Proceedings of V Brazilian Conference on Neural Networks - V Congresso Brasileiro de Redes Neurais
pp. 247-252, April 2-5, 2001 – Rio de Janeiro - RJ - Brazil

Navegação de Robôs Autônomos Utilizando Redes Neurais, com Planejamento de Trajeto por Algoritmos Genéticos baseados em um Mapa Fuzzy

João Alberto Fabro, Heitor Silvério Lopes, L.V.R. Arruda
Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial
Centro Federal de Educação Tecnológica do Paraná – CEFET-PR
Av. Sete de Setembro, 3165 80.230-901 Curitiba-PR
joao@cpgei.cefetpr.br, hslopes@cpgei.cefetpr.br, arruda@cpgei.cefetpr.br

Abstract

This work describes a system for navigation of autonomous robots. The system uses a genetic algorithm for path planning and a fuzzy model of the environment in which the navigation is accomplished by means of a reactive subsystem controlled by neural networks.

1. Introdução

O problema de navegação autônoma de robôs consiste em conduzir um robô, partindo de sua posição inicial, até uma determinada posição final, através do ambiente. É necessário realizar esta navegação evitando colisões com obstáculos do ambiente (por exemplo: paredes, objetos, pessoas, ou outros robôs autônomos). Existem várias propostas para a solução deste problema. As duas classes de abordagens principais são o planejamento de trajeto baseado em um modelo do ambiente, e a navegação utilizando informações de sensores [1].

As abordagens baseadas em modelos utilizam um modelo do ambiente para definir um trajeto de movimentação em direção ao alvo, evitando obstáculos. A eficácia destes métodos depende principalmente da precisão do modelo, para possibilitar a geração de trajetos que evitem colisões. O problema referente à utilização destas técnicas decorre da dificuldade de se obter modelos precisos dos ambientes. Estas técnicas são mais apropriadas para planejamento de trajetos em ambientes estáticos e controlados.

As abordagens baseadas em sensores utilizam leituras obtidas diretamento do ambiente, para determinar o próximo movimento do robô em direção ao seu objetivo, evitando obstáculos. Uma classe de metodologias baseadas em sensores utiliza o conceito de *comportamentos* [2]. Cada comportamento é baseado em leituras de sensores específicos. Exemplos de comportamentos incluem *alcançar alvo* e *desviar obstáculos*. A principal vantagem da navegação baseada em sensores é que o robô pode navegar em ambientes dinâmicos e desconhecidos, pois as únicas informações necessárias para a movimentação são obtidas dos próprios sensores. Entretanto, as dificuldades apresentadas no projeto de tais abordagens são grandes devido à imprevisibilidade do ambiente, e às inúmeras situações reais com as quais o sistema poderá se defrontar.

As abordagens baseadas em modelos e em sensores podem ser combinadas em abordagens híbridas[3], nas quais o planejamento global do trajeto é realizado com o uso de um modelo incompleto do ambiente, enquanto que técnicas de navegação por sensores são utilizadas para desviar obstáculos que possam ser encontrados no caminho planejado.

No atual trabalho, é proposta uma abordagem híbrida para o problema do planejamento do trajeto e de sua execução. O planejamento do trajeto é feito através de um **algoritmo genético** que, utilizando informações provenientes de um **mapa nebuloso (*fuzzy)*** do ambiente, procura caminhos que levem o robô de sua posição inicial até a posição final, sem colidir com os obstáculos mapeados. Para a execução do trajeto planejado, o robô deve se basear nas leituras dos sensores, e para isto foi desenvolvida uma **rede neural**. Através do controle neural é possível detectar os obstáculos e desvia-los, permitindo que o robô encontre uma rota alternativa para voltar ao trajeto planejado pelo algoritmo genético, não colidindo com possíveis obstáculos não mapeados, ou obstáculos dinâmicos que atravessem seu caminho. Desta forma, esta proposta é híbrida no sentido de utilizar uma combinação de abordagens baseadas em modelos e abordagens baseadas em sensores. Também é híbrida no sentido de utilizar diversas técnicas da inteligência computacional: algoritmos genéticos, sistemas *fuzzy* e redes neurais. Para os experimentos, foi utilizado o simulador de robôs móveis Khepera [4].

