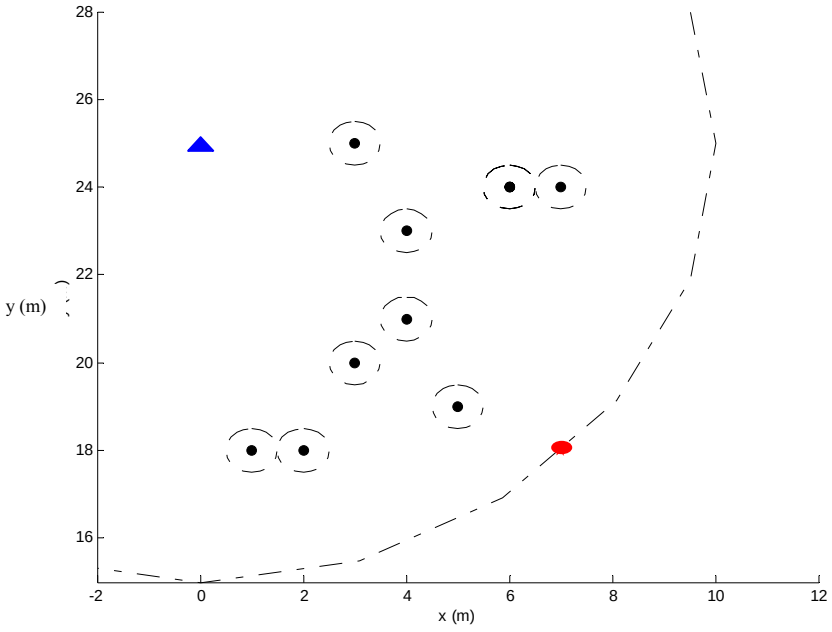


Figura 4.1 - Cenário hipotético de navegação com obstáculos.

4.2 - Algoritmo proposto

As características principais do algoritmo são as seguintes:

- A busca será conduzida no espaço de possíveis trajetórias do robô. Cada trajetória é definida pelo segmento de reta compreendido entre dois

pares ordenados (***x,y***), onde ***x*** e e ***y*** são as coordenadas do pontos de guinada;

- As soluções candidatas são codificadas como *strings* binárias;
- Uma função de avaliação é usada para verificar a qualidade das soluções candidatas.

4.3 - Implementação do algoritmo proposto

A seguir são descritos em detalhes os elementos usados na implementação do AG proposto.

4.3.1 - População inicial

Ponto de partida do algoritmo genético, a POPULAÇÃO INICIAL é formada por um número predeterminado de indivíduos (***I***), que representam possíveis soluções para o problema. Na presente dissertação, a população de indivíduos consiste de caminhos completos, ligando os pontos de partida e de destino do robô. A figura 4.2 demonstra uma população inicial com 10 indivíduos.

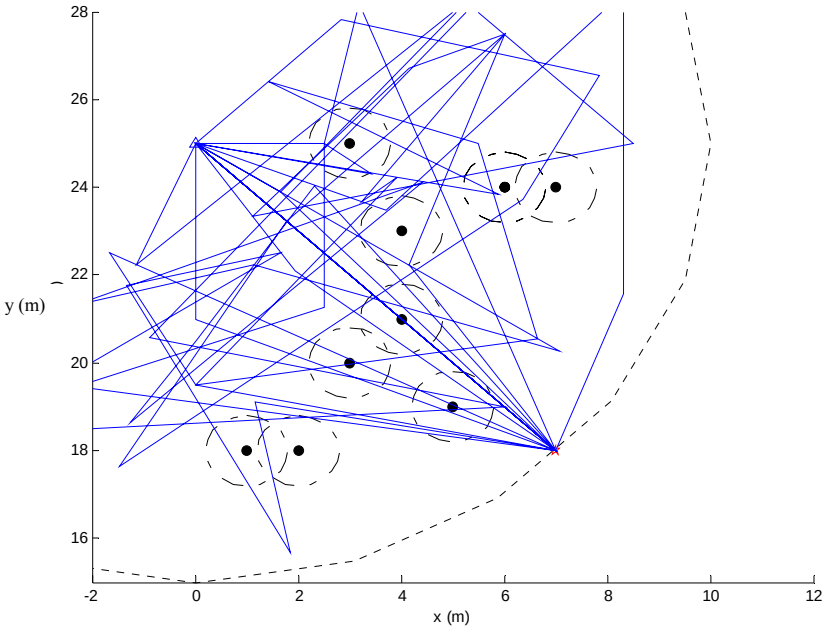


Figura 4.2– Todos os indivíduos de uma população inicial no ambiente de navegação

Os GENES dos indivíduos representam pares ***(x,y)*** no ambiente de navegação, por onde o robô deve se passar. Cada gene é responsável, portanto, por um trecho da trajetória, ou como será chamado neste trabalho, um SEGMENTO.

Dada a natureza do problema, não é conveniente estipular uma mesma quantidade fixa de genes por indivíduos, pois, dependendo da quantidade e da disposição dos obstáculos, o robô pode realizar uma trajetória com maior ou menor quantidade de segmentos.

Um ambiente com poucos obstáculos ou com obstáculos muito afastados um dos outros permite realizar o percurso do robô praticamente sem desvios, levando a uma trajetória com poucos segmentos. Por outro lado, uma seqüência desfavorável de obstáculos pode resultar em vários desvios, aumentando a quantidade de segmentos.

Assim, cada indivíduo da população inicial terá uma quantidade de segmentos sorteada entre um e dez.

A figura 4.3 mostra um indivíduo qualquer da população, que contem N segmentos.

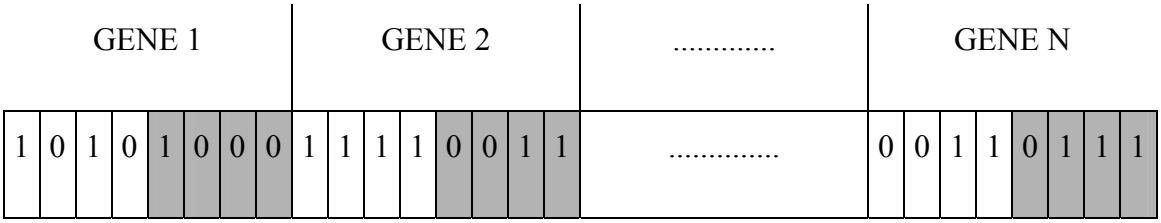


Figura 4.3 – Esquema de codificação do cromossomo

Cada gene está composto por 8 bits, sendo que os primeiros quatro bits são usados para determinar a coordenada em x e os outros quatro a coordenada y.

O primeiro segmento da trajetória é determinado ligando o ponto de partida ao ponto (x_1,y_1), obtido a partir da decodificação do gene1. O segundo segmento é aquele compreendido entre (x_1,y_1) e o ponto (x_2,y_2), obtido a partir da decodificação do gene2 e assim por diante até atingir o enésimo gene, correspondendo ao enésimo segmento, que termina no ponto de destino.

Um exemplo de população hipotética composta por cinco indivíduos, contendo um mínimo de um segmento e um máximo de cinco segmentos, é apresentada na tabela 4.1. O primeiro indivíduo tem três segmentos e, por conseguinte, três pontos de guinada. O segundo ficou com cinco segmentos e, portanto tem cinco pontos de guinada.

Tabela 4.1– Indivíduos de uma população inicial hipotética

indivíduo	População	Qtd de segmentos
1	10001111001101101000101 1	3
2	10001011010011011001101000000100 10001000	5
3	11000001110100110010110000101001	4
4	00001000110010111 1001011	3
5	00000101000010001111001100011110	4

Convertendo-se a população acima, obtém-se os pares ordenados de todos os pontos de guinada para cada indivíduo.

A tabela 4.2 resume os pontos de guinada para a população considerada. A quantidade de pares ordenados corresponde à quantidade de segmentos.

Tabela 4.2– Pontos de guinada para a população considerada

indivíduo	Ponto inicial	Pontos de guinada	Ponto final
1	(0,25)	(-3.3334 20.0112) (3.2336 23.6606) (1.1705 19.1153)	(7,18)
2	(0,25)	(1.1705 19.1153) (-0.7804 21.0769) (2.4874 18.9948) (2.0000 25.0000) (8.0000 25.0000)	(7,18)
3	(0,25)	(6.6518 29.4446) (8.3367 26.6583) (0.0000 22.0000) (1.6667 22.5056)	(7,18)
4	(0,25)	(1.4142 23.5858) (1.5607 17.1537) (1.5607 17.1537)	(7,18)
5	(0,25)	(1.9616 24.6098) (1.4142 23.5858) (9.3175 26.8534) (-0.9567 22.6903)	(7,18)

4.3.2 - Verificação da aptidão do indivíduo

A otimização envolvida é multi-objetivo, pois se deseja não somente minimizar o percurso, mas também evitar que o robô colida ou se aproxime a menos de um determinado limite dos obstáculos. Assim, são duas as grandezas consideradas para se avaliar a aptidão de um determinado indivíduo.

O algoritmo proposto usa a informação de comprimento da trajetória gerada (***l***) e da quantidade de vezes (***q***) em que a trajetória gerada impacta ou se aproxima dos obstáculos.

A quantidade e a disposição dos obstáculos, bem como a mínima distância permitida entre o robô e o obstáculo podem levar a uma maior dificuldade na geração do percurso.

Tome-se como exemplo, as figuras 4.4 e 4.5. Na primeira a distância mínima entre os segmentos de reta e os obstáculos é de 0,5 unidade de comprimento (uc). Na segunda essa distância foi de 1 unidade de comprimento. Percebe-se que na segunda, as possibilidades de se obter trajetórias mínimas e livre de colisão são menores do que na primeira.

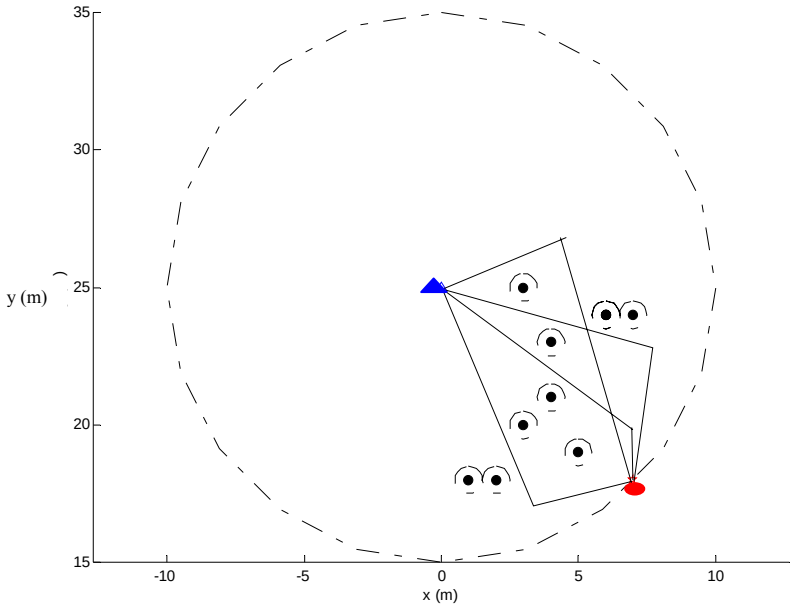


Figura 4.4– caminhos possíveis quando a distância mínima é de 0,5 uc

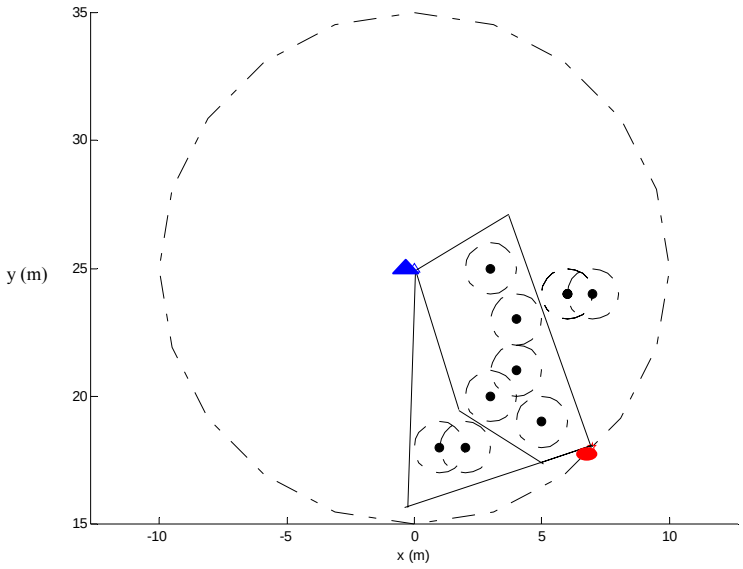


Figura 4.5– caminhos possíveis quando a distância mínima é de 1 uc

Uma forma de se avaliar a aptidão de um indivíduo é obter o seu CUSTO [2]. O custo do indivíduo pode ser entendido como sendo a diferença entre o que se desejava obter e o que foi efetivamente obtido. A aptidão do indivíduo é a função inversa do custo, ou seja, quanto menor o custo maior a aptidão.

