

Departamento de Computação e Matemática  
Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto (FFCLRP)  
Universidade de São Paulo (USP)

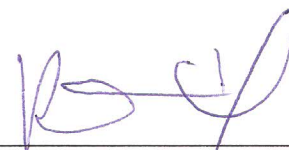
Relatório Final – Iniciação Científica  
Projeto Número: 2017/06757-7  
Período relatado: fevereiro de 2018 a agosto de 2018

# **Algoritmos Genéticos Aplicados na Busca de Soluções Robustas para Robôs Móveis em Ambientes Dinâmicos**



---

**Rafael Silva Del Lama** (Aluno)



---

**Prof. Dr. Renato Tinós**  
(Orientador)

Ribeirão Preto, setembro de 2018.

---

## SEÇÃO A - INTRODUÇÃO

---

Este é o relatório final do projeto de iniciação científica FAPESP de número 2017/06757-7 desenvolvido pelo bolsista Rafael Silva Del Lama. Não se utilizou os recursos da reserva técnica.

De acordo com o cronograma proposto, foram desenvolvidas as seguintes etapas:

- Levantamento Bibliográfico: estudo das referências relacionadas ao tema da pesquisa, principalmente otimização robusta em ambientes dinâmicos.
- Implementação da rede neural Echo State Network (ESN) para controle do robô móvel.
- Estudo e implementação dos algoritmos para busca de soluções robustas considerando-se que a robustez não será avaliada durante o processo de otimização.
- Estudo e implementação dos algoritmos para busca de soluções robustas considerando-se que a robustez será avaliada durante o processo de otimização.
- Testes com os algoritmos desenvolvidos.
- Comparação e análise dos resultados.

Este relatório está organizado da seguinte maneira:

- Seção A: fornece uma breve introdução ao conteúdo do relatório;
- Seção B: apresenta um resumo do plano inicial da pesquisa;
- Seção C: resumo das atividades realizadas durante o semestre;
- Seção D: relatório detalhado das atividades realizadas;
- Seção E: descrição e avaliação do apoio institucional;
- Seção F: conclusão
- Seção G: referências bibliográficas.

---

## SEÇÃO B – RESUMO DO PROJETO PROPOSTO

---

Este trabalho propõe um algoritmo de controle de robôs móveis nos quais o ambiente e a existência de obstáculos não são previamente conhecidos. A grande dificuldade do projeto advém da impossibilidade de prever as situações que serão confrontadas pelos robôs em tais ambientes.

Com as mudanças, soluções (leis de controle) encontradas podem se tornar ruins no novo ambiente, devido ao fato das soluções serem otimizadas considerando-se superfície de fitness estáticas.

Este projeto tem por objetivo o estudo do problema de controle de robôs móveis em ambientes dinâmicos via leis de controle robustas definidas por redes neurais artificiais otimizadas por algoritmos genéticos. Deseja-se estudar a robustez de soluções geradas por algoritmos genéticos desenvolvidos para ambientes dinâmicos, ou seja, no qual a superfície de fitness é alterada durante o processo de otimização.

---

## SEÇÃO C – RESUMO DAS ATIVIDADES DESENVOLVIDAS

---

No primeiro semestre de pesquisa foram estudadas referências bibliográficas relacionadas aos algoritmos genéticos (AGs), técnicas de manutenção de diversidade de soluções, rede ESN, otimização robusta e robótica em ambientes dinâmicos. Através do estudo destas referências foi obtida uma implementação do algoritmo da rede ESN e do AG real para ser usado como base desta pesquisa.

Foram implementados também duas técnicas para manutenção da diversidade da população no algoritmo genético: hipermutação e imigrantes aleatórios.

Os resultados produzidos pelo AG Padrão e pelas duas técnicas de manutenção da diversidade foram analisados e comparados levando em conta a robustez.

Além das referências relacionadas ao tema, foi desenvolvido um algoritmo para realizar a comunicação com o robô, baseado em algoritmos utilizados em projetos anteriores.

O simulador, desenvolvido em um projeto anterior, foi otimizado e adaptado para ser utilizado nesta pesquisa.

Durante o segundo semestre, foi implementado o algoritmo para busca de soluções robustas considerando-se a robustez durante o processo de otimização. Os experimentos foram refeitos utilizando este novo algoritmo.

Os resultados foram analisados e comparados.