O planejamento do trajeto é feito através de algoritmos genéticos (AG's). Abordagens utilizando algoritmos genéticos para evoluir bases de regras capazes de controlar o comportamento reativo de um robô foram inicialmente propostas por Grefenstette e colaboradores [5][6]. Também Koza [7] propôs a utilização de programação genética para o mesmo fim, onde são evoluídos programas capazes de controlar a navegação reativa de robôs. A utilização de AG's para o planejamento de trajetos de robôs móveis foram propostas por Page [8], e, mais recentemente, por Michalewicz [9]. Nesta última proposta, o mapa interno representa os obstáculos em formato de polígonos fechados, e o algoritmo genético é utilizado tanto para o planejamento inicial do caminho, como para o tratamento de obstáculos não previamente mapeados.

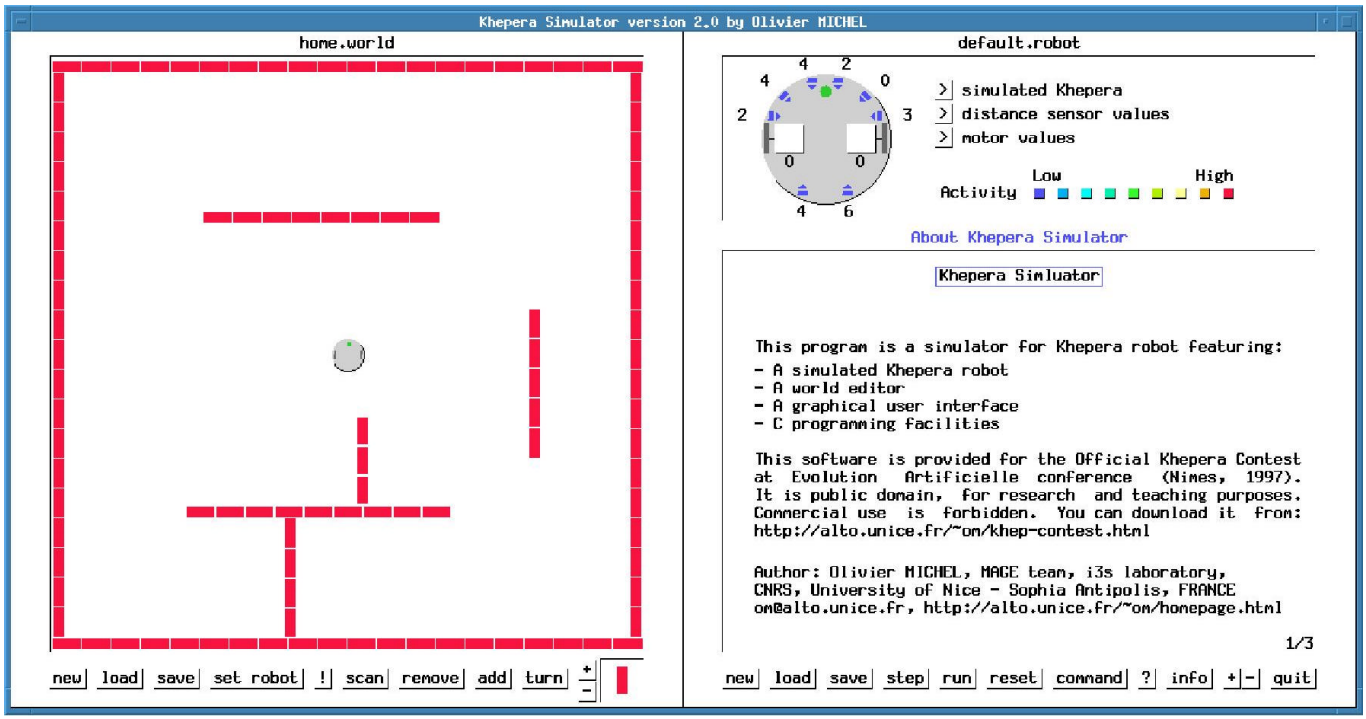


Figura 1 – O Simulador de Robôs Autônomos Khepera

Deste modo, o mesmo algoritmo genético têm execução contínua, durante o trajeto, para, em caso de encontro com obstáculos não conhecidos, replanejar a trajetória. Em um trabalho mais recente [10], é apresentada uma melhoria da proposta anterior [9], utilizando um algoritmo genético incremental que é capaz de tratar uma maior gama de situações. Entretanto, ambas as propostas, por utilizarem os AG's para o re-planejamento da trajetória, tratam apenas obstáculos estáticos e completamente conhecidos.