No levantamento do custo do individuo é conveniente realizar a normalização e/ou a ponderação das variáveis envolvidas. A sugestão para a normalização de uma variável qualquer ***V*** é confrontá-la com os valores mínimo e máximo que ela pode assumir. A equação normalizada e ponderada para a variável ***V*** é mostrada na equação 4.1.

$$f(\boldsymbol{V}) = \boldsymbol{w}\frac{V - V_{\min}}{V_{\max} - V_{\min}} \tag{4.1}$$

Onde os subscritos ***min*** e ***max*** se referem aos valores mínimos e máximos que a variável ***V*** pode assumir e w é o fator de ponderação. Usando a idéia de normalização e de ponderação, o custo proposto para o indivíduo considerado nesta dissertação é a soma das parcelas devido a cada variável do problema de otimização. A equação 4.2 apresenta a função custo estabelecida para o problema considerado neste trabalho.

$$\boldsymbol{Custo(q,l)} = w_q \frac{q - q_{\min}}{q_{\max} - q_{\min}} + w_l \frac{l - l_{\min}}{l_{\max} - l_{\min}} \tag{4.2}$$

Onde $\boldsymbol{q_{min}}$, $\boldsymbol{q_{max}}$, $\boldsymbol{l_{min}}$ ***e*** $\boldsymbol{l_{max}}$ se referem aos valores mínimo e máximo da quantidade de impactos (***q***) e ao comprimento da trajetória (***l***).

Para se determinar os valores mínimos e máximos dessas variáveis, o algoritmo foi executado diversas vezes, da seguinte forma: gerava-se um grupo de indivíduos e dentro deste grupo verificava-se qual o comprimento máximo e qual a quantidade máxima de vezes em que houve colisão ou aproximação dos obstáculos. Foi observado experimentalmente que a maior quantidade de impactos ou proximidade (***q***) foi 30 e que os maiores caminhos tinham 60 unidades de comprimento. As quantidades mínimas para essas duas variáveis são $\boldsymbol{q_{min}}$=0 e $\boldsymbol{l_{min}}$=R, onde R é o comprimento da reta que une o ponto de partida ao ponto de chegada.

Uma vez que se deseja trajetória livre de colisões (q=0), o custo do indivíduo deve ser tão mais alto quanto maior for o valor de q. Por essa razão, foi escolhido um fator de ponderação w_q alto. Por outro lado, quanto menor for o comprimento da trajetória (l) menor deve ser o custo do indivíduo. Assim, o valor de w_l deve ser escolhido de forma a resultar em um custo pequeno além de ser capaz de promover uma variação contínua do custo, visando a fornecer uma boa separação entre esses valores. Os fatores de ponderação escolhidos foram $w_q = 4$ e $w_l = 0{,}8$.

Esses valores resolvem o problema proposto e determinam bem a importância das duas heurísticas da função custo.

Assim, a função custo utilizada pelo algoritmo, considerando $l_{min} = 10$ e $q_{min} = 0$, é a apresentada na equação 4.3:

$$custo(q,l) = 4\frac{q}{30} + 0{,}8\frac{l-10}{50} \tag{4.3}$$

A aptidão é escrita como sendo a função inversa do custo, que também deve ser normalizada :

$$f_k = \frac{1}{custo + 1} \tag{4.4}$$

A figura 4.6 mostra a representação gráfica da aptidão em função dos valores de q e l. Verifica-se que os valores de aptidão são maiores para o caso em que $q = 0$, e l está mais próximo de dez (l_{min}). A aptidão máxima é obtida quando $q = 0$ e $l = 10$.

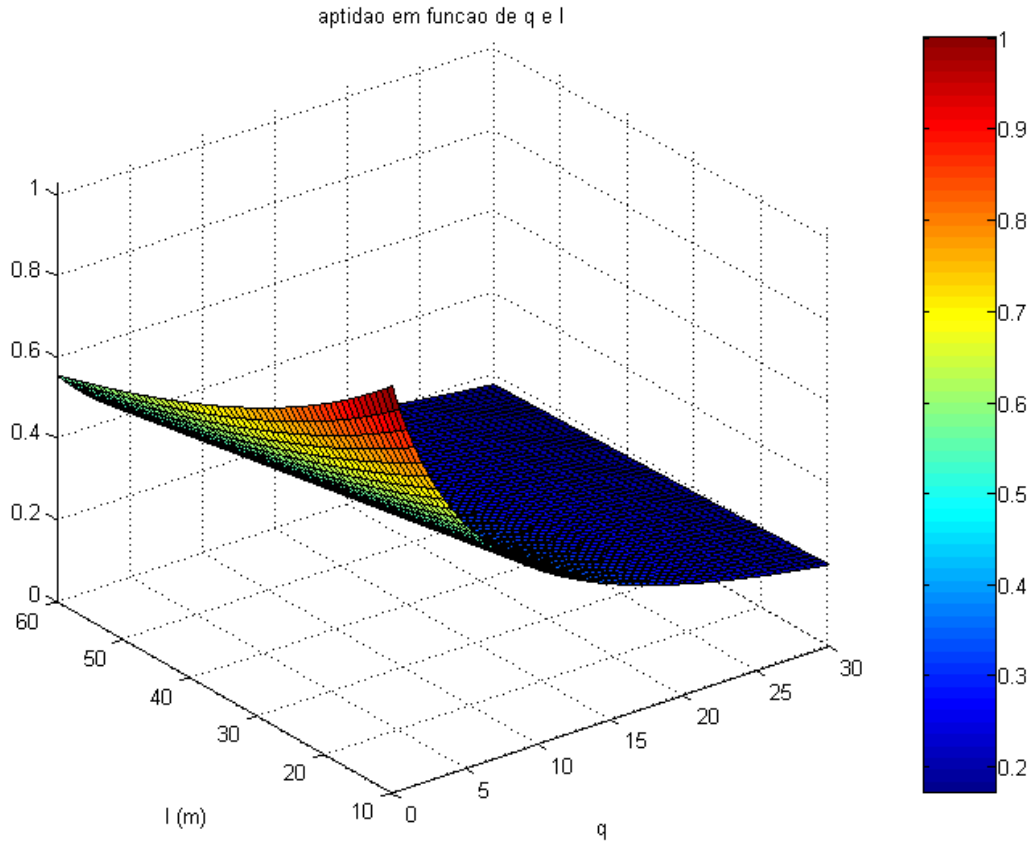


Figura 4.6– aptidão em função de *q* e *l*.

A tabela 4.3 apresenta alguns valores de aptidão valores de ***q*** e ***l***, considerando a função das equações 4.3 e 4.4. Verifica-se, por exemplo, que para um dado ***q***, o valor de aptidão decresce de forma progressiva. Verifica-se ainda, que quando q é diferente de zero, a maior aptidão não supera o valor de 0,882.

Tabela 4.3 – valores de aptidão para alguns valores de q e l.

q	***l***	custo	aptidão
0	10	0	1
	11	0,016	0,984
	12	0,033	0,968
	14	0,065	0,939
	18	0,128	0,886
	20	0,160	0,862
1	10	0,133	0,882
	11	0,149	0,870
	12	0,165	0,858
	14	0,197	0,835
	18	0,263	0,792
	20	0,293	0,773
2	10	0,267	0,789
	11	0,283	0,779
	12	0,299	0,770
	14	0,331	0,751
	18	0,394	0,717
	20	0,429	0,7

4.3.3 - A seleção dos indivíduos e as operações genéticas

Para a seleção dos indivíduos foram utilizados dois métodos tradicionalmente empregados nos AG. O primeiro método de seleção empregado foi o SUS, no qual a roleta é construída tomando-se por base os valores de aptidão dos indivíduos, os quais definem proporcionalmente cada uma das seções da roleta. A amostragem é feita utilizando-se apenas um giro da roleta. A APTIDÃO RELATIVA de cada um dos indivíduos é a proporção existente entre a aptidão individual e a aptidão total da população, que vem a ser a soma de todas as aptidões de indivíduos. Para exemplificar como os

indivíduos são selecionados, considere a população hipotética de dez indivíduos com as aptidões apresentadas na tabela 4.4.

Tabela 4.4– Aptidão e aptidão relativa da população em uma dada geração

Indivíduo	Aptidão	Aptidão relativa
1	0,5828	11,58%
2	0,4235	8,42%
3	0,5155	10,24%
4	0,3340	6,64%
5	0,4329	8,60%
6	0,2259	4,49%
7	0,5798	11,50%
8	0,7604	15,10%
9	0,5297	10,53%
10	0,6405	12,70%
Total	**5,0295**	**100%**

Os dados acima permitem construir, com base na aptidão relativa, os setores da roleta, apresentados na figura 4.7. A posição do primeiro ponteiro é aleatoriamente gerada. Os demais ponteiros são levantados somando-se a unidade ao valor do primeiro ponteiro.

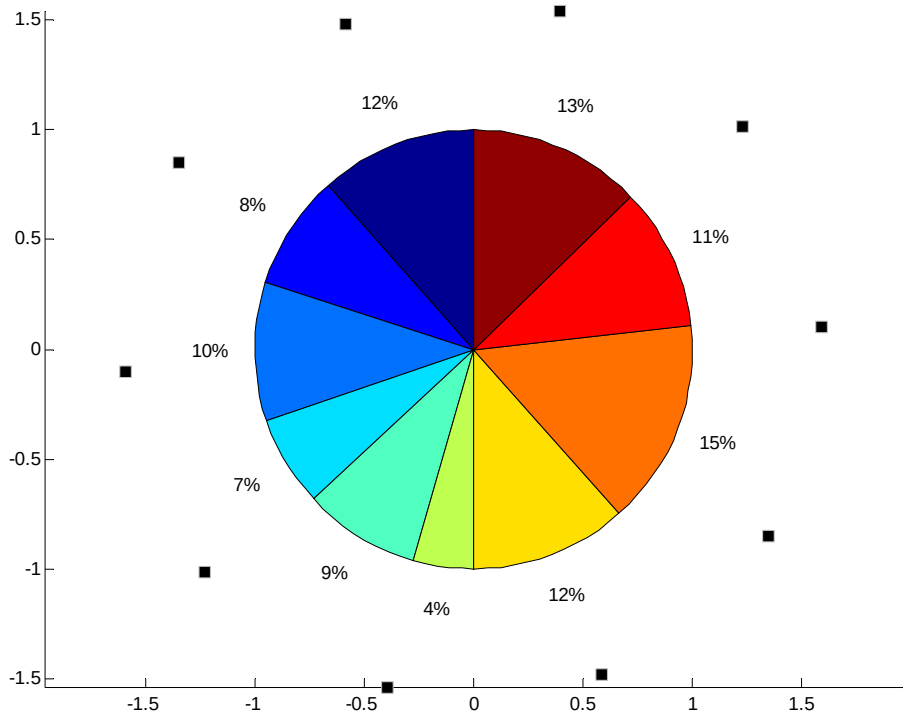


Figura 4.7– Seleção por roleta. Os quadrados pretos são os ponteiros.

Um indivíduo será selecionado tantas vezes quantas forem a quantidade de ponteiros dentro do setor correspondente. Se em um dado setor não houver ponteiro, o indivíduo correspondente não será selecionado.

Após a seleção pela roleta, a população selecionada realiza as operações de *crossover* e de mutação.

O segundo método de seleção escolhido foi o do torneio. Neste método, a população é ordenada por ordem decrescente de aptidão (*ranking*). A metade mais apta é separada para a realização dos torneios. Os torneios ocorrem da seguinte forma: a partir dos indivíduos da metade mais apta da população, são gerados subconjuntos de tamanho fixo (de modo geral, o tamanho do subconjunto é 2 [2]) Os elementos desses subconjunto são selecionados aleatoriamente. Os dois indivíduos realizam o torneio e vence aquele que tiver maior aptidão ou aquele que, segundo um dado critério,

ultrapassa um valor estabelecido com limite. O vencedor do torneio passa a ser considerado um dos genitores.

Serão realizados tantos torneios quantos forem os necessários para se recompor a quantidade de indivíduos da população original. A tabela 4.5 mostra como é empregado o método, considerando a mesma população da tabela 4.4. A população se encontra ordenada e a metade menos apta (destacada) é abandonada.

Tabela 4.5– população ordenada pela aptidão

Indivíduo	Aptidão
8	0,7604
10	0,6405
1	0,5828
7	0,5798
9	0,5297
3	0,5155
5	0,4329
2	0,4235
4	0,3340
6	0,2259

Para a população considerada, são sorteados cinco subconjuntos de dois elementos para a realização dos torneios. Serão realizados cinco torneios, de forma a se obter três casais de genitores, que gerarão cinco descendentes, recompondo a quantidade de indivíduos da população. Os subconjuntos sorteados para a realização do torneio, bem como os genitores resultantes, estão resumidos na tabela 4.6.

Tabela 4.6– Levantamento dos genitores na seleção por torneio

Torneio	Subconjuntos sorteados	genitores
1	(8,9)	8
2	(7,7)	7
3	(8,10)	8
4	(9,1)	1
5	(7,10)	10

Ao final, os casais de genitores serão (8,7); (8,1); e (1,10). Por meio do processo de *crossover*, cada um dos casais gera dois filhos. Um deles será abandonado para se recompor a população de dez indivíduos.

A população da próxima geração será composta pelos cinco genitores e pelos cinco filhos gerados. Por fim a população sofre a mutação.

4.3.4 - Critério de parada

O critério de parada do algoritmo foi estipulado como sendo o valor da aptidão. O AG é colocado para evoluir até que o valor da aptidão máxima supere 0,98. Para um $\boldsymbol{q}$ nulo, este valor de aptidão ocorre com um comprimento de cerca de 11,2 unidades de comprimento.

5 - Testes e resultados obtidos

Quando se trabalha com AG, algumas considerações com relação aos parâmetros do algoritmo devem ser feitas. A análise mais detalhada da quantidade de indivíduos, taxa de *crossover*, taxa de mutação e da forma de seleção podem levar a resultados melhores, mais estáveis e com menor tempo de execução.

Para uma quantidade fixa de gerações, por exemplo, uma população inicial composta por 10 indivíduos será executada mais rapidamente do que uma com 80 indivíduos. Porém, a solução obtida com 10 indivíduos pode não ser tão boa quanto a obtida com os 80 indivíduos.

Analogamente, uma taxa de mutação muito baixa pode não produzir a diversidade necessária na população para evitar a convergência prematura para um mínimo local. Por outro lado, uma taxa muito alta pode comprometer a herança genética da população.

