UM SISTEMA EVOLUTIVO EMBARCADO PARA CONTROLAR UMA POPULAÇÃO DE ROBÔS MÓVEIS USANDO PROGRAMAÇÃO GENÉTICA

Anderson Luiz Fernandes Perez*, Guilherme Bittencourt*, Mauro Roisenberg†

*Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Santa Catarina
Florianópolis, SC, Brasil

†Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, SC, Brasil

Emails: anderson@das.ufsc.br, gb@das.ufsc.br, mauro@inf.ufsc.br

Abstract— In this paper, an embodied evolutionary system, able to control a population of mobile robots, is proposed. This system should be able to execute tasks such as collision-free navigation, box pushing and predator and prey. The proposed system has the following characteristics: i) it extends the traditional genetic programming algorithm to allow the evolution in a population of physical robots; ii) the evolutionary process occurs in an asynchronously way among the robots in the population; iii) it is fail-safe, therefore it allows the continuation of the evolutionary process even if only one robot remains in the population of robots; iv) it saves the information about the more adapted individuals in a kind of memory; v) it has an execution and management environment that is independent of the evolutionary process.

Keywords— Genetic Programming, Evolutionary Robotic, Embodied Evolutionary System.

Resumo— Neste artigo é proposto um sistema evolutivo embarcado capaz de controlar uma população de robôs móveis para executar tarefas do tipo navegação livre de colisões, empurrar uma caixa e presa e predador. O sistema proposto possui as seguintes características: i) estende o algoritmo tradicional da programação genética para suportar a evolução em uma população de robôs reais; ii) o processo evolutivo acontece de forma assíncrona entre os robôs da população; iii) é tolerante a falhas, pois permite a continuação do processo evolutivo mesmo que só reste um único robô na população de robôs; iv) guarda informações sobre os indivíduos mais adaptados em uma espécie de memória; v) possui um ambiente de execução e gerenciamento independente do processo evolutivo.

Keywords— Programação Genética, Robótica Evolutiva, Sistema Evolutivo Embarcado.

1 Introdução

Programar um robô para executar uma determinada tarefa, muitas vezes, exige que o programador tenha um bom conhecimento sobre o domínio do problema, ou seja, o programador é responsável por descrever todos os passos necessários que o robô deverá tomar para executar a tarefa.

Muitas vezes o programador não possui meios de prever possíveis problemas que o robô poderá enfrentar, principalmente se o ambiente no qual o robô irá atuar for dinâmico ou não estruturado. Nesse caso, é importante que o robô tenha um alto grau de autonomia e seja dotado de mecanismos que permitam sua auto-adaptação para poder tomar decisões em situações para as quais ele não foi programado.

Para tornar o sistema de controle de um robô móvel mais dinâmico, ou seja, adaptável, é necessário utilizar alguma técnica de desenvolvimento que permita que o sistema se modifique ao longo de sua execução. A Computação Evolutiva (CE) (Fogel, 2000), através de seus Algoritmos Evolutivos, possibilita o desenvolvimento de sistemas de controle adaptativos para robôs móveis (Pollack et al., 2000).

A Robótica Evolutiva (RE) (Nolfi and Floreano, 2002) visa o desenvolvimento de sistemas de controle adaptativos baseado nas técnicas de CE. Uma das áreas de pesquisa da RE é a Evolução Embarcada (EE) (Watson et al., 2002), que une a RE à robótica cooperativa (Ficici et al., 1999), (Cao et al., 1997). Na EE o processo evolutivo acontece entre os robôs, de uma população de robôs móveis, ou seja a reprodução acontece entre os indivíduos que fazem parte da população de robôs.

A Programação Genética (PG) é uma técnica da CE que visa a geração automática de programas de computador (Koza, 1992). O principal objetivo da PG é ensinar computadores a se programar, isto é, a partir de especificações de comportamentos primários, o computador deve ser capaz de gerar um programa que satisfaça algumas condições que visam a solução de alguma tarefa ou problema.