Neste trabalho, são utilizados os conceitos dos sistema *fuzzy* na representação interna do mapa do ambiente. Além disto, a execução da trajetória é realizada por duas redes neurais *feedforward*, treinadas pelo algoritmo *Back-Propagation* [11]. Uma rede é responsável pela navegação reativa, desviando os eventuais obstáculos desconhecidos (estáticos ou dinâmicos) que apareçam no caminho. A outra rede é responsável por dirigir o robô em direção ao seu alvo no ambiente, através do caminho planejado. O comportamento de "desvio de obstáculos", precedente ao comportamento de "seguir alvo", permite ao robô caminhar por seu trajeto e alcançar sua posição objetivo, sem entretanto colidir com obstáculos desconhecidos com que se depare durante sua navegação.

2. O Simulador Khepera

Este simulador [4] divide-se em duas partes: o "mundo", que simula um ambiente de 1 metro quadrado, ocupando a parte esquerda da tela, e o "robô", que simula o robô Khepera real de 5 cm de diâmetro, na parte direita da tela (figura 1). Na parte do mundo pode-se observar o desempenho do robô enquanto este navega pelo ambiente, e na parte do robô pode-se visualizar os valores dos sensores e motores do robô, além de outras informações pertinentes. Este simulador permite que sejam construídos labirintos, utilizando-se de primitivas gráficas (tijolos) retangulares. Apresenta uma interface de programação para a implementação de algoritmos de controle de navegação usando as linguagens C ou C++.

3.O Mapa *Fuzzy* do Ambiente

Neste trabalho é necessário armazenar o modelo do ambiente em um formato que possa ser facilmente manipulado pelo algoritmo genético, de modo a determinar a intersecção do trajeto com os obstáculos. O ambiente do simulador é armazenado internamente, sendo representado por uma lista encadeada de objetos do tipo tijolo, que formam as paredes e os obstáculos. A partir desta lista de objetos, optou-se por representar o mapa do ambiente, internamente, como uma matriz com 1000x1000 posições, representando o mundo simulado, de 1 metro quadrado.

Em uma proposta anterior[12], esta matriz era binária e preenchida com um valor 0 se não houvesse obstáculo na posição, e 1 se houvesse. Varrendo a lista interna de tijolos, esta matriz era preenchida e utilizada como um modelo preciso da localização dos obstáculos. Para realizar o planejamento do trajeto, utilizava-se um algoritmo de busca heurística. Este algoritmo dependia completamente da perfeição do ambiente. Assim, paredes "rugosas" (isto é, com alguns tijolos não perfeitamente encaixados) impossibilitavam que a busca tivesse sucesso.

Neste trabalho é utilizada uma abordagem mais robusta, através do uso de obstáculos nebulosos(*fuzzy*). Este novo tipo de abordagem permite que cada ponto do ambiente possua pertinência a algum obstáculo. A pertinência 0.0 indica que o ponto não pertence a nenhum obstáculo. A pertinência 1.0 indica que o ponto é o centro de um

obstáculo. Um obstáculo *fuzzy*, se assemelha à uma pirâmide. Levando em conta que os sensores do robô Khepera atingem valor máximo em suas leituras antes de entrar em contato com os tijolos, foi definida uma zona de incerteza ao redor de cada tijolo, englobando a incerteza das leituras dos sensores. Os tijolos presentes no ambiente são convertidos em obstáculos *fuzzy*, como apresentado a seguir.

O obstáculo *fuzzy* é uma composição de dois conjuntos *fuzzy*, X e Y, um vertical outro horizontal, com ponto mediano (x_m, y_m) idêntico. As pertinências são:

-no ponto central do obstáculo (x_m, y_m), a pertinência é máxima(1.0);

-nas bordas do obstáculo, a pertinência é 0.0.

-nos pontos intermediários(x_i, y_i), é necessário calcular o valor de pertinência aos dois conjuntos nebulosos, X e Y, e então obter a t-norma [13] entre os dois valores de pertinência, para obter a pertinência do ponto (x_i, y_i). Neste trabalho foi utilizada a t-norma mínimo (min).

