Um Algoritmo Evolutivo para um Sistema de Controle de Navegação de Multi-Robôs Móveis Autônomos

Mauro Miazaki e Eduardo do Valle Simões
Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
Cx. Postal 668 - CEP 13560-970 -São Carlos - SP - Brasil
{mauro, simoes}@icmc.usp.br

Abstract

This paper presents the application of the Genetic Algorithms (GAs) in robotics for the development of adaptive robots able to explore their working environment. The interactions between robots and environment will determine the best adjusts of their behavior, without designer intervention. A simulator was implemented to test the system and to find adequate values for the genetic parameters. Differently than simulations, GA Systems with real robots work with very small populations and the generation time (evaluation phase) is considerably long. To optimize GA performance, considering the small number of individuals in the population, it is necessary to choose the correct parameters. Therefore, this paper presents an evaluation of the effects of modifying GA parameters on system performance, considering the algorithm convergence using different techniques of crossover, mutation and selection.

1. Introdução

A Computação Evolutiva consiste em procedimentos de busca e otimização inspirados na biologia. Estes tentam abstrair e imitar algumas das características da evolução natural com o propósito de encontrar soluções ótimas para problemas que requerem adaptação, busca e otimização [1]. A Robótica Evolutiva surgiu da utilização de técnicas de Computação Evolutiva [2] para sintetizar automaticamente controladores embarcados para equipes de robôs, com o propósito de treiná-los para desenvolver tarefas específicas. Algoritmos Genéticos têm sido empregados com sucesso no desenvolvimento automatizado de controladores para robôs em diversas formas de abordagem [3], utilizando-se simuladores [4] e um ou mais robôs reais [5].

Assim, o objetivo deste trabalho é otimizar o desempenho do AG para produzir um sistema evolutivo com robôs reais que se adaptem continuamente às variações do meio ambiente e não apenas para encontrar uma solução momentânea para um problema específico. Portanto, o AG atua continuamente em tempo de execução, adaptando os indivíduos às modificações do ambiente de trabalho. Dessa maneira, o objetivo não é desenvolver uma solução fixa e reproduzi-la várias vezes, e sim, gerar um sistema constantemente adaptativo [6].

Este artigo apresenta os resultados encontrados no estudo da aplicação de técnicas de Computação Evolutiva na robótica, onde percepção, cognição e evolucionismo são combinados para abordar aprendizado, execução de tarefas e planejamento, em ambientes adversos, complexos, imprevisíveis e dinamicamente mutáveis [7]. Foi desenvolvido um sistema evolutivo para robôs móveis autônomos adaptativos, utilizando técnicas de Computação Evolutiva para que fossem capazes de explorar ambientes e evitassem colisões com obstáculos. As próprias interações entre os robôs e o ambiente determinaram os melhores ajustes de suas configurações. Também foi desenvolvido um simulador de um sistema evolutivo com uma população de robôs. Foram implementadas e avaliadas técnicas de seleção de indivíduos para reprodução, crossover e mutação.

Este trabalho é uma continuação dos experimentos originais propostos em [8], onde foi demonstrado que um sistema evolutivo era capaz de ser aplicado na evolução do comportamento de navegação para um time de seis robôs móveis autônomos. Dando seqüência a este trabalho, a pesquisa aqui realizada analisa diversas variações do sistema evolutivo proposto, experimentando-o com várias técnicas de crossover, seleção e mutação, para adquirir dados que ajudassem a especificar corretamente o conjunto de operadores genéticos e os parâmetros para configurar um sistema evolutivo com uma população de robôs reais. É necessário ajus-

tar o sistema evolutivo para que se possa operar em tempo real em um ambiente físico, onde qualquer perda de tempo desnecessária pode representar muitas horas extras de experimentos [9].

2. Sistema Evolutivo

Para a condução dos experimentos, foi implementado um sistema evolutivo, mais precisamente, um AG, em linguagem C. Foram utilizados da aplicação desenvolvida em [8], as rotinas de transmissão e recepção de dados do robô e a interface com o usuário. Foi adicionada a esta interface uma tela de configuração dos parâmetros do AG implementado. No robô, foi utilizado um algoritmo, desenvolvido em linguagem de máquina, responsável pela sua pontuação e por dirigi-lo, conforme especificado por um cromossomo. Este algoritmo foi adaptado para permitir sua comunicação com o programa principal e para executar os experimentos.