Para determinar os valores adequados destes parâmetros do AG para a classe de problemas estudados, foram executados vários testes, com várias combinações de tamanhos da população, taxas de *crossover* e mutação, tanto no algoritmo utilizando a seleção por roleta quanto na variante utilizando o torneio. O intuito foi o de verificar as combinações que permitem obter maior eficiência do algoritmo na solução do problema de trajetória ótima, isto é, menor comprimento da trajetória e menor tempo de execução.

As rotinas foram implementadas em MATLAB 6.1 e todos os testes foram realizados em um computador pessoal, com processador ATHLON XP 2000 (1,25 GHz) com 256 MB de memória RAM.

5.1 – Avaliação do algoritmo

Visando a avaliação do algoritmo, foi proposto o ambiente de navegação mostrado na figura 5.1, que tem as seguintes características: nove obstáculos; menor distância entre o ponto inicial e o ponto final é de 10 metros; e a distância mínima de aproximação entre o robô e os obstáculos é de 0,5 m.

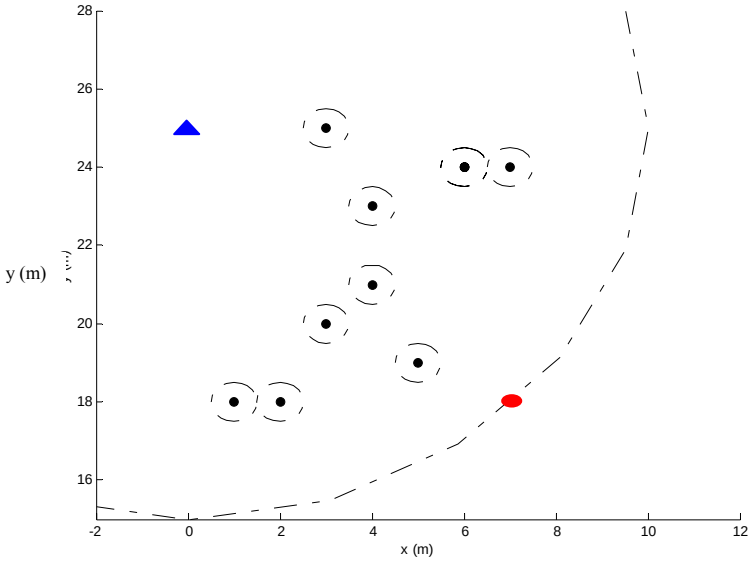


Figura 5.1 - Ambiente de navegação utilizado nos testes.

O objetivo é fazer o algoritmo evoluir até superar o valor de aptidão máxima de 0,98. Este valor foi escolhido para fazer o algoritmo evoluir até obter uma trajetória livre de impactos e bem próxima da menor possível. Caso a população não atinja o valor de aptidão determinado, o algoritmo para ao atingir 1000 gerações.

O menor tamanho da população foi estabelecido em 10 indivíduos, sendo progressivamente aumentado para se verificar a influência desse parâmetro na resposta desejada.

Foi verificado experimentalmente que, para um tamanho de população da ordem de 10 indivíduos, os resultados obtidos com valores de mutação de 1%, resultam em uma rápida estagnação do algoritmo indicada pela pequena variação da aptidão média e pela estagnação da aptidão máxima logo nas primeiras gerações, como é mostrado na Fig. 5.2. O exemplo mostrado nesta figura corresponde a uma população de dez indivíduos e uma taxa de mutação de 1%. A trajetória resultante, mostrada na figura 5.3 ficou com um comprimento de 12,10 metros. Portanto, a pouca diversidade da população força o algoritmo a convergir precocemente para um mínimo local.

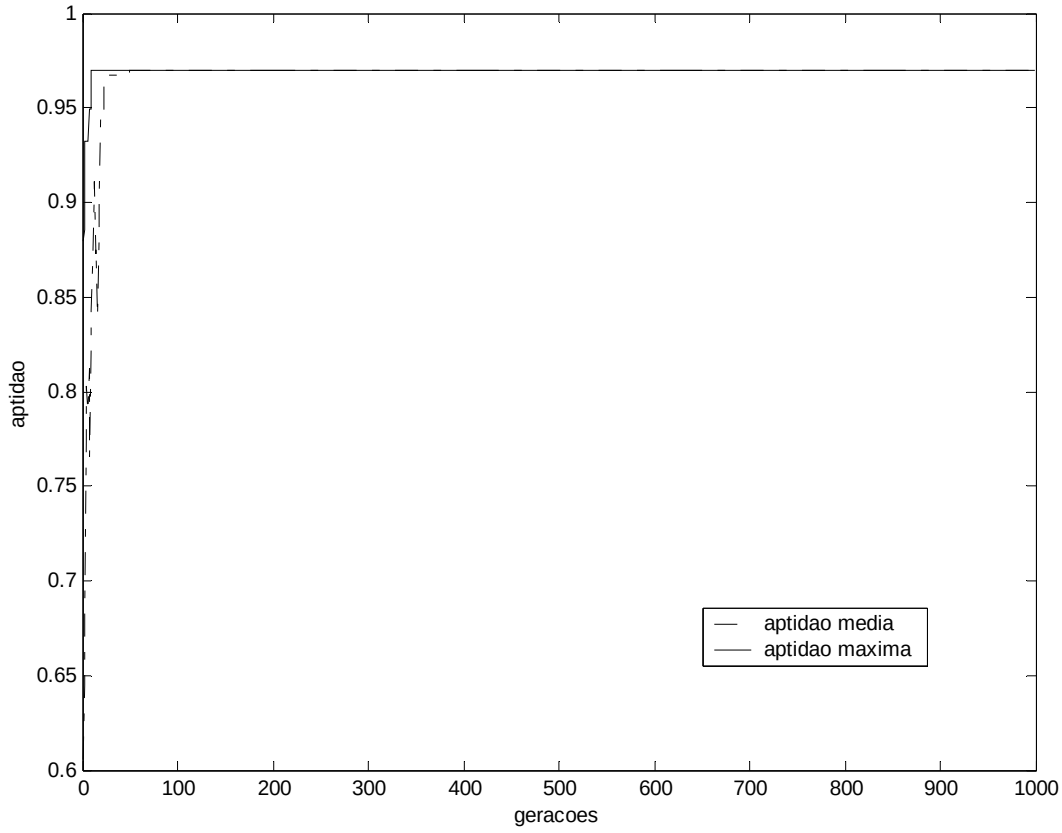


Figura 5.2 – Exemplo de convergência para mínimo local, devido a pequena população e baixa taxa de mutação.

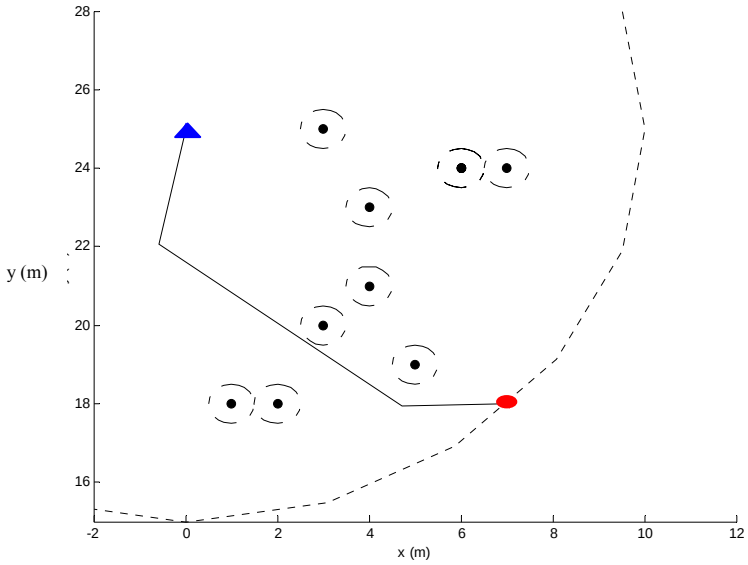


Figura 5.3 – Trajetória resultante com pequena população e baixa taxa de mutação.

Outra característica importante a ser considerada no AG é o uso ou não do elitismo. Foi observado que sem o elitismo, a solução final apresentava trajetórias visivelmente mais longas. Nas figuras 5.4 e 5.5 são mostradas, respectivamente, uma trajetória e a aptidão correspondente usando a seleção por roleta sem elitismo.

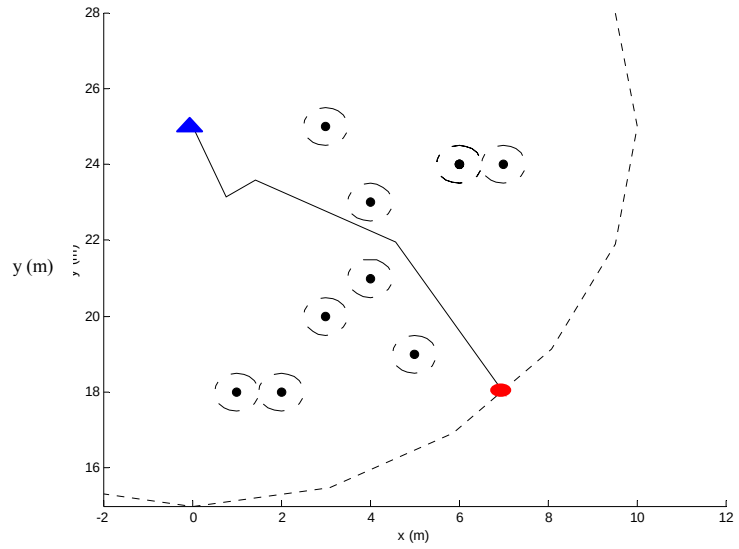


Figura 5.4 – Trajetória resultante de seleção sem elitismo

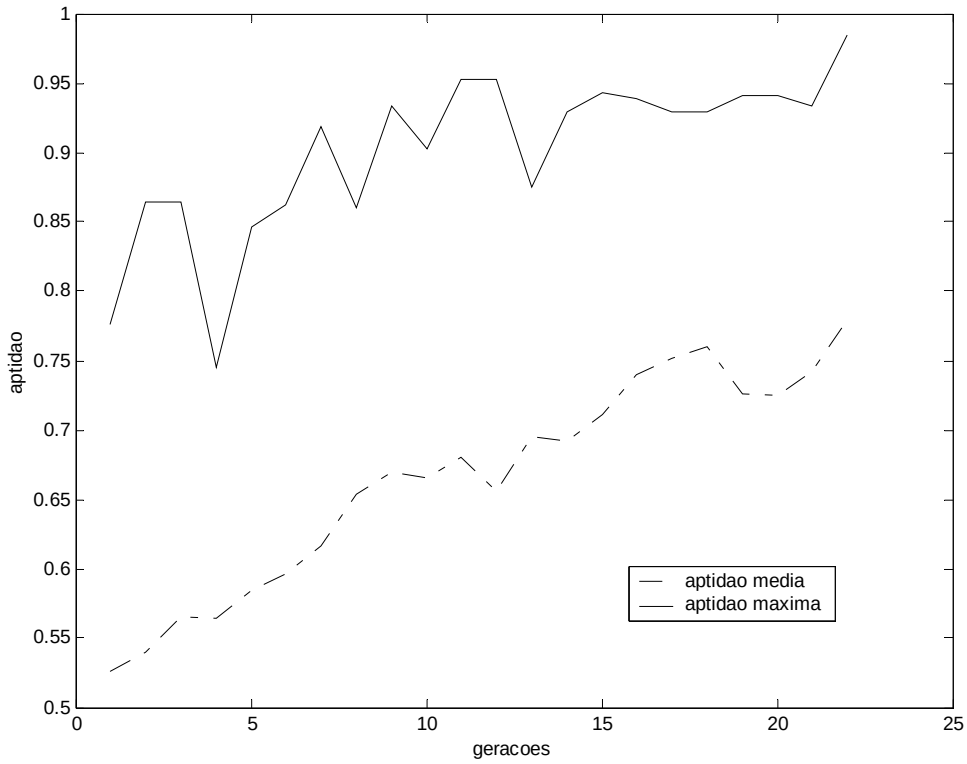


Figura 5.5 – Aptidão máxima e média resultante de seleção sem elitismo.

Assim, foram escolhidos valores da taxa de mutação variando em função do tamanho da população inicial. A tabela 5.1 mostra a combinação dos valores escolhidos para a realização dos testes.

Tabela 5.1 - Valores empregados no algoritmo para teste com seleção por roleta.

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação
10, 15, 20, 25 e 50	40%	10%
		15%
	60%	10%
		15%
	80%	10%
		15%

Os testes foram realizados com a seleção por roleta e com torneio e ambos com elitismo. Contudo, cabe ressaltar que como no método de seleção por torneio tendo em vista que os genitores selecionados permanecem na população a taxa de *crossover* é 50% e neste caso apenas a taxa de mutação é variada. Além disso, o tamanho da população para esse tipo de seleção foi escolhido de forma ligeiramente diferente, sendo sempre par. A tabela 5.2 resume os valores usados.

Tabela 5.2 - Valores empregados no algoritmo para teste com seleção por torneio.

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação
12, 16, 20, 24,52 e 64	50%	10%
		15%
		10%
		15%
		10%
		15%

Para cada conjunto proposto de parâmetros, o algoritmo foi executado oito vezes, permitindo avaliar a estabilidade da solução obtida em função da geração aleatória da população inicial. As tabelas que se seguem apresentam de forma resumida os valores médios e os melhores valores para cada configuração e método de seleção.