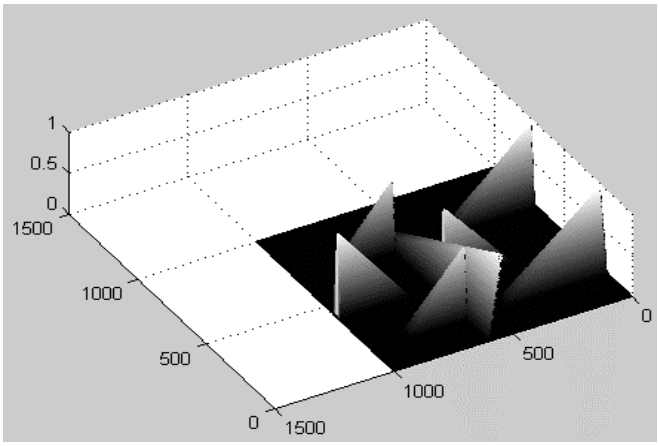
3.1 -Fusão Nebulosa de Obstáculos

Quando dois obstáculos estiverem alinhados entre si (mas não necessariamente exatamente alinhados), e próximos (isto é, houver sobreposição de suas bordas; na verdade, devem estar tão próximos que o robô não possa passar entre eles), há a possibilidade de fundi-los em um único obstáculo. Para isto, basta criar um novo obstáculo que englobe os dois outros obstáculos, preenchendo toda a área que os dois anteriores preenchiam - mesmo que seja mais largo ou mais comprido que os dois - e, portanto, os substitua sem perda de capacidade de representação do mapa.

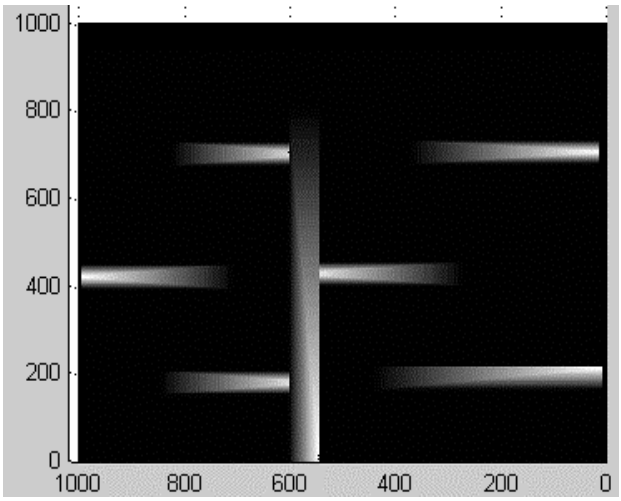
Com esta capacidade de fusão, é possível fundir vários obstáculos em seqüência, transformando paredes inteiras em um único obstáculo *fuzzy*. Isto representa uma grande vantagem com relação à representação binária, pois o obstáculo *fuzzy* pode fornecer mais claramente informação sobre quão próximo um ponto do obstáculo está em relação às bordas do mesmo. Esta informação, quando utilizada pelo algoritmo genético, traz uma grande carga de conhecimento heurístico para a busca, acelerando a localização de caminhos factíveis, isto é, caminhos que não passam sobre nenhum obstáculo.

Ainda com relação à fusão de obstáculos, se estes estiverem muito próximos, mas não for possível fundi-los(por exemplo, são obstáculos ortogonais entre si), há a possibilidade de incluir no próprio obstáculo esta informação. Como não há a possibilidade de um caminho passar entre os dois obstáculos, este lado do obstáculo deixa de estar próximo à sua borda. Para representar isto, basta mover o centro do obstáculo *fuzzy* para próximo da borda onde há proximidade com outro obstáculo. Deste modo, quanto mais longe o robô passar desta borda, maior será a possibilidade dele realmente contornar este obstáculo.

A definição dos obstáculos *fuzzy*, bem como suas capacidades de fusão e interação, permitiu construir um mapa *fuzzy* do ambiente, que é utilizado pelo algoritmo genético para encontrar o trajeto para a navegação do robô. Os obstáculos nebulosos são percorridos, e suas pertinências são mapeadas para uma matriz 1000x1000 de números de ponto flutuante, indicando as pertinências de cada ponto a obstáculos. Uma visualização tridimensional de um ambiente representado neste formato é apresentada na figura 2.



(a)



(b)

Figura 2: Mapa *Fuzzy* de um Ambiente: visão 3D (a) e superior (b).

*4.*Planejamento *off-line* do trajeto através de algoritmos genéticos

Uma solução para o problema da navegação autônoma é um caminho através do qual o robô possa chegar ao alvo. Como o número de segmentos do trajeto é desconhecido, a solução foi representada por um genoma de tamanho variável. O genoma é implementado como uma lista encadeada, onde cada elemento é uma coordenada (ponto x,y) do ambiente.

