Evolutionary Optimization of Robust Control Laws for Mobile Robots in Dynamic Environments

Conference Paper · October 2018

DOI: 10.5753/eniac.2018.4439

4 authors, including:

Rafael Del Lama
University of São Paulo
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Renato Tinós
University of São Paulo
113 PUBLICATIONS **846** CITATIONS
SEE PROFILE

Evolutionary Optimization of Robust Control Laws for Mobile Robots in Dynamic Environments

Rafael S. Del Lama, Raquel M. Candido, Luciana T. Raineri, Renato Tinós

[1]Departamento de Computação e Matemática, FFCLRP, Universidade de São Paulo (USP)
Ribeirão Preto, SP, Brasil

rafael.lama@usp.br, raquel.candido@usp.br,luciana.traineri@usp.br,

rtinos@ffclrp.usp.br

Abstract. *The problem of controlling mobile robots in dynamic environments is an interesting challenge. This paper investigates the problem of controlling mobile robots in dynamic environments through robust control laws defined by echo state networks (ESN). The output weights of the ESN are optimized by genetic algorithms (GAs). Different GAs developed for optimization in dynamic environments are compared in the problem of searching for robust solutions. Two approaches are investigated: through dynamic evolutionary optimization and robust evolutionary optimization. In the experiments, the GA evolved in the static environment produces good trajectories in environments that resemble the static environment (without obstacles). However, it presents unsatisfactory performance in environments that are very different from the static environment. Both GAs evolved in the dynamic and robust optimization approaches present good results in environments that differ from the static environment.*

1. Introdução

Algoritmos genéticos (AGs), e outros algoritmos evolutivos, têm sido utilizados em diversas áreas da robótica, quer para o desenvolvimento da arquitetura do robô, quer para a otimização de leis de controle e de estratégias de navegação e planejamento [Floreano and Nolfi 2000]. Tal fato ocorre principalmente porque o projeto de robôs autônomos e seus controladores para ambientes não-estruturados, dinâmicos e/ou parcialmente desconhecidos é uma tarefa difícil para um projetista humano [Siegwart et al. 2011, Romero et al. 2014].

Utilizando algoritmos evolutivos, o ambiente e a tarefa a ser executada passam a ser os fatores principais no desenvolvimento do robô e de seu controlador, tirando o posto que antes cabia ao projetista. Tal é a perspectiva do uso de algoritmos evolutivos em robótica que se cunhou um termo especialmente para designar os mecanismos por eles criados: robôs evolutivos (REs) [Floreano and Nolfi 2000]. Salienta-se que a conexão entre robótica e biologia não têm um sentido único neste caso: robôs autônomos podem servir como uma importante ferramenta para o desenvolvimento e teste de modelos comportamentais, de habilidades cognitivas e de modelos evolutivos de organismos vivos [Webb 2001, Shimo et al. 2010].

A aplicação crescente de robótica em ambientes não-estruturados ocasionou um aumento no interesse por algoritmos evolutivos que produzam soluções que mudam com o tempo [Branke 2002] ou soluções robustas. As mudanças que podem ocorrer em robótica

são causadas, entre outras, pela ocorrência de falhas [Tinós and de Carvalho 2006] e por mudanças nas características do ambiente [Billard et al. 1999]. Tais mudanças representam variações nas restrições das soluções, no número de variáveis e/ou na avaliação das soluções (fitness), afetando o processo de otimização devido às alterações na superfície de fitness [Tinós and Yang 2014].

A área de pesquisa em algoritmos evolutivos para ambientes dinâmicos tem recebido grande atenção recentemente [Jin and Branke 2005], com diversos trabalhos em teoria e no desenvolvimento de algoritmos e aplicações. Uma dificuldade que pode ser enfrentada quando o problema muda é a perda da diversidade da população de soluções. Isto geralmente ocorre devido à convergência da população para ótimos locais. Quando o problema muda, é difícil escapar destes ótimos utilizando-se um algoritmo genético padrão. Para solucionar esse problema, diversos mecanismos têm sido criados para controlar ou aumentar a diversidade da população em AGs aplicados a problemas em ambientes incertos [Jin and Branke 2005]. Dois dos mais utilizados são a hipermutação e os imigrantes aleatórios [Cobb and Grefenstette 1993].

Mais recentemente, pesquisadores na área de computação evolutiva dinâmica começaram a estudar o problema de busca por soluções robustas para ambientes dinâmicos [Beyer and Sendhoff 2007, Fu et al. 2015]. Enquanto que em problemas de otimização dinâmica se está preocupado em adaptar rapidamente a solução durante o processo de otimização, em otimização robusta busca-se encontrar uma solução que seja robusta para mudanças futuras, ou seja, após o processo de otimização.