Tabela 5.3– Valores obtidos para 10 indivíduos com seleção por roleta

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
			Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
10	40%	10%	10,97	10,15	0,984	0,998	272,12	23	2,88	2	7,54	0,64
		15%	11,00	10,88	0,984	0,986	79,50	3	2,75	2	2,17	0,25
	60%	10%	10,56	10,13	0,991	0,998	174,37	10	2,75	2	5,32	0,34
		15%	10,63	10,05	0,990	0,999	64,25	2	2,75	2	2,12	0,09
	80%	10%	10,93	10,02	0,985	0,999	296,25	7	2,50	2	11,62	0,25
		15%	10,06	10,15	0,990	0,998	59,125	10	2,63	2	1,62	0,36

Tabela 5.4– Valores obtidos para 15 indivíduos com seleção por roleta

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
			Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
15	40%	10%	11,12	10,57	0,982	0,990	59,75	5	127,62	2	12,04	0,31
		15%	10,81	10,15	0,988	0,998	21,38	3	2,37	2	0,95	0,20
	60%	10%	10,88	10,25	0,986	0,996	31,75	2	2,38	2	1,27	0,14
		15%	10,85	10,15	0,987	0,998	31	4	2,50	2	1,29	0,30
	80%	10%	10,90	10,15	0,986	0,998	17,13	3	2,75	2	0,97	0,20
		15%	10,48	10,15	0,992	0,998	41,63	4	2,44	2	2,02	0,21

Tabela 5.5– Valores obtidos para 20 indivíduos com seleção por roleta

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
			Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
20	40%	10%	10,36	10,03	0,994	0,999	136,25	2	2,38	2	7,08	0,16
		15%	10,54	10,02	0,992	0,999	9,25	2	2,38	2	0,691	0,33
	60%	10%	10,60	10,06	0,990	0,999	34,25	2	2,75	2	2,41	0,16
		15%	10,68	10,13	0,989	0,998	14,12	2	2,25	2	0,83	0,16
	80%	10%	10,69	10,03	0,989	0,999	29	4	2,75	2	1,72	0,36
		15%	10,40	10,05	0,994	0,999	10,25	4	2,25	2	0,68	0,27

Tabela 5.6– Valores obtidos para 25 indivíduos com seleção por roleta

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
			Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
25	40%	10%	10,56	10,03	0,991	0,999	42,25	3	2,37	2	2,52	0,39
		15%	10,55	10,15	0,991	0,998	13,50	2	2,37	2	1,02	0,20
	60%	10%	10,80	10,03	0,987	0,999	38,25	3	2,50	2	2,89	0,34
		15%	10,53	10,05	0,992	0,999	8,75	2	2,63	2	0,83	0,20
	80%	10%	10,51	10,03	0,991	0,999	18,75	2	2,5	2	1,58	0,19
		15%	10,90	10,15	0,986	0,998	7,12	2	2,25	2	0,63	0,17

Tabela 5.7 - Valores obtidos para 50 indivíduos com seleção por roleta

Quantidade de indivíduos	Taxa de crossover	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
			Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
50	40%	10%	10,44	10,13	0,993	0,998	6,50	4	2,12	2	1,21	0,81
		15%	10,53	10,13	0,991	0,998	10,50	5	2,13	2	1,83	1,03
	60%	10%	10,54	10,03	0,991	0,999	5	2	2,13	2	0,93	0,33
		15%	10,71	10,15	0,989	0,997	6,25	2	2,37	2	1,19	0,34
	80%	10%	10,60	10,02	0,991	0,999	7,62	3	2,37	2	1,48	0,59
		15%	11,01	10,43	0,984	0,993	6,25	2	2,37	2	1,20	0,33

Tabela 5.8– Valores obtidos para 12 indivíduos com seleção por torneio

Quantidade de indivíduos	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
		Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
12	10%	11,33	10,15	0,979	0,998	383	2	2,5	2	12,55	0,11
	15%	10,51	10,03	0,992	0,999	4	3	2,25	2	0,199	0,14

Tabela 5.9– Valores obtidos para 16 indivíduos com seleção por torneio

Quantidade de indivíduos	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
		Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
16	10%	11,88	10,43	0,971	0,993	628	2	2,5	2	21,44	0,13
	15%	11,56	10,43	0,975	0,993	502	2	2,38	2	18,25	0,13

Tabela 5.10– Valores obtidos para 20 indivíduos com seleção por torneio

Quantidade de indivíduos	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
		Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
20	10%	11,7519	10,18	0,973	0,997	630,75	2	2,38	2	28,67	0,16
	15%	10,6429	10,05	0,989	0,999	130,87	2	2,25	2	4,39	0,14

Tabela 5.11– Valores obtidos para 24 indivíduos com seleção por torneio

Quantidade de indivíduos	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
		Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
24	10%	10,77	10,1312	0,988	0,998	6,88	2	2,5	2	0,46	0,19
	15%	10,29	10,06	0,995	0,999	4,13	2	2	2	1,52	0,17

Tabela 5.12 - Valores obtidos para 52 indivíduos com seleção por torneio

Quantidade de indivíduos	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
		Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
52	10%	10,72	10,03	0,988	0,999	11,75	3	2,37	2	1,62	0,59
	15%	10,57	10,05	0,991	0,999	8,12	3	2,11	2	1,27	0,45

Tabela 5.13 - Valores obtidos para 64 indivíduos com seleção por torneio

Quantidade de indivíduos	Taxa de mutação	Comprimento (m)		Aptidão		Gerações		Quantidade de Segmentos		Tempo execução (s)	
		Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor	Médio	Melhor
64	10%	10,64	10,02	0,989	0,999	4,37	2	2,25	2	0,83	0,39
	15%	10,28	10,05	0,995	0,999	3,87	2	2,12	2	0,79	0,40

5.2 – Discussão dos resultados

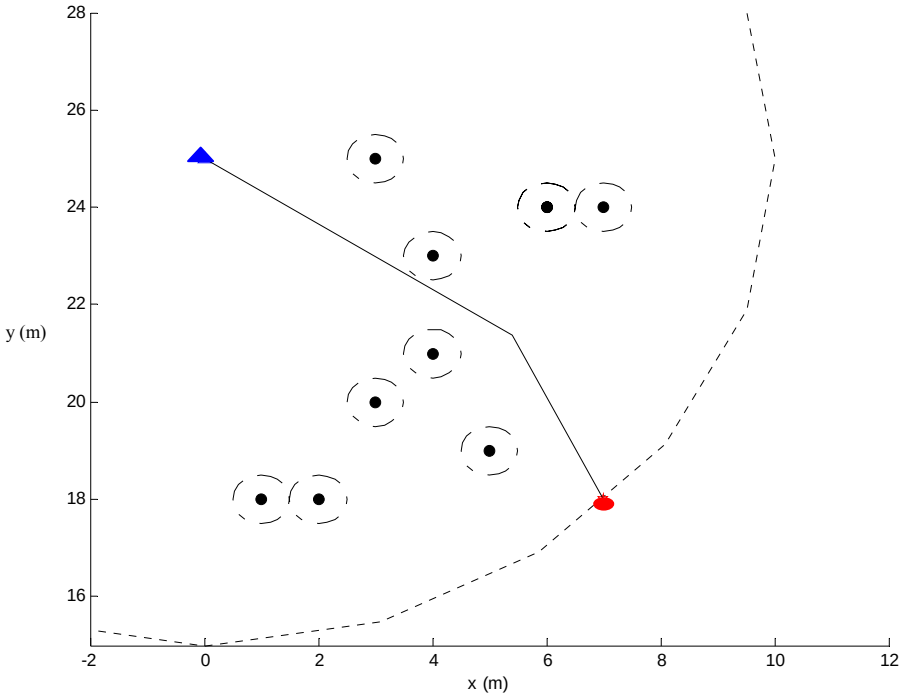
A análise dos resultados permite concluir que em ambas as formas de seleção, o tamanho da população tem influencia considerável no resultado. Observa-se que o resultado fica mais estável com o aumento do tamanho da população. Foi verificado experimentalmente que no caso da seleção por torneio, essa quantidade foi maior do que na seleção por roleta. Ainda no caso da seleção por roleta, maiores valores para a taxa de *crossover* resultaram em resultados mais estáveis, sendo que o valor de 60% foi o de melhor resultado. O melhor valor obtido para a taxa de mutação foi 10% para a seleção por roleta e 15% para a seleção por torneio.

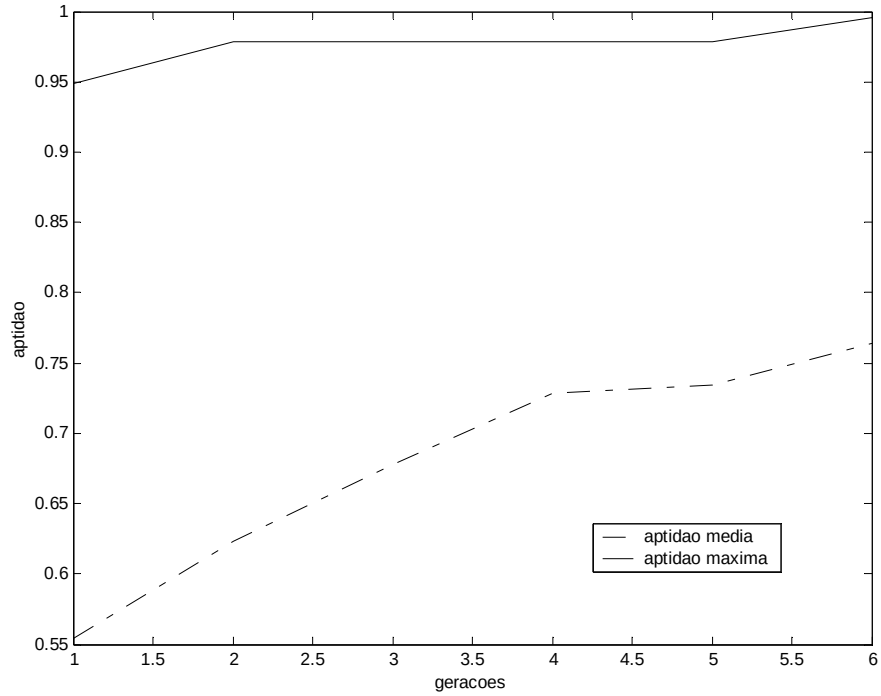
Por fim verifica-se, como previsto, que, a quantidade de segmentos da trajetória reduz-se ao longo da evolução. Para o cenário testado a quantidade média foi de 2,3 segmentos, o que reduz a quantidade de manobras do robô facilitando desta forma a navegação.

Os melhores valores de taxa de *crossover* e mutação obtidos a partir dos testes realizados, foram utilizados na solução de alguns exemplos com tamanho da população fixado em 50 indivíduos, quando a seleção foi feita por roleta, e 64 quando foi feito seleção por torneio.

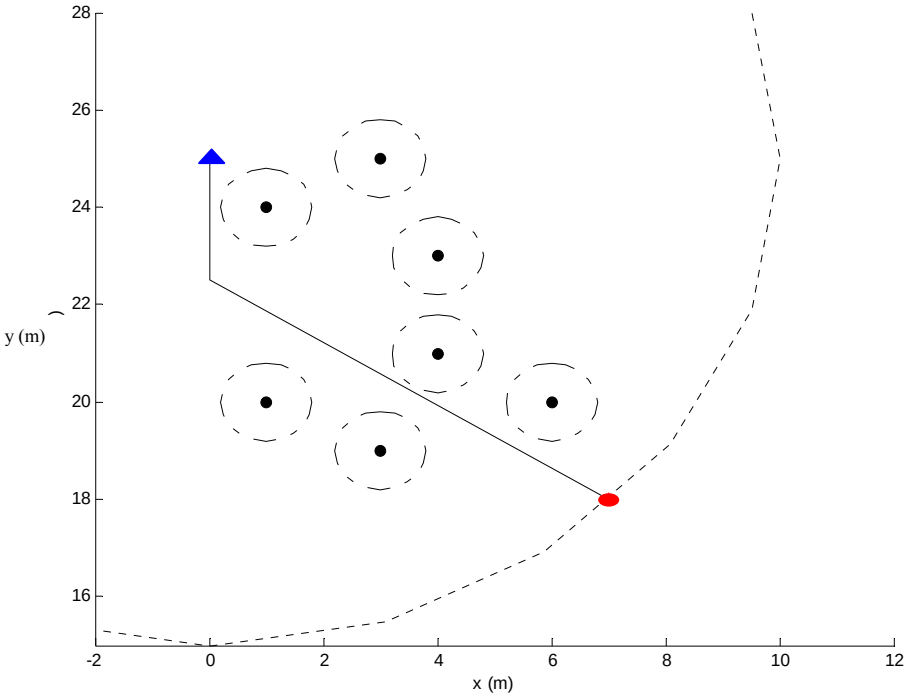
A seguir são apresentados os resultados para seleção por roleta com elitismo