Neste artigo é descrito um Sistema de Controle Evolutivo (SCE) para uma população de robôs móveis. O SCE é composto de duas partes principais, que são: o PGD (Programação Genética Distribuída) e o SEGS (Sistema de Execução, Gerenciamento e Supervisão). O artigo está organizado da seguinte forma: na seção 2 apresenta-se a descrição do Sistema de Controle Evolutivo; nas seções 3 e 4 são descritos detalhes

do mecanismo do PGD e do SEGS, respectivamente; na seção 5 são descritos alguns problemas escolhidos para a avaliação do sistema proposto; na última seção, apresentam-se as conclusões.

2 Sistema de Controle Evolutivo

O Sistema de Controle Evolutivo (SCE) proposto é baseado no algoritmo da Programação Genética (PG) (Koza, 1992). O objetivo é que somente com o uso da PG seja possível construir um sistema de controle para uma população de robôs móveis que interajam entre si para realizarem alguma tarefa.

O SCE é composto de dois módulos principais. O primeiro, chamado de PGD (Programação Genética Distribuída), é o algoritmo responsável por todo o processo evolutivo do sistema de controle dos robôs. O PGD é uma extensão do algoritmo tradicional da PG para suportar a evolução do sistema de controle dos robôs que fazem parte da população de robôs móveis. O segundo módulo, chamado de SEGS (Sistema de Execução, Gerenciamento e Supervisão), é responsável pela execução e o gerenciamento do PGD.

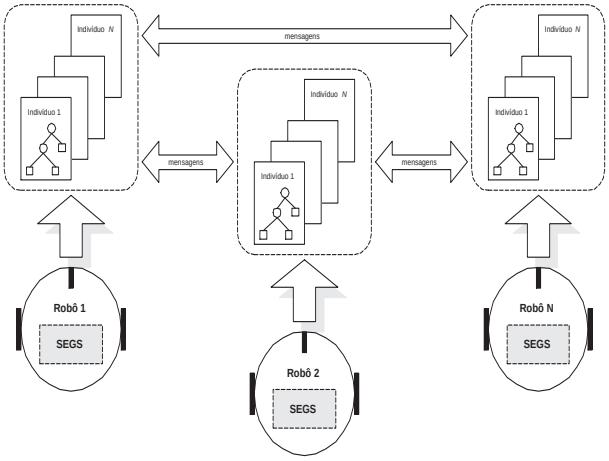

Figura 1: Representação esquemática do SCE.

A Figura 1 ilustra, genericamente, o funcionamento do SCE proposto. Em cada robô é executado o SEGS, responsável pelo gerenciamento do sistema de controle evolutivo. Cada robô possui uma população local [1] que interage com a população local dos outros robôs. A cada geração, partes do melhor indivíduo de cada robô são enviadas para todos os outros robôs.

A vantagem em utilizar PG para o desenvolvimento de sistemas de controle para robôs móveis é que as estruturas (funções e terminais) manipuladas são de alto nível, o que pode acarretar num melhor desempenho no processo evolutivo. O espaço de busca tende a ser menor, quando comparado com outras técnicas em CE que manipulam estruturas de mais baixo nível como os Algoritmos Genéticos (AGs), por exemplo.

Uma desvantagem da PG com relação a outras técnicas de CE é que o processo de definição dos terminais e principalmente das funções requer maior atenção e experiência por parte do programador. Funções e terminais desenvolvidos para um problema em particular podem não ser aplicados a outros tipos de problemas. Portanto, o conjunto de funções e de terminais devem ser projetados para serem o mais abrangentes possível, ou seja que com poucos modificações ou até mesmo nenhuma, seja possível utilizar as mesmas funções e terminais para vários tipos de problemas, bastando apenas redefinir a função de avaliação.

As Seções 3 e 4 descrevem detalhadamente o funcionamento do PGD e do SEGS, respectivamente.

3 Programação Genética Distribuída - PGD

O PGD (Programação Genética Distribuída) é uma extensão do algoritmo tradicional da PG. O PGD é baseado no Microbial GA (Harvey, 2001), uma variação do AG, seu funcionamento é semelhante a recombinação (infecção) genética que acontece nas bactérias onde segmentos do DNA são transferidos entre dois membros da população.

