# Controle Inteligente do Caminhar de Robôs Móveis Utilizando Algoritmos Genéticos e Redes Neurais Artificiais

**Milton Roberto Heinen**[1] **e Fernando Santos Osório**[2]

[1]Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15064 – CEP 91501-970 – Porto Alegre – RS – Brasil

[2]Computação Aplicada - Universidade do Vale do Rio dos Sinos (UNISINOS)
CEP 93022-000 – São Leopoldo – RS – Brasil

mrheinen@inf.ufrgs.br, fosorio@unisinos.br

***Resumo.*** *Este artigo descreve o simulador LegGen, que realiza a configuração automática do caminhar em robôs simulados dotados de pernas. No simulador LegGen, algoritmos genéticos são utilizados para a evolução dos parâmetros do caminhar em robôs móveis simulados através da biblioteca de simulação baseada em física Open Dynamics Engine (ODE). Diversos experimentos foram realizados utilizando duas estratégias de controle: (i) um autômato finito; (ii) uma rede neural do tipo Elman. Diversos experimentos estatisticamente válidos foram realizados, que permitiram constatar a superioridade do controlador baseado em redes neurais artificiais na tarefa em questão.*

## 1. Introdução

Os robôs móveis autônomos tem atraído a atenção de um grande número de pesquisadores, devido ao desafio que este novo domínio de pesquisas propõe: dotar sistemas de uma capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos [Medeiros 1998]. Atualmente os robôs móveis atuam em diferentes áreas, como desarmamento de bombas, exploração de ambientes hostis, e a condução de veículos de forma semi-autônoma [Batavia et al. 1996, Heinen et al. 2006]. A maioria dos robôs móveis desenvolvidos até o momento se deslocam através de rodas, o que facilita bastante o controle, mas impede que eles sejam capazes de se deslocarem em ambientes irregulares que possuam desníveis e degraus [Knight and Nehmzow 2002]. Assim, para que um robô móvel possa se deslocar livremente em ambientes irregulares, ele precisa do mesmo mecanismo de locomoção utilizado pelos seres humanos e um grande número de seres vivos, ou seja, ele precisa ser dotado de pernas ou articulações [Bekey 2005, Pfeifer and Scheier 1999].

Entretanto, o desenvolvimento de robôs com pernas que consigam se deslocar livremente em ambientes irregulares é uma tarefa bastante árdua, que exige a configuração de diversos parâmetros relativos ao caminhar. A configuração manual destes parâmetros exige muitas horas de um especialista humano, e os resultados obtidos são sub-ótimos e dependentes da arquitetura específica do robô [Chernova and Veloso 2004]. Desta forma, seria bastante útil realizar a configuração do caminhar de forma automática, através do uso de técnicas de aprendizado de máquina (*machine learning* – ML) [Mitchell 1997]. Uma das técnicas de ML mais adequadas para este tipo de problema são os algoritmos genéticos (*genetic algorithms* – GA) [Goldberg 1989], pois segundo a teoria da evolução natural das

espécies [Darwin 1859], os mecanismos de locomoção utilizados pelos seres vivos são um resultado da Evolução Natural, o que torna o uso de GAs uma solução biologicamente inspirada [Pfeifer and Scheier 1999]. Do ponto de vista computacional, os GAs também são bastante adequados, pois conseguem realizar uma busca multi-critério em um espaço multi-dimensional, e não necessitam de informações locais para a correção do erro nem do cálculo do gradiente [Mitchell 1996].

Em trabalhos anteriores, foi realizado um estudo comparativo entre modelos de robôs de quatro e seis pernas [Heinen and Osório 2006c, Heinen and Osório 2007] e entre diferentes funções de *fitness* [Heinen and Osório 2006a, Heinen and Osório 2006b]. Este artigo descreve um estudo comparativo entre as seguintes estratégias de controle:

- Controle baseado em um autômato finito;
- Controle baseado em redes neurais artificiais.

Em ambas as estratégias, a otimização dos parâmetros é realizada através de algoritmos genéticos. Este artigo está estruturado da seguinte forma: A Seção 2. descreve o uso de simulação baseada em física e a biblioteca *Open Dynamics Engine* (ODE); A Seção 3. descreve diversos trabalhos do estado da arte na área em questão; A Seção 4. descreve o modelo proposto, bem como protótipo implementado para a validação do mesmo; A Seção 5. descreve os experimentos realizados e os resultados obtidos; Por último, a Seção 6. traz as conclusões finais e as perspectivas futuras.