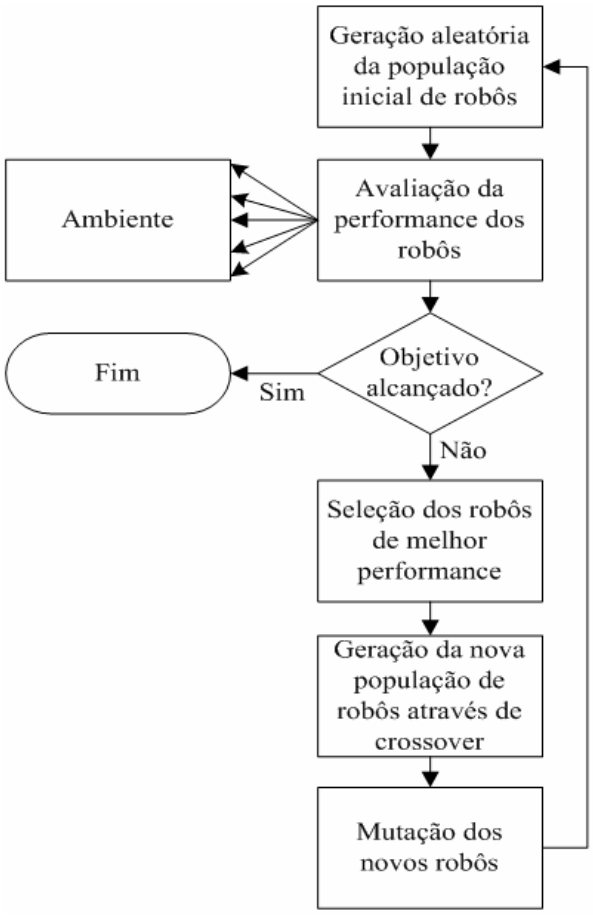

Figura 1. Fluxograma do Sistema Evolutivo Implementado

No fluxograma do sistema evolutivo apresentado na Figura 1, a população inicial é formada a partir da criação de indivíduos com cromossomos gerados aleatoriamente. Em seguida, os indivíduos são avaliados. No caso dos robôs reais, os indivíduos são transferidos para os robôs, que são avaliados em seu ambiente enquanto estiverem em operação. A exploração do ambiente, ou seja, a movimentação, é premiada com o recebimento de pontos, enquanto que colisões são penalizadas com perda de pontos [8]. Nas simulações, foi estabelecido um cromossomo meta, que deve ser alcançado pelo sistema, ou seja, a simulação continua até que pelo menos um dos indivíduos desenvolva este cromossomo. O cromossomo meta foi projetado manualmente, com os parâmetros considerados como ideais ou pelo menos próximo ao ideal. Sua finalidade é servir como objetivo a ser alcançado pela simulação, reduzindo, assim, o tempo de testes para encontrar a configuração ideal dos parâmetros do AG. Para a pontuação na simulação, foi utilizada a distância de Hamming entre o cromossomo do indivíduo e o cromossomo meta. Quanto menor a distância, maior a pontuação. Se todos os bits do cromossomo do indivíduo forem iguais aos do meta, a máxima pontuação obtida é 34 pontos.

2.1. Cromossomo

Cada robô é representado como um cromossomo de 34 genes, ou seja, uma *string* de 34 *bits*:

$$B_{01}\ B_{02}\ B_{03}\ B_{04}\ B_{05}\ B_{06}\ B_{07}\ B_{08}\ B_{09}\ B_{10}\ B_{11}\ B_{12}$$
$$B_{13}\ B_{14}\ B_{15}\ B_{16}\ B_{17}\ B_{18}\ B_{19}\ B_{20}\ B_{21}\ B_{22}\ B_{23}\ B_{24}$$
$$B_{25}\ B_{26}\ B_{27}\ B_{28}\ B_{29}\ B_{30}\ B_{31}\ B_{32}\ B_{33}\ B_{34}$$