Para avaliar um trajeto, foi definida uma função objetivo, que é função da distância a ser percorrida neste caminho. Esta função objetivo irá minimizar o tamanho do trajeto, portanto o melhor trajeto será sempre o menor possível. Entretanto, caso este trajeto atravesse algum obstáculo, ele não poderá ser executado. Deste modo, é necessário incluir uma penalidade na avaliação do trajeto planejado, utilizando para isto a informação sobre os obstáculos presentes no ambiente (o mapa *fuzzy*).

Para obter esta penalidade, calcula-se a intersecção entre o trajeto planejado e os obstáculos que estão representados no mapa do ambiente. Este cálculo é realizado percorrendo todos os passos intermediários do caminho por onde o robô deve passar para percorrer o trajeto, somando a pertinência de cada ponto passado onde se encontra um obstáculo. Também é utilizado como fator de penalização da função objetivo o tamanho (em segmentos) do caminho proposto, para que o algoritmo procure encontrar soluções com a quantidade mínima de passos intermediários. Assim, a equação para a obtenção da função objetivo do trajeto *t* é:

$$f_{obj}(t) = ctte - (t_{comp} + p_1 * t_{obs} + t_{seg}) \qquad (1)$$

Sendo:

- t_{comp} =comprimento do caminho (soma dos comprimentos dos segmentos);
- t_{obs} =soma das pertinências dos pontos intermediários do caminho à obstáculos do ambiente;
- t_{seg} =número de segmentos do caminho;
- p_1 =peso da penalidade;
- $ctte$ =constante utilizada para transformar a função em uma maximização, onde o melhor caminho possui melhor avaliação.

Nos experimentos realizados, a constante *ctte* possui valor 1000, e o peso da penalidade foi 20.

Foram implementados operadores específicos para este problema, contendo informação heurística sobre o mesmo. O primeiro operador desenvolvido é o operador de mutação por inserção. Este operador insere um nó de posição aleatória no caminho. Entretanto, esta inserção ocorre sempre após um nó não factível. Um nó não factível da lista encadeada é um nó do qual parte um segmento do trajeto que passa por dentro de um obstáculo. Inserindo um outro nó, aleatoriamente, existe a possibilidade de que o segmento saindo deste nó deixe de ser infactível, como no exemplo apresentado na figura 3.

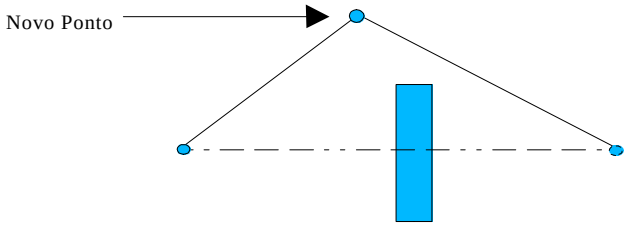


Figura 3: Exemplo do operador de inserção de ponto para transformar caminho não factível em factível.

Para o correto funcionamento deste operador, entretanto, o primeiro elemento da lista deve ser sempre um ponto alcançável a partir da posição inicial onde se encontra o robô. Para que isto seja possível, foi desenvolvido um processo específico de inicialização da população, que gera um caminho intermediário com número de nós variável, entre 0 e 15, porém garantindo que o primeiro nó é sempre alcançável diretamente a partir da posição onde se encontra inicialmente o robô. A geração dos nós para inclusão na lista é aleatória, a partir do segundo ponto da lista. O trajeto pode ter comprimento 0 pois a posição inicial e final do caminho, por serem constantes e conhecidos para cada execução do algoritmo, não são inseridos na lista, que contém apenas os pontos intermediários pelos quais o robô deve passar para caminhar pelo trajeto planejado.

Outro operador desenvolvido neste trabalho foi o operador de mutação por remoção de nó. Este operador remove o nó que se encontra logo após um ponto infactível na lista encadeada, procurando, desta forma, tornar o caminho factível. Um exemplo de seu funcionamento é apresentado na figura 4.

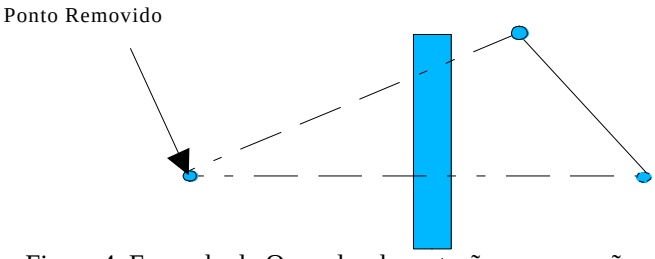


Figura 4: Exemplo do Operador de mutação por remoção

Foi desenvolvido também um operador específico de cruzamento (*crossover*) para substituir o *crossover* tradicional de um ponto. Este novo operador é ilustrado na figura 5 e se comporta da seguinte forma para gerar um novo descendente:

- seleciona-se dentre a população dois cromossomos para cruzamento, pai1 e pai2;
- varre-se os dois cromossomos, calculando o número de pontos infactíveis em cada pai;
- corta-se o pai1 no último ponto infactível, isto é, o ponto infactível mais próximo do fim do cromossomo;
- corta-se o pai2 no primeiro ponto infactível, isto é, o mais próximo do início do cromossomo;
- concatena-se a parte inicial do pai1 com a parte final do pai2, gerando o descendente;

A cada cruzamento, apenas um descendente é gerado. Este descendente terá no máximo um segmento infactível, justamente no ponto de concatenação. O outro descendente não é gerado, pois os dois outros segmentos dos pais, se concatenados, teriam um número muito grande de segmentos infactíveis.

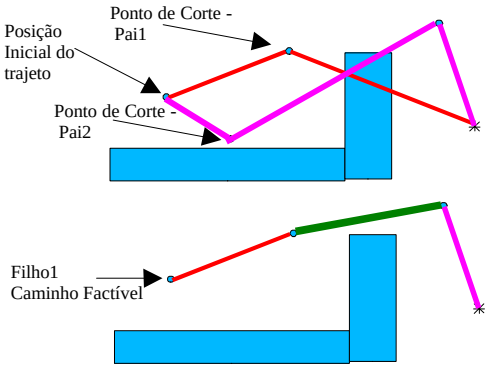


Figura 5: Exemplo do operador de cruzamento

A população utilizada pelo algoritmo é de 100 indivíduos. As probabilidades de aplicação dos operadores é de 15%, definida empiricamente.

Nas figura 6 é apresentado um exemplo de caminho gerados pelo algoritmo, e sua execução.

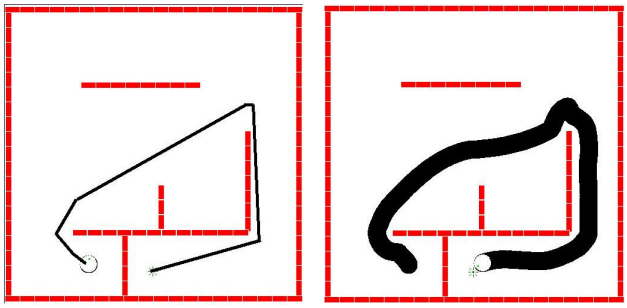


Figura 6 : Exemplo de trajetória gerada e executada em um ambiente exemplo.

Entretanto, quando se tentou aumentar a complexidade dos ambientes (figuras 8 e 9), o AG apresentou problemas, principalmente devido à convergência prematura. Mesmo alterando o método de seleção para Torneio, ou Resto Estocástico, houveram casos onde não ocorreu a convergência para caminhos factíveis.

Na busca por uma solução a este problema, optou-se pelo uso de um fator de *crowding*. Apesar da biblioteca utilizada[14] não disponibilizar diretamente esta opção, existe uma implementação de uma variação do Algoritmo Genético denominada *Deterministic Crowding*, que se comporta da seguinte maneira:

- seleciona-se (aleatoriamente) dois indivíduos de uma população;
- realiza-se o *crossover* entre ambos, gerando um descendente;
- substitui-se um dos dois indivíduos selecionados pelo descendente: esta substituição se dá baseada no grau de similaridade entre os indivíduos -o mais similar ao descendente é descartado e substituído;

Esta variação do algoritmo apresentou uma convergência muito menos acentuada e uma maior capacidade de encontrar soluções factíveis, mesmo em ambientes mais complexos, como os apresentados nas figura 8 e 9(a). Entretanto, por sua maior demora na convergência, o tempo de processamento também aumentou significativamente.

5.Execução do Trajeto por Redes Neurais

Para a execução do trajeto planejado, o robô deve se basear nas leituras dos sensores, e para isto foi utilizada uma rede neural que detecta os obstáculos e os desvia[12]. Isto permite que o robô encontre uma rota alternativa para voltar ao trajeto planejado pelo algoritmo genético e não colida com possíveis obstáculos não mapeados. No caso da detecção de um obstáculo não conhecido durante a execução da trajetória, o robô, guiado por sua rede neural de navegação reativa, tenta "contornar" este obstáculo. Isto foi realizado por duas redes neurais com treinamento *back-propagation*, cada uma com 8 entradas, 27 elementos na camada oculta e 2 na camada de saída (controle dos dois motores do robô). Uma das redes é responsável pelo desvio dos obstáculos e a outra por buscar alvos pré-estabelecidos no ambiente.