Este artigo tem por objetivo a investigação de AGs aplicados na otimização de leis de controle de robôs móveis que sejam robustas a mudanças no ambiente que ocorram após o processo de otimização. No problema estudado, o robô móvel deve explorar ao máximo o ambiente sem se chocar com os obstáculos e retornar periodicamente para uma área definida *a priori*. Em um problema real, mudanças que afetam o hardware ou o ambiente poderão ocorrer após o processo de otimização. A grande dificuldade do projeto de robôs nestes casos advém da impossibilidade de prever as situações que serão confrontadas pelos robôs em ambientes não-estruturados ou desconhecidos. Com as mudanças, soluções encontradas pelo AG podem se tornar ruins no novo ambiente devido ao fato das soluções serem otimizadas considerando-se superfícies de fitness estáticas. Neste trabalho, diferentes algoritmos genéticos desenvolvidos para a otimização em ambientes dinâmicos são comparados no problema da busca por soluções robustas, ou seja, que apresentem um bom desempenho mesmo que o robô ou o ambiente sofram alterações. Particularmente, são investigados os mecanismos de imigrantes aleatórios e hipermutação. De acordo com o conhecimento dos autores, tais mecanismos não foram investigados no contexto de otimização robusta. Além disso, estratégias próprias para se encontrar soluções robustas são investigadas [Fu et al. 2015]. Uma outra contribuição deste trabalho é o uso de uma rede neural recorrente do tipo echo state network (ESN) [Jaeger and Haas 2004] para o controle do robô móvel.

A metodologia proposta é apresentada na Seção 2. A Seção 3 apresenta os resultados obtidos nos experimentos, enquanto que a Seção 4 apresenta as conclusões deste trabalho.

2. Metodologia

Neste trabalho, considera-se um robô móvel que pode realizar em cada instante uma das seguintes ações: andar em frente 10 cm, girar -45 graus, girar +45 graus, ou girar 90 graus. O robô possui quatro sensores, sendo um posicionado para frente, um para cima, um na diagonal direita e um na diagonal esquerda (ambos formando um ângulo de 45° com o sensor da frente). O sensor voltado para cima permite detectar a área de recarga de bateria (ver adiate), já que esta é a única área coberta da arena em que o robô deve navegar.

Aqui, o robô móvel é controlado por uma rede neural recorrente do tipo ESN [Jaeger and Haas 2004]. Para cada posição da arena em que o robô estiver navegando, a ESN gera uma saída correspondente a uma das quatro ações. Cada neurônio de saída é relacionado a uma destas ações, sendo que o neurônio com máxima ativação define a ação tomada pela ESN em cada instante de tempo. As entradas da rede correspondem às leituras dos sensores do robô. As conexões recorrentes são necessárias para que o robô tenha comportamentos não simplesmente reativos. A Figura 1 mostra a rede neural utilizada neste trabalho.

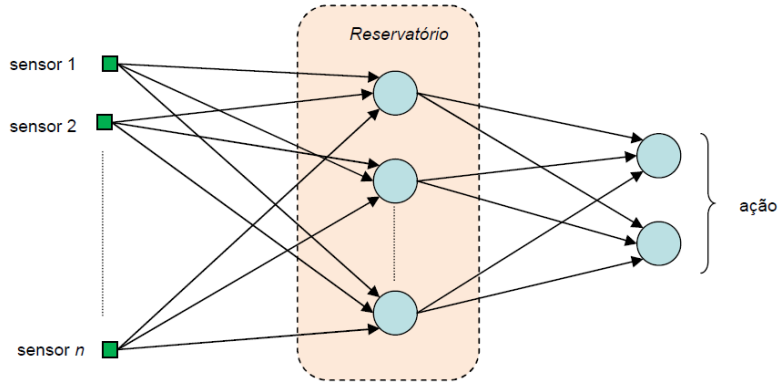

Figura 1. Rede neural ESN utilizada para controlar o robô. Por simplicidade, as conexões recorrentes dos neurônios localizados no reservatório não são mostradas. Os únicos pesos a serem ajustados (treinados) são aqueles entre os neurônios do reservatório e da camada de saída. Aqui, propõe-se otimizar estes pesos utilizando-se um algoritmo genético.

A ESN usa um reservatório com neurônios esparsamente conectados através de pesos gerados aleatoriamente e posteriormente normalizados. Os pesos que conectam as unidades sensoriais (entradas) aos neurônios do reservatório são também aleatórios. O fato de utilizar um grande reservatório de neurônios possibilita a obtenção de comportamentos dinâmicos complexos. Em ESN supervisionadas, os pesos entre os neurônios do reservatório e os neurônios da camada de saída são ajustados utilizando-se o método dos mínimos quadrados, o que torna o aprendizado bastante rápido [Jaeger and Haas 2004]. Entretanto, as saídas desejadas não são conhecidas para o problema estudado neste trabalho, i.e., não se conhece *a priori* a ação que o robô deve tomar em cada instante de sua trajetória ou para posições e orientações específicas.