Os primeiros nove genes representam os níveis de velocidade, que variam de 1 a 10. O nível de velocidade é calculado da seguinte forma:

$$V = (B_{01} + B_{02} + B_{03} + B_{04} + B_{05} + B_{06} + B_{07} + B_{08} + B_{09}) + 1$$

Assim, a soma do número de uns entre os bits B_{01} a B_{09} mais um, indicará o nível de velocidade do robô. Exemplo:

$$V = (0 + 1 + 1 + 0 + 1 + 0 + 0 + 0 + 1) + 1 = 5$$

No exemplo acima, temos que $B_{01} = 0$, $B_{02} = 1$, e assim por diante. Somando-se os valores dos *bits*, obtemos a velocidade, que, neste caso, será de nível 5.

Os nove genes seguintes representam a posição dos sensores de proximidade de obstáculos habilitados, que serão no máximo três entre os sensores existentes. Há oito sensores no robô, cada um identificado por um número binário (Figura 2). Cada conjunto de três bits forma um número binário correspondente a um sensor no robô. Os bits B_{10}, B_{11} e B_{12} representam o sensor S1. Já os bits B_{13}, B_{14} e B_{15}, representam o sensor S2.

E, por fim, o sensor S3 é representado por B_{16}, B_{17} e B_{18}. Poderá haver sobreposição dos sensores, caso dois ou todos os sensores coincidam (S1 = S2, S1 = S2 = S3 , por exemplo). Nestes casos, haverá apenas dois ou um sensor habilitado, ao invés de três.

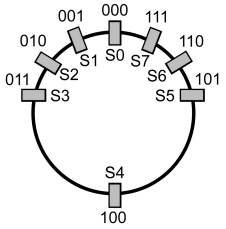

Figura 2. Disposição e representação binária dos sensores de proximidade de obstáculos no robô

Tabela 1. Representação binária dos movimentos

Sigla do Movimento	Movimento	Representação binária
F	Frente	00, 11
E	Esquerda	01
D	Direita	10

Tabela 2. Exemplo de tabela de controle

S_1	S_2	S_3	Representação binária	Movimento
0	0	0	00	F
0	0	1	01	E
0	1	0	10	D
0	1	1	10	D
1	0	0	01	E
1	0	1	01	E
1	1	0	00	F
1	1	1	01	E

2.2. Sistema Evolutivo

Na seleção dos indivíduos para o cruzamento, foram utilizadas duas técnicas encontradas na literatura, elitismo [10] e roleta [11], e algumas variações e combinações entre elas, resultando em 5 tipos de seleção:

1) **Elitismo com apenas um dos indivíduos de maior nota selecionado:** O indivíduo de maior nota é selecionado. Caso haja mais de um com a maior pontuação, um deles é escolhido aleatoriamente. O indivíduo escolhido é passado inalterado para a próxima geração e seu código genético é combinado com todos os outros da população, através de crossover, gerando os novos indivíduos que irão substituí-los.
2) **Elitismo com todos os indivíduos empatados com a maior nota selecionados:** Neste caso, todos os indivíduos empatados com a maior nota são selecionados e passados inalterados para a próxima geração. O restante da população é substituída pelos seus descendentes, gerados por crossover entre cada um deles e um dos selecionados, escolhido aleatoriamente para cada cruzamento.
3) **Roleta:** Para cada indivíduo I_j é selecionado um outro I_k para o cruzamento, gerando um descendente D_j que comporá a próxima geração, ocupando o lugar de I_j. Cada indivíduo tem uma chance de ser selecionado (ser I_k) proporcional à sua pontuação. Quanto maior a sua nota, maiores são suas chances.
4) **Roleta com Elitismo, onde apenas um dos indivíduos de maior nota é selecionado:** É uma mistura dos tipos de seleção 1 e 3. A diferença deste tipo com a roleta original é que um dos indivíduos de maior pontuação é passado inalterado para a próxima geração.
5) **Roleta com Elitismo, onde todos os indivíduos empatados com a maior nota são selecionados:** É uma mistura dos tipos de seleção 2 e 3. A diferença deste tipo com a roleta original é que todos os indivíduos empatados com a maior pontuação são passados inalterados para a próxima geração.

