Algoritmos Genéticos e Redes Neurais Artificiais no Controle de Robôs Móveis em Ambientes Dinâmicos

Eder A. B. de Oliveira[1], **Rafael Del Lama**[1], **Renato Tinós**[1]

[1]Departamento de Computação e Matemática
Faculdade de Filosofia, Ciência e Letras de Ribeirão Preto
Universidade de São Paulo (USP)
Ribeirão Preto, SP, Brasil

eder_oliveira@usp.br, rafael.lama@alumni.usp.br, rtinos@ffclrp.usp.br

Abstract. *The development of robust control laws for mobile robots in Dynamic Environments is a challenging problem. The objective of this work is the search for control solutions for mobile robots that adapt to changes in the environment. The proposed strategy starts from an algorithm developed by the programmer to perform a task in a predefined environment. Then, an Echo-State Network (ESN) is trained in a supervised way to reproduce the input-output mapping of the algorithm developed by the programmer. Finally, the trained ESN is adapted every time the environment changes. For this, the ESN weights are optimized using Genetic Algorithms (GAs). Evolutionary techniques developed for dynamic environments (GAs with random immigrants and hypermutation) are compared in experiments in which ten new environments are considered. The robustness of the control laws found is tested by presenting five new environments. In the simulations, the use of GAs allowed the adaptation of the control laws to the new environments. However, the solutions found were not necessarily robust to new changes in the environment.*

Resumo. *O desenvolvimento de leis de controle robustas para robôs móveis em Ambientes Dinâmicos é um problema desafiador. O objetivo deste trabalho é o busca de soluções de controle de robôs móveis que se adaptem às mudanças no ambiente. A estratégia proposta parte de um algoritmo desenvolvido pelo programador para realizar uma tarefa em um ambiente pré-definido. Então, Redes Neurais Recorrentes (RNN) do tipo Echo-State Network (ESN) são treinadas de maneira supervisionada para reproduzir o mapeamento entrada-saída do algoritmo desenvolvido pelo programador. Finalmente, adapta-se a ESN treinada toda vez que o ambiente muda. Para isso, os pesos da ESN são otimizados por meio de Algoritmos Genéticos (AGs). Técnicas evolutivas desenvolvidas para ambientes dinâmicos (AGs com imigrantes aleatórios e hipermutação) são comparadas em experimentos no qual dez novos ambientes são considerados. Por fim, testa-se a robustez das leis de controle encontradas apresentando-se cinco novos ambientes. Nas simulações, o uso de AGs permitiu a adaptação das leis de controle para os novos ambientes. Entretanto, as soluções encontradas não se mostraram necessariamente robustas a novas mudanças de ambiente.*

1. Introdução

Na Robótica Evolutiva, Algoritmos Evolutivos (AEs) são aplicados para o projeto de robôs e seus algoritmos. AEs podem ser utilizados, entre outros, para a definição da

arquitetura dos robôs móveis, para a a otimização de suas leis de controle, no planejamento de rotas e de estratégias de navegação. Algoritmos Evolutivos são inspirados em princípios da Genética e da Teoria da Evolução por Seleção Natural [Mitchell 1996].

O projeto de estratégias de controle para robôs autônomos em ambientes não-estruturados, dinâmicos ou parcialmente desconhecidos é uma tarefa bastante difícil para um projetista humano. Tal dificuldade é oriunda da imprevisibilidade das situações que poderão ser confrontadas pelos robôs nestes ambientes [Meyer 1998]. Robôs cujo design e/ou algoritmos são projetados por AEs são denominados Robôs Evolutivos (REs) [Floreano and Nolfi 2000].

Com utilização de AEs, o ambiente e a tarefa a ser executada tornam-se os principais fatores no desenvolvimento do robô e de suas leis de controle, reduzindo a interferência do projetista. O uso de AEs em robôs móveis conecta a Robótica e a Biologia. A Robótica é beneficiada pelo uso de algoritmos com inspiração biológica, bem como robôs autônomos podem ser utilizados como ferramenta de estudo, teste e modelagem de estruturas comportamentais, habilidades cognitivas e modelos evolutivos de organismos vivos [Webb 2001] [Shimo et al. 2010]. O crescente uso de robôs em aplicações que envolvem ambientes não estruturados implica muitas vezes na busca por estratégias que produzam soluções que mudam com o tempo [Branke 2001] ou solução robustas. No geral, busca-se soluções robustas às variações de ambiente e/ou soluções que se adaptam às possíveis mudanças ambientais

Mudanças nas leis de controle podem também ser necessárias quando falhas ocorrem. Robôs Autônomos são dependentes das informações obtidas por seus sensores e da ação de seus atuadores. Entretanto, falhas em componentes mecânicos ou eletrônicos podem ocorrer, levando os robôs a se comportarem de maneira inesperada [Parasuraman 2015]. Falhas podem implicar em mudanças na função de avaliação (*fitness*) utilizadas pelos AEs na otimização dos robôs e seus algoritmos. Quando mudanças no hardware ou ambiente ocorrem, a otimização deixa de ser estática, tornando-se dinâmica [Branke 2001]. A Computação Evolutiva Dinâmica visa estudar AEs para tais Problemas de Otimização Dinâmica.

Nos últimos anos, pesquisadores da área Computação Evolutiva Dinâmica têm também estudado algoritmos para gerar soluções robustas para ambientes dinâmicos [Beyer and Sendhoff 2007, Fu et al. 2015]. Duas estratégias principais se destacam quando AEs são utilizados para produzir tais soluções: *Tracking Moving Optimal* (TMO) [Aragón and Esquivel 2004] e, mais recentemente, *Robust Optimization Over Time* (ROOT) [Yu et al. 2010]. Na primeira, busca-se as melhores soluções sempre que o problema muda. Já em ROOT, busca-se uma única solução que apresentará melhor desempenho médio para vários ambientes.