## 2. Simulação de robôs móveis

Quando se deseja realizar experimentos em robótica móvel, duas alternativas são possíveis [Pfeifer and Scheier 1999]: (i) realizar os experimentos diretamente em um robô real; ou (ii) realizar os experimentos utilizando um robô simulado em um ambiente virtual realista. A utilização de um robô real possui a vantagem de tornar reais os resultados obtidos, mas o uso de simulação possui as seguintes vantagens [Law and Kelton 2000]:

- Na simulação não existe o risco de se danificar o robô;
- A troca ou recarga de baterias e a manutenção do robô não são necessárias;
- O reposicionamento do robô pode ser realizado sem a intervenção humana;
- O relógio da simulação pode ser acelerado, reduzindo o tempo de aprendizado;
- Pode-se testar várias arquiteturas e modelos diferentes de robôs antes da construção física, e assim descobrir com antecedência o modelo de robô mais eficiente.

Para o desenvolvimento de um simulador de robôs móveis, o uso de uma biblioteca de simulação baseada em física é bastante útil, como pode ser visto na próxima seção.

### 2.1. Simulação baseada em física

Para que uma simulação de robôs móveis seja realista, diversos elementos do mundo real precisam estar presentes no modelo de simulação, para que os corpos se comportem de forma similar à realidade. Em especial, é necessário que um robô sofra quedas se não for bem controlado ou se não estiver bem posicionado, e que colida contra os objetos do ambiente de forma realista. Para que isto ocorra, é necessário que as leis da física sejam modeladas no ambiente de simulação (gravidade, inércia, fricção e colisão) [Osório et al. 2006]. Atualmente existem várias bibliotecas de software disponíveis para a implementação de simulações baseadas em física. Após o estudo de diversas possibilidades, optou-se pela utilização de uma biblioteca de código aberto e gratuita chamada

*Open Dynamics Engine* (ODE)[1], que permite a realização de simulações da dinâmica de corpos rígidos articulados com bastante realismo. Utilizando a ODE, é possível criar diversos corpos rígidos, e estes podem ser conectados através de juntas de vários tipos. Para a movimentação das juntas, é possível que sejam aplicadas forças diretamente aos corpos, ou podem ser utilizados os motores angulares que estão disponíveis no ambiente ODE. Um motor angular é um elemento que pode ser conectado a dois corpos articulados, e que possui uma velocidade e uma força máxima. Com o uso de juntas e motores angulares, é possível que sejam reproduzidas as diversas articulações presentes em um robô real, em seres humanos e nos animais com um alto nível de precisão [Osório et al. 2006].

## 3. Trabalhos relacionados

O controle de locomoção em robôs com pernas é um problema de busca em um espaço de estados multi-dimensional que vem desafiando os pesquisadores a várias décadas [Bekey 2005]. Este controle requer a especificação e a coordenação dos movimentos de todas as pernas do robô, enquanto são considerados fatores como a estabilidade e a fricção em relação a superfície de contato (solo) [Kohl and Stone 2004]. Esta área de pesquisas possui uma clara ligação com o controle de locomoção realizado pelos animais, e muitos das pesquisas realizadas até o momento se inspiram no caminhar realizado por animais como os mamíferos e os insetos [Reeve and Hallam 2005]. Como trabalhos pioneiros desta área podemos destacar os primeiros robôs com pernas realmente independentes, como o "Phony Pony" desenvolvido por Frank e McGhee [McGhee 1976], onde cada junta foi controlada por uma simples máquina de estados finita, até o controle algorítmico bem sucedido desenvolvido por Raibert [Raibert 1986], que era capaz de manter a estabilidade dinâmica em robôs de uma (*monopod*), duas e quatro pernas.