No crossover, os cromossomos (*strings*) de dois indivíduos são combinados para criar um novo indivíduo, que ocupará o lugar de um dos pais na próxima geração. Para cada posição da *string* do novo indivíduo, é escolhido aleatoriamente um *bit* de posição correspondente na *string* de um dos pais. Exemplo:

Pai : 01001111000101011000111010010111101
Mãe : 11011001011100100101110111011001100110
Filho : 11001101001100110100110010011001101

O sistema implementado permite ajustar o percentual de chance de que o *bit* escolhido seja do pai ou da mãe, favorecendo um ou outro. Este valor deve estar entre 0.000 (0.0%) e 1.000 (100%). Quanto maior a taxa, maiores são as chances do pai passar seus genes. Entende-se por pai neste trabalho os indivíduos escolhidos de maior *fitness* (pontuação de aptidão) para combinar seus genes, e como a mãe, o indivíduo que será substituído pelo filho na próxima geração.

Após a geração dos novos indivíduos por crossover, é aplicado neles a mutação. Cada *bit* do filho tem uma chance p_m de ser mutado, ou seja, invertido, onde p_m é um percentual escolhido entre 0.000 (0.0%) e 1.000 (100.0%).

3. Resultados Experimentais

Foram realizados diversos experimentos buscando-se encontrar a melhor combinação entre os parâmetros do sistema evolutivo. A população utilizada foi constituída de seis indivíduos e foram testadas 200 gerações. Para cada gráfico mostrado a seguir, o programa foi executado com os mesmos parâmetros 10 vezes. Desta forma, foi feita uma média dos resultados obtidos, ou seja, para cada geração, foi pega a média, a maior e a menor nota nas 10 execuções e calculou-se o valor médio de cada um. No eixo horizontal, estão as gerações e, no eixo vertical, encontra-se a pontuação de aptidão. São descritos a seguir os experimentos mais relevantes.

3.1. O Operador de Crossover

O objetivo destes experimentos foi estudar o comportamento do operador genético de crossover, sem a interferência da mutação e com os demais parâmetros fixos. Para o método de seleção, foi escolhido o elitismo, com seleção de apenas um indivíduo (tipo 1). Foram testados os percentuais de 10% a 100%, com intervalos de 10 em 10. A Figura 3 exibe alguns dos gráficos obtidos. Em nenhum dos experimentos foi alcançado o cromossomo meta. Pode-se perceber pelos gráficos que quanto maior a taxa, maior são as chances da população convergir para um máximo local, ou seja, um valor de pico no espaço de busca que, porém, não é o maior. Para percentuais baixos (10% e 30%), a população demora mais para convergir, porém atinge resultados melhores que aqueles obtidos com percentuais altos (50% e 80%), onde a convergência é rápida, mas o resultado final é bem pior. Isso acontece porque o pai possui melhor desempenho que a mãe e quando o percentual é alto, têm-se mais chances de selecionar os *bits* mais adaptados do pai. Este fato, porém, leva a uma perda da diversidade da população, que passa a ser cada vez mais similar ao pai.

3.2. O Operador de Mutação

Nestes experimentos, foi estudado o comportamento do operador genético de mutação. O operador de crossover foi desabilitado e o método de seleção utilizado foi o elitismo, com seleção de apenas um indivíduo (tipo 1). Para compor o restante da população, o operador de mutação é aplicado ao indivíduo de melhor nota selecionado diversas vezes, gerando diversos filhos, até atingir o número de indivíduos requerido. Foram realizados diversos testes com várias taxas de mutação, sendo alguns destacados nos gráficos da Figura 4. Pode-se observar nos gráficos que com taxas de mutação pequenas, em curto prazo, o crescimento da curva é lento. Porém, consegue chegar a altos valores de aptidão em longo prazo. Já no caso de taxas altas, no início a curva sobe rapidamente, mas em longo prazo, tende a estagnar em valores mais baixos do que quando utilizadas taxas mais baixas. Isto ocorre porque no início, quando é gerada a população inicial randômica, há mais genes ruins do que bons. Por isso, há uma probabilidade maior de que um gene mutado origine um gene bom. Assim, quanto maior for a taxa de mutação, maior é a chance de um gene ser mutado, o que acelera o aparecimento de genes bons. Mas em longo prazo, ao contrário, tende-se a haver mais genes bons do que ruins. Assim, quanto maior for a chance de ocorrer mutação maior é a probabilidade de originar um gene ruim.