Neste trabalho, explora-se o uso combinado de Redes Neurais Artificiais (RNAs) e Algoritmos Genéticos (AGs) na otimização e adaptação das leis de controle de um robô móvel em ambientes sujeitos a mudanças. Diferentemente de outras estratégias [Del Lama et al. 2018, Raineri et al. 2019], a solução é evoluída a partir de um algoritmo de controle simples definido pelo projetista para uma tarefa específica. Na tarefa requerida aqui, o robô deve navegar em uma área definida utilizando o algoritmo projetado pelo programador.

Durante a navegação, são mapeados os valores das leituras dos sensores e saídas dos atuadores em cada instante da trajetória percorrida pelo robô. Com estes dados, é realizado o treinamento supervisionado de uma Rede Neural do tipo *Echo State Network* (ESN) [Jager and Haas 2004], que posteriormente deverá controlar o robô. A ESN treinada pelo algoritmo supervisionado para um ambiente específico é então otimizada por um AG toda vez que o ambiente muda. ESNs são interessantes para este problema por apresentar memória e devido ao fato de possuírem poucos parâmetros ajustáveis (pesos sinápticos), reduzindo portanto o espaço de busca a ser explorado pelo AG [Raineri et al. 2019, Del Lama et al. 2018]. Investiga-se ainda o impacto de técnicas de manutenção de diversidade, como Hipermutação e Imigrantes Aleatórios [Cobb and Grefenstette 1993], para o AG utilizado para a adaptação dos pesos da ESN. Os métodos desenvolvidos são analisados em experimentos na abordagens TMO e ROOT.

2. Metodologia

O robô móvel utilizado nas simulações possui quatro sensores, sendo um sensor frontal, um voltado para cima, um na diagonal direita e um na diagonal esquerda (ambos formando um angulo de 45 graus com o sensor frontal). O sensor voltado para cima é utilizado para detectar o ponto de recarga da bateria, que é a única área coberta do ambiente. Os outros sensores são utilizados para a detecção de obstáculos. Esta arquitetura já foi utilizado em outros trabalhos do grupo de pesquisa [Del Lama et al. 2018]. O robô deve navegar o máximo possível em linha reta no ambiente, evitando obstáculos e retornando à área de recarga da bateria sempre que necessário. A cada iteração, o robô pode realizar as seguintes ações: andar para frente 10cm, girar -45 graus, girar +45 graus, ou girar +90 graus.

Aqui, na fase chamada de 'pré-treinamento', o robô simulado é colocado em um ambiente de 100cm x 200cm, sem obstáculos (além das paredes). O robô é controlado pelo Algoritmo 1, desenvolvido pelo programador. O algoritmo orienta o robô a desviar de obstáculos simples, desviando para esquerda ou direta caso um obstáculo seja detectado pelo sensor frontal; caso contrário, movimenta-se para frente. Não se considera neste algoritmo a necessidade de retorno para a área de recarga da bateria.

Na estratégia proposta, mapeia-se as entradas (sensores) e saídas (ações) do robô controlado pelo algoritmo definido pelo programador. Uma ESN é pré-treinada para mapear as entradas e saídas utilizadas no controle do robô. Nesta etapa o treinamento ocorre de modo supervisionado. Na etapa de pré-treinamento da ESN, consideram-se 500 repetições de trajetórias iniciadas em pontos distintos do ambiente e um limite de 300 iterações (ações do robô) por repetição. O objetivo é posteriormente adaptar a ESN para atuar em ambientes com recarga da bateria, com obstáculos e diferentes configurações. A adaptação da ESN ocorre por meio do AG, sendo que neste caso não existem saídas desejadas. O AG otimiza o vetor de pesos entre a camada intermediaria (reservatório) e a camada de saída da ESN.

2.1. Echo State Network

Tradicionalmente uma ESN é composta por um reservatório com neurônios esparsamente conectados através de pesos gerados aleatoriamente e posteriormente normalizados [Jaeger 2001]. Os pesos das conexões entre as entradas e o reservatório também são

Algoritmo 1: ALGORITMO DEFINIDO PELO PROGRAMADOR

```
1  início
2      inicialização;
3      lerSensores();
4      if (sensorFrente == 0) then
5          andarFrente();
6      end
7      else if (sensorDireita == 0) then
8          virarDireita(45 graus);
9      end
10     else if (sensorEsquerda == 0) then
11         virarEsquerda(45 graus);
12     end
13     else
14         virarEsquerda(90 graus);
15     end
16 fim
```

aleatórios. Ao utilizar um reservatório grande, é possível a obtenção de comportamentos dinâmicos complexos. Em uma rede ESN supervisionada , como a aqui utilizada na fase de pré-treinamento, o ajuste dos pesos entre os neurônios do reservatório e os neurônios da camada de saída é realizado utilizando-se regressão linear, o que torna o aprendizado mais rápido [Jager and Haas 2004]. Diferentemente de redes neurais recorrentes tradicionais, não é necessário realizar o treinamento de todos os pesos de todas as conexões entre os nós que compõem a rede neural, o que torna o treinamento mais simples.

Na ESN utilizada neste trabalho, cada sensor do robô corresponde a uma entrada da ESN e suas saídas representam cada uma das 4 ações possíveis de serem executadas. A Figura 1 mostra a ligação entre os nós que compõem a ESN de controle do robô.

A arquitetura aqui utilizada é a mesma empregada em [Del Lama et al. 2018]. A ESN é composta de 4 entradas (um para cada sensor), 50 neurônios no reservatório e 4 saídas (ações). A densidade de conexão do reservatório é definida em 0.15. O raio espectral, utilizado para normalizar os pesos aleatórios da camada de entrada até o reservatório, é igual a 0.95.