Na área de controle inteligente de robôs com pernas, os primeiros trabalhos datam do final dos anos 80 e início dos anos 90, como por exemplo o trabalho de Lewis [Lewis et al. 1992], que utilizou algoritmos genéticos para a evolução dos controladores de um robô de seis pernas (*hexapod*). Neste trabalho, o controlador foi evoluído em um robô cujo caminhar era inspirado no caminhar dos insetos. Através de vários estágios de evolução, seu comportamento foi sendo modificado até atingir um caminhar razoavelmente satisfatório. Bongard [Bongard and Pfeifer 2002] evoluiu os parâmetros de uma rede neural artificial dinâmica utilizada para controlar diversos tipos de robôs simulados. Busch [Busch et al. 2002] utilizou Programação Genética para evoluir os parâmetros de controle de diversos tipos de robôs, simulados utilizando o pacote de software DynaMechs[2]. Jacob [Jacob et al. 2005] utilizou aprendizado por reforço para o controle de um robô de quatro pernas (*tetrapod*) simulado através da biblioteca de software ODE. Reeve [Reeve and Hallam 2005] utilizou algoritmos genéticos para a evolução dos parâmetros de diversos modelos de redes neurais utilizadas para o controle de diversos *tetrapods* simulados utilizando o DynaMechs.

Na maioria das abordagens descritas acima, a função de *fitness* utilizada foi a distância percorrida pelo robô durante um certo período de tempo. Embora esta função de *fitness* seja largamente utilizada, ela pode fazer com que a evolução privilegie formas de caminhar pouco estáveis em detrimento de soluções um pouco mais lentas porém muito

[1]ODE – http://www.ode.org

[2]DynaMechs – http://dynamechs.sourceforge.net/

mais estáveis [Golubovic and Hu 2003]. Em nossos estudos, além da distância percorrida pelo robô, foram utilizadas como critério de *fitness* informações sensoriais, provenientes de um giroscópio simulado, a fim de se garantir que os caminhares obtidos fossem tanto rápidos quanto estáveis [Heinen and Osório 2006a, Heinen and Osório 2006b].

## 4. Modelo proposto

Esta seção descreve o simulador LegGen[3], [Heinen and Osório 2006a], proposto e implementado para a validação de diferentes estratégias de controle do caminhar de robôs com pernas. Este simulador é composto de diversos módulos, mostrados na Figura 1.

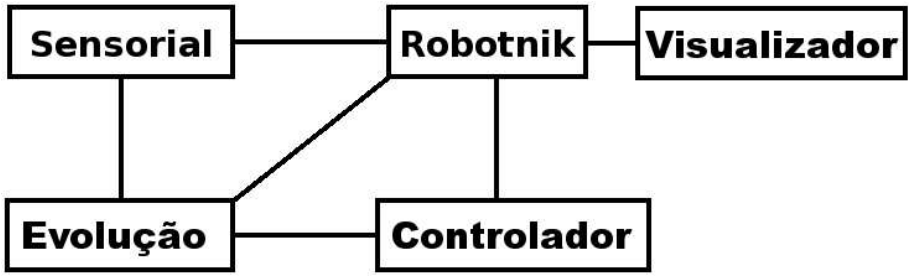



**Figura 1. Módulos do LegGen**

O módulo *Robotnik* é responsável pela criação do robô e do ambiente virtual através do uso da biblioteca ODE. O módulo *Evolução* é responsável pela evolução dos parâmetros do controlador através de algoritmos genéticos. O módulo *Sensorial* é responsável pela leitura de informações sensoriais durante a simulação, utilizadas no cálculo do *fitness* de cada indivíduo. O módulo *Visualizador* é responsável pela visualização dos resultados em um ambiente gráfico tridimensional. O módulo *Controlador* é responsável pelo controle das juntas do robô durante o caminhar. As próximas seções descrevem as estratégias de controle das juntas utilizadas nos experimentos da Seção 5..

### 4.1. Controle baseado em um autômato finito

Esta estratégia de controle consiste em uma máquina de estados, onde cada estado determina o ângulo desejado para cada uma das juntas do robô. Desta forma, o controlador continuamente realiza a leitura dos ângulos de cada uma das juntas, para verificar se elas já atingiram os valores desejados. Em robôs reais, os ângulos das juntas podem ser obtidos através da leitura de sensores (encoders) instalados nas mesmas [Dudek and Jenkin 2000]. O controle do caminhar é realizado da seguinte forma: inicialmente o controlador verifica se as juntas já atingiram os ângulos desejados. As juntas que não tiverem atingido são movimentadas (os motores são ativados), e quando todas elas tiverem atingido os seus respectivos ângulos, o autômato passa para o estado seguinte.