Desta forma, propõe-se utilizar um AG para otimizar o vetor de pesos entre a camada intermediária (reservatório) e a camada de saída. Como dito anteriormente, os pesos

da camada de entrada e do reservatório são aleatórios (estes últimos são normalizados para que a ESN possua certas propriedades dinâmicas inerentes a esta arquitetura). Após alguns testes iniciais, a arquitetura adotada foi de 4 entradas (sensores), 50 neurônios no reservatório e 4 saídas (movimentos) com um valor de densidade de conexão de 0,15 (15%) no reservatório. O raio espectral de 0,95 é utilizado para normalizar os pesos aleatórios.

A aptidão (fitness) de cada indivíduo é obtida testando-se as leis de controle (ESN) definidas pelo cromossomo do indivíduo. Devido ao longo tempo necessário para avaliar uma solução no robô real, um simulador foi utilizado durante a otimização (Seção 2.6). A seguir, algumas das características dos AGs implementados são discutidas. A população inicial dos AGs é gerada aleatoriamente de acordo com uma distribuição uniforme.

2.1. Codificação

Cada indivíduo é representado por um vetor real (cromossomo), na qual cada posição representa o peso correspondente na camada de saída da rede ESN. Cada indivíduo codifica um vetor de pesos, que por sua vez define uma ESN. Cada ESN, por sua vez, define um comportamento para o robô (trajetória) no ambiente. O tamanho do cromossomo varia de acordo com a quantidade de neurônios no reservatório e na camada de saída.

2.2. Reprodução e Seleção

Os indivíduos são reproduzidos pelos operadores de *crossover* de dois pontos e mutação gaussiana. O método do torneio, no qual K_t indivíduos da população são aleatoriamente escolhidos e aquele com maior fitness é selecionado, é empregado. Este método é interessante pois permite o controle da pressão seletiva por meio do parâmetro K_t , além de representar um procedimento computacionalmente mais simples, quando comparado com o método da roleta [Mitchell 1996]. Elitismo também é empregado, sendo que os dois melhores indivíduos da população atual são copiados para a próxima população.

2.3. Avaliação

Diferentes ESNs geram comportamentos diferentes (trajetórias) do robô. Assim, para avaliar um indivíduo, o robô com a ESN dada pelo cromossomo do indivíduo que está sendo avaliado deve navegar pelo ambiente, andando o máximo possível em linha reta, até atingir algum dos seguintes critérios de parada:

- Colidir com um obstáculo;
- Não voltar para carregar a bateria dentro de uma determinada quantidade de movimentos. Esta condição simula o processo de descarga da bateria do robô;
- Atingir o número máximo de movimentos.

Assim, o número de iterações do robô, com a ESN definida pelo indivíduo **x**, é dado por:

$$t_{max}(\mathbf{x}) = min(300, t_{choque}, t_{bateria}) \quad (1)$$

sendo t_{choque} o instante em que o robô chocou-se com um obstáculo e $t_{bateria}$ o instante em que a bateria do robô ficou totalmente descarregada. É importante observar que $t_{bateria}$ muda cada vez que a bateria é recarregada. A carga da bateria é simulada utilizando-se uma função linear cujos valores decrescem com o tempo [Floreano and Nolfi 2000]. O tempo de carga é considerado instantâneo, sendo que a bateria é recarregada sempre

que o robô entra em uma área pré-definida da arena (a pontuação não é contada quando o robô está na área de recarga). O tempo de descarga simulado é de 80 iterações, ou seja, a bateria fica descarregada após 80 iterações depois do início do experimento ou 80 iterações depois que o robô saiu da área de recarga.

A seguinte função de avaliação (fitness) é utilizada:

$$f(\mathbf{x}) = \sum_{t=1}^{t_{max}} \alpha(\mathbf{x}, t) \tag{2}$$

na qual $\mathbf{x}$ é o vetor de pesos (da camada de saída) da ESN armazenado no cromossomo do indivíduo, t é a iteração (em cada iteração, o robô executa uma ação), e:

$$\alpha(\mathbf{x}, t) = \begin{cases} 1, & \text{se o robô andou para frente na interação } t \\ 0, & \text{caso contrário} \end{cases} \tag{3}$$

Para atingir o *fitness* máximo, o robô deve navegar no ambiente em linha reta o máximo possível, sem se chocar com os obstáculos e retornar à área de recarga sempre que a bateria estiver quase descarregada. A tarefa de simplesmente andar na arena sem se chocar com os obstáculos exigi apenas ações reativas. Entretanto, ao colocar a restrição para o tempo de bateria, exigindo assim que o robô volte periodicamente para a área de recarga, faz com que ações que envolvem memória sejam necessárias [Floreano and Nolfi 2000].

O pseudo-código apresentado na Figura 2 mostra como a função de fitness é calculada para uma dado indivíduo do AG (ou seja, uma rede neural ESN com pesos dados pelo cromossomo do indivíduo).

```
Algoritmo: Fitness (indivíduo)
Inicio
    Enquanto (qtdMovExec < qtdMov) {
        Realiza a leitura dos sensores
        Executa a rede ESN definida pelo indivíduo de acordo com a leitura dos sensores
        Determina o neurônio de maior ativação e executa a ação correspondente
        qtdMovExec++;

        Se a ação executada foi andar para frente
            qtdFrente++;
        Fim se
    }
 Retorna qtdFrente/qtdMov;
Fim
```

Figura 2. Pseudo-código para computar o fitness de um indivíduo do AG. A variável $qtdMovExec$ representa a quantidade de movimentos executados pelo robô durante a avaliação do indivíduo, e deve ser menor que $qtdMov$ (neste trabalho, $qtdMov = 300$). Já a variável $qtdFrente$ representa a quantidade de movimentos em linha reta que o robô executou.