---

## SEÇÃO D – RELATÓRIO DETALHADO DAS ATIVIDADES DESENVOLVIDAS

---

A evolução das pesquisas no campo da robótica nos últimos anos teve uma grande influência nas atividades humanas em diversos setores da sociedade. Em relação a robótica móvel, o principal objetivo é desenvolver robôs inteligentes autônomos capazes de planejar efetivamente sua movimentação em ambientes estáticos e dinâmicos desconhecidos [SIEGWART & NOURBAKHS, 2011].

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) [NOLFI & FLOREANO, 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].

Métodos envolvendo Inteligência Artificial (IA) estão dentre os mais utilizados para esse propósito. Técnicas baseadas em Algoritmos Genéticos (AGs) têm sido muito utilizadas por causa de sua robustez, tanto em ambientes estáticos quanto dinâmicos [OLIVEIRA, 2015]. A aplicação de robótica em ambientes desconhecidos exige que robôs consigam se adaptar às mudanças que ocorrem no ambiente, fazendo com que os problemas de otimização se tornem dinâmicos [BRANKE, 2002].

Dentre as mudanças que podem ocorrer em robótica, devemos destacar para o problema estudado: ocorrência de falhas [TINÓS & CARVALHO, 2006] e mudanças nas características do ambiente [BILLARD et al., 2000]. 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 & YANG, 2014].

Uma dificuldade que pode ser enfrentada é a perda da diversidade da população de soluções, devido a convergência prematura da população para ótimos locais. 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 & BRANKE, 2005]. Dois dos mais utilizados são a hipermutação [COBB & GREFENSTETTE, 1993] e os imigrantes aleatórios [TINÓS & CARVALHO, 2007], [SILVA et al., 2008].

Muitas vezes, robôs evolutivos são controlados por redes neurais artificiais [NOLFI & FLOREANO, 2000]. Neste trabalho de iniciação Científica, a rede Echo State Network (ESN) é utilizada para o controle do robô móvel. A ESN [JAEGER & HAAS, 2004], [MANTAS, 2012], [ROMERO et al, 2006] é uma rede neural recorrente [ZUBEN, 2013] que 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 e conexões recorrentes possibilita que a rede tenha comportamentos dinâmicos complexos.

No problema estudado nesta pesquisa, as saídas desejadas da rede ESN não são conhecidas, i.e., não se conhece a ação que o robô deve tomar em cada posição de sua trajetória.

Desta forma, aqui são empregados os AGs para otimizar o vetor de pesos entre a camada intermediária (reservatório) e a camada de saída. A aptidão (fitness) de cada indivíduo é obtida testando-se as leis de controle (ESN) definidas pelo cromossomo do indivíduo.

O plano inicial do projeto definia a realização das seguintes atividades:

1. Levantamento Bibliográfico: estudo das referências relacionadas ao tema da pesquisa, principalmente otimização robusta em ambientes dinâmicos.
2. Implementação da rede neural ESN para controle do robô móvel.
3. Estudo e implementação dos algoritmos para busca de soluções robustas considerando-se que a robustez não será avaliada durante o processo de otimização.
4. Testes com os algoritmos desenvolvidos na Fase 3.
5. Estudo e implementação dos algoritmos para busca de soluções robustas considerando-se que a robustez será avaliada durante o processo de otimização.
6. Testes com os algoritmos desenvolvidos na Fase 5.
7. Comparação e análise dos resultados obtidos nas Fases 3 e 5.

Estas etapas foram executadas e são a seguir descritas.

## **1. LEVANTAMENTO BIBLIOGRÁFICO: ESTUDO DAS REFERÊNCIAS RELACIONADAS AO TEMA DA PESQUISA, PRINCIPALMENTE OTIMIZAÇÃO ROBUSTA EM AMBIENTES DINÂMICOS**

O estudo dos temas foi baseado em livros e artigos científicos.

O estudo sobre algoritmo genético real foi baseado principalmente no livro de Ricardo Linden “Algoritmos Genéticos” [LINDEN, 2012], que fornece uma descrição detalhada sobre o AGs, com exemplos em Java. Já os métodos de manutenção de diversidade em ambientes

incertos foram estudados baseando-se no texto [JIN & BRANKE, 2005], que trata de otimização de problemas com ampla gama de incertezas. Já as técnicas de hipermutação foram estudadas em [COBB & GREFENSTETTE, 1993], [SILVA et al., 2008] e os imigrantes aleatórios em [TINÓS & CARVALHO, 2007], [SILVA et al., 2008], COBB & GREFENSTETTE, 1993], textos nos quais tratam de estratégias para a manutenção da diversidade em AGs aplicados em ambientes dinâmicos.