Para que haja sincronia nos movimentos, é importante que todas as juntas atinjam os ângulos desejados praticamente ao mesmo tempo, o que é possível com a aplicação de uma velocidade angular específica para cada uma das juntas, calculada através da fórmula:

$$V_{ij} = Vr_i(\alpha_{ij} - \alpha_{ij-1}) \quad (1)$$

[3]LegGen – http://www.inf.unisinos.br/~osorio/leggen

onde $V_{ij}$ é a velocidade aplicada ao motor da junta $i$ no estado $j$, $\alpha_{ij}$ é o ângulo da junta $i$ no estado $j$, $\alpha_{ij-1}$ é o ângulo da junta $i$ no estado anterior $(j-1)$, e $Vr_i$ é a velocidade referencial do estado $i$, utilizada para controlar a velocidade do conjunto. A velocidade referencial $Vr$ é um dos parâmetros do caminhar otimizados pelo algoritmo genético. Os outros parâmetros são os ângulos desejados de cada uma das juntas para cada estado. Para limitar o espaço de busca, o algoritmo genético gera apenas valores dentro do intervalo máximo e mínimo de cada junta.

### 4.2. Controle baseado em uma rede neural

Outra forma de se controlar o deslocamento de robôs com pernas é através do uso de redes neurais artificiais (*artificial neural networks* – ANN) [Haykin 2001]. Esta abordagem possui uma limitação: não é possível se obter de antemão informações locais para o cálculo do gradiente e a correção dos erros, e isto impede a utilização dos algoritmos de aprendizado supervisionado tradicionais (*back-propagation* e similares). Por isto, foram utilizados algoritmos genéticos para a evolução dos pesos sinápticos. A vantagem de se utilizar GAs para o ajuste dos pesos é que eles não necessitam de informações locais para a correção dos erros, ou seja, eles não necessitam de uma base com os dados de treinamento. As principais vantagens de se utilizar uma ANN no controle do caminhar são: (i) elas são mais robustas em relação a situações novas e inesperadas; (ii) possuem um alto grau de generalização.

Como entradas da ANN, foram utilizados os ângulos atuais das juntas do robô, normalizados entre -1 ($\alpha_{mim}$) e 1 ($\alpha_{max}$). Na saída da ANN, são obtidos os ângulos desejados para as juntas no instante $t+1$, normalizados entre -1 e 1. Após alguns testes preliminares, optou-se por utilizar as redes neurais recorrentes do tipo Elman [Elman 1990], que são redes neurais do tipo *multi layer perceptron* (MLP) que possuem conexões de realimentação (*feedback*) na camada oculta. Estas conexões permitem que as redes de Elman aprendam a reconhecer e gerar padrões temporais. A função de ativação utilizada foi a tangente hiperbólica. O intervalo de valores possíveis para os pesos sinápticos foi limitado em $[-1;1]$, que se mostrou bastante adequado para a representação do problema em questão.

### 4.3. Evolução dos parâmetros de controle

Em ambas as estratégias de controle, a evolução dos parâmetros é realizada através de algoritmos genéticos, implementados através da biblioteca GAlib[4]. A Tabela 1 mostra os parâmetros utilizados pelo algoritmo genético durante as simulações.

**Tabela 1. Parâmetros do algoritmo genético**

| Parâmetro | Valor |
|---|---|
| Taxa de cruzamentos | 0,80 |
| Taxa de mutação | 0,08 |
| Tamanho da população | 350 |
| Número de gerações | 700 |