2.4. Robustez

Neste trabalho, considera-se que o ambiente pode mudar periodicamente. Deseja-se uma estratégia de controle (dada pela rede neural) que seja robusta às mudanças. Duas estratégias são investigadas. Na primeira, o AG otimiza a ESN considerando-se mudanças periódicas no ambiente ocorridas durante a otimização. Esta é a estratégia utilizada em

otimização evolutiva dinâmica [Tinós and Yang 2014]. A população do AG deve, neste caso, se adaptar às mudanças do ambiente. Estratégias para controle da diversidade (ver próxima subseção) são geralmente adotadas neste caso para facilitar a fuga de ótimos locais na otimização de um dado ambiente. Neste trabalho, estamos interessados em obter soluções robustas. Assim, a primeira hipótese que será investigada é a de que técnicas de manutenção de diversidade desenvolvidas em otimização evolutiva dinâmica podem ser interessantes para obter soluções robustas para o robô sujeito a mudanças no ambiente.

A segunda hipótese que será investigada é a de que tais técnicas de manutenção de diversidade podem ser interessantes quando técnicas de avaliação de fitness desenvolvidas em otimização evolutiva robusta são empregadas para se obter soluções robustas para o robô sujeito a mudanças no ambiente. Neste caso, a seguinte função de fitness [Fu et al. 2015] é utilizada:

$$f(x) = \frac{1}{n} \sum_{s=1}^{n} \sum_{t=1}^{t_{max}} \alpha(\mathbf{x}, t) \tag{4}$$

na qual s é o índice do ambiente em que o robô está sendo avaliado e n é o número total de ambientes. Esta estratégia de avaliação é bastante simples: o robô é avaliado em n ambientes, sendo que o fitness é dado pela média do fitness (Eq. 2) considerando-se todos os ambientes. Ambas as estratégias (por otimização evolutiva dinâmica ou robusta) são aqui avaliadas, ao fim da otimização, apresentando-se o robô em novos ambientes não vistos durante o processo de otimização. A avaliação média nestes novos ambientes é utilizada como medida de eficiência.

2.5. Manutenção da diversidade

Aqui, técnicas para manutenção ou aumento da diversidade das soluções são empregadas no AG. Os ambientes são dinâmicos devido à introdução de mudanças de configurações dos obstáculos. Além do AG padrão, as seguintes estratégias são investigadas:

- **Hipermutação:** nesta estratégia, a taxa de mutação é aumentada toda vez que a diversidade da população de soluções atinge um patamar ou quando o algoritmo converge para uma solução. Aumentando-se a taxa de mutação, aumenta-se a chance de o algoritmo escapar do ótimo local em que ele se encontra [Cobb and Grefenstette 1993].
- **Imigrantes aleatórios:** a estratégia dos imigrantes aleatórios é inspirada no fluxo de indivíduos que entram e saem de uma população entre duas gerações na natureza [Cobb and Grefenstette 1993]. O AG com imigrantes aleatórios é bastante simples e interessante, sendo que em cada geração do processo de otimização, alguns indivíduos da população corrente são substituídos por indivíduos aleatórios. Uma estratégia de substituição, como por exemplo, substituir indivíduos aleatoriamente (estratégia utilizada aqui) ou os indivíduos menos aptos, define quais indivíduos são substituídos. Por meio da introdução de novos indivíduos, a estratégia tenta manter o nível de diversidade em um patamar razoável.

O AG padrão estático (sem que ocorram alterações durante a otimização) é comparado com: i) o AG padrão com alterações no problema ocorrendo durante o processo de otimização (enfoque por otimização evolutiva dinâmica); ii) o AG com função de fitness para problemas robustos (enfoque por otimização evolutiva robusta). Para os dois

enfoques, são testadas duas estratégias de manutenção de diversidade: i) Hipermutação; ii) Imigrantes Aleatórios.

2.6. Simulador

Como o processo de otimização em um robô real é muito demorado, foi desenvolvido um simulador para reduzir o tempo de otimização. Durante a simulação não é considerado que possam existir ruídos e ações imperfeitas. Entretanto, ruídos e ações imperfeitas são intrínsicas a experimentos que envolvam robôs reais. Assim, após encontradas as soluções no simulador, pode ser necessário evoluir estas soluções por mais algumas gerações no robô real.