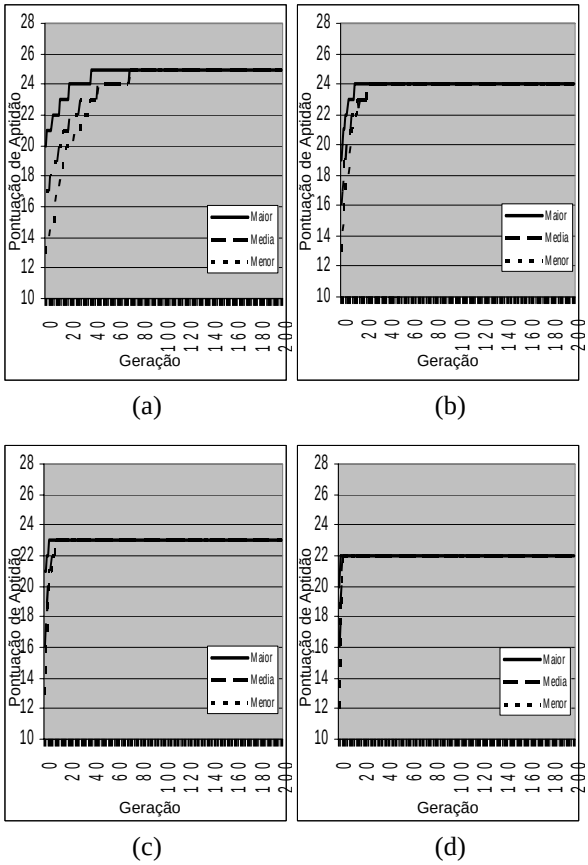

Figura 3. Experimentos com variação na taxa de crossover: (a) Taxa de crossover a 10%; (b) Taxa de crossover a 30%; (c) Taxa de crossover a 50%; (d) Taxa de crossover a 80%

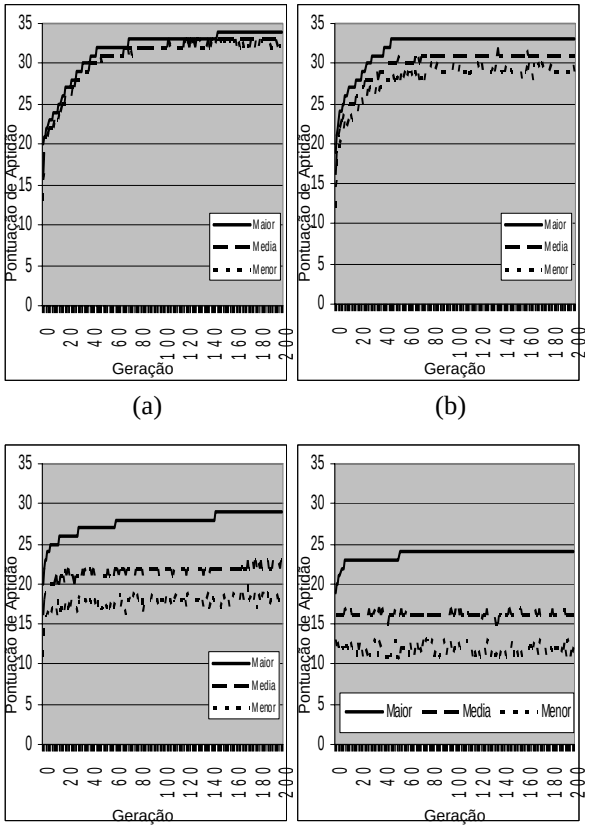

Figura 4. Experimentos com variação na taxa de mutação: (a) Taxa de mutação a 1%; (b) Taxa de mutação a 7%; (c) Taxa de mutação a 30%; (d) Taxa de mutação a 60%