[4]GAlib – http://www.lancet.mit.edu/ga/

A função de *fitness* $F$ utilizada pelo GA é calculada através da fórmula:

$$F = \frac{D}{1 + B + a \times G} \tag{2}$$

onde $D$ é a distância percorrida pelo robô em relação ao eixo $x$, $B$ é o índice dos *bumpers*, $G$ é a taxa de instabilidade, e $a$ é uma constante que serve para alterar a influência de $G$ na função de *fitness*. Nos experimentos realizados, foi utilizado $a = 10$. O *bumpers* $B$ é calculado através da equação:

$$B = \sum_{i=1}^{P} \left( \frac{n_i}{N} - \frac{1}{2} \right)^2 \tag{3}$$

onde $P$ é o número de *endpoints* (patas), $n_i$ é a quantidade de amostras sensoriais nas quais o *endpoint* $i$ estava em contato com o solo, e $N$ é o número total de leituras sensoriais realizadas. Nesta função de *fitness*, o valor de $B$ tenderá a zero quando o robô mantiver as patas no chão por aproximadamente 50% do tempo, que é o comportamento desejado durante o caminhar. Já se o robô mantiver todas as patas no chão durante o período de simulação, o valor de $B$ será igual a 1. O mesmo ocorrerá se o robô mantiver todas as patas no ar durante o período de simulação.

Para o cálculo da taxa de instabilidade, várias leituras de um giroscópio simulado são realizadas durante a simulação, e ao final da mesma é calculada a taxa de instabilidade do robô $G$ (Gyro) [Golubovic and Hu 2003]:

$$G = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2 + \sum_{i=1}^{N}(y_i - \overline{y})^2 + \sum_{i=1}^{N}(z_i - \overline{z})^2}{N}} \tag{4}$$

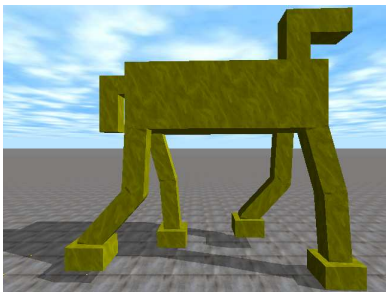
onde $N$ é o número de amostras coletadas, $x_i$, $y_i$ e $z_i$ são os dados coletados pelo giroscópio simulado no tempo $i$, e $\overline{x}$, $\overline{y}$ e $\overline{z}$ são as médias das leituras do giroscópio, calculadas através das equações:

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}, \quad \overline{y} = \frac{\sum_{i=1}^{N} y_i}{N}, \quad \overline{z} = \frac{\sum_{i=1}^{N} z_i}{N} \tag{5}$$

Utilizando esta função de *fitness*, o indivíduo mais bem qualificado é aquele que possui a melhor relação entre velocidade e estabilidade, de forma que as melhores soluções são aquelas mantém o compromisso entre estes dois critérios de avaliação.

### 4.4. Robô modelado

Conforme consta em sua documentação, a biblioteca ODE possui uma complexidade computacional de ordem $O(n^2)$, onde $n$ é o número de corpos presentes no mundo físico simulado. Deste modo, para manter a velocidade da simulação em um nível aceitável, é preciso modelar os corpos da forma mais simples possível. Por este motivo, todos os robôs simulados foram modelados com objetos simples, como retângulos e cilindros, e eles possuem apenas as articulações necessárias para a tarefa de caminhar. Para manter o projeto do robô simples, as juntas utilizadas nos membros se movimentam apenas em torno do eixo $z$ em relação ao robô (o mesmo movimento do nosso joelho), pois as simulações realizadas até o momento foram todas com o robô a caminhando em linha reta. No futuro, o modelo será estendido para aceitar modelos de robôs com juntas mais complexas. Inicialmente foram modelados e testados diversos tipos de robôs, até que se chegou ao modelo final, mostrado na Figura 2. As dimensões destes robôs são aproximadamente as de um cachorro.



| | Dimensões | | |
|---|---|---|---|
| **Parte** | $x$ | $y$ | $z$ |
| Corpo | 45,0cm | 15,0cm | 25,0cm |
| Coxa | 5,0cm | 15,0cm | 5,0cm |
| Canela | 5,0cm | 15,0cm | 5,0cm |
| Pata | 8,0cm | 5,0cm | 9,0cm |