O AG executado nos experimentos com o simulador é o mesmo aplicado nos experimentos com o robô real, caso este seja utilizado. A única diferença é que, ao invés do fitness ser calculado utilizando o robô real, ele é calculado utilizando o robô simulado. É interesante notar que, ao transferir as soluções evoluídas no simulador e continuar os experimentos em robôs reais, mudanças no problema de otimização ocorrem. Assim, a busca por soluções robustas é de grande relevância do ponto de vista prático.

No simulador, antes de iniciar a avaliação de um indivíduo, são sorteadas uma posição e um ângulo inicial. As novas posições e orientações do robô após cada ação são calculadas através de modelos cinemáticos. Por meio de cálculos envonvendo geometria, são simuladas as saídas produzidas pelos sensores. Os critérios de parada do AG são os mesmos citados anteriormente. A posição e o ângulo inicial aleatórios são importantes pois, se um indivíduo for passado de geração em geração, ele será avaliado várias vezes iniciando em configurações distintas.

Todos os códigos utilizados no trabalho foram desenvolvidos em C++. As simulações foram executadas em um servidor com 2 processadores Intel Xeon E5-2620 v2 (com 15 MB Cache e 2.10 GHz) e 32 GB de memória RAM.

3. Resultados

Nos experimentos realizados nas abordagens por otimização evolutiva dinâmica e robusta, todos os indivíduos de cada uma das populações foram apresentados a 10 ambientes diferentes durante o processo de otimização. Estes ambientes diferem no número de obstáculos e na posição destes em uma arena retangular de 2 m por 1,20 m. Para cada um dos AGs, foram realizadas 25 execuções com diferentes sementes aleatórias. Cada execução demorou cerca de 40 horas na plataforma computacional utilizada. De modo a reduzir o tempo total, diversas execuções ocorreram em paralelo.

Todos os AGs utilizam: população composta por 100 indivíduos, taxa de crossover definida como 0,6, taxa de mutação definida como $1/m$, sendo m o tamanho do cromossomo. Na seleção por torneio, $K_t = 3$. Foi utilizado também o método do elitismo, onde os dois melhores indivíduos da população atual passam para a próxima geração. Na mutação Gaussiana foi utilizado desvio padrão igual a 1,0.

Quando a hipermutação é ativada, a taxa de mutação é aumentada, ficando 4 vezes maior que a taxa de mutação inicial por 10 gerações. A taxa de substituição de indivíduos para o AG com Imigrantes Aleatórios é 0,05, o que significa que 5% dos indivíduos são substituídos por novos indivíduos aleatórios em cada geração. Os indivíduos a serem

substituídos são escolhidos aleatoriamente e com distribuição uniforme, com exceção dos dois indivíduos selecionados pelo método do elitismo. Todos estes parâmetros foram obtidos realizando-se experimentos iniciais.

3.1. Enfoque por Otimização Evolutiva Dinâmica

Nos experimentos em que são inseridas mudanças no ambiente durante a otimização, estas ocorrem a cada 200 gerações. O processo de otimização ocorre durante 2.200 gerações da seguinte forma: nas 200 primeiras gerações é utilizado o ambiente inicial sem nenhum obstáculo; nas 1.800 gerações seguintes são utilizados 9 ambientes com diferentes obstáculos; nas últimas 200 gerações é utilizado o ambiente inicial sem nenhum obstáculo. A Figura 3 apresenta o fitness médio (para 25 execuções) do melhor indivíduo de cada geração para cada um dos quatro algoritmos estudados.

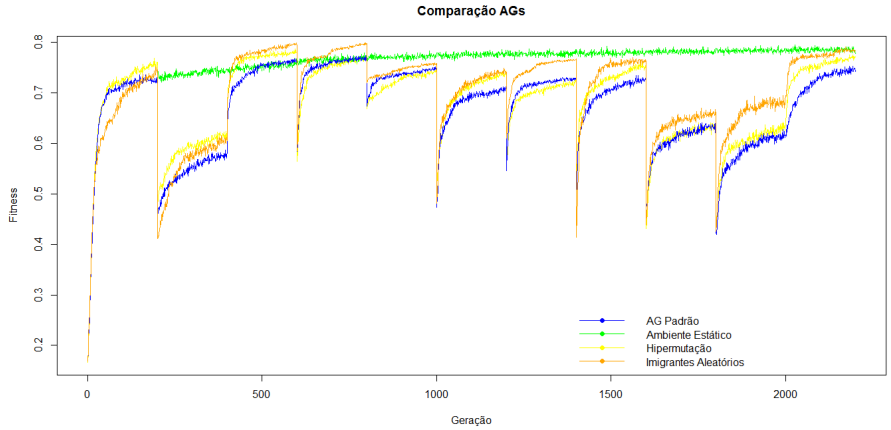

Figura 3. Média do fitness do melhor indivíduo (considerando 25 execuções) para: i) AG estático (sem considerar mudança no ambiente durante otimização); ii) AG padrão; iii) AG com hipermutação; iv) AG com imigrantes aleatórios. Considerou-se aqui o Enfoque por Otimização Evolutiva Dinâmica.