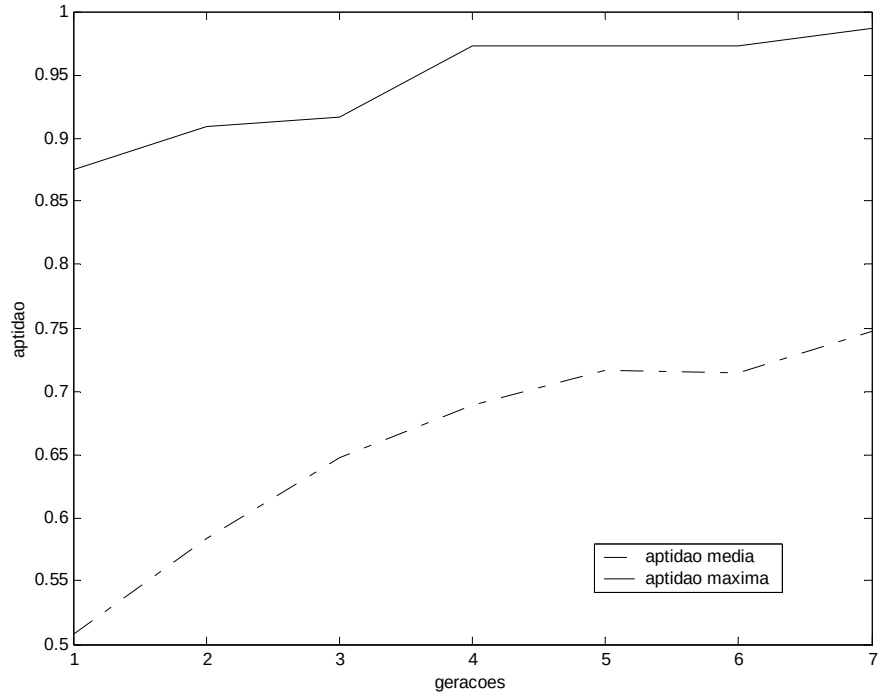
Comprimento do percurso (m)	Ponto de guinada	Tempo de processamento (s)	Total de gerações	Valor da aptidão
10,24	(5,40 , 21,38)	1,59	6	0,996



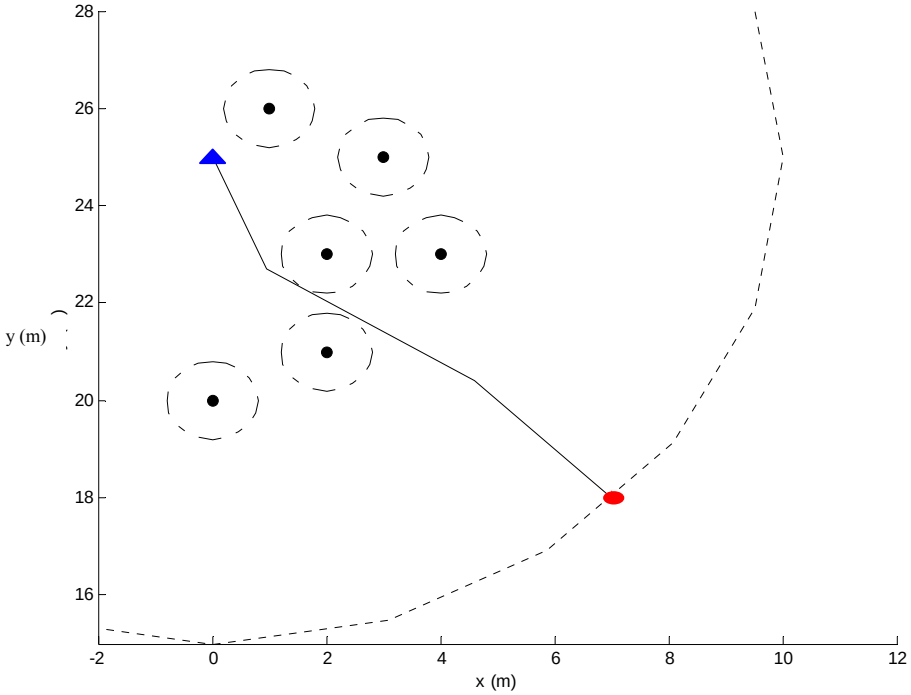


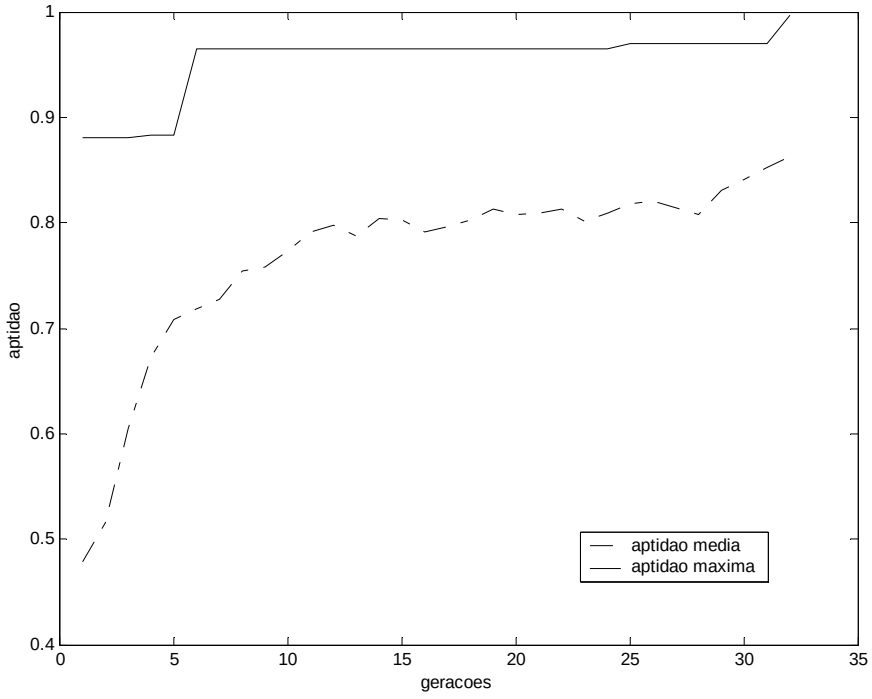
Comprimento do percurso (m)	Ponto de guinada	Tempo de processamento (s)	Total de gerações	Valor da aptidão
10,82	(0 , 22,50)	1,10	8	0,987





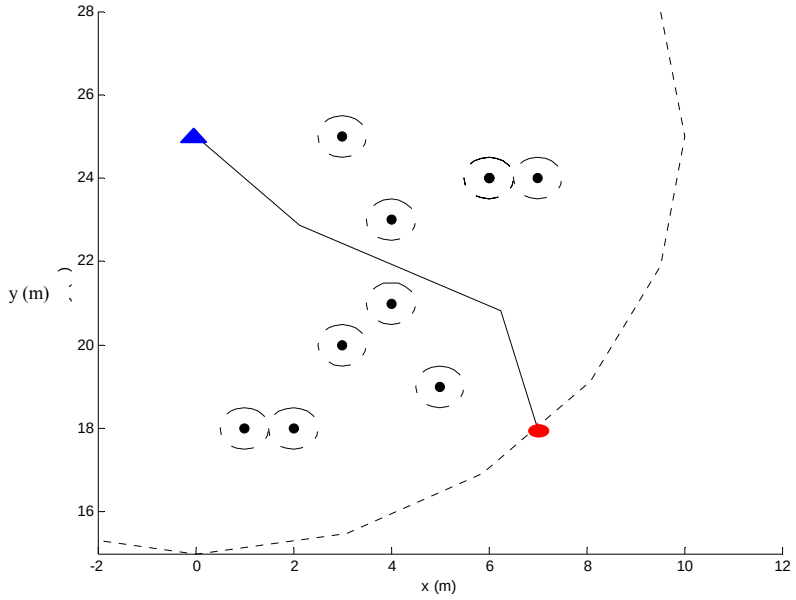
Comprimento do percurso (m)	Ponto de guinada	Tempo de processamento (s)	Total de gerações	Valor da aptidão
10,20	(0,96 , 22,69) (4,59 , 20,40)	3,54	33	0,996

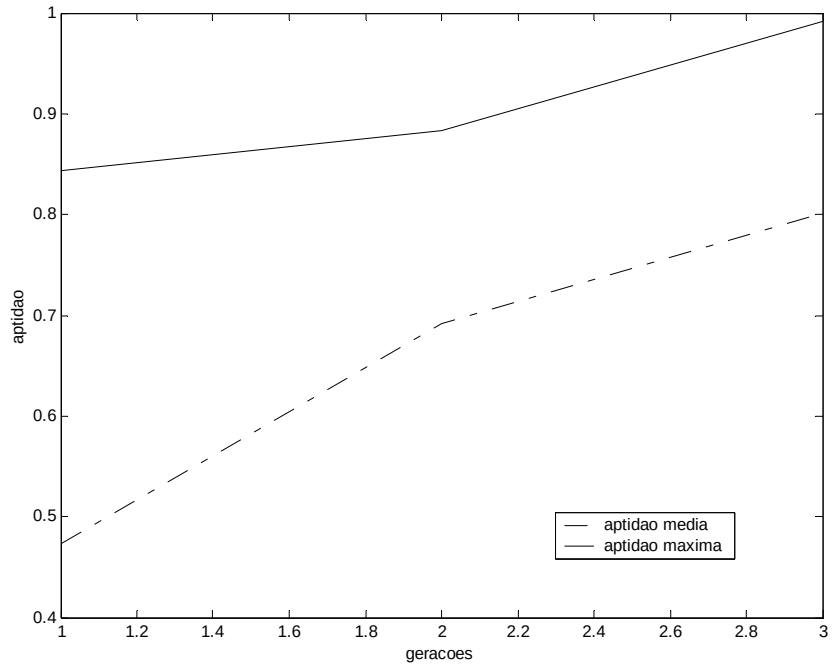




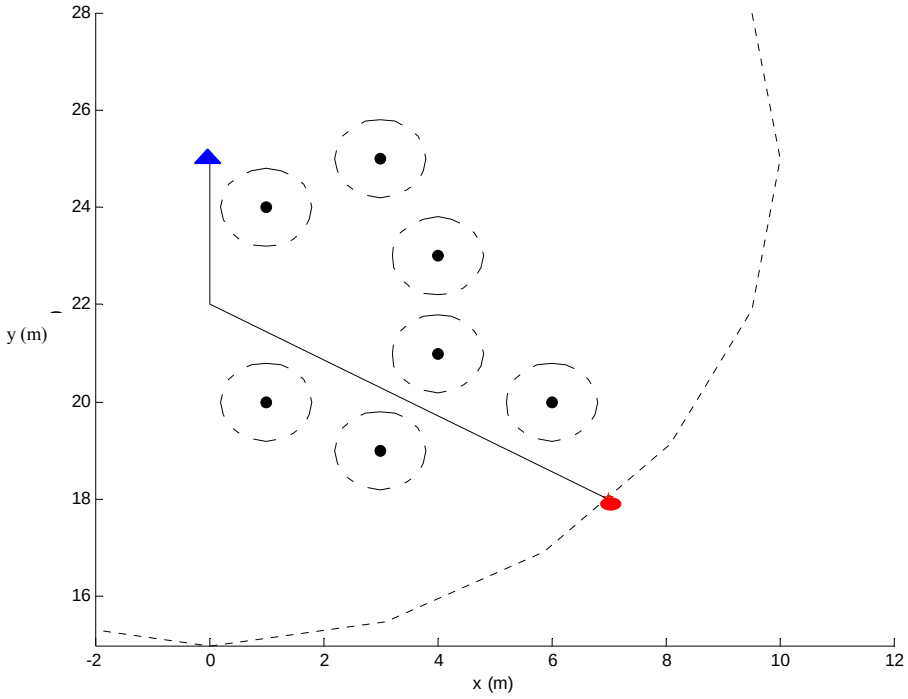
A seguir é apresentado o resultado obtido com seleção por torneio e com elitismo.

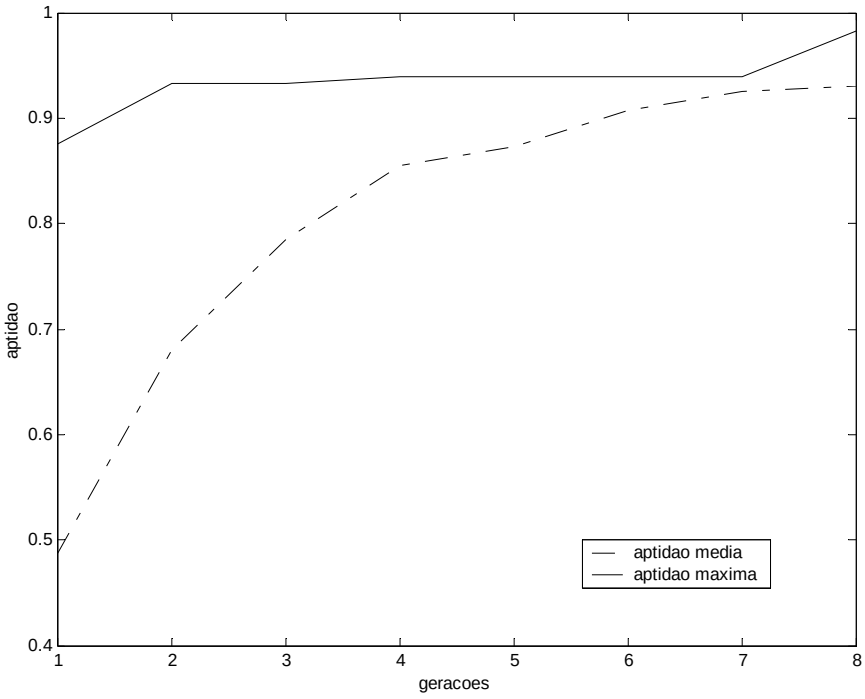
Comprimento do percurso (m)	Ponto de guinada	Tempo de processamento (s)	Total de gerações	Valor da aptidão
10,52	(2,12 , 22,88) (6,23 , 20,83)	0,67	3	0,991