2.2. Algoritmos Genéticos

O pseudo-código do AG geracional padrão [Goldberg 1989] utilizado aqui é apresentado no Algoritmo 2, sendo $Pop(t)$ a população na geração t. O AG geracional padrão cria uma população inicial de indivíduos (cromossomos) aleatórios, sendo cada indivíduo composto por genes que representam possíveis partes da solução para o problema de otimização. A população então é submetida a uma avaliação, utilizando para isso uma função de avaliação. Então são aplicados operadores de seleção e transformação. Os detalhes do AG empregado para otimizar a ESN são a seguir apresentados.

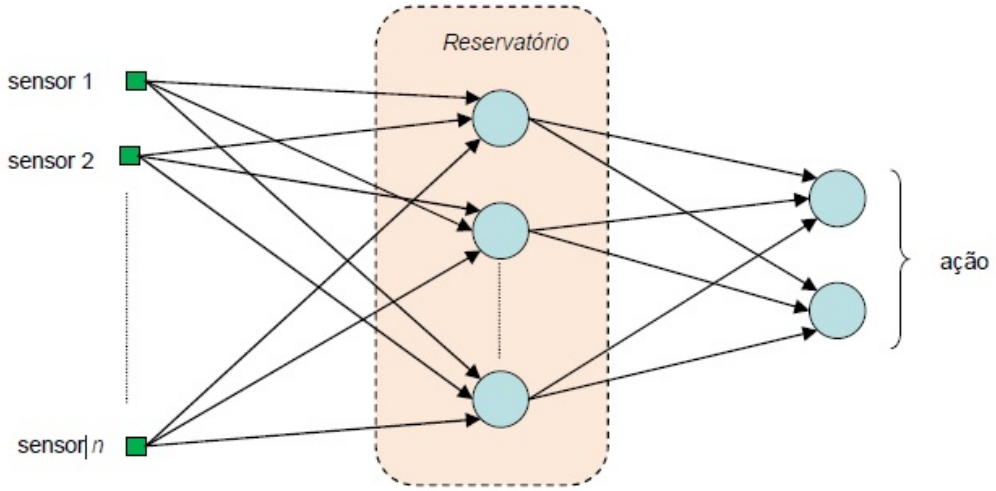

Figura 1. Rede ESN utilizada para controle do Robô.

2.2.1. Codificação

Após o treinamento inicial, a ESN é otimizada (treinada) por meio do AG. Portanto, o conjunto de pesos entre o reservatório e a última camada da ESN é codificado em um vetor de pesos reais (cromossomo) contido no indivíduo do AG. Deste modo, cada cromossomo gera uma ESN, geralmente implicando em um comportamento diferente para o robô e, portanto, em uma trajetória diferente. A primeira população do AG é formada pela cópia, com mutações, da solução obtida pelo treinamento supervisionado da ESN para o ambiente inicial.

2.2.2. Reprodução e Seleção

Utiliza-se para reprodução durante o processo de otimização, operadores de *crossover* de dois pontos e mutação gaussiana. Também é empregado o método de torneio para seleção, onde, um número K_t de indivíduos da população são aleatoriamente escolhidos e aquele com maior *fitness* é selecionado. Com isso, é possível controlar a pressão seletiva variando-se o parâmetro K_t. O método do torneio é computacionalmente mais simples que o método da roleta [Mitchell 1996]. O elitismo, onde os dois melhores indivíduos da população atual são passados para a população seguinte, é também empregado.

2.3. Avaliação

Após realizado o pré-treinamento supervisionado da ESN de controle do robô, busca-se uma solução que permita ao robô se locomover sem colisões, em linha reta, voltando eventualmente para a área de recarga da bateria. O *fitness* de cada solução é obtido analisando-se a trajetória do robô cuja lei de controle (ESN) está codificada no cromossomo do indivíduo (que define a ESN). A trajetória é definida no ambiente corrente. Assim, os seguintes critérios são adotados para finalizar a avaliação do indivíduo, i.e., da trajetória do robô dada pela ESN codificada no cromossomo:

- Colidir com um obstáculo;
- Não retornar a área de carregamento dentro de uma determinada quantidade de iterações (ações do robô);
- Atingir o número máximo de 300 iterações, i.e., 300 ações do robô podem ocorrer.

Portanto, o número de iterações do robô na trajetória para a solução (ESN) $\mathbf{x}$ é dado por:

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

sendo t_{choque} o instante no qual o robô colide com um obstáculo, e $t_{bateria}$ o instante o qual a carga da bateria é totalmente descarregada. A carga da bateria é simulada utilizando uma função linear cujos valores decrescem com cada iteração [Floreano and Nolfi 2000]. O tempo de descarga da bateria simulado é de 80 ou 160 iterações. O tempo de carga é considerado instantâneo, toda vez que o robô se encontra na área de recarga.

Para calcular o fitness da solução (ESN) $\mathbf{x}$, é utilizada a seguinte equação:

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

sendo que a cada iteração t o robô executa uma ação e:

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

Para obter o máximo valor de *fitness*, o robô deve ser capaz de navegar pelo ambiente em linha reta o máximo possível, sem colidir com obstáculos, retornando à área de carga, sempre que a bateria estiver quase descarregada. A tarefa de andar pelo ambiente sem se chocar é puramente reativa. Porém a restrição de carga e descarga da bateria, exige ações envolvendo memória.