3.3. Os Métodos de Seleção

Os experimentos realizados para o estudo dos métodos de seleção utilizaram a taxa de crossover em 50% e a de mutação em 3%. O método que apresentou melhores resultados foi o elitismo com apenas um dos indivíduos de maior nota selecionado (tipo 1). Em geral, este método consegue encontrar o cromossomo meta. Os tipos 2 (elitismo com todos os indivíduos de maior nota selecionados) e 5 (roleta com elitismo, onde todos os indivíduos de maior nota são selecionados) tenderam a convergir a um máximo local, principalmente com populações pequenas. Isto acontece devido a uma tendência dos indivíduos de maior nota dominarem a população, reduzindo a diversidade genética. O tipo 4 (roleta com elitismo, onde apenas um dos indivíduos de maior nota é selecionado) apresentou um desempenho razoável, embora inferior ao tipo 1.

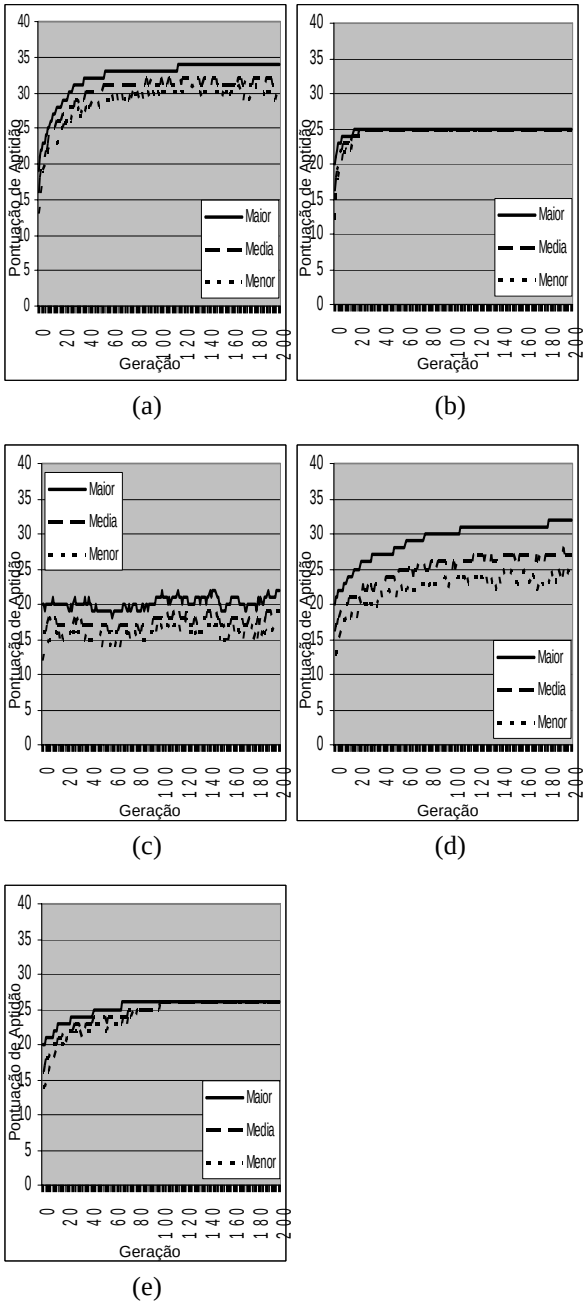

Figura 5. Experimentos com os cinco tipos de seleção: (a) Tipo 1: Elitismo com apenas um dos indivíduos de maior nota selecionado; (b) Tipo 2: Elitismo com todos os indivíduos empatados com a maior nota selecionados; (c) Tipo 3: Roleta; (d) Tipo 4: Roleta com Elitismo, onde apenas um dos indivíduos de maior nota é selecionado; (e) Tipo 5: Roleta com Elitismo, onde todos os indivíduos empatados com a maior nota são selecionados

Os três métodos anteriores algumas raras vezes conseguem alcançar o cromossomo meta. Já a roleta (tipo 3), foi o método de seleção que apresentou o pior desempenho, nunca conseguindo alcançar o cromossomo meta. A Figura 5 apresenta os gráficos dos cinco experimentos.