No PGD são considerados dois conjuntos de populações. O primeiro, chamado de conjunto local ou $P_{local_{R_i}}$, refere-se a população local de cada robô R_i, isto é, o conjunto de indivíduos ou soluções candidatas que estão embarcadas no robô. Cada $x \in P_{local_{R_i}}$ representa uma solução candidata a um problema. O segundo conjunto, chamado de conjunto total ou P_{total}, é formado pela união de todas as populações locais de cada robô, onde: $P_{total} = P_{local_{R_1}} \cup P_{local_{R_2}} \cup \cdots \cup P_{local_{R_n}}$. O processo evolutivo ocorre sempre considerando a população total, ou seja, partes (subárvores) de um indivíduo local de um determinado robô podem ser consideradas no processo evolutivo da população local de outro robô.

Ao contrário de outras abordagens em Evolução Embarcada (EE), no PGD o processo evolutivo é assíncrono, isto é, não é necessário que dois robôs se sincronizem para se reproduzirem. No PGD partes de um indivíduo mais adaptado são enviados para todos os outros robôs. A seqüência de passos do PGD é a seguinte:

1. Criar aleatoriamente uma população de programas;
2. Executar iterativamente os seguintes passos até que algum critério de parada seja satisfeito:

[1] A população local é o conjunto de programas candidatos a resolverem um problema. Esses programas estão embarcados no robô. Um problema é uma tarefa qualquer que o robô deve executar. Por exemplo, navegação livre de colisões.

(a) Avaliar cada programa da população através de uma função heurística (*fitness*), que expressa a sua aptidão, ou seja, o quão próximo o programa está da solução ideal;

(b) Recebe[2] partes de um indivíduo remoto[3] enviadas por outro robô;

(c) Seleciona os t melhores indivíduos da população local usando o método de seleção por torneio;

(d) Seleciona aleatoriamente uma parte do melhor indivíduo local (mais adaptado) e a envia em *broadcast* (difusão) para os outros robôs;

(e) Compara se o *fitness* do pior indivíduo selecionado localmente é menor que o *fitness* do indivíduo remoto. Se sim, executa o operador de mutação substituindo partes do indivíduo local pelas partes recebidas de um indivíduo remoto;

(f) Executa os operadores de cruzamento e mutação;

3. Retornar com o melhor programa encontrado.

O método de seleção empregado no PGD é a seleção por torneio com a manutenção dos pais após o cruzamento. Esta é uma técnica elitista conhecida na bibliografia da área como *steady-state genetic programming*. As partes recebidas remotamente são adicionados a árvore do pior indivíduo, dos t melhores selecionados, obedecendo a equação 1:

$$M(A) = \left\{ \begin{array}{ll} Muta(A) & \text{if } FitnessR > FitnessL \\ A & \text{if } FitnessR \leq FitnessL \end{array} \right\} \quad (1)$$

onde, *FitnessR* é o valor do *fitness* do indivíduo remoto, que enviou uma parte de sua árvore. *FitnessL* é o valor do *fitness* do indivíduo local. Muta(A) é função de mutação, onde aleatoriamente é escolhido uma parte da árvore A do indivíduo local para ser substituída pela parte da árvore do indivíduo remoto.

As mensagens trocadas entre os robôs devem conter o *fitness* e uma parte da árvore (sub-árvore) do indivíduo da população local. Para isso, todas as funções e terminais recebem uma identificação numérica única, um número par para cada função e um número ímpar para cada terminal. Por exemplo, na tarefa de forageamento, onde um conjunto de robôs deve navegar por um ambiente a procura de comida, o conjunto de funções e terminais poderiam ser definidos conforme a Tabela 1.

Na Tabela 1 a coluna Id. representa a identificação das funções e dos terminais. A Figura 2 ilustra um exemplo do funcionamento do PGD.