Algoritmo 2: AG PADRÃO

```
1  início
2      inicializar Pop(t)
3      avaliar Pop(t)
4      enquanto critério de convergência não for satisfeito faça
5          aplicar seleção e crossover em Pop(t) para criar Pop(t+1)
6          aplicar mutação em Pop(t + 1)
7          t ← t + 1
8          avaliar Pop(t)
9      fim
10 fim
```

2.4. Técnicas de Aumento e Manutenção da Diversidade

Um das principais dificuldades do AGs padrão em sua utilização em ambientes dinâmicos é a perda da diversidade da população de soluções. Isto geralmente ocorre devido à convergência da população para ótimos locais. Quando as características do problema mudam, aumenta a dificuldade de escapar destes ótimos. Diversos mecanismos têm sido propostos para controle ou aumento a diversidade populacional em AGs de modo a melhorar

o desempenho em ambientes dinâmicos [Jin and Branke 2005]. Dentre as estratégias mais utilizadas pode se destacar as técnicas de hipermutação e de uso de imigrantes aleatórios [Cobb and Grefenstette 1993]. Estas técnicas são testadas e comparadas neste trabalho.

A **Hipermutação** é uma estratégia na qual a taxa de mutação é aumentada quando a diversidade da população de soluções atinge um patamar ou quando o algoritmo converge para uma solução. Com o aumento da taxa de mutação, aumenta-se a chance do AG escapar de um ótimo local [Cobb and Grefenstette 1993]. Ja a técnica de **Imigrantes Aleatórios** é inspirada na migração de indivíduos em uma população [Cobb and Grefenstette 1993]. Nesta estratégia, a cada geração, alguns indivíduos são substituídos por indivíduos aleatórios. Podem haver variações nessa estratégia, como a substituição de indivíduos menos aptos (*fitness* menor), ou a substituição de indivíduos aleatórios, sendo essa última a estratégia a ser empregada neste trabalho.

3. Resultados

Nas simulações, são comparadas as seguintes estratégias: ESN apenas com o pré-treinamento (ESN Puro): AG Tradicional (ESN+AG); AG com Hipermutação (ESN+AG+HIP); AG com Imigrantes Aleatórios (ESN+AG+IMI); AG com a combinação da Hipermutação e Imigrantes Aleatórios (ESN+AG+HIP+IMI). Nos AGs, a ESN obtida no pré-teinamento é adaptada durante a otimização. Todos os códigos utilizados no trabalho foram desenvolvidos em C++ e as simulações foram executadas em um computador com processador Intel i7-10700 e 16GB de memória RAM.

No simulador, em um primeiro momento é realizado o pré-treinamento da ESN para o robô controlado pelo Algoritmo 1, i.e., efetua-se o treinamento supervisionado. Então, simula-se as mudanças no ambiente. Antes de iniciar a avaliação de um indivíduo em cada ambiente, é sorteada uma posição e um ângulo inicial para o robô. Durante a trajetória, as novas posições e orientações do robô após cada ação são calculadas através de modelos cinemáticos e, por meio de cálculos geométricos, são simuladas as saídas produzidas pelos sensores. O uso de posições e ângulos iniciais aleatórios na inicialização é importante pois, assim, cada indivíduo é avaliado em condições iniciais distintas.

Todos os AGs utilizam para otimização os seguintes parâmetros: população composta por 100 indivíduos, taxa de *crossover* definida com valor de 0.6, taxa de mutação definida como $\frac{1}{m}$, sendo m tamanho do cromossomo. Para seleção por torneio utiliza-se $K_t = 3$. Além deste método de seleção, também é utilizado o método de 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. Durante a otimização são feitas 25 execuções (repetições).

Na hipermutação, a taxa de mutação aumenta toda vez que o ambiente muda, ficando quatro vezes maior que a taxa de mutação (desvio da mutação Gaussiana) durante 10 gerações. No AG com imigrantes aleatórios, a taxa de substituição de indivíduos é de 0.05, o que significa que 5% dos indivíduos da população atual são substituídos por indivíduos aleatórios em cada geração. Os parâmetros dos AGs foram obtidos em experimentos iniciais ou em trabalhos anteriores do grupo de pesquisa [Del Lama et al. 2018, Raineri et al. 2019].

3.1. Experimento com adaptação das soluções (abordagem TMO)

Neste experimento, testa-se o poder dos AGs em produzir soluções eficientes quando o ambiente muda. Durante a otimização, são inseridas mudanças ambientais a cada 200 gerações. Nas primeiras 200 gerações é utilizado o ambiente inicial sem obstáculos empregado durante o pré-treinamento da ESN, mas com área de recarga da bateria. A área de recarga tem 0.3x0.3m, sendo localizada sempre no canto inferior esquerdo de cada ambiente. Para simular as alterações ambientais, nas 1800 gerações seguintes são utilizados 9 ambientes com obstáculos variados e pré-definidos. Nas últimas 200 gerações é utilizado novamente o ambiente inicial sem obstáculos. Os nove ambientes com obstáculos são apresentados na Figura 2.

Simulações com dois valores para o tempo de bateria foram executadas. As figuras 3 e 4 apresentam, respectivamente, as médias (para 25 execuções) do fitness médio e do melhor fitness (para o melhor indivíduo) de cada geração para cada um dos algoritmos estudados, com bateria com carga para 80 iterações. As figuras 5 e 6 apresentam, respectivamente, as médias (para 25 execuções) do fitness médio e do melhor fitness (para o melhor indivíduo) de cada geração para cada um dos algoritmos estudados, com bateria com carga para 160 iterações.

Os resultados indicam que o AG foi capaz de gerar novas soluções adaptadas ao ambiente quando este mudou. As soluções obtidas pelo AG são superiores àquelas obtidas pela ESN pré-treinada (ESN Puro). Nota-se também que, como era se se esperar, melhores soluções foram obtidas quando a duração máxima (nível) da bateria aumentou. Por fim, nota-se que os AGs com controle ou manutenção de diversidade (AGs com hipermutação e/ou imigrantes aleatórios) geram melhores soluções do que aquelas obtidas pelo AG padrão (ESN+AG). O AG que apresentou o melhor resultado foi o que combinou hipermutação e imigrantes aleatórios (ESN+AG+HIP+IMI).