Para comparar a robustez dos algoritmos, foram selecionados o melhor indivíduo entre todas as execuções para cada um do quatro algoritmos estudados. Este indivíduos foram avaliados 10 vezes (com diferentes posições e orientações iniciais) em 5 ambientes diferentes dos apresentados durante o processo de otimização. Os novos ambientes podem ser vistos na Figura 5. Como os novos ambientes apresentam diferentes níveis de dificuldade, os resultados de robustez são analisados por ambiente. A Tabela 1 mostra os resultados obtidos. Para avaliar a significância estatística, foi utilizado o teste de Wilcoxon Signed-Rank com um nível de significância de 5%. Os resultados do AG com imigrantes aleatórios foram comparados com os resultados dos outros algoritmos.

3.2. Enfoque por Otimização Evolutiva Robusta

A diferença deste experimento para o anterior é que ao invés do ambiente ser alterado depois de um certo número de gerações (Enfoque por Otimização Evolutiva Dinâmica), cada indivíduo é avaliado nos 10 ambientes ao mesmo tempo, ou seja, a Eq. 4 é utilizada para a avaliação do fitness ao invés da Eq. 2. Os ambientes utilizados durante a otimização e para posterior teste da robustez são os mesmos utilizados anteriormente. A evolução do fitness do melhor indivíduo para cada algoritmo é apresentada na Figura 4.

Tabela 1. Resultados para o enfoque por Otimização Evolutiva Dinâmica. São mostrados os resultados da média e desvio padrão dos fitness do melhor indivíduo da última geração de todas as execuções de cada um dos quatro AGs avaliado nos 5 novos ambientes. A letra S indica que a comparação de resultados do AG com Imigrantes Aleatórios com o respectivo algoritmo é estatisticamente significante considerando-se o Teste Wilcoxon Signed-Rank . Os símbolos $+$ e $-$ significam respectivamente que médias do AG com Imigrantes Aleatórios foi maior ou menor que a média do respectivo algoritmo.

Ambiente	AG Padrão	Ambiente Estático	Hipermutação	Imigrantes Aleatórios
1	0,4385 ± 0,1921 (S+)	0,4346 ± 0,1918 (S+)	0,4855 ± 0,1974 (+)	0,5304 ± 0,1738
2	0,6776 ± 0,1459 (S+)	0,7174 ± 0,1192 (S+)	0,6821 ± 0,1418 (S+)	0,7295 ± 0,1026
3	0,4532 ± 0,1824(-)	0,4886 ± 0,2161(-)	0,4009 ± 0,1609 (+)	0,4316 ± 0,2090
4	0,6068 ± 0,1853 (S+)	0,6527 ± 0,1866 (+)	0,6698 ± 0,1562 (+)	0,7215 ± 0,1348
5	0,4963 ± 0,1869(-)	0,4137 ± 0,1790 (+)	0,4468 ± 0,2006 (-)	0,4446 ± 0,2182

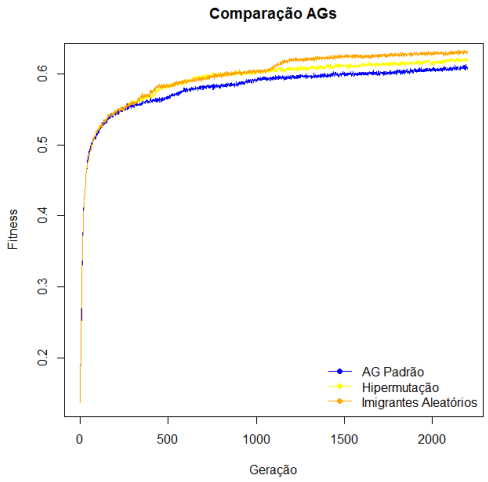

Figura 4. Média do fitness do melhor indivíduo de cada execução de cada um dos três AGs para o Enfoque por Otimização Evolutiva Robusta.

Do mesmo modo que para o enfoque anterior, foram selecionados o melhor indivíduo da última geração de todas as execuções dos quatro algoritmos estudados, e em seguida cada um desses indivíduos foram avaliados 10 vezes em 5 ambientes diferentes dos apresentados durante o processo de otimização. A Tabela 2 mostra os resultados obtidos.

Tabela 2. Resultados para o enfoque por Otimização Evolutiva Robusta.

Ambiente	AG Padrão	Ambiente Estático	Hipermutação	Imigrantes Aleatórios
1	0,3898 ± 0,1837(+)	0,4346 ± 0,1918 (+)	0,3915 ± 0,1889 (+)	0,4605 ± 0,2133
2	0,7427 ± 0,0450 (+)	0,7174 ± 0,1192 (+)	0,7509 ± 0,0453(-)	0,7433 ± 0,0473
3	0,5581 ± 0,1842 (+)	0,4886 ± 0,2161 (+)	0,5562 ± 0,1822 (+)	0,5582 ± 0,1693
4	0,6097 ± 0,1792(-)	0,6527 ± 0,1866 (S-)	0,6053 ± 0,1791 (+)	0,6091 ± 0,1870
5	0,6240 ± 0,1405 (+)	0,4137 ± 0,1790 (S+)	0,6307 ± 0,1227 (+)	0,6690 ± 0,1135

3.3. Comparação e análise dos resultados obtidos

Para o Enfoque por Otimização Evolutiva Dinâmica, o AG com imigrantes aleatórios se destaca por ter obtido geralmente o maior fitness entre os AGs na maioria dos ambientes de simulação. Isto ocorre tanto durante a otimização (Figura 3) quanto para o teste de

robustez (Tabela 1). Os resultados do AG foram superiores em 11 dos 15 casos, sendo estatisticamente significante em 6.