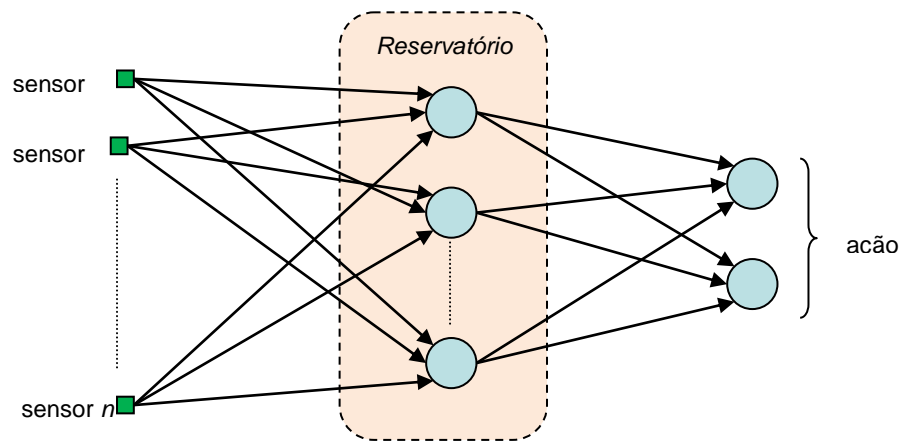
O estudo sobre a rede ESN, que é uma rede neural recorrente, foi baseado inicialmente no texto [ZUBEN, 2013] que aborda redes neurais recorrentes e em seguida os estudos foram baseados nos textos [JAEGER & HAAS, 2004], [MANTAS, 2012], [ROMERO et al, 2006] que tratam da Echo State Network.

Em relação aos algoritmos evolutivos em ambiente dinâmico, o estudo foi baseado na Tese de Livre Docência do orientador [TINÓS, 2012] e em [TINÓS & CARVALHO, 2006], [BILLARD et al., 2000], que tratam das mudanças que podem ocorrer em robótica e nas alterações que podem provocar nas soluções (fitness), afetando o processo de otimização [TINÓS & YANG, 2014].

Já o estudo relacionado à otimização robusta foi baseado em [BEYER & SENDHOFF, 2007], [FU et al., 2015], [OLIVEIRA, 2015], [BRANKE, 2002], que estudaram problema de busca por soluções robustas para ambientes dinâmicos. Mais detalhes sobre as técnicas estudadas são dadas nas seções a seguir.

## **2. IMPLEMENTAÇÃO DA REDE NEURAL ESN PARA CONTROLE DO ROBÔ MÓVEL**

A rede neural ESN foi implementada em C++ de acordo com a estrutura mostrada na Figura 1, e baseada no código desenvolvido em outro trabalho pelo orientador desta pesquisa em [WHITLEY et al., 2015]. As entradas da rede serão as leituras dos sensores, o reservatório terá um tamanho variável e cada saída da rede determina uma ação, sendo que o neurônio com maior ativação define a ação a ser tomada. A rede foi implementada de forma genérica, a fim de facilitar a modificação das configurações.



**Figura 1.** RNA recorrente (*Echo State Network*) utilizada para controlar o robô. Por simplicidade, as conexões recorrentes dos neurônios localizados no reservatório não são mostradas.

A recorrência ocorre apenas entre os neurônios do reservatório. Inicialmente foi adotada a configuração de 4 entradas (sensores), 50 neurônios no reservatório e 4 saídas (movimentos) com um valor de densidade de conexão de 0,15 (15%).

Cada ESN é definida por um indivíduo do AG. Assim, o cromossomo do indivíduo é um vetor de números reais que especifica o conjunto de pesos da camada de saída da ESN.

A ESN possui os seguintes métodos:

- ESN (int inputSize, int repSize, int outputSize, int n\_rec): construtor – responsável por realizar as alocações de memória de acordo com os parâmetros que definem a configuração da rede, e chamar o método responsável por gerar os pesos da camada de entrada e as recorrências.
- ~ESN(): destrutor – responsável por desalocar a memória utilizada na rede.
- double\* Execute (double \*in): método responsável por determinar a saída da rede de acordo com o conjunto de entradas.
- void setResWeight (double \*weight): método responsável por definir os pesos da camada de saída de acordo com o cromossomo do indivíduo que está sendo testado.
- double FuncAtivacao (double x): método responsável por calcular a ativação do neurônio de acordo com a função tangente hiperbólica.
- void weightInput(): método responsável por gerar aleatoriamente e normalizar os pesos da camada de entrada e as recorrências.