3.2. Experimento para testar a robustez das soluções evoluídas (abordagem ROOT)

No experimento anterior, verificou-se a capacidade de adaptar as soluções durante a evolução do AG. Aqui, avalia-se a robustez das soluções geradas no experimento anterior em 5 novos ambientes. Em outras palavras, a melhor solução de cada execução do experimento anterior é testada em 5 novos ambientes. Busca-se assim verificar se as soluções evoluídas pelo AG são robustas. Os novos ambientes podem ser vistos na Figura 7. Como os novos ambientes apresentam diferentes níveis de dificuldade, os resultados são analisados por ambiente.

As tabelas 1 e 2 mostram os valores médios para os melhores indivíduos obtidos nas 25 execuções do experimento anterior. Os resultados são para a avaliação dos novos ambientes. Para avaliar a significância estatística das comparações entre os algoritmos e o ESN+AG+HIP+IMI, foi utilizado o teste de *Wilcoxon Signed-Rank* com um nível de significância de 5%. Em nenhuma vez detectou-se significância estatística. Este resultado indica que, apesar de os algoritmos baseados em AG permitirem a adaptação das soluções (ver seção anterior), as soluções obtidas não foram mais robustas que aquelas obtidas pela ESN pré-treinada.

4. Conclusões

No experimento com ambientes dinâmicos (abordagem TMO), os AGs foram capazes de encontrar leis de controle eficientes quando o ambiente mudou. Os AGs com técnicas

Tabela 1. Média e desvio-padrão da avaliação do melhor indivíduo (para cada execução) para 5 novos ambientes, para o experimento com duração máxima de bateria igual a 80 iterações. As comparações em relação ao algoritmo ESN+AG+HIP+IMI são apresentadas. Sinais +, -, = respectivamente indicam que o respectivo algoritmo apresentou resultado melhor, pior ou igual quando comparado com ESN+AG+HIP+IMI. O melhor resultado médio para cada execução é destacado.

Amb.	ESN Puro	ESN+AG	ESN+AG+HIP	ESN+AG+IMI	ESN+AG+HIP+IMI
10	0.082±0.124(+)	**0.098±0.142**(+)	0.091±0.138(+)	0.082±0.113(+)	0.077±0.166
11	**0.109±0.180**(+)	0.089±0.114(+)	0.098±0.152(+)	0.102±0.152(+)	0.064±0.135
12	0.075±0.121(-)	0.081±0.118(+)	0.086±0.142(+)	**0.096±0.153**(+)	0.078±0.152
13	**0.085±0.131**(+)	0.084±0.119(+)	0.084±0.143(+)	0.084±0.132(+)	0.063±0.143
14	0.075±0.119(+)	0.091±0.152(+)	0.088±0.158(+)	**0.092±0.149**(+)	0.053±0.124

Tabela 2. Média e desvio-padrão da avaliação do melhor indivíduo (para cada execução) para 5 novos ambientes, para o experimento com duração máxima de bateria igual a 160 iterações.

Amb.	ESN Puro	ESN+AG	ESN+AG+HIP	ESN+AG+IMI	ESN+AG+HIP+IMI
10	0.083±0.131(-)	**0.098±0.174**(+)	0.058±0.134(-)	0.084±0.129(-)	0.086±0.159
11	0.093±0.149(-)	0.095±0.170(-)	0.081±0.166(-)	0.124±0.191(-)	**0.127±0.237**
12	0.058±0.122(-)	**0.090±0.173**(+)	0.064±0.121(-)	0.086±0.143(+)	0.083±0.148
13	0.074±0.128(+)	**0.099±0.188**(+)	0.077±0.168(+)	0.087±0.148(+)	0.069±0.160
14	0.075±0.127(+)	**0.093±0.160**(+)	0.073±0.123(+)	0.056±0.084(-)	0.072±0.157

de aumento e manutenção de diversidade (AG com hipermutação e imigrantes aleatórios) apresentaram resultados superiores ao do AG padrão no primeiro experimento, no qual as soluções eram adaptadas conforme o ambiente mudava. O destaque foi o AG que combinou hipermutação e imigrantes aleatórios. Entretanto, o segundo experimento (abordagem ROOT) mostrou que as soluções evoluídas pelos AG não foram mais robustas, i.e., quando foram apresentadas para novos ambientes.

No futuro, experimentos com robôs reais devem ser realizados. Outras estratégias robustas devem ser testadas, como aquelas descritas em [Fu et al. 2015]. Finalmente, devem ser considerados prolemas em que ocorrem falhas de hardware durante a otimização.