Para integrar esta proposta puramente reativa com um planejamento global de trajeto, é suficiente definir uma série de pontos intermediários. Os sensores de alvo são ativados utilizando cada um destes pontos, para que o robô se dirija para estas posições intermediárias do caminho planejado. Sempre que uma posição intermediária for alcançada pelo robô, deve-se passar a utilizar a próxima posição para ativar os sensores e assim por diante, até o objetivo ser alcançado (ponto final da navegação).

Na figura 7 é apresentado um diagrama de blocos do sistema completo. O bloco "Mapa *Fuzzy*" é responsável por obter a lista de obstáculos do simulador(2) e construir seu mapa *fuzzy* do ambiente(3). O algoritmo genético obtém do simulador as posições inicial e final do trajeto(1), utiliza o mapa *fuzzy*(3) e resulta na lista de pontos intermediários do trajeto(4). O subsistema de navegação por redes neurais percorre este caminho atuando sobre os motores do robô(5), e tratando as leituras dos sensores de obstáculos do ambiente(6).

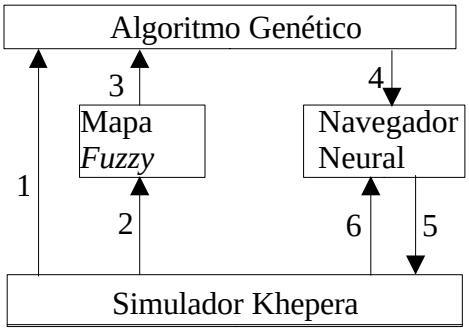


Figura 7: Diagrama de Blocos do Sistema

6.Experimentos

Foram realizados experimentos com os ambientes descritos anteriormente, por representarem um desafio aos algoritmos genéticos iniciais. As soluções encontradas, assim como as trajetórias executadas pelo navegador neural, são apresentados nas figuras 6, 8 e 9. Algumas diferenças entre o caminho planejado e o executado se devem ao caráter neural da execução. O subsistema de navegação neural também possui a capacidade de tratar localmente possíveis obstáculos não pertencentes ao mapa, como apresentado na figura 9(b).

Com relação à evolução dos algoritmos, houve uma variação no tempo de convergência proporcional à complexidade do caminho a ser encontrado. As execuções variaram dos casos onde um caminho factível existia na própria população inicial, até casos onde foram necessárias até 500 gerações para se encontrar um caminho factível (ambiente da figura 9a).

O critério de parada do algoritmo precisou ser alterado, para considerar a grande gama de situações e tempos de convergência dos algoritmos. Foi decidido finalizar o

algoritmo 20 gerações após a localização do primeiro caminho factível. Entretanto, pelo algoritmo utilizado ser o *Deterministic Crowding*, a seleção dos indivíduos é feita aleatoriamente, e este algoritmo não apresenta bom resultado na otimização dos caminhos factíveis encontrados. Para resolver este problema, foi incluído, ao final da execução do algoritmo, uma fase de pós-processamento, responsável apenas pela melhora do caminho encontrado, que é realizada através de sucessivas aplicações de um operador de mutação, que altera cada ponto para algum outro ponto na vizinhança (de distância aleatória entre 0 e 10 pontos). Este operador é aplicado múltiplas vezes, enquanto for capaz de melhorar o caminho. Após sua aplicação por 10 vezes sem melhora no caminho, o pós-processamento termina e o caminho resultante pode então ser percorrido pelo robô.

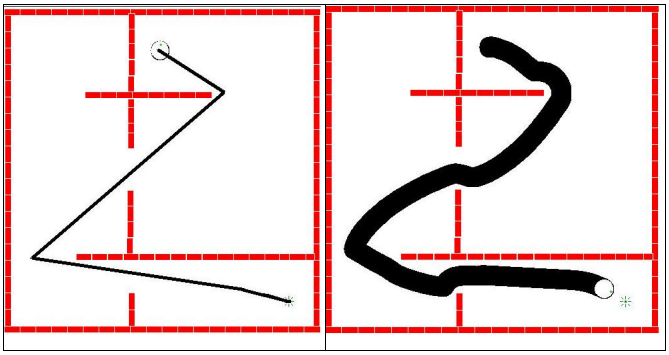


Figura 8 - Exemplo de planejamento e execução do trajeto em um ambiente de maior complexidade

7.Conclusões

Com o desenvolvimento deste projeto, confirma-se a dificuldade do problema de planejamento de trajetórias. O não conhecimento dos tipos de ambientes com os quais o robô irá se defrontar torna muito difícil incluir informações heurísticas suficientes para o planejamento de trajetória eficaz em todos os tipos de ambientes, e torna o processo de busca por soluções complexo. Entretanto, mesmo com pouca informação heurística disponível, é possível ao algoritmo genético encontrar soluções factíveis nos mais diversos ambientes.