Na Figura 2 (a), o robô 1, está enviando parte de seu árvore para o robô 2 (Figura 2 (b)). Nesse exemplo, o indivíduo da população local do robô 1, que está sendo utilizado, possui um valor de *fitness* 15. E o indivíduo do robô 2 possui o valor de *fitness* 10. Aleatoriamente, parte da árvore do indivíduo do robô 1 é enviada para o robô 2. A mensagem enviada é formada pelos seguintes elementos: **M = {15,2,3,7}**. Isto é, o valor do *fitness*, ComidaFrente, VirarEsquerda e Retornar. Após a comparação dos valores de *fitness*, parte da árvore do indivíduo do robô 2, que também é escolhida aleatoriamente, é substituída pela parte da árvore do robô 1 (Figura 2 (c)).

É importante ressaltar que para um correto funcionamento do PGD todos os robôs devem conter os mesmos conjuntos de funções e terminais, ou pelo menos, devem utilizar as mesmas funções e terminais para um problema em particular.

Diferentemente de outras abordagens em EE, o PGD garante a continuidade do processo evolutivo do sistema de controle, mesmo quando houver algum problema com os outros robôs que fazem parte da população de robôs. Isso é possível porque cada robô possui uma população local de programas, o que garante a continuidade do processo evolutivo.

4 Sistema de Execução, Gerenciamento e Supervisão - SEGS

O SEGS (Sistema de Execução, Gerenciamento e Supervisão) é o sistema responsável pelo gerenciamento do processo evolutivo que acontece de maneira embarcada em cada robô. Além do PGD o SEGS possui outros componentes necessários para um correto funcionamento do sistema de controle de cada robô. A Figura 3 ilustra os componentes do SEGS e as relações entre eles.

Abaixo segue a descrição completa do objetivo e o funcionamento de cada componente que faz parte do SEGS e da interligação entre eles (quando houver).

Controle Evolutivo (CE): é o componente principal do SEGS, é nele que está implementado o PGD. O CE funciona como um interpretador e é responsável por criar, aleatoriamente, a população local de indivíduos através das bibliotecas de funções e terminais. A cada nova geração os indivíduos gerados são testados, isto é, executados pelo CE. As FADs (Funções Automaticamente Definidas ou ADFs (*Automatically Defined Functions*) do Inglês) também são gerenciadas por esse componente. O CE está ligado ao Gerenciador de

[2]Em cada ciclo de execução o PGD considera somente uma mensagem recebida. Cada nova mensagem, contendo partes de um indivíduo remoto, é armazenada em buffer local. O buffer é sobrescrito sempre que uma nova mensagem é recebida.

[3]Um indivíduo remoto é um programa que faz parte da população local de outro robô.

Tabela 1: Conjunto de funções e terminais

Funções			
Nome	**Aridade**	**Id.**	**Definição**
ComidaFrente	2	2	Se encontrou comida, executa nodo (função ou terminal) da esquerda; senão, executa nodo da direita.
Prog2	2	4	Executa dois ramos da árvore
Prog3	3	6	Executa três ramos da árvore
Terminais			
VirarDireita	0	1	Faz o robô virar a direita (15 graus).
VirarEsquerda	0	3	Faz o robô virar a esquerda (15 graus).
SeguirEmFrente	0	5	Faz o robô seguir em frente (300ms).
Retornar	0	7	Faz o robô retornar, dar a ré (300ms).