**Figura 2. Modelo de robô utilizado nas simulações**

## 5. Resultados

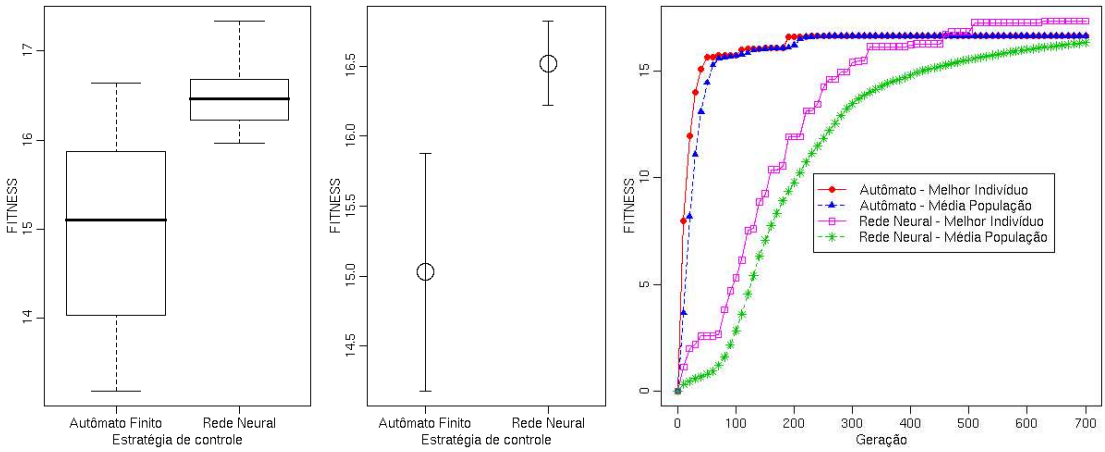
Esta seção descreve os experimentos realizados visando avaliar o desempenho de ambas as estratégias de controle propostas na Seção anterior. Para cada estratégia de controle, foram realizados 10 experimentos distintos, e ao final foi calculada a média e o desvio padrão dos resultados. A Tabela 2 mostra os resultados obtidos nestes experimentos.

**Tabela 2. Avaliação as estratégias de controle**

| | Autômato Finito | | | | Rede Neural | | | |
|---|---|---|---|---|---|---|---|---|
| **E** | $F$ | $D$ | $B$ | $G$ | $F$ | $D$ | $B$ | $G$ |
| 1 | 14,0371 | 32,1705 | 0,0126 | 0,1279 | 16,2651 | 29,1890 | 0,0019 | 0,0793 |
| 2 | 14,2773 | 32,3815 | 0,0092 | 0,1259 | 16,6349 | 28,3056 | 0,0070 | 0,0695 |
| 3 | 13,1820 | 30,3278 | 0,0065 | 0,1294 | 16,9914 | 27,8498 | 0,0085 | 0,0631 |
| 4 | 15,8724 | 26,8063 | 0,0032 | 0,0686 | 16,6781 | 27,9149 | 0,0017 | 0,0672 |
| 5 | 16,6450 | 36,5972 | 0,0022 | 0,1196 | 16,1571 | 28,2006 | 0,0076 | 0,0738 |
| 6 | 16,4784 | 27,6930 | 0,0028 | 0,0678 | 15,9652 | 31,1260 | 0,0218 | 0,0928 |
| 7 | 14,8847 | 31,6925 | 0,0097 | 0,1120 | 17,3346 | 29,6290 | 0,0048 | 0,0704 |
| 8 | 13,7677 | 29,0248 | 0,0115 | 0,1097 | 16,6541 | 29,0379 | 0,0060 | 0,0738 |
| 9 | 15,3287 | 34,4105 | 0,0022 | 0,1243 | 16,2892 | 30,1511 | 0,0022 | 0,0849 |
| 10 | 15,8022 | 37,0133 | 0,0025 | 0,1340 | 16,2274 | 29,8084 | 0,0039 | 0,0833 |
| $\mu$ | **15,0275** | **31,8118** | **0,0062** | **0,1119** | **16,5197** | **29,1212** | **0,0065** | **0,0758** |
| $\sigma$ | **1,1871** | **3,4828** | **0,0041** | **0,0242** | **0,4197** | **1,0753** | **0,0059** | **0,0091** |

A primeira coluna (**E**) representa o índice do experimento. As demais colunas representam, respectivamente, os resultados obtidos utilizando o controlador baseado em um autômato finito e o controlador baseado em uma rede neural. As sub-colunas representam, respectivamente, o *fitness* ($F$), a distância percorrida pelo robô ($D$) em 30 segundos, o índice dos *bumpers* ($B$) e a taxa de instabilidade ($G$). As duas últimas linhas da tabela trazem a média ($\mu$) e o desvio padrão ($\sigma$) de cada coluna.