Vale ressaltar que esta parte da pesquisa foi realizada em conjunto com outras duas alunas de Iniciação Científica: Luciana Raineri e Raquel Cândido.



### 3. ESTUDO E IMPLEMENTAÇÃO DOS ALGORITMOS PARA BUSCA DE SOLUÇÕES ROBUSTAS CONSIDERANDO-SE QUE A ROBUSTEZ NÃO SERÁ AVALIADA DURANTE O PROCESSO DE OTIMIZAÇÃO

Os AGs são métodos adaptativos, inspirados nos processos genéticos de organismos biológicos e na teoria da evolução por seleção natural, que podem ser utilizados para resolver problemas de busca e otimização [MITCHELL, 1996], [LINDEN, 2012].

Neste projeto utilizamos o algoritmo genético real com o objetivo de otimizar o vetor que define os pesos da camada de saída da ESN aplicada para controlar o robô. O código foi baseado no algoritmo genético binário desenvolvido em uma pesquisa anterior a qual o aluno fez parte. O pseudo-código simplificado do AG padrão é apresentado na Figura 2.

#### Algoritmo: AG padrão

##### Início

Inicialize a população

Avalie a população inicial

##### Repita

Se critério de convergência for satisfeito

Interrompa

##### Fim se

Selecione indivíduos para a nova população

Aplique mutação e cruzamento nos indivíduos selecionados

Avalie os indivíduos da nova população

##### Fim repita

##### Fim

Figura 2. Pseudo-código do AG padrão.

A Figura 3 mostra como o robô foi utilizado.



Figura 3: Esquema do AG utilizado para otimizar o vetor de controle (cromossomo) do RE. Na etapa de avaliação, as leis de controle definidas pelo cromossomo do indivíduo são aplicadas no robô real.

#### I. Codificação

Cada indivíduo é representado por um vetor real (cromossomo), na qual cada posição representa um peso da 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 vetor varia de acordo com a quantidade de neurônios no reservatório e na camada de saída.

A população inicial é gerada aleatoriamente de acordo com a distribuição uniforme.

## II. 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.

## III. 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 deverá navegar pelo ambiente, andando o máximo possível em linha reta, até atingir algum dos critérios de parada:

- I. Robô colidir com o obstáculo;
- II. Não voltar para carregar a bateria dentro de uma quantidade de movimentos;
- III. Atingir o número máximo de movimentos.

O *fitness* da solução é a medida de desempenho do robô na tarefa executada (ver Figura 2).

A seguinte função de avaliação para tarefas de navegação foi utilizada:

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

na qual  $\mathbf{x}$  é cromossomo do indivíduo,  $t$  é a iteração (em cada iteração, o robô executa uma ação),  $t_{\max}(\mathbf{x})$  é o número de iterações executadas pelo robô, e:

$$\alpha(\mathbf{x}, t) = \begin{cases} 1, & \text{se o robô andou para a frente na interação } t \\ 0, & \text{caso contrário} \end{cases} \quad (2)$$

Na função de avaliação dada na Eq. (1), o número de iterações  $t_{\max}(x)$  é dado por:

$$t_{\max}(\mathbf{x}) = \min_t(300, t_{\text{choque}}, t_{\text{bateria}}) \quad (3)$$

sendo  $t_{choque}$  o instante em que o robô chocou-se com um obstáculo (caso tenha havido o choque; caso contrário,  $t_{choque} = 300$ ) e  $t_{bateria}$  o instante em que a bateria do robô ficou totalmente descarregada. A carga da bateria é simulada utilizando-se uma função linear cujos valores vão decrescer com o tempo [NOLFI & FLOREANO, 2000]. O tempo de carga foi considerado instantâneo, sendo que a bateria era recarregada sempre que o robô entrar 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.

O robô pode realizar 4 ações: seguir em frente (por uma distância pré-definida), girar 45°, -45° ou 90°. Cada neurônio de saída define uma destas ações, sendo que o neurônio com máxima ativação define a ação tomada pela ESN em cada instante de tempo.