Já para o Enfoque por Otimização Evolutiva Robusta, houve a mudança na função de fitness. Com esta alteração, ao invés do ambiente se alterar de tempos em tempos, os indivíduos são avaliados nos 10 ambientes durante toda a fase de otimização (exceto para o AG no Ambiente Estático). É possível observar que aqui também a estratégia dos Imigrantes Aleatórios produz os melhores resultados durante a otimização (Figura 4). Os melhores resultados da estratégia por Imigrantes Aleatórios são explicados pela maior diversidade da população propiciada durante o processo de otimização. Tal diversidade é portanto importante em problemas de otimização robusta. No teste da robustez, o AG com imigrantes aleatórios foi superior em 12 dos 15 casos; entretanto foi significativamente superior em apenas um caso (Tabela 2).

Ambiente	AG Padrão	Hipermutação	Imigrantes Aleatórios
1	+	+	+
2	-	-	-
3	-	S-	S-
4	-	+	S+
5	S-	S-	S-

Tabela 3. Comparação das duas estratégias. Os símbolos $+$ e $-$ aparecem respectivamente quando a estatégia por otimização evolutiva dinâmica é melhor ou pior que a estratégia por otimização evolutiva robusta.

Na Tabela 3, são comparados os resultados das duas estratégias. Nota-se que, para o AG com Imigrantes Aleatórios, a segunda estratégia (otimização robusta) é melhor que a primeira (otimização dinâmica) nos ambientes 2, 3 e 5. Na Figura 5 são apresentados algumas soluções produzidas pelo AG com Imigrantes Aleatórios obtidas pelo enfoque robusto. Verifica-se que 1, 3 e 5 são os mais desafiadores pois são os que mais diferem do ambiente estático (sem obstáculos). O AG evoluído por ambiente estático produziu em geral soluções em que a solução evoluída foi a de andar rente a parede. Nota-se que tal estratégia não é boa para alguns ambientes, como por exemplo o ambiente 3. É possível observar na Figura 5, que o AG com imigrantes aleatórios produziu uma solução que produz bons resultados para este ambiente.

3.4. Teste das soluções em um robô real

Algumas das soluções evoluídas no robô simulado foram testadas em um robô real. Os experimentos ocorreram utilizando a plataforma robótica Curumim desenvolvida pela empresa XBot. O robô simulado possui as mesmas características da configuração do robô real utilizada nesta pesquisa (ver Seção 2). Como dito na Seção 2.6, pode ser necessário evoluir por mais algumas gerações a população de soluções obtidas durante os experimentos com o simulador. Entretanto, esta evolução adicional não foi necessária, mostrando que as soluções evoluídas são robustas a mudanças no problema. Neste caso, a mudança ocorreu devido a transferência do soluções do robô simulado para o robô real.

Nos experimentos com o robô real, a melhor solução obtida nos experimentos com o AG com Imigrantes Aleatórios no Enfoque por Otimização Evolutiva Robusta foi testada em quatro experimentos na arena sem obstáculos. Nestes quatro experimentos, a posição inicial do robô era diferente. O AG com Imigrantes Aleatórios no Enfoque por Otimização Evolutiva Robusta foi escolhido pois este apresentou os melhores resultados