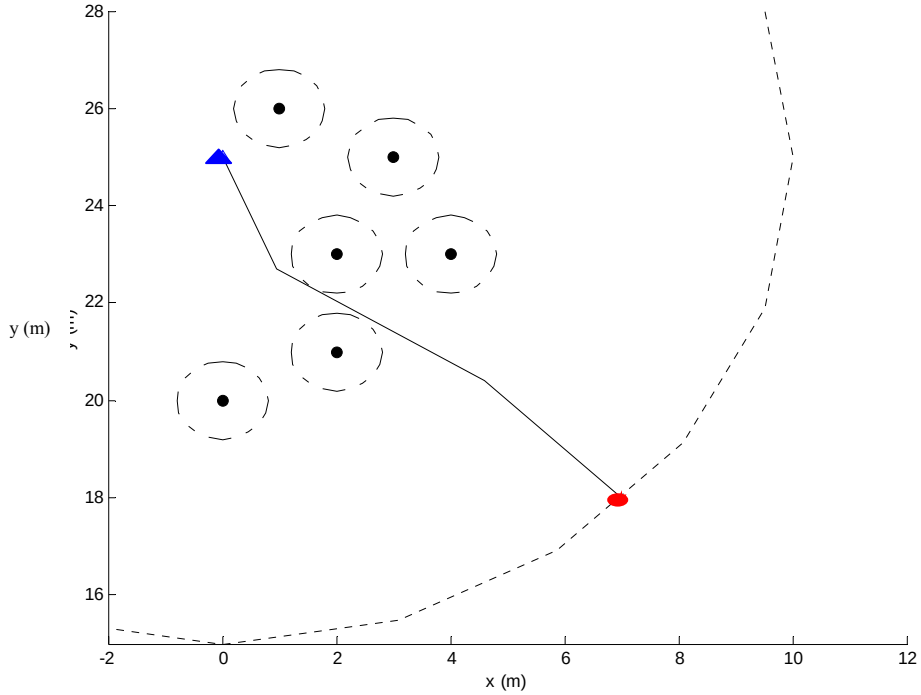


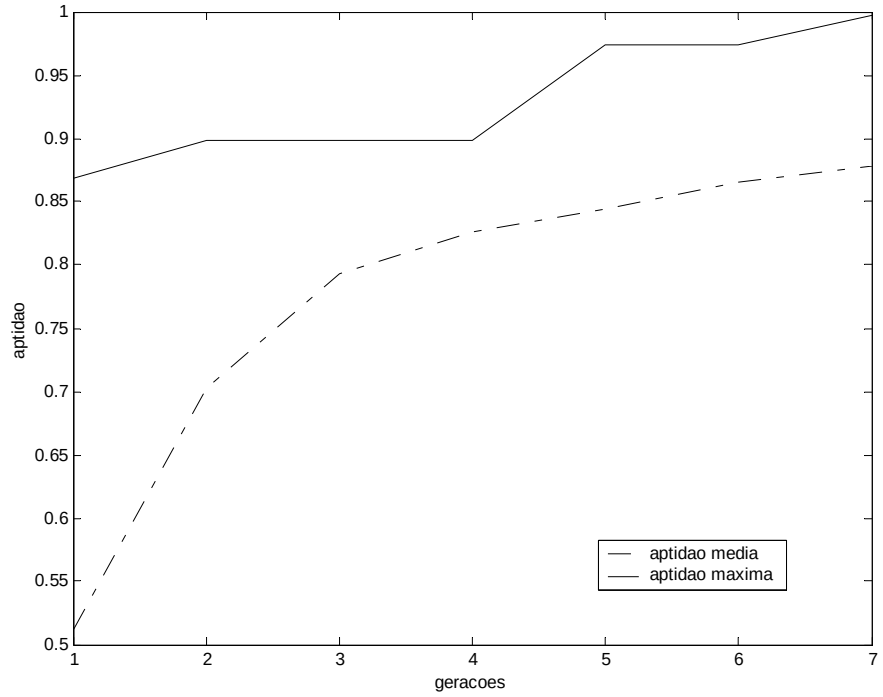
Comprimento do percurso (m)	Ponto de guinada	Tempo de processamento (s)	Total de gerações	Valor da aptidão
11,06	(0 , 22)	1,16	9	0,983





Comprimento do percurso (m)	Ponto de guinada	Tempo de processamento (s)	Total de gerações	Valor da aptidão
10,20	(0,96 , 22,70) (4,60 , 20,40)	1,00	8	0,997





5.3 – Aplicação do algoritmo proposto

Para demonstrar a potencialidade do algoritmo proposto, é feita uma comparação entre o resultado obtido no artigo de Márquez et al [14] e com o algoritmo proposto em por Sung-Keun Kim et al [32].

O primeiro artigo foi escolhido para a comparação de resultados devido a sua similaridade com o ambiente proposto nesta dissertação.

Conforme foi discutido no Capítulo 3 os autores propõem a geração de *waypoints*, a partir de um Diagrama de Voronoi, que serão utilizados pelo robô para a navegação por campos potenciais. Cada um dos *waypoints* gerados terá seu potencial atrativo calculado, sendo considerado como um destino local.

Um AG é usado para otimizar um total de dez variáveis relacionadas aos campos potenciais gerados pelos *waypoints*, obstáculos e destino. Os *waypoints* e o destino serão representados por campos de atração e os obstáculos serão campos repulsivos.

Na formulação do algoritmo, diversos *waypoints* são considerados, mas durante sua execução, são abandonados, não sendo usados efetivamente para a navegação. Com isto, é consumido tempo durante o cálculo do campo potencial de *waypoints* que não são utilizados efetivamente. Outro ponto a ser comentado é que dentre os parâmetros dos campos potenciais evoluídos no AG existe um que representa a distância mínima entre o robô e o *waypoint* para a qual o *waypoint* pode ser desconsiderado, isso também implica em custo computacional desnecessário.

Em seu artigo os autores usaram um simulador virtual de robô (*VirBot*), que simula um tipo específico de robô (*Khepera*) navegando em ambientes criados por eles. Um ambiente é apresentado e suas dimensões são apresentadas. A área de navegação

possui 1,20 x 0,90m. Como o diâmetro do robô também é conhecido (55 mm) foi possível, a partir dessas informações, reproduzir em escala, no MATLAB, o ambiente apresentado.

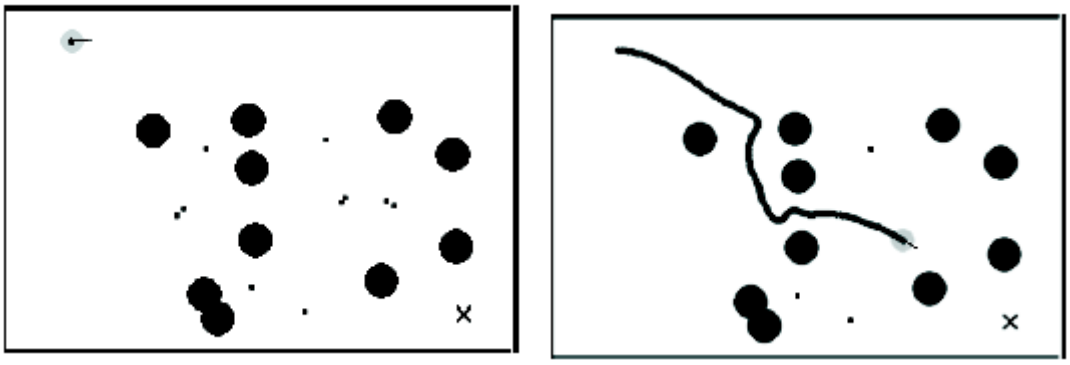


Figura 5.6 - O ambiente de navegação do artigo.

A figura 5.6 mostra o ambiente de navegação do robô, com a posição inicial do robô no canto superior esquerdo, sendo o destino o ponto "x" no canto inferior direito. Os obstáculos são representados pelos pontos maiores e os *waypoints* pelos menores. A medida que o robô se desloca, alguns *waypoints* são apagados.

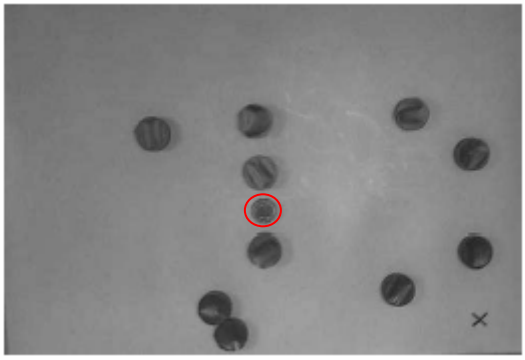


Figura 5.7 – O ambiente real de navegação. O robô está assinalado em vermelho.

Para a simulação dessa comparação, o AG empregado tem as seguintes características:

Total de indivíduos: 50

Tempo de execução: 0,2500s

Comprimento da trajetória com AG: 1,29 m

Comprimento estimado do percurso escolhido no artigo: 1,34

Ponto de guinada (0,60 , 0,71) (0,77 , 0,61)

Seleção por roleta

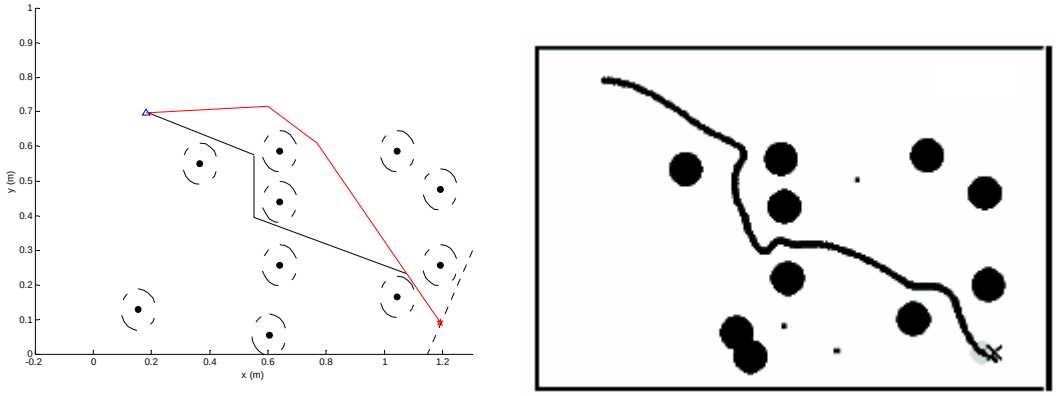


Figura 5.8 – Comparação dos resultados. A linha em vermelho é a trajetória determinada pelo AG proposto.

Percebe-se que a trajetória obtida com o algoritmo proposto nesta dissertação, passa por uma área com menor densidade de obstáculos, realizando menos manobras. A navegação do robô pode ocorrer pelo método de campos potenciais, nos mesmos moldes propostos pelos autores do artigo, incluindo a otimização por AG. A diferença, e maior vantagem da solução proposta, é a de reduzir o número de campos potenciais de *waypoints* que seria necessário calcular a apenas dois. Como vantagem adicional, o número de variáveis incluídas no AG de otimização dos campos potenciais também é reduzido.

O algoritmo proposto também pode ser empregado para resolver o problema apresentado por Sung-Keun Kim et al. [32]. Neste trabalho, os autores desenvolvem um método para o planejamento do caminho de robôs que circulam em construções, onde existem obstáculos fixos, geralmente materiais de construção ou escombros, cuja localização é conhecida. O robô é assumido como sendo um ponto e usa o GPS para calcular a direção de movimento e a sua distância para o destino. O algoritmo proposto pelos autores, *SensBug,* que realiza as seguintes ações no robô:

- percorra a linha principal até atingir um obstáculo ou o ponto de chegada;

- ao se aproximar de um obstáculo, determine os pontos intermediários necessário para se obter o ponto para o qual o obstáculo será superado;

- contorne o obstáculo;

- se possível retome o caminho principal, senão refaça o procedimento.

A figura 5.9 mostra o ambiente proposto pelos autores para navegação do robô.

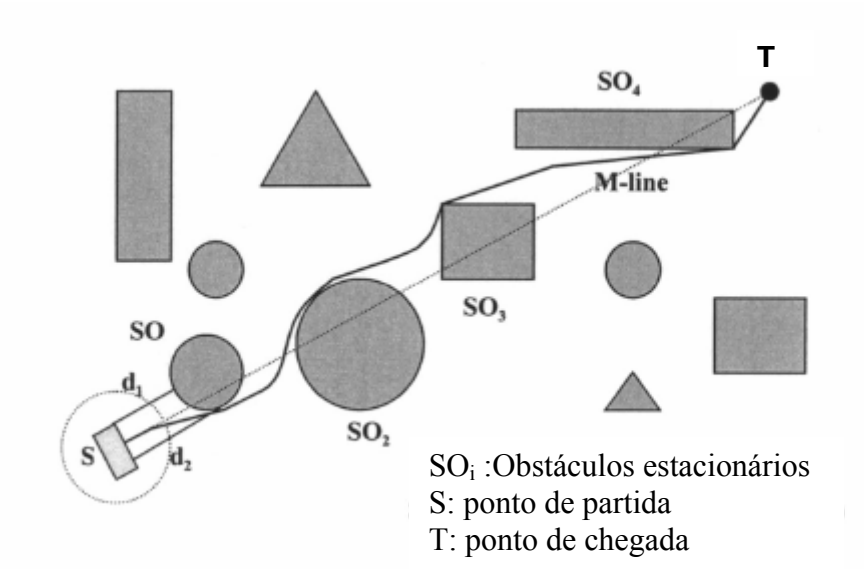


Figura 5.9 – Ambiente proposto com obstáculos estacionários

O método proposto pelos autores planeja a trajetória, mas sem garantias de que será o caminho ótimo.

Os autores não apresentam resultados de simulações ou de implementação do algoritmo em robôs. Mas o problema apresentado possui similaridade com o proposto nesta dissertação, o que permite assumir que ele pode ser resolvido com o AG proposto, com a vantagem de se obter uma trajetória ótima.

6 - Conclusão e trabalhos futuros

Esta dissertação apresentou o problema de planejamento do caminho de um robô móvel como sendo um problema de otimização. Foram propostos uma formulação e um método para a solução do problema de determinação da trajetória ótima utilizando Algoritmos Genéticos.