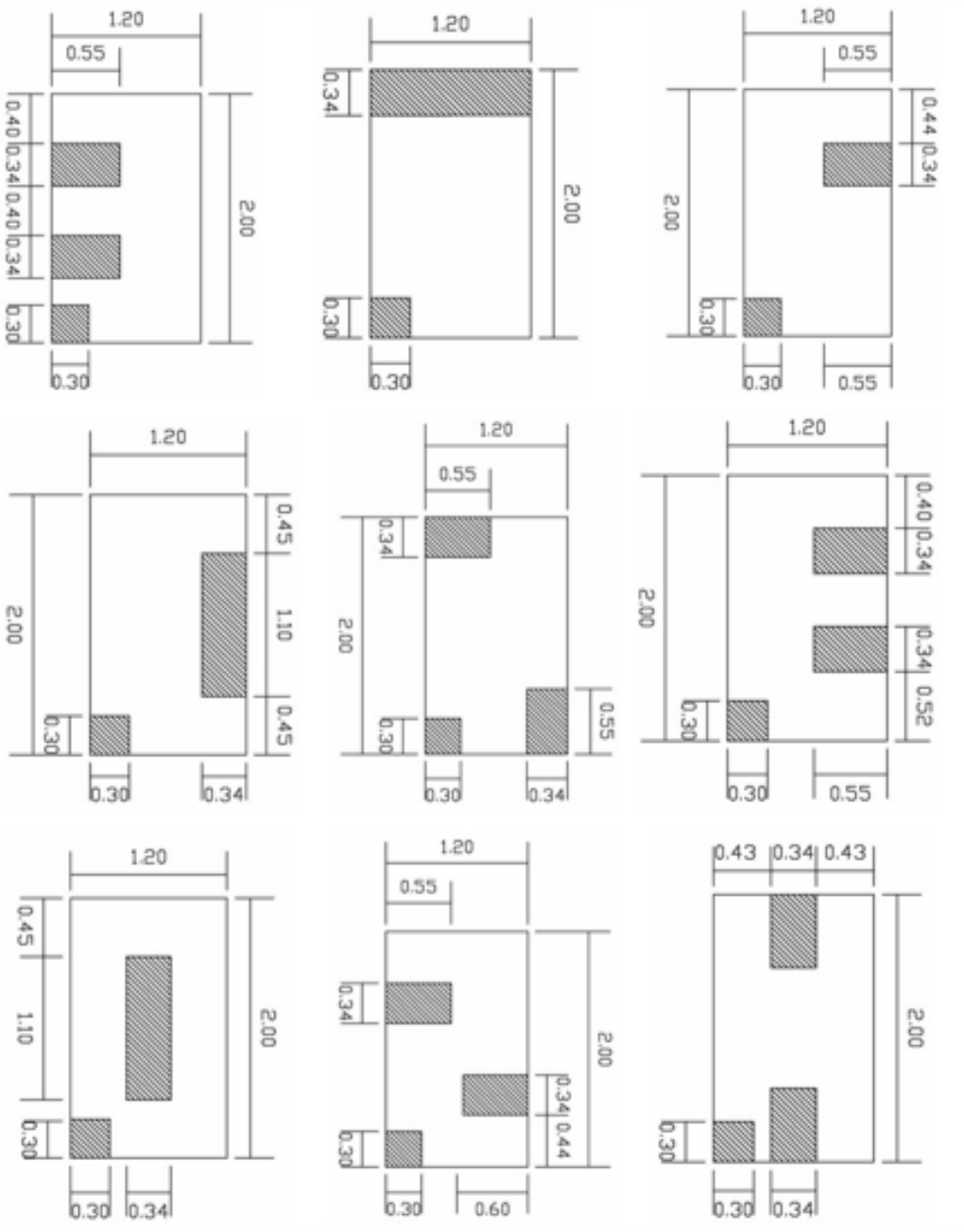

Figura 2. Ambientes dinâmicos.

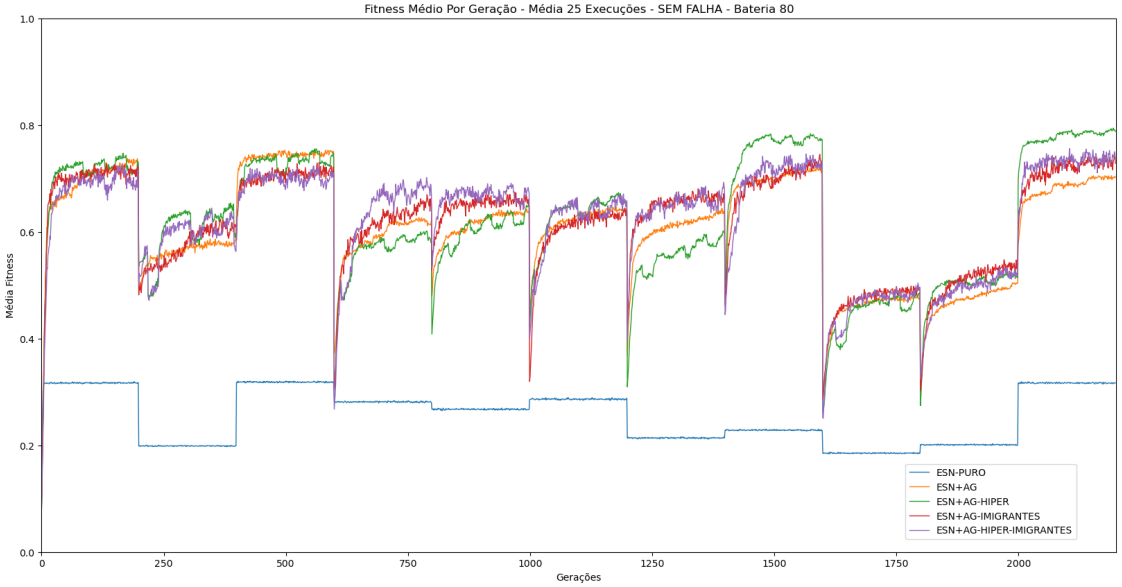

Figura 3. Fitness médio por geração nos ambientes dinâmicos para os experimentos com nível de bateria igual a 80 iterações.