Para atingir o fitness máximo, o robô navegar no ambiente em linha reta o máximo possível, sem se chocar com os obstáculos e retornar à área de recarga apenas quando a bateria estiver quase descarregada. A tarefa de simplesmente andar na arena sem se chocar com os obstáculos exige 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 [NOLFI & FLOREANO, 2000].

O pseudo-código apresentado na Figura 4 mostra como a função de fitness é calculada para um dado indivíduo do AG (ou seja, uma rede neural ESN com pesos dados pelo cromossomo do indivíduo).

**Algoritmo: Fitness (indivíduo)**

**Início**

**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 4: Pseudo-código para computar a função de fitness. 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 andando em linha reta o robô executou.

#### IV. Manutenção/Aumento da diversidade

O desempenho de AGs em problemas dinâmicos podem ser afetados pela convergência dos indivíduos para ótimos locais, e a probabilidade de se escapar destes ótimos é baixa. Este fator ocorre devido, principalmente, à crescente falta de diversidade na população. Para evitar essa dificuldade, são utilizadas neste trabalho duas técnicas de manutenção de diversidade, a Hipermutação e Imigrantes Aleatórios.

##### a. 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 & GREFENSTETTE, 1993].

No AG com hipermutação [COBB & GREFENSTETTE, 1993], inicialmente também é criada uma população aleatória e seus indivíduos são avaliados. O ciclo acontece por  $n$  gerações, sendo  $n$  predefinido inicialmente. No final do ciclo, é feita uma média dos valores de fitness dos melhores indivíduos do ciclo. Em seguida o ciclo se repete até a geração  $2n$  e a média dos valores de fitness dos melhores indivíduos das  $n$  últimas gerações é calculada novamente para este período. Neste momento são comparadas as médias obtidas do período  $(1, n)$  e  $(n, 2n)$ .

Caso a média no período  $(n, 2n)$  seja maior, nada é alterado.

Caso seja menor, o valor da taxa de mutação, inicialmente definida como  $p_m$ , é aumentada em uma porcentagem  $r$ . Feito isso, o ciclo prossegue até a geração  $3n$  com a taxa de mutação aumentada. Na geração  $3n$ , a taxa de mutação é ajustada para o valor inicialmente definido ( $p_m$ ). Assim, quando é detectado que a média dos melhores indivíduos diminuiu em relação ao último valor calculado, a hipermutação aplicada por um período de  $n$  gerações procura ocasionar maior diversidade genética, sendo posteriormente avaliado se os seus efeitos foram produtivos ou não quando seu antigo valor é retomado [SILVA et al., 2008].

##### b. Imigrantes Aleatórios

No AG com imigrantes aleatórios [COBB & GREFENSTETTE, 1993], a cada nova população gerada, uma porcentagem dos indivíduos desta nova população é selecionada aleatoriamente (com distribuição uniforme) e substituída por indivíduos gerados aleatoriamente.

Esta técnica procura manter alguns indivíduos com características diferentes da população geral que, provavelmente, está convergindo para uma solução local. Com isso, quando ocorrem mudanças neste problema, os imigrantes aleatórios podem possuir características que possibilitem readaptar a população a novas soluções, aumentando assim a diversidade genética [SILVA et al., 2008].

Neste trabalho, são gerados os novos indivíduos e estes são automaticamente inseridos na geração seguinte, sem realizar crossover na geração atual e sem serem avaliados. A avaliação destes indivíduos só será efetuada na geração seguinte.

## V. Simulador

Como o processo de otimização é demorado, foi desenvolvido em um projeto anterior no qual o aluno fez parte, um simulador para otimizar o processo de aprendizagem.

O algoritmo genético executado no simulador é o mesmo aplicado no robô, a única diferença que ao invés do fitness ser calculado utilizando o robô, ele é calculado utilizando esse simulador.

Antes de iniciar a avaliação de um indivíduo, é sorteado aleatoriamente uma posição e um ângulo inicial para o robô iniciar e, através de métodos matemáticos são calculados os resultados dos sensores. Os critérios de parada são os mesmos citados anteriormente.

A posição e o ângulo inicial aleatórios são importantes pois, se um indivíduo passou de geração em geração, ele foi avaliado várias vezes iniciando em configurações distintas, garantindo que este indivíduo é realmente uma boa solução para o problema.

Para utilizar o simulador neste projeto, algumas adaptações tiveram que ser feitas:

- O robô era controlado por um autômato finito e não por uma rede neural.
- O simulador recebia como parâmetro o indivíduo a ser avaliado e retornava seu fitness; já neste projeto, o simulador retorna a leitura dos sensores e executa a ação que é informada por parâmetro para função responsável por realizar a ação. Desta forma o simulador não fica responsável por executar as leis de controle, apenas simula a localização do robô no tablado e a leitura dos sensores.