Nos experimentos realizados utilizando o autômato finito, foram utilizados 4 estados no autômato, e no controle neural, foram utilizados 3 neurônios na camada oculta. Estes parâmetros foram determinados através da realização de diversos experimentos estatisticamente válidos, nos quais eles se mostraram bastante adequados. O tempo total despendido na realização dos experimentos da Tabela 2 foi de 149,22 horas (6,22 dias). A Figura 3 mostra os gráficos de *boxplot* e do intervalo de confiança (CI) a 95%, relativos aos valores de *fitness* obtidos nos experimentos da Tabela 2.

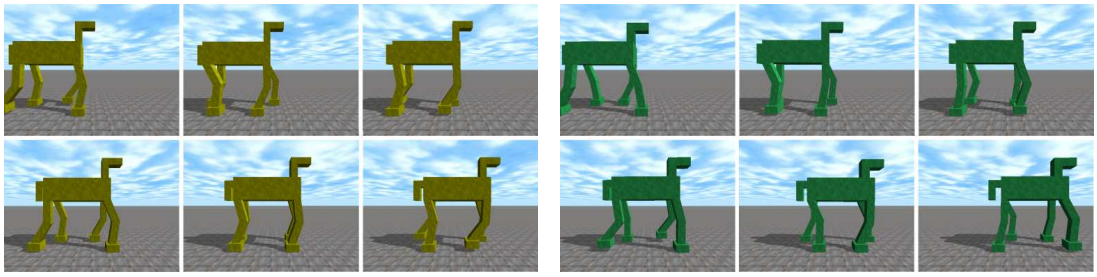



(a) Gráfico boxplot (b) Int. confiança (c) Evolução

**Figura 3. Gráfico de *boxplot* e CI dos experimentos da Tabela 2**

Pela análise dos gráficos da Figura 3, pode-se afirmar que os resultados obtidos com o controlador baseado em um autômato finito são inferiores aos resultados obtidos com o controlador neural, pois os intervalos de confiança não se sobrepõe. Além disso, os resultados obtidos com o autômato finito possuem uma maior variabilidade. A Figura 3(c) mostra um gráfico que compara a evolução das soluções (*fitness* do melhor indivíduo e da média da população) obtidas com as duas estratégias de controle. Os experimentos descritos neste gráfico foram os que levaram aos melhores resultados em cada estratégia.

Percebe-se que o controlador neural precisa de muito mais épocas de evolução para atingir resultados satisfatórios. Isto se deve ao tamanho maior do espaço de estados (o autômato finito possui 13 parâmetros livres, enquanto que a ANN possui 44). A Figura 4(a) mostra um exemplo de caminhar obtido utilizando o autômato finito, e a Figura 4(b) mostra um exemplo de caminhar obtido utilizando o controlador neural.



(a) Robô controlado por um autômato (b) Robô controlado por uma ANN

**Figura 4. Exemplos de robôs controlados por ambas as estratégias**

## 6. Conclusões e perspectivas

O objetivo deste artigo foi descrever o simulador LegGen, que é um simulador desenvolvido para realizar a configuração automática do caminhar de robôs móveis dotados de

pernas. Neste simulador, a configuração do caminhar é realizada utilizando algoritmos genéticos, que evoluem os parâmetros do caminhar através do uso de robôs simulados através da biblioteca ODE. Para o controlar as juntas do robô, duas estratégias foram utilizadas: (i) um autômato finito; (ii) redes neurais artificiais. Através da realização de diversos experimentos estatisticamente válidos, foi possível constatar que o controlador neural é muito mais eficiente que o controlador baseado em um autômato finito.

As perspectivas futuras incluem tornar o caminhar possível em superfícies irregulares e a subida de escadas, bem como a construção de um robô físico conforme as especificações de um dos melhores modelos aprendidos, para assim validar o modelo em condições reais.

## Referências


Batavia, P., Pomerleau, D., and Thorpe, C. (1996). Applying advanced learning algorithms to alvinn. Technical Report CMU-RI-TR-96-31, Carnegie Mellon Univ. (CMU), Pittsburgh, PA.

Bekey, G. A. (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, Cambridge, MA.