O algoritmo consiste em fazer evoluir uma população inicial de indivíduos, gerada aleatoriamente, que representa possíveis caminhos para o robô. A idéia é determinar as coordenadas no ambiente de deslocamento do robô que sirvam de pontos de guinada (*waypoints*) na trajetória, permitindo que ele possa se deslocar usando o processo de *dead reckoning* por hodometria ou por meios inerciais.

Para avaliar o algoritmo foram utilizadas duas formas de seleção dos indivíduos, a roleta e o torneio. Foi verificado que, para o problema estudado, a seleção por torneio necessitou de um tamanho de população maior do que o da roleta para apresentar um resultado mais estável. Contudo, após a estabilização, os valores apresentados pela seleção por torneio foram melhores do que os obtidos por roleta, resultando em trajetórias médias e tempo médio de processamento menores.

Os resultados obtidos mostram que o algoritmo é adequado para determinar a trajetória mínima em um planejamento global, onde os obstáculos existentes são estáticos e com posição previamente conhecida.

O algoritmo não foi empregado em um robô real, mas para demonstrar a sua potencialidade foi realizada uma comparação com o resultado obtido em [14]. Foi mostrado que, uma vez que o ambiente de navegação é conhecido e seus obstáculos têm sua posição determinada, não é necessário calcular uma série de pontos usando o

diagrama de Voronoi, para se conhecer os pontos de guinada que auxiliam a navegação. O algoritmo proposto planejou uma trajetória que pode ser cumprida usando o mesmo método de navegação desenvolvido em [14], usando apenas dois *waypoints*, reduzindo com isto o custo computacional.

Nesta dissertação é apresentada ainda a possibilidade de uso do algoritmo como alternativa ao proposto em [32] para a realização do planejamento do caminho de um robô empregado em áreas de construção ou em áreas de desabamentos. Os cenários propostos pelos autores do artigo guardam grande similaridade com os estudados neste trabalho porém o método de planejamento apresentado no artigo não garante o menor caminho.

Por fim, conclui-se com esta dissertação que o uso do AG para a solução problema de planejamento do caminho em ambientes onde os obstáculos são estáticos, resulta em um procedimento de simples implementação, sem a necessidade de se obter os pontos de um diagrama de Voronoi ou de se determinar um grafo, cuja complexidade aumenta com a quantidade de vértices.

A seguir são apresentadas sugestões para trabalhos futuros:

1 – Acrescentar na formulação do algoritmo a dinâmica do robô, levando em consideração para o planejamento do caminho, a sua velocidade de deslocamento.

2 – Formular o algoritmo para a situação na qual o robô se desloca no espaço; e

3 – Aproveitar o algoritmo apresentado, para a realização de roteamento de placas de circuito impresso.

7 - Referências Bibliográficas

[1] TOMATIS, N., *Hybrid, Metric, Topological Mobile Robot Navigation*, Tese de Dotourado, Ècole Polytechnique Federale de Lausanne, Lausanne, 2001.

[2] HAUPT, R. L., *Practical Genetic Algorithms*. 1 ed. New York, John Wiley & Sons, Inc., 1998.

[3] HOLLAND, J. H., *Adaptation in Natural and Artificial Systems*. Ann Harbour, University of Michigan Press, 1975.

[4] GEN, M. e CHENG, R., *Genetic Algorithm & Engeneering Design*. 1 ed. New York, John Wiley & Sons, Inc, 1997.

[5] PIO, J. L. S. e M., C. M. F., "Navegação Robótica" in *Jornadas de Atualização em Informática - Livro Texto*, vol. II, S.B.C., Ed., 1ª ed: Campinas, 2003, pp. 389-438.

[6] ARKIN, R. C., *Based-Behavior Robotics*, MIT Press, 1998.

[7] BORENSTEIN, J. e KOREN, Y., "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots", *IEEE Journal of Robotic and Automation*, v. 7, n. 3, pp. 278-288, 1991.

[8] THRUN, S. e BÜCKEN, A., "Integrating Grid-Based and Topological Maps for Mobile Robot Navigation", In: *Proceedings of 13th National Conference on Artificial Intelligence AAAI*, pp. 944-950, Portland, Oregon, 1996.

[9] CHATILA, R., "Mobile Robot Navigation Algorithms" in *Algorithms Foundations of Robotics*: A. K. Peters, 1995, pp. 97-107.

[10] DIXON, J. e HENLICH, O., 1997, *Mobile Robot Navigation*. In: Final Report Imperial College Of London, London.

[11] BORENSTEIN, J., EVERETT, H. R., FENG, L., e WEHE, D., "Mobile Robot Positioning - Sensors and Techniques", *Journal of Robotic Systems*, v. 14, n. 4, pp. 231-249, 1996.

[12] SMITH, R., SELF, M., e CHEESEMAN, P., "Estimating Uncertain Spatial Relationships in Robotics" in *Autonomous Robot Vehicles*: Springer, 1990.

[13] RIBEIRO, M. I., 2005, *Obstacle Avoidance*. In: Instituto Superior Técnico, Lisboa.

[14] MARQUEZ, E., SAVAGE, J., PETTERSSON, J., TRYGG, N., PETERSSON, A., e WAHDE, M., "Optimization of Waypoints-Guided Potential Field Navigation Using Evolutionary Algorithms", In: *Proceedings of International Conference on Intelligent Robots and Systems*, pp. 7, Sendai, Japan, 2004.

[15] SAVIOLI, C. E., *Criação de um Vetor de Teste Usando Algoritmos Genéticos*, Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, 2005.

[16] WHITLEY, D., "A Genetic Algorithm Tutorial", *Statistics and Computing*, v. 4, pp. 65-85, 1994.

[17] MILLER, B. L. e GOLDBERG, D. E., 1995, *Genetic Algorithms, Tournament Selection and the Effects of Noise*. In: IlliGAL Report 95006, University of Illinois at Urbana-Champaign, Urbana.

[18] MAN, K. F., TANG, K. S., e KWONG, S., *Genetic Algorithm*. 1 ed. London, Springer-Verlag, 1999.

[19] HILL, S. e O'RIORDAN, C., 2001, *Genetic Algorithms, their Operators and the NK Model*. In: Final Report National University of Ireland, Galway, Ireland.

[20] DIJKSTRA, E. W., "A note on two problems in connection with graphs", *Numerische Mathematik*, v. 1, pp. 269-271, 1959.

[21] BRIGGS, A. J., DETWEILER, C., e SCHARSTEIN, D., "Expected Shortest Paths for Landmark-Based Robot Navigation", *International journal of Robotics Resarch*, v. 23, pp. 717-728, 2004.

[22] ERSSON, T. e HU, X., "Path Planning and Navigation of Mobile Robots in Unknown Environments", In: *Proceedings of International Conference on Intelligent Robots and Systems*, 2001.

[23] THULASIRAMAN, K. e SWAMY, M. N. S., *Graphs: Theory and Algorithms*, John Willey & Sons, 1992.

[24] AHUJA, R. K., MAGNATI, T. L., e ORLIN, J. B., *Network Flows: Theory, algorithm and Application*, Prentice Hall, 1993.

[25] STENTZ, A., "Optimal and Efficient Path Planning for Partially-Known Enviroments", In: *Proceedings of IEEE - International Conference on Robotic and Automation*, pp. 3310-3317, 1994.

[26] STENTZ, A., "The Focussed D* Algorithm for Real-Time Replanning", In: *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 1652-1659, 1995.

[27] SZABO, R., "Topological Navigation of Simulated Robots using Occupancy Grid", *International Journal of Advanced Robotic Systems*, v. 1, n. 3, pp. 201-206, 2004.

[28] AHUACTZIN, J. M., TALBI, E.-G., BESSIERE, P., e MAZER, E., "Using Genetic Algorithms for Robot Motion Planning", In: *Proceedings of 10th European Conference on Artificial Intelligence (ECAI92)*, pp. 672-675, 1992.

[29] DUCKETT, T., "A Genetic Algorithm for Simultaneous Localization and Mapping", In: *Proceedings of IEEE International Conference on Robotic and Automation (ICRA2003)*, pp. 434-439, 2003.

[30] FLOREANO, D. e MONDALA, F., "Evolution of Homing Navigation in a Real Mobile Robot", *IEEE Transaction on Systems, Man and Cybernetic*, v. 3, n. 26, pp. 396-407, 1996.

[31] H-SEDIGHI, K., ASHENAYI, K., MANIKAS, T. W., e WAINGWRIGHT, R. L., "Autonomous Local Path-Planning for a Mobile Robot Using a Genetic Algorithm", In: *Proceedings of Congress on Evolutionary Computation (CEC 2004)*, pp. 1338-1345, Portland, OR, 2004.

[32] KIM, S.-K., RUSSELL, J. S., e KOO, K.-J., "Construction Robot Path-Planning for Earthwork Operations", *Journal of Computing in Civil Engineering*, v. 17, n. 2, pp. 97-104, 2003.