Os algoritmos genéticos, bem como todos os algoritmos utilizados para controle e supervisão do robô móvel, 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.

#### 4. TESTES COM OS ALGORITMOS DESENVOLVIDOS NA FASE 3.

Aqui são apresentados resultados de experimentos utilizando o simulador já modificado, para trabalhar com a rede neural (ESN).

Nos experimentos realizados, todos os indivíduos de cada uma das populações foram apresentados a 10 ambientes diferentes durante o processo de otimização. Os ambientes podem ser vistos na figura 7 do Apêndice A.

As mudanças no ambiente ocorrem a cada 200 gerações. O processo de otimização ocorreu durante 2.200 gerações da seguinte forma: nas 200 primeiras gerações foi utilizado o ambiente inicial sem nenhum obstáculo, e nas próximas 1.800 gerações foram utilizados 9 ambientes com diferentes obstáculos. Nas últimas 200 gerações foi utilizado o ambiente inicial sem nenhum obstáculo.

Cada solução foi avaliada dez vezes, e seu fitness foi a média dos fitness das avaliações. Para cada um dos AGs, foram realizadas 25 simulações variando em cada uma delas o valor da semente aleatória. Cada execução demorou cerca de 4 horas na plataforma computacional utilizada. De modo a reduzir o tempo total, diversas execuções ocorreram em paralelo.

##### a. Parâmetros AG Padrão

O tamanho da população pode afetar o desempenho e a eficiência dos AGs. Populações muito pequenas têm grandes chances de perder a diversidade necessária para convergir a uma boa solução; entretanto, se a população for muito grande, o algoritmo perde a sua eficiência devido à demora em avaliar a função de aptidão de todos os indivíduos em cada interação. Nos experimentos realizados, cada população era composta de 100 indivíduos.

Em relação ao *crossover*, quanto maior for esta taxa, mais rapidamente novos indivíduos serão introduzidos na população. Mas se esta for muito alta, bons indivíduos poderão ser rapidamente substituídos, perdendo boas soluções. Por outro lado, um valor baixo pode tornar o algoritmo muito lento.

Tendo em vista o melhor desempenho, a taxa de *crossover* empregada foi 0,6 (60%).

A taxa de mutação foi definida sendo  $1/m$ , onde  $m$  é o tamanho do cromossomo. Esta taxa é empregada para que em média haja uma mutação por indivíduo por geração.

O método de seleção utilizado foi o do torneio, no qual  $k_t$  indivíduos da população são aleatoriamente escolhidos e aquele com maior fitness é empregado. Neste trabalho, o valor de  $k_t$  foi determinado como 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.

### **b. Hipermutação**

Na técnica de hipermutação, o tamanho do ciclo foi definido como sendo 10 gerações. Quando a hipermutação é ativa, a taxa de mutação é aumentada, ficando 4 vezes maior que a taxa de mutação inicial.

### **c. Imigrantes Aleatórios**

A taxa de substituição de indivíduos pelo AG com Imigrantes Aleatórios é um problema importante. Uma taxa pequena pode não atingir o objetivo desejado de manter a diversidade da população de soluções; por outro lado, uma taxa muito alta impede que características boas sejam propagadas ao longo das gerações.

Neste trabalho, foi utilizada uma taxa de substituição de 0,05. Com esta estratégia, em média 5% dos indivíduos são substituídos por novos indivíduos em cada geração.