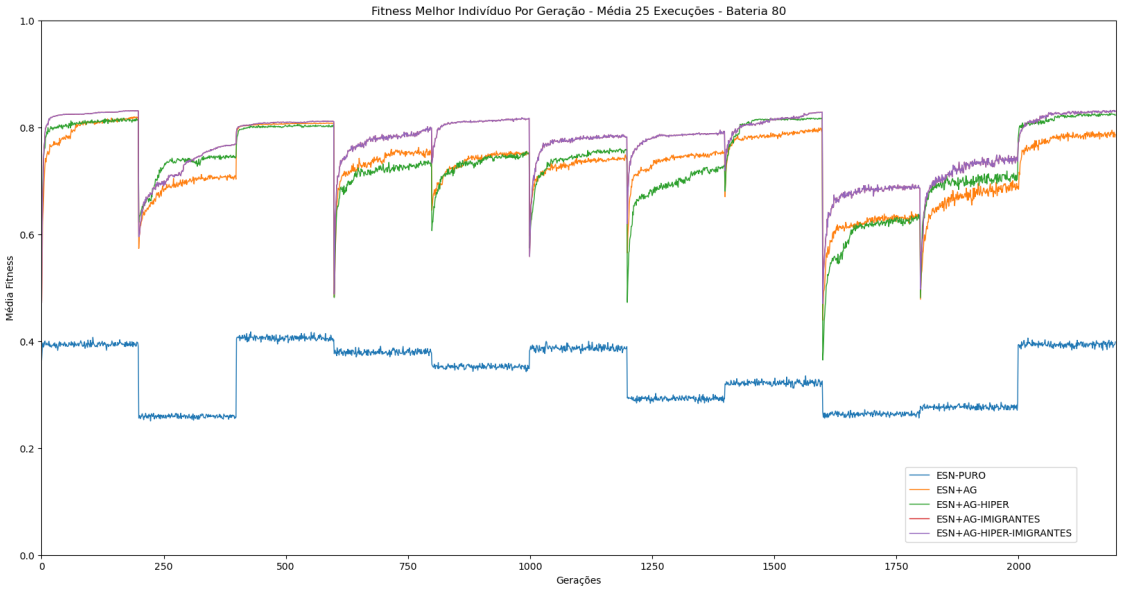

Figura 4. Fitness do melhor indivíduo por geração nos ambientes dinâmicos para os experimentos com nível de bateria igual a 80 iterações.

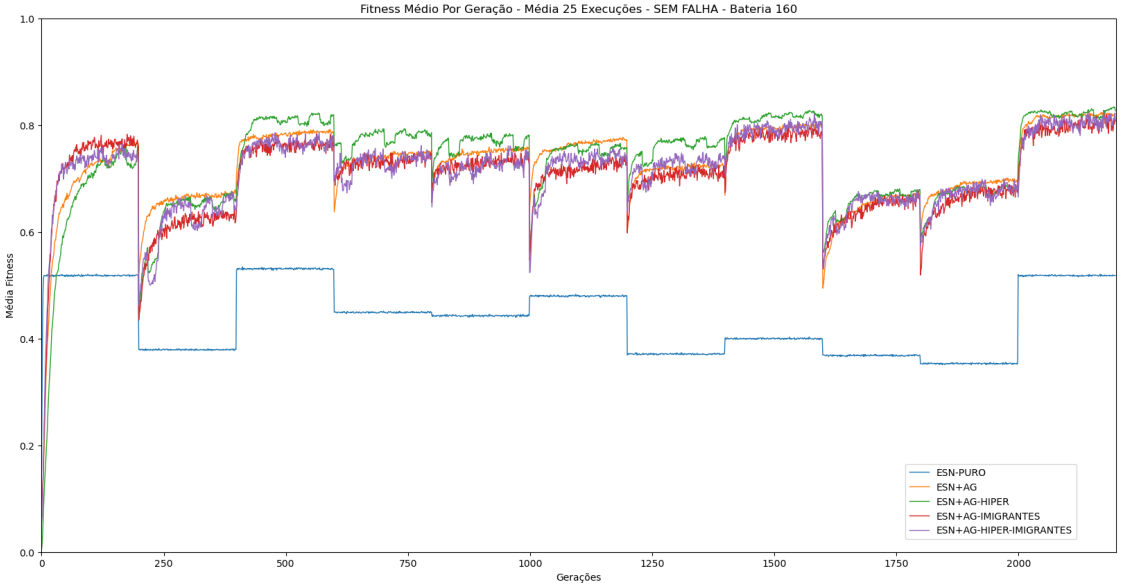

Figura 5. Fitness médio por geração nos ambientes dinâmicos para os experimentos com nível de bateria igual a 160 iterações.

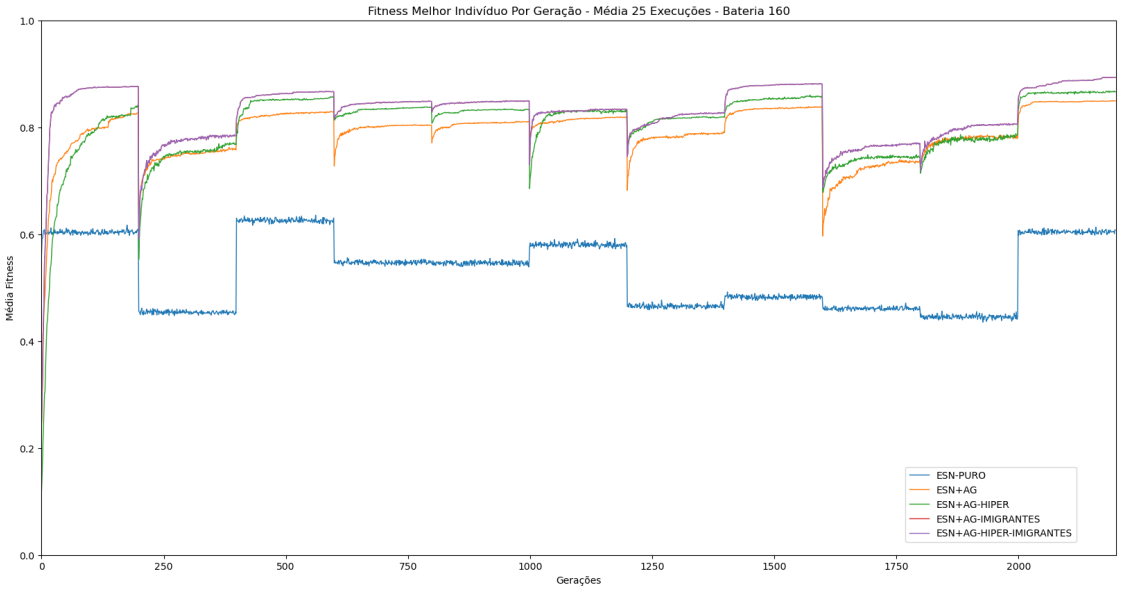

Figura 6. Fitness do melhor indivíduo por geração nos ambientes dinâmicos para os experimentos com nível de bateria igual a 160 iterações.