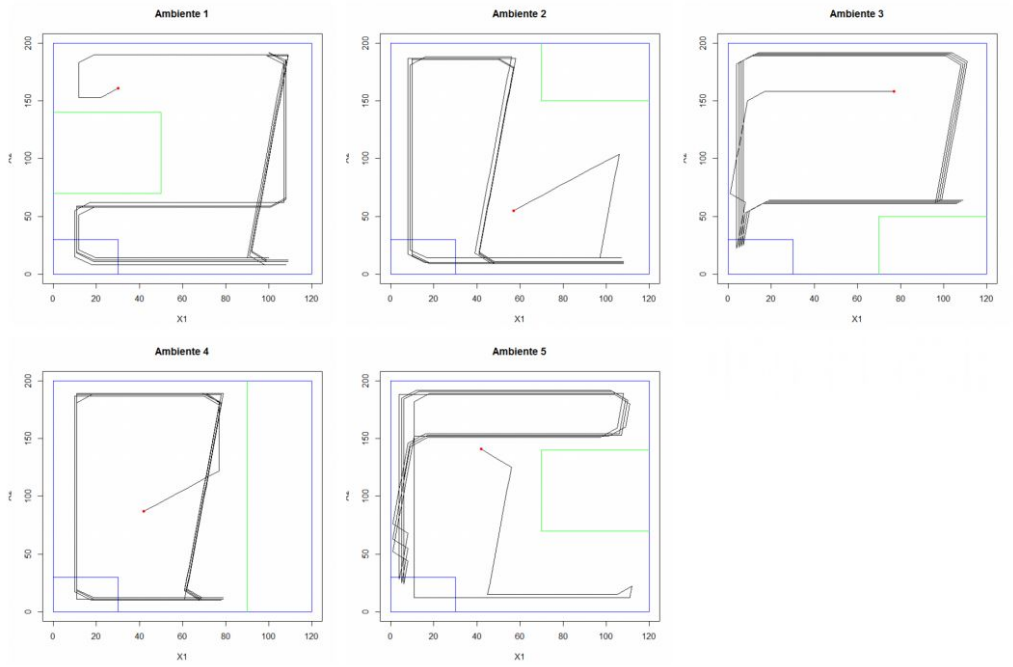

Figura 5. Trajeto percorrido pela solução produzida pelo AG com Imigrantes Aleatórios (Enfoque por Otimização Evolutiva Robusta). A área de recarga é representada pelo retângulo azul na parte inferior da arena. Os obstáculos são representados pelos retângulos em verde.

nas simulações considerando-se a robustez das soluções. Os valores de fitness obtidos nos experimentos foram: 0,7; 0,74; 0,73 e 0,69. Quando comparados com os valores obtidos na simulações (Tabela 2), é possível perceber a boa qualidade das soluções. O robô foi capaz de percorrer a arena sem se chocar com as paredes, retornando periodicamente à área de recarga. Um vídeo dos experimentos pode ser visto em `https://www.youtube.com/watch?v=v8oEpdo4dc8&feature=youtu.be`.

4. Conclusões

Todos os AGs estudados conseguiram encontrar leis de controle robustas à mudanças no ambiente, obtendo soluções que não colidiram em nenhum dos ambientes utilizados na fase de teste de robustez. Em relação às técnicas estudadas, o destaque foi para o Algoritmo Genético com Imigrantes Aleatórios, que se mostrou superior na maioria dos testes realizados. Os resultados encontrados pelo segundo experimento, na qual a robustez era considerada durante o processo de otimização, foram geralmente superiores aos resultados encontrados no primeiro onde a robustez não era considerada.

Apesar de o AG evoluído para o ambiente estático produzir boas trajetórias em ambientes que lembram o ambiente estático (sem obstáculos), ele apresenta trajetórias não satisfatórias em ambientes muito diferentes do ambiente estático. Tanto o AG evoluído no enfoque por otimização dinâmica, como o AG evoluído no enfoque por otimização robusta, apresentaram bons resultados em ambientes que diferem do ambiente estático. Isto mostra que tais enfoques conseguiram produzir soluções robustas às mudanças nos ambientes. Além disso, as técnicas de manutenção de diversidade se mostraram bastante interessantes no problema estudado. No futuro, mais experimentos com robô real devem

ser realizados. Também, outras estratégias robustas descritas em [Fu et al. 2015] devem ser investigadas.

Agradecimentos

Os autores agradecem à FAPESP (processos 2017/06757-7, 2017/11139-0 e 2015/06462-1) e ao CNPq pelo apoio financeiro.

Referências

Beyer, H.-G. and Sendhoff, B. (2007). Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33-34):3190–3218.

Billard, A., Ijspeert, A. J., and Martinoli, A. (1999). A multi-robot system for adaptive exploration of a fast-changing environment: Probabilistic modeling and experimental study. *Connection Science*, 11(3-4):359–379.

Branke, J. (2002). *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers.

Cobb, H. G. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. Technical report, Naval Research Lab Washington DC.

Floreano, D. and Nolfi, S. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press/Bradford Books.

Fu, H., Sendhoff, B., and Tang, K. (2015). *Robust optimization over time: Problem difficulties and benchmark problems*. IEEE Transactions on Evolutionary Computation.

Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80.

Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation*, 9(3):303–317.

Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press.

Romero, R. A. F., Prestes, E., Osório, F., and Wolf, D. (2014). *Robótica Móvel*. São Paulo: LTC.

Shimo, H. K., Roque, A. C., Tinós, R., Tejada, J., and Morato, S. (2010). Use of evolutionary robots as an auxiliary tool for developing behavioral models of rats in an elevated plus-maze. In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, pages 217–222. IEEE.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Tinós, R. and de Carvalho, A. C. P. L. F. (2006). Use of gene dependent mutation probability in evolutionary neural networks for non-stationary problems. *Neurocomputing*, 70(1-3):44–54.

Tinós, R. and Yang, S. (2014). Analysis of fitness landscape modifications in evolutionary dynamic optimization. *Information Sciences*, 282:214–236.

Webb, B. (2001). Can robots make good models of biological behaviour? *Behavioral and brain sciences*, 24(6):1033–1050.