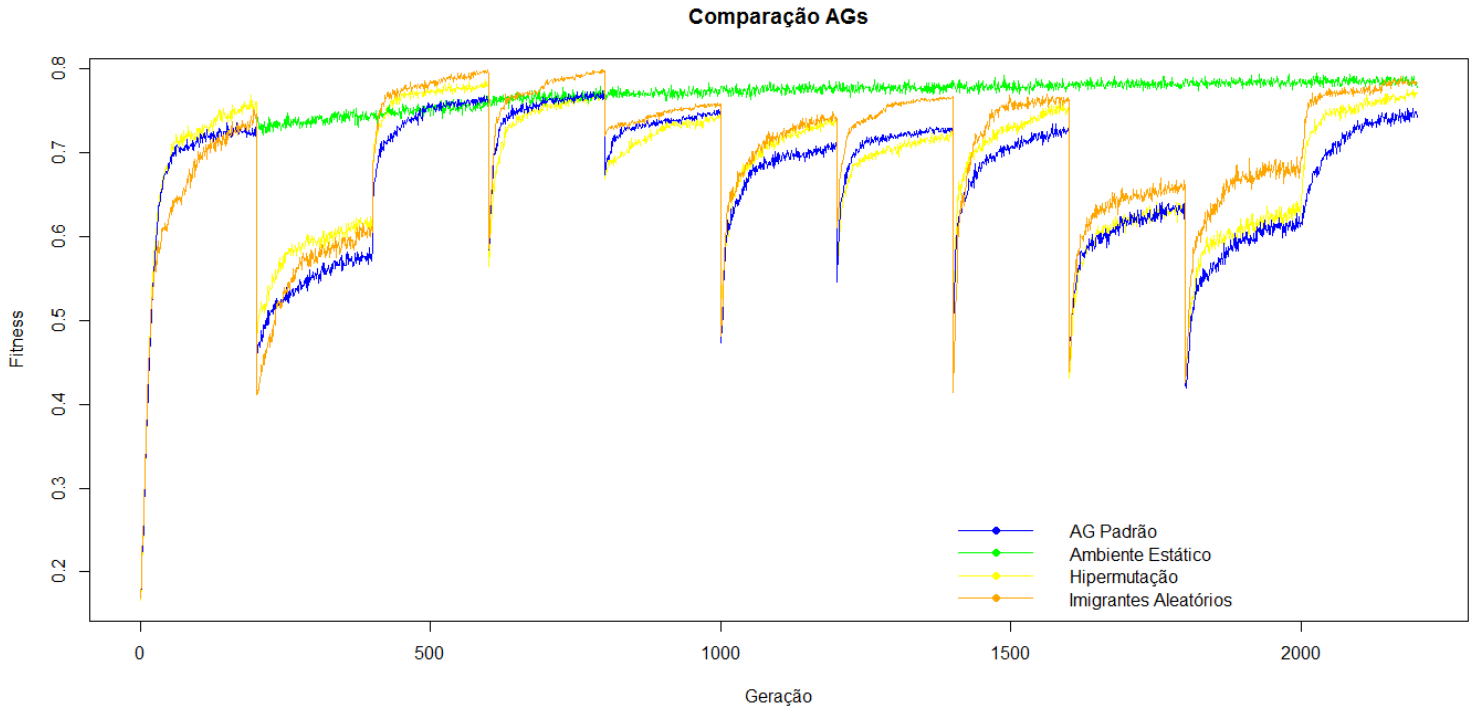
Os indivíduos a serem substituídos foram escolhidos aleatoriamente e com distribuição uniforme, com exceção dos dois indivíduos selecionados pelo método do elitismo.

### **d. Ambiente estático**

Nesta abordagem, o ambiente permanente inalterado e sem obstáculos durante todo o processo de otimização. O intuito de utilizar esta abordagem nos experimentos foi para verificar se a utilizar vários ambientes no processo de treinamento influencia na robustez da solução encontrada pelo AG.

### **e. Resultados**

Para comparar a evolução dos algoritmos e a sua adaptação após uma alteração no ambiente, foi calculada a média do fitness do melhor indivíduo de cada geração para cada um dos quatro algoritmos estudados. Os resultados estão representados na figura 5.



**Figura 5: 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.**

Para comparar a robustez dos algoritmos, foram selecionados o melhor indivíduo entre todas as execuções para cada um dos quatro algoritmos estudados. Estes 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 8 do Apêndice A. 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.



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

**Tabela 1:** 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 significativa 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.

## 5. ESTUDO E IMPLEMENTAÇÃO DOS ALGORITMOS PARA BUSCA DE SOLUÇÕES ROBUSTAS CONSIDERANDO-SE QUE A ROBUSTEZ SERÁ AVALIADA DURANTE O PROCESSO DE OTIMIZAÇÃO

A diferença deste algoritmo para o algoritmo anterior está na função de fitness, ao invés do ambiente ser alterado depois e um certo número de gerações, cada indivíduo é avaliado nos 10 ambientes ao mesmo tempo, ou seja, a nova função de avaliação para tarefas de navegação é:

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

na qual  $x$  é cromossomo do indivíduo,  $s$  é o índice do ambiente em que o robô está sendo avaliado,  $n$  é o número total de ambientes  $t$  é a iteração.