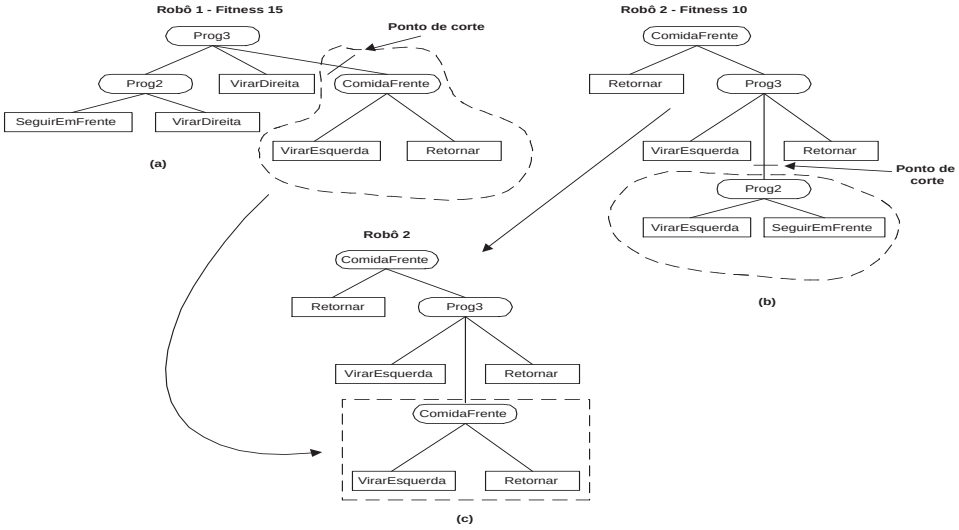

Figura 2: Exemplo de funcionamento do PGD.

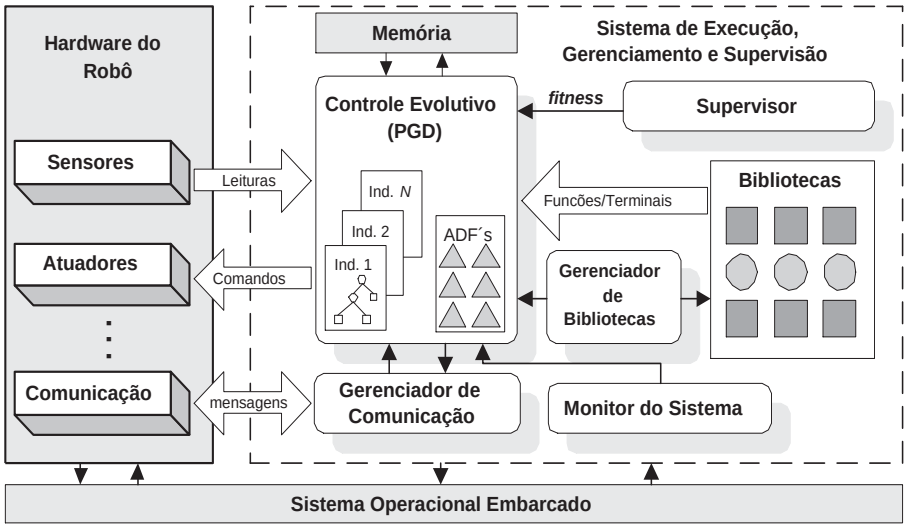

Figura 3: Estrutura básica do SEGS.

Comunicação, ao Gerenciador de Bibliotecas e à Memória.

Memória: o objetivo da memória é armazenar a representação dos indivíduos mais adaptados em cada geração. Por exemplo, considerando o exemplo descrito na Seção 3, o mel-

hor indivíduo do Robô 1, de acordo com as regras de representação para as funções e os terminais, poderia ser representado da seguinte forma: **{6, 4, 5, 1, 1, 2, 3, 7}**, o que representa a seguinte estrutura em termos de funções e terminais: (Prog3, Prog2, SeguirEmFrente, VirarDireita, ComidaFrente, VirarEsquerda, Retornar). A representação **{6, 4, 5, 1, 1, 2, 3, 7}** mais o valor do *fitness* são armazenados em memória para poderem ser utilizados num processo de recuperação em caso de falhas do CE ou até mesmo para otimizar tarefas que envolvem mais de uma competência, como por exemplo, desviar de obstáculos e deslocar um objeto de um lugar para outro. Nesse caso, a memória funciona como uma espécie de "fotografia" do sistema, podendo ser utilizada para acelerar o processo de aprendizagem através de experiências realizadas no passado.

Gerenciador de Comunicação (GC): é o componente responsável pelo envio e o recebimento das mensagens que são trocadas entre os robôs. No processo de envio de mensagens, o CE repassa para o GC a seqüência de funções e terminais e o valor do *fitness* que serão enviados para os outros robôs. O GC monta uma mensagem contendo essas informações e a envia para os outros robôs. Na recepção, o GC recebe as mensagens enviadas pelos outros robôs e as repassa para o CE.