APÊNDICE

Neste apêndice são apresentadas as rotinas, escritas em MATLAB, utilizadas para a realização dos testes.

```
%Programa para identificar o caminho minimo V2.2
%versao com selecao por roleta e com elitismo
%
%ALEXANDRE DE VASCONCELOS SICILIANO
%05/06/2006

clear all;
clc;
clear fun;
clf;
%=========================================================
%ponto inicial e final, posicao dos obstaculos
%=========================================================
ptoinicial=[0,25];
ptofinal=[7,18];
phi=abs(atan((ptofinal(1,2)-ptoinicial(1,2))/(ptofinal(1,1)-ptoinicial(1,1))));
vetor_obst=[6 24;5 19;2 18;4 21;3 20;4 23;7 24;1 18;6 24;3 25];
[qtd_obst,col]=size(vetor_obst);
raio_busca=10;
Rmax=10;
rmin=2;
%=========================================================
%populacao inicial
%=========================================================
qtd_ind=50; %quantidade de individuos na populacao
qtd_bit=4; %quantidade de bits por gene
pop_inicial=zeros(qtd_ind,80);
for indidx=1:qtd_ind
```

```
    qtd_seg(indidx)=round((rand(1)*9)+1);
pop_inicial(indidx,1:(qtd_seg(indidx)*qtd_bit*2))=randint(1,(qtd_seg(indidx)*qtd_bit*
2));
end
populacao=pop_inicial;
geracao=1;
fit_max=0;
melhor_fit=0;
tic
%=========================================================
%inicio da rotina principal
%=========================================================
while fit_max<0.98
   if geracao==1000
      break
   end

   for ind=1:qtd_ind
      qtd_gene_ind(ind)=qtd_seg(ind)*2;
      novo_x=ptoinicial(1,1);
      novo_y=ptoinicial(1,2);
      q(ind)=0;
      for seg=1:qtd_seg(ind)
         rho(ind,seg)=abs(rmin+(((Rmax-rmin)/2^4)*bi2de(populacao(ind,(2*seg-
2)*4+1:(2*seg-2)*4+4),'left-msb')));
         teta(ind,seg)= ((pi/2)-phi)-((pi/2^4)*bi2de(populacao(ind,(2*seg-2)*4+5:(2*seg-
2)*4+8),'left-msb'));
         novox(ind,seg)=ptoinicial(1,1)+(rho(ind,seg)*cos(teta(ind,seg)));
         novoy(ind,seg)=ptoinicial(1,2)+(rho(ind,seg)*sin(teta(ind,seg)));

      end
      %calculo do comprimento do individuo
      matriz_wp=[ptoinicial;novox(ind,1:qtd_seg(ind))'
novoy(ind,1:qtd_seg(ind))';ptofinal];
      comprimento_tot(ind)=compr(matriz_wp);
```

```
%============================================================
        %rotina para verificar se o caminho gerado impacta ou passa prx a um obstaculo
%============================================================
              for segm=1:qtd_seg(ind)+1
                 if matriz_wp(segm+1,1)== matriz_wp(segm,1)
                   a(ind,segm)=0;
                   b(ind,segm)=matriz_wp(segm,1);
                 else
                   a(ind,segm)=(matriz_wp(segm+1,2)-
matriz_wp(segm,2))/(matriz_wp(segm+1,1)-matriz_wp(segm,1));
                   b(ind,segm)=((matriz_wp(segm,2)*matriz_wp(segm+1,1))-
(matriz_wp(segm+1,2)*matriz_wp(segm,1)))/(matriz_wp(segm+1,1)-matriz_wp(segm,1));
              end

              for obst=1:qtd_obst
                 if matriz_wp(segm,1)<matriz_wp(segm+1,1)
                    infx=matriz_wp(segm,1)-1;
                    supx=matriz_wp(segm+1,1)+1;
                 else
                    infx=matriz_wp(segm+1,1)-1;
                    supx=matriz_wp(segm,1)+1;
                 end
                  if matriz_wp(segm,2)<matriz_wp(segm+1,2)
                    infy=matriz_wp(segm,2)-1;
                    supy=matriz_wp(segm+1,2)+1;
                 else
                    infy=matriz_wp(segm+1,2)-1;
                    supy=matriz_wp(segm,2)+1;
                 end
                 if vetor_obst(obst,1)>=infx & vetor_obst(obst,1)<=supx &
vetor_obst(obst,2)>=infy & vetor_obst(obst,2)<=supy
                    dist_reta(ind,segm,obst)=abs(((-
a(ind,segm)*vetor_obst(obst,1))+(vetor_obst(obst,2))-b(ind,segm))/(sqrt((a(ind,segm)^2)+1)));
                    if dist_reta(ind,segm,obst)<=0.5
                      q(ind)=q(ind)+1;
```

```
                    end
                end
            end%loop obst
        end %loop segm

%=============================================================
        %determinando aptidao do individuo
%=============================================================
        Wl=0.8;%peso
        Wq=4;
        obtido(ind)=(Wq*q(ind)/30)+ (Wl*((comprimento_tot(ind)-10)/50));
        fk(ind)=1/(obtido(ind)+1);

    end%loop individuo

    fit_medio(geracao)=mean(fk);
    [Fkord(geracao,:),Iind(geracao,:)]=sort(fk);

    if melhor_fit<=Fkord(geracao,qtd_ind)
        melhor_fit=Fkord(geracao,qtd_ind);

melhor_caminho=[ptoinicial;novox(Iind(geracao,qtd_ind),1:qtd_seg(Iind(geracao,qtd_ind)))'
novoy(Iind(geracao,qtd_ind),1:qtd_seg(Iind(geracao,qtd_ind)))';ptofinal];
        melhor_ind=populacao(Iind(geracao,qtd_ind),:);
        qtd_seg_melhor_ind=qtd_seg(Iind(geracao,qtd_ind));
    end
    fit_max(geracao)=melhor_fit;

%=============================================================
        %obtendo os setores da roleta
%=============================================================
        [num_ind,col]=size(populacao);
        acc_set=0;
        for set=1:num_ind
            acc_set=acc_set+fk(set);
            setor(set)=acc_set/fit_medio(geracao);
```

```
        end

%=================================================================
        %usando a funcao roleta para selecionar os melhores individuos
%=================================================================
        [best_pop,best_qtd_seg]=roletaT(setor,fk,populacao,qtd_seg);
%=================================================================
        %embaralhando a populacao de melhores individuos
%=================================================================
        [pop_shuffle,qtd_seg_shuffle]=shuffleT(best_pop,best_qtd_seg);
%=================================================================
        %usando a funcao crossover para realizar o crossover e a mutacao
%=================================================================
        [nova_pop,nova_qtd_seg]=crossoverT(pop_shuffle,qtd_seg_shuffle,60,10);
        populacao=nova_pop;
        qtd_seg=nova_qtd_seg;
        populacao(1,:)=melhor_ind;
        qtd_seg(1)=qtd_seg_melhor_ind;
        geracao=geracao+1;
    end%geracoes
    toc
    %=============================================================
    %apresentacao dos resultados
    %=============================================================
    comprimento=compr(melhor_caminho)
    melhor_fit
    geracao
    melhor_caminho
    hold on
    for obsta=1:qtd_obst
    auxo=1;
    Xoc=vetor_obst(obsta,1);
    Yoc=vetor_obst(obsta,2);
    for alfao=0:0.1*pi:2*pi;
        xo(auxo)=Xoc+0.5*sin(alfao);
        yo(auxo)=Yoc+0.5*cos(alfao);
```

```
   auxo=auxo+1;
end
plot(xo,yo,'k-.');
end
Xc=ptoinicial(1,1);
Yc=ptoinicial(1,2);
aux=1;
for alfa=0:0.1*pi:2*pi;
   x(aux)=Xc+raio_busca*sin(alfa);
   y(aux)=Yc+raio_busca*cos(alfa);
   aux=aux+1;
end

axis([-2 12 15 28]);
%axis('equal');
plot (x,y,'k:');
plot(vetor_obst(:,1),vetor_obst(:,2),'ko');
plot(ptofinal(1,1),ptofinal(1,2),'rp');
plot(ptoinicial(1,1),ptoinicial(1,2),'b^');
line(melhor_caminho(:,1),melhor_caminho(:,2))
xlabel('x (m)');
ylabel('y (m)');
hold off
figure(2);
g=[1:geracao-1];
plot(g,fit_medio,'k-.',g,fit_max,'k-');
xlabel('geracoes');
ylabel('aptidao');
legend('aptidao media','aptidao maxima');
```

```
%Programa para identificar o caminho minimo V2.3
%versao com selecao por torneio e com elitismo
%
%ALEXANDRE DE VASCONCELOS SICILIANO
%05/06/2006

clear all;
clc;
clear function;
clf;
%================================================================
%ponto inicial e final, posicao dos obstaculos
%================================================================
ptoinicial=[0,25];
ptofinal=[7,18];
phi=abs(atan((ptofinal(1,2)-ptoinicial(1,2))/(ptofinal(1,1)-ptoinicial(1,1))));
vetor_obst=[6 24;5 19;2 18;4 21;3 20;4 23;7 24;1 18;6 24;3 25];
[qtd_obst,col]=size(vetor_obst);
raio_busca=10;
Rmax=10;
rmin=2;
%================================================================
%populacao inicial
%================================================================
qtd_ind=64; %quantidade de individuos na populacao
qtd_bit=4; %quantidade de bits por gene
pop_inicial=zeros(qtd_ind,80);
for indidx=1:qtd_ind
    qtd_seg(indidx)=round((rand(1)*9)+1);
pop_inicial(indidx,1:(qtd_seg(indidx)*qtd_bit*2))=randint(1,(qtd_seg(indidx)*qtd_bit*
2));
end
populacao=pop_inicial;
geracao=1;
fit_max=0;
melhor_fit=0;
```

```
tic
%=================================================================
%inicio da rotina principal
%=================================================================
while fit_max<0.98
   if geracao==1000
      break
   end

   for ind=1:qtd_ind
      qtd_gene_ind(ind)=qtd_seg(ind)*2;
      novo_x=ptoinicial(1,1);
      novo_y=ptoinicial(1,2);
      q(ind)=0;
      for seg=1:qtd_seg(ind)
         rho(ind,seg)=abs(rmin+(((Rmax-rmin)/2^4)*bi2de(populacao(ind,(2*seg-
2)*4+1:(2*seg-2)*4+4),'left-msb')));
         teta(ind,seg)= ((pi/2)-phi)-((pi/2^4)*bi2de(populacao(ind,(2*seg-2)*4+5:(2*seg-
2)*4+8),'left-msb'));
         novox(ind,seg)=ptoinicial(1,1)+(rho(ind,seg)*cos(teta(ind,seg)));
         novoy(ind,seg)=ptoinicial(1,2)+(rho(ind,seg)*sin(teta(ind,seg)));

      end
      %calculo do comprimento do individuo
      matriz_wp=[ptoinicial;novox(ind,1:qtd_seg(ind))'
novoy(ind,1:qtd_seg(ind))';ptofinal];
      comprimento_tot(ind)=compr(matriz_wp);

%=================================================================
      %rotina para verificar se o caminho gerado impacta ou passa prx a um obstaculo
%=================================================================
      for segm=1:qtd_seg(ind)+1
         if matriz_wp(segm+1,1)== matriz_wp(segm,1)
           a(ind,segm)=0;
           b(ind,segm)=matriz_wp(segm,1);
         else
```

```
                    a(ind,segm)=(matriz_wp(segm+1,2)-
matriz_wp(segm,2))/(matriz_wp(segm+1,1)-matriz_wp(segm,1));
                    b(ind,segm)=((matriz_wp(segm,2)*matriz_wp(segm+1,1))-
(matriz_wp(segm+1,2)*matriz_wp(segm,1)))/(matriz_wp(segm+1,1)-matriz_wp(segm,1));
                end

                for obst=1:qtd_obst
                    if matriz_wp(segm,1)<matriz_wp(segm+1,1)
                        infx=matriz_wp(segm,1)-1;
                        supx=matriz_wp(segm+1,1)+1;
                    else
                        infx=matriz_wp(segm+1,1)-1;
                        supx=matriz_wp(segm,1)+1;
                    end
                     if matriz_wp(segm,2)<matriz_wp(segm+1,2)
                        infy=matriz_wp(segm,2)-1;
                        supy=matriz_wp(segm+1,2)+1;
                    else
                        infy=matriz_wp(segm+1,2)-1;
                        supy=matriz_wp(segm,2)+1;
                    end
                    if vetor_obst(obst,1)>=infx & vetor_obst(obst,1)<=supx &
vetor_obst(obst,2)>=infy & vetor_obst(obst,2)<=supy
                        dist_reta(ind,segm,obst)=abs(((-
a(ind,segm)*vetor_obst(obst,1))+(vetor_obst(obst,2))-b(ind,segm))/(sqrt((a(ind,segm)^2)+1)));
                        if dist_reta(ind,segm,obst)<=0.5
                          q(ind)=q(ind)+1;
                        end
                    end
                end%loop obst
            end %loop segm

%=====================================================================
            %determinando aptidao do individuo
%=====================================================================
            W=0.8;%peso
```

```
            obtido(ind)=(4*q(ind)/30)+ (W*((comprimento_tot(ind)-10)/50));
            fk(ind)=1/(obtido(ind)+1);

        end%loop individuo

        fit_medio(geracao)=mean(fk);
        [Fkord(geracao,:),Iind(geracao,:)]=sort(fk);

        if melhor_fit<=Fkord(geracao,qtd_ind)
            melhor_fit=Fkord(geracao,qtd_ind);

melhor_caminho=[ptoinicial;novox(Iind(geracao,qtd_ind),1:qtd_seg(Iind(geracao,qtd_ind)))'
novoy(Iind(geracao,qtd_ind),1:qtd_seg(Iind(geracao,qtd_ind)))';ptofinal];
            melhor_ind=populacao(Iind(geracao,qtd_ind),:);
            qtd_seg_melhor_ind=qtd_seg(Iind(geracao,qtd_ind));
        end
        fit_max(geracao)=melhor_fit;

%=========================================================================
        %inicio da rotina do torneio
%=========================================================================
        for ap=1:qtd_ind
            pop_ord(ap,:)=populacao(Iind(geracao,ap),:);
            qtd_seg_ord(ap)=qtd_seg(Iind(geracao,ap));
        end
        rank_pop=flipud(pop_ord);
        rank_qtd_seg=fliplr(qtd_seg_ord);

        for pair=1:qtd_ind/2
            torn(pair,:)=[round(((qtd_ind/2)*rand(1))+1) round(((qtd_ind/2)*rand(1))+1)];
        end
        %matriz de pais que reproduzirao (parents)
        for sel=1:qtd_ind/2
            if torn(sel,1)<=torn(sel,2) %VRF qual dos 2 competidores tem a maior aptidao
                parents(sel,:)=rank_pop(torn(sel,1),:);
                parqtd_seg(sel)=rank_qtd_seg(torn(sel,1));
```

```
        else
            parents(sel,:)=rank_pop(torn(sel,2),:);
            parqtd_seg(sel)=rank_qtd_seg(torn(sel,2));

        end
    end

%=====================================================================
    %realizando o cruzamento dos competidores (pais) vencedores
%=====================================================================
    nova_pop=zeros(qtd_ind,80);
    nova_pop(1:qtd_ind/2,:)=parents(1:qtd_ind/2,:);
    nova_qtd_seg(1:qtd_ind/2)=parqtd_seg(1:qtd_ind/2);
    ptocorte=(parqtd_seg.*8)-1;
    for lin=1:2:qtd_ind/2
        if ptocorte(lin)<=ptocorte(lin+1)
            corte=round(((ptocorte(lin)-1)*rand(1))+1);
        else
            corte=round(((ptocorte(lin+1)-1)*rand(1))+1);
        end
    nova_pop(lin+(qtd_ind/2),1:corte)=parents(lin,1:corte);
    nova_pop(lin+(qtd_ind/2),corte+1:80)=parents(lin+1,corte+1:80);
    nova_pop(lin+1+(qtd_ind/2),1:corte)=parents(lin+1,1:corte);
    nova_pop(lin+1+(qtd_ind/2),corte+1:80)=parents(lin,corte+1:80);
    nova_qtd_seg(lin+(qtd_ind/2))=parqtd_seg(lin);
    nova_qtd_seg(lin+1+(qtd_ind/2))=parqtd_seg(lin+1);
    end
    populacao=nova_pop;
    qtd_seg=nova_qtd_seg;

%=====================================================================
    % rotina de mutacao
%=====================================================================
    tx_mut=15;
    qtd_mut=round((tx_mut*qtd_ind)/100);
    max_mut=qtd_ind-qtd_mut;
```

```
    inicio_mut=round((((max_mut)-1)*rand(1))+1);

    for mut = inicio_mut:inicio_mut+qtd_mut-1
        mutcorte=(qtd_seg(mut)*8)-1;
        mutacao=round(((mutcorte-1)*rand(1))+1);
        if populacao(mut,mutacao)==1
            populacao(mut,mutacao)=0;
        else
            populacao(mut,mutacao)=1;
        end
    end%mut
    populacao(1,:)=melhor_ind;
    qtd_seg(1)=qtd_seg_melhor_ind;
    geracao=geracao+1;
end%geracoes
toc
%================================================================
%apresentacao dos resultados
%================================================================
comprimento=compr(melhor_caminho)
melhor_fit
geracao
melhor_caminho

hold on
for obsta=1:qtd_obst
auxo=1;
Xoc=vetor_obst(obsta,1);
Yoc=vetor_obst(obsta,2);
for alfao=0:0.1*pi:2*pi;
    xo(auxo)=Xoc+0.5*sin(alfao);
    yo(auxo)=Yoc+0.5*cos(alfao);
    auxo=auxo+1;
end
plot(xo,yo,'k-.');
end
```

```
Xc=ptoinicial(1,1);
Yc=ptoinicial(1,2);
aux=1;
for alfa=0:0.1*pi:2*pi;
   x(aux)=Xc+raio_busca*sin(alfa);
   y(aux)=Yc+raio_busca*cos(alfa);
   aux=aux+1;
end

axis([-2 12 15 28]);
%axis('equal');
plot (x,y,'k:');
plot(vetor_obst(:,1),vetor_obst(:,2),'ko');
plot(ptofinal(1,1),ptofinal(1,2),'rp');
plot(ptoinicial(1,1),ptoinicial(1,2),'b^');
line(melhor_caminho(:,1),melhor_caminho(:,2))
xlabel('x (m)');
ylabel('y (m)');
hold off
figure(2);
g=[1:geracao-1];
plot(g,fit_medio,'k-.',g,fit_max,'k-');
xlabel('geracoes');
ylabel('aptidao');
legend('aptidao media','aptidao maxima');
```