4. Conclusão

Através da comparação de diversas técnicas de Algoritmos Genéticos e do estudo da configuração de seus parâmetros, este trabalho demonstrou que pequenas alterações podem significar grandes mudanças de desempenho. Outra coisa percebida é a importância de se ajustar corretamente os parâmetros do AG para cada aplicação específica com que se deseja trabalhar, a fim de alcançar os resultados esperados. Este estudo levou à sugestão da utilização de um novo AG para ajustar os parâmetros do AG original, conforme a aplicação. Este novo AG fará o papel do projetista, que tem de testar várias configurações de parâmetros para depois selecionar aqueles que propiciam melhor desempenho. E esta é exatamente uma função típica que os AGs realizam muito bem. Então, nada mais lógico que utilizá-los para fazer este ajuste de forma automática.

Através dos experimentos, foi determinado que o melhor conjunto de parâmetros é a utilização do método de seleção elitismo com apenas um dos indivíduos de maior nota selecionado (tipo 1), com baixas taxas de mutação e crossover.

Foi constatado nos experimentos o amplo potencial de AGs como ferramenta de busca de soluções para problemas complexos, pois os AGs permitem estudar o comportamento de uma solução no contexto do problema e ajustar os seus parâmetros sem que se precise compreender o funcionamento da solução, bastando a análise comportamental do sistema. A próxima fase para a continuidade deste trabalho será a aplicação de um AG com os melhores parâmetros selecionados pelos experimentos aqui reportados a um time de 10 robôs reais. Isso deverá permitir o funcionamento eficiente do AG ao evoluir os controladores de navegação embarcados dos robôs, uma vez que a fase de avaliação com robôs reais pode levar até 5 minutos e qualquer perda de eficiência no AG devido à má configuração pode resultar em várias horas a mais de experimento.

Referências

[1] M. Tomassini, “A Survey of Genetic Algorithms”, *Annual Reviews of Computational Physics*, World Scientific, 1995, pp. 87-118.

[2] P. J. Bentley, *Evolutionary Design by Computers*, Morgan Kaufmann, San Francisco, 1999.

[3] E. D. V. Simões, “Robótica Evolutiva”, *Sociedades Artificiais: A nova fronteira das máquinas*, Editora ARTMED, Porto Alegre, RS, 2003, pp. 251-275.

[4] G. Baldassarre, S. Nolfi, and D. Parisi, “Evolving Mobile Robots Able to Display Collective Behaviour”, *International Workshop on Self-Organisation and Evolution of Social Behaviour*, C. K. Hemelrijk, Swiss Federal Institute of Technology, Zurich Switzerland, 2002.

[5] E. D. V. Simões, and K. R. Dimond, “An Evolutionary Controller for Autonomous Multi-Robot Systems”, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, v. 6, Tokyo, Japan, 1999, pp. 596-601.

[6] R. A. Watson, S. G. Ficici, and J. B. Pollack, “Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots”, *Robotics and Autonomous Systems*, 2002, v. 39, pp. 1-18.

[7] S. Nolfi, G. Baldassarre, D. Marrocco, “The Importance of Viewing Cognition as the Result of Emergent Processes Occurring at Different Time Scales”, *International Series on Advanced Intelligence – Dynamic Systems Approach for Embodiment and Sociality*, vol. 6, K. Murase and T. Asakura, World Scientific Publishing, Singapore, 2003.

[8] E. D. V. Simões, *Development of An Embedded Evolutionary Controller to Enable Collision-Free Navigation of a Population of Autonomous Mobile Robots*, PhD Thesis, University of Kent at Canterbury, Inglaterra, 2000, 289 p.

[9] P. Hartono, and S. Hashimoto, “Migrational GA That Preserves Solutions in Non-Static Optimization Problems”, *IEEE International Conference on Systems, Man and Cybernetics*, Tucson-AZ, USA, ISBN 0-7803-7089-9, 2001, pp. 255–260.

[10] K. De Jong, *The Analysis and Behaviour of a Class of Genetic Adaptative Systems*, PhD thesis, University of Michigan, 1975.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.