Gerenciador de Bibliotecas (GB): esse componente gerencia os conjuntos de terminais e funções de cada robô. Nesse componente todas as funções e terminais são identificadas através de um identificador único, um número par para as funções e um número ímpar para os terminais, para poderem ser enviadas para outros robôs pelo GC. Ter as bibliotecas separadas do sistema evolutivo representa uma vantagem adicional, pois é possível, a qualquer momento adicionar ou remover funções ou terminais sem a necessidade de redefinições no sistema de controle. Por exemplo, caso o robô receba um conjunto de sensores novos com novas funcionalidades, basta adicionar essas informações ao conjunto de terminais nas bibliotecas do SEGS. Isso faz com que os indivíduos gerados pelo CE se adaptem às mudanças de características no hardware do robô (morfologia).

Supervisor: é responsável por avaliar e atribuir um valor de *fitness* para cada indivíduo, isso é feito através de um método de punição e recompensa. Para um correto funcionamento desse método, para cada tarefa a ser realizada pelo robô, deve ser definido como e quando acontece a recompensa e a punição. Por exemplo, se a tarefa é a navegação livre de colisões, a recompensa e a punição podem ser definidas de acordo com a quantidade de colisões do robô. A cada choque com um obstáculo, o valor do *fitness* é decrementado (punição), caso contrário, o valor vai sendo incrementado de tempos em tempos (recompensa).

Monitor: é o componente responsável por avaliar a execução do sistema. Caso o sistema fique inativo, isto é, o robô fique parado por um longo período de tempo, o monitor ativa um processo de re-inicialização do CE. Quando acontece a reinicialização do CE, a imagem do melhor indivíduo, que está armazenada na memória, é recuperada e o processo evolutivo inicia-se a partir deste indivíduo, isto é, a população local é completada através do uso de partes da árvore deste indivíduo ao invés de começar de uma população aleatória como acontece no algoritmo tradicional da PG.

O SEGS pode ser implementado diretamente sobre o hardware do robô ou como uma tarefa executando sobre um sistema operacional embarcado, tal como o CubeOS (Kenn, 2001), o Arena (Kingsbury et al., 1998) e o Robios (Bräunl, 2006). A vantagem em executar sobre um sistema operacional embarcado é poder tirar proveito das funcionalidades deste, como controle de concorrência, gerência de memória e a gerência do hardware.

5 Avaliação do Sistema

Com o objetivo de avaliar o sistema de controle proposto pretende-se realizar alguns experimentos, tanto em simulador como em robôs reais, em três problemas distintos da robótica móvel.

Navegação Livre de Colisões: nesse tipo de problema os robôs devem vagar por um ambiente contendo vários obstáculos com diferentes tamanhos e formatos. O objetivo de cada robô é detectar e desviar dos obstáculos e também dos outros robôs.

Empurrar a Caixa: esse é um típico problema de cooperação, onde um grupo de robôs deve empurrar uma caixa de um local para outro no ambiente. Cada robô deve, primeiramente, localizar e se aproximar da caixa, depois, com o auxílio dos outros robôs (cooperação), deslocar a caixa para uma outra posição. Esse problema pode ser implementado de duas maneiras distintas. Na primeira, considerar que no ambiente somente existirão os robôs e a caixa. Na segunda, considerar que além dos robôs e a caixa, também existirão no ambiente alguns obstáculos que devem ser evitados pelos robôs.

Presa e Predador: nesse tipo de problema existem duas classes distintas de robôs. A primeira é formada pelos predadores, que têm por objetivo primário "caçar" as presas. A segunda, é formada pelas presas que devem evitar a captura pelo predador. O comportamento de uma classe de robô afeta o comportamento da outra (co-evolução), pois a adaptação do sistema de controle da Presa para fugir acarretará a adaptação do sistema de controle do predador para "caçar" e vice versa.

Os três problemas descritos acima foram escolhidos para os experimentos de avaliação por serem bastante difundidos na área de robótica móvel. Por conta disso, existem diferentes abordagens para solucionar tais problemas, como por exemplo as descritas por: (Floreano and Nolfi, 1997), (Simões and Dimond, 2001) e (Zhang and Cho, 2000). O objetivo é confrontar os resultados obtidos com o sistema proposto com os resultados obtidos com outras abordagens.