A representação do ambiente com obstáculos *fuzzy* permitiu ao algoritmo genético uma melhor base de comparação entre os múltiplos caminhos avaliados, tornando o processo de localização de caminhos factíveis mais rápido e eficaz.

Pretende-se continuar o desenvolvimento deste trabalho, desenvolvendo novos operadores heurísticos para o algoritmo genético. Será também incluída a capacidade de inserir no mapa *fuzzy* os obstáculos não mapeados encontrados durante a navegação, de modo a atualizar o mapa durante a interação com o ambiente.

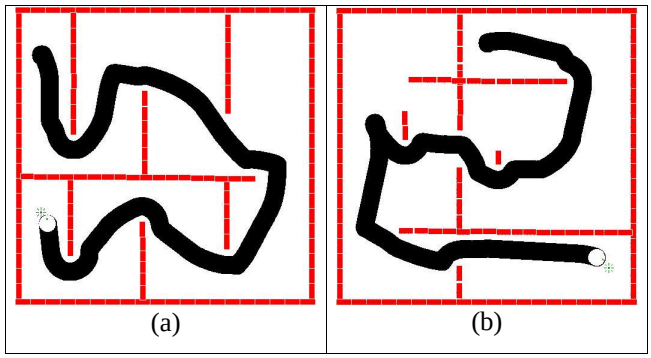


Figura 9 -Execução do trajeto em ambiente sinuoso(a), e com a presença de obstáculos não mapeados(b).

Referências

[1] Fabro, J.A., Sistemas Nebulosos e Grupos Neurais: Aplicação à Navegação Autônoma de Veículos-*Tese de Mestrado*- DCA - FEE - UNICAMP, Fev. 1996.

[2] Brooks, R.A., A robust layered control system for a mobile robot, *IEEE Journal of Robotics Automation*, 2(1), 14-23, 1986.

[3] Verschure, P. F. M. J., Formal Minds and Biological Brains, *IEEE Expert*, 66-75, October 1993.

[4] Michel, O., *Khepera Simulator version 2.0*, Homepage:http://diwww.epfl.ch/lami/team/michel/khep-sim/ , 1996.

[5] Grefenstette, J., Ramsey, C. L. and Schultz, A.C., Learning sequential decision rules using simulation models and competition, *Machine Learning* 5(4), 355-381.

[6] Grefenstette, J., Evolutionary Algorithms in Robotics, *Robotics and Manufacturing:Recent Trends in Research, Education and Applications*, M. Janshidi, C. Nguyem, eds., 65-72, ASME Press, 1994

[7] Koza, J. R., *Genetic Programming*, MIT Press, Cambridge, MA, 1996

[8] Page, W. C., McDonnell, J.R, and Anderson, B., An evolutionary programming approach to multi-dimensional path planning, *Proc. of First Conf. on Evolutionary Programming*, Evolutionary Programming Society, San Diego, CA, pages 63-70, 1992.

[9] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs* , Springer Verlag, New York, 1996.

[10] Xiao, J., Michalewicz, Z., Zhang, L. and Trojanowski, K. Adaptive Evolutionary Planner/Navigator for Mobile Robots, *IEEE Trans. on Evol. Computation*, (1) 1, 1997.

[11] Rumelhart, D.E., and McClelland, J.L., Parallel *Distributed Processing: Explorations of the microstructure of cognition*, vols. 1 and 2, MIT Press, 1986.

[12] Vaz, J. M. e Fabro, J.A., SNNAP -Sistema Neural para Navegação em Ambientes Pré-Mapeados, *Anais do IV Congresso Brasileiro de Redes Neurais*, São José dos Campos ITA/CTA, pages 118-123, 1999.

[13] Zadeh, L.A., Fuzzy Sets, *Information and control*, 8(3), 338-353, 1965.

[14] Wall, M.-*GALIB Programming Library Version 2.4.5*, HomePage: http://lance.mit.edu/ga, Feb 2000