Bongard, J. C. and Pfeifer, R. (2002). A method for isolating morphological effects on evolved behaviour. In *Proc. 7th Int. Conf. Simulation of Adaptive Behaviour (SAB)*, pages 305–311, Edinburgh, UK. MIT Press.

Busch, J., Ziegler, J., Aue, C., Ross, A., Sawitzki, D., and Banzhaf, W. (2002). Automatic generation of control programs for walking robots using genetic programming. In *Proc. 5th European Conf. Genetic Programming (EuroGP)*, volume 2278 of *LNCS*, pages 258–267, Kinsale, Ireland. Springer-Verlag.

Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sendai, Japan.

Darwin, C. (1859). *Origin of Species*. John Murray, London, UK.

Dudek, G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*. Cambridge Univ. Press, Cambridge, UK.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Golubovic, D. and Hu, H. (2003). Ga-based gait generation of sony quadruped robots. In *Proc. 3th IASTED Int. Conf. Artificial Intelligence and Applications (AIA)*, Benalmadena, Spain.

Haykin, S. (2001). *Redes Neurais: Princípios e Prática*. Bookman, Porto Alegre, RS, Brazil, 2 edition.

Heinen, M. R. and Osório, F. S. (2006a). Applying genetic algorithms to control gait of physically based simulated robots. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada. IEEE World Congress on Computational Intelligence (WCCI), IEEE Press.

Heinen, M. R. and Osório, F. S. (2006b). Gait control generation for physically based simulated robots using genetic algorithms. In Schiman, J., Coelho, H., and Rezende, S. O., editors, *Proceedings of the International Joint Conference 2006, 10th Ibero-American Conference on AI (IBERAMIA), 18th Brazilian Symposium on AI (SBIA)*, Lecture Notes in Computer Science, Ribeirão Preto - SP, Brazil. Springer-Verlag.

Heinen, M. R. and Osório, F. S. (2006c). Uso de algoritmos genéticos para a configuração automática do caminhar em robôs móveis. In *Anais do Encontro de Robótica Inteligente (EnRI)*, Campo grande, MS, Brazil. UFMS, UCDB.

Heinen, M. R. and Osório, F. S. (2007). Evolving gait control of physically based simulated robots. *Revista de Informática Teórica e Aplicada (RITA) – to appear*.

Heinen, M. R., Osório, F. S., Heinen, F. J., and Kelber, C. (2006). SEVA3D: Using artificial neural networks to autonomous vehicle parking control. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, Vancouver, Canada. IEEE World Congress on Computational Intelligence (WCCI), IEEE Press.

Jacob, D., Polani, D., and Nehaniv, C. L. (2005). Legs than can walk: Embodiment-based modular reinforcement learning applied. In *Proc. IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 365–372, Espoo, Finland.

Knight, R. and Nehmzow, U. (2002). Walking robots - a survey and a research proposal. Technical Report CSM-375, Univ. Essex, Essex, UK.

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2619–2624, New Orleans, LA.

Law, A. M. and Kelton, D. W. (2000). *Simulation Modeling and Analysis*. McGraw-Hill, New York.

Lewis, M. A., Fagg, A. H., and Solidum, A. (1992). Genetic programming approach to the construction of a neural network for control of a walking robot. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2618–2623, Nice, France.

McGhee, R. B. (1976). Robot locomotion. *Neural Control of Locomotion*, pages 237–264.

Medeiros, A. (1998). Introdução à robótica. In *Anais do XVII Encontro Nacional de Automática*, volume 1, pages 56–65, Natal, RN, Brazil.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA.

Mitchell, T. (1997). *Machine Learning*. McGrall-Hill, New York.

Osório, F. S., Musse, S. R., Vieira, R., Heinen, M. R., and Paiva, D. C. (2006). *Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation*, volume 2, pages 1–45. Springer-Verlag, Berlin, Germany.

Pfeifer, R. and Scheier, C. (1999). *Understanding Intelligence*. MIT Press, Cambridge, MA.

Raibert, M. H. (1986). *Legged Robots That Balance*. MIT Press, Cambridge, MA.

Reeve, R. and Hallam, J. (2005). An analysis of neural models for walking control. *IEEE Trans. Neural Networks*, 16(3):733–742.