6 Conclusão

Nesse artigo foi descrita a proposta de um Sistema Evolutivo Embarcado, baseado no algoritmo da Programação Genética, para controlar uma população de robôs móveis.

O sistema proposto possui as seguintes características: (i) estende o algoritmo tradicional da PG para suportar a evolução em uma população de robôs reais; (ii) o processo evolutivo acontece de forma assíncrona entre os robôs e continua mesmo quando acontecer um problema com os outros robôs da população de robôs móveis, pois cada robô possui sua própria população local de programas; (iii) o uso da memória para guardar informações sobre a estrutura da árvore dos indivíduos mais adaptados provê um mecanismo de recuperação de falhas e permite a maximização do processo evolutivo em tarefas que exigem mais de uma competência; (iv) ambiente de execução e gerenciamento independente do processo evolutivo.

O sistema descrito está em fase de desenvolvimento e será avaliado tanto em ambiente simulado quanto em robôs reais. Os comportamentos primitivos como ComidaFrente, VirarDireita, VirarEsquerda, SeguirEmFrente e Retornar já foram implementados em robôs do tipo Eyebot (Bräunl, 2006) do Departamento de Automação e Sistemas da UFSC. Para os experimentos também serão utilizados os robôs do Instituto de Controle de Processos e Robótica da Universidade de Karlsruhe na Alemanha.

Agradecimentos

O autor, Anderson Luiz Fernandes Perez, agradece ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) pela bolsa de doutorado.

Referências

Bräunl, T. (2006). Eyebot online documentation, http://robotics.ee.uwa.edu.au/eyebot/.

Cao, Y. U., Fukunaga, A. S. and Kahng, A. B. (1997). Cooperative mobile robotics: Antecedents and directions, *Autonomous Robots* **4**: 7–27.

Ficici, S., Watson, R. and Pollack, J. (1999). Embodied evolution: A response to challenges in evolutionary robotics, pp. 14–22.

Floreano, D. and Nolfi, S. (1997). Adaptive behavior in competing co-evolving species, *in* P. Husbands and I. Harvey (eds), *Fourth European Conference on Artificial Life*, The MIT Press, Cambridge, MA, pp. 378–387.

Fogel, D. (2000). What is evolutionary computation?, *IEEE Spectrum* **37**(2): 28–32.

Harvey, I. (2001). Artificial evolution: A continuing saga in evolutionary robotics: From intelligent robots to artificial life, *in* T. Gomi (ed.), *8th International Symposium on Evolutionary Robotics (ER2001)*, Springer-Verlag Lecture Notes in Computer Science LNCS 2217 2001.

Kenn, H. (2001). *CubeOS: A Component-based operating system for autonomous system*, PhD thesis, Vrije University, Brussels.

Kingsbury, S., Mayes, K. and Warboys, B. (1998). Real-time arena: A user-level operating system for co-operating robots, *of The Interantional Conference on Prallel and Distributed Processing Techniques and Applications (PDPTA)*, pp. 1844–1850.

Koza, J. R. (1992). *Genetic Programming: A Paradigm for Genetically Breeding Computer Population of Computer Programs to Solve Problems*, MIT Press.

Nolfi, S. and Floreano, D. (2002). Synthesis of autonomous robots through evolution, *Trends in Cognitive Science* **6**(1): 31–36.

Pollack, J. B., Lipson, H., Ficci, S., Funes, P., Hornby, G. and Watson, R. A. (2000). Evolutionary techniques in physical robotics, *ICES*, pp. 175–186.

Simões, E. D. V. and Dimond, K. R. (2001). Embedding a distributed evolutionary system into population of autonomous mobile robots, *Proceeding of The 2001 IEEE System, Man, and Cybernetics Conference.*

Watson, R., Ficici, S. and Pollack, J. (2002). Embodied evolution: Distributing an evolutionary algorithm in a population of robots, *Robotics and Autonomous Systems* **39**(1): 1–18.

Zhang, B.-T. and Cho, D.-Y. (2000). Evolving complex group behaviors using genetic programming with fitness switching, *Artificial Life and Robotics* **4**(2): 103–108.