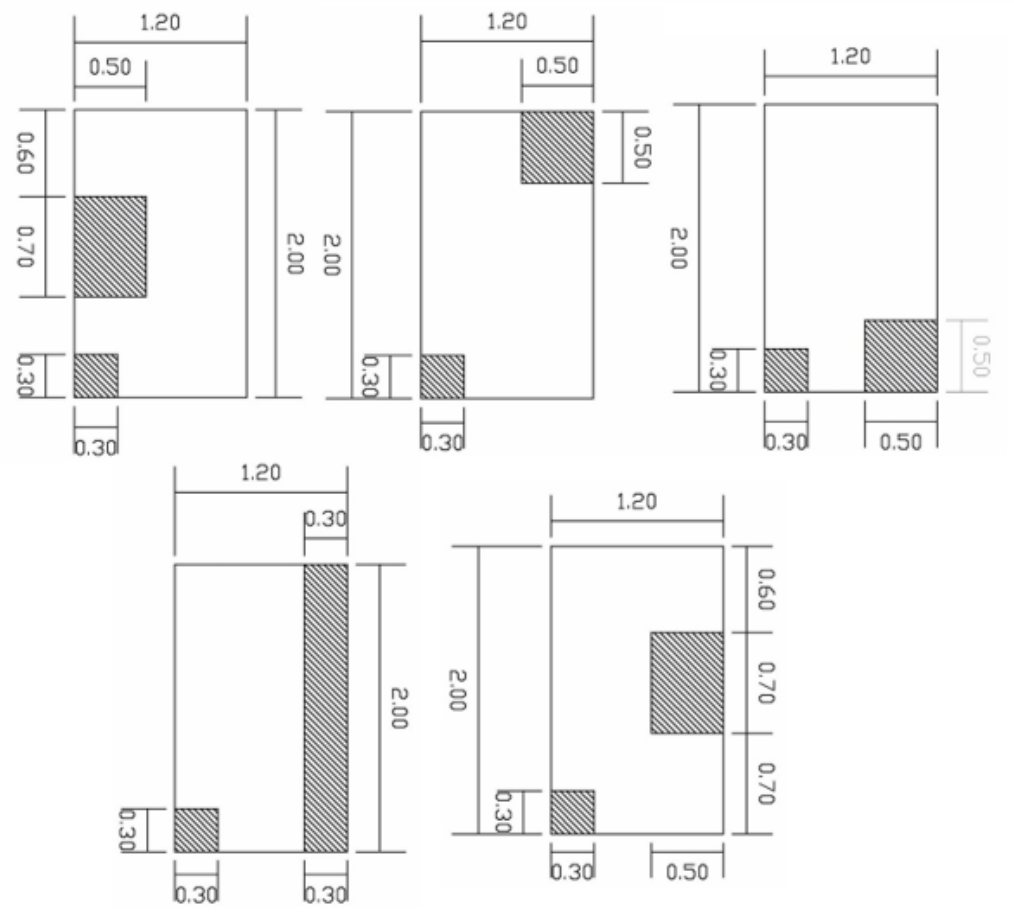

Figura 7. Ambientes (novos) utilizados para testes de robustez.

Agradecimentos

Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (projetos #2021/09720-2 e #2013/07375-0), Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (projeto #306689 /2021-9) e Centro de Inteligência Artificial - C4AI (apoiado pela FAPESP, por meio do projeto #2019/07665-4, e IBM Corporation).

Referências

Aragón, V. S. and Esquivel, S. C. (2004). An evolutionary algorithm to track changes of optimum value locations in dynamic environments. *Journal of Computer Science and Technology*, 4(3):127–133.

Beyer, H.-G. and Sendhoff, B. (2007). Robust optimization – a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196:3190–3218.

Branke, J. (2001). *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, USA.

Cobb, H. G. and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In *Proc. of the 5th International Conference on Genetic Algorithms*, page 523–530, San Francisco, CA, USA.

Del Lama, R. S., Candido, R. M., Raineri, L. T., and Tinós, R. (2018). Evolutionary optimization of robust control laws for mobile robots in dynamic environments. In *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*, pages 461–472, Porto Alegre, RS, Brasil. SBC.

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., K. Tang, K., and Yao, X. (2015). Robust optimization over time: Problem difficulties and benchmark problems. *IEEE Transactions on Evolutionary Computation*, 19:731–745.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA.

Jaeger, H. (2001). The"echo state"approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148.

Jager, 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.

Meyer, J. (1998). Evolutionary approaches to neural control in mobile robots. In *Proc. of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2418–2423.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press/Bradford Books.

Parasuraman, R. (2015). Few common failure cases in mobile robots. *CoRR*, abs/1508.03000.

Raineri, L., Lama, R. D., Candido, R., Costa, A., and Tinós, R. (2019). Robôs evolutivos para a investigação do comportamento de ratos no teste de campo aberto. *Revista Eletrônica De Iniciação Científica Em Computação*, 17(2).

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 *Proc. of the 2010 Eleventh Brazilian Symposium on Neural Networks*, pages 217–222. IEEE.

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

Yu, X., Jin, Y., Tang, K., and Yao, X. (2010). Robust optimization over time — a new perspective on dynamic optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 1–6.