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. 4) 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.

## 6. TESTES COM OS ALGORITMOS DESENVOLVIDOS NA FASE 5.

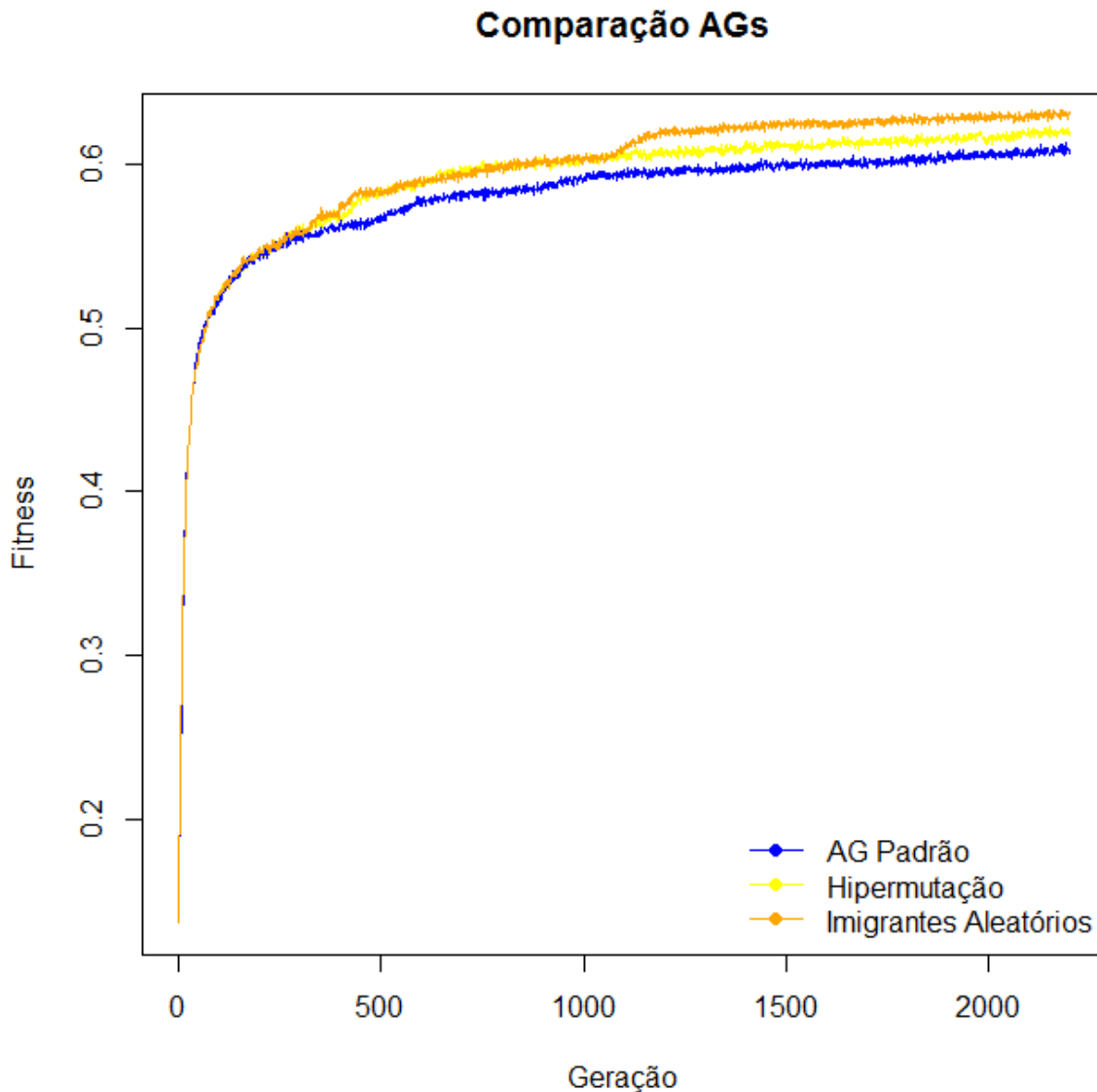
Nos experimentos realizados, todos os indivíduos de cada uma das populações foram apresentados a 10 ambientes diferentes durante o processo de otimização. A diferença para o algoritmo anterior é que ao invés do ambiente se alterar, cada indivíduo é avaliado nos 10 ambientes ao mesmo tempo, e seu fitness é a média do fitness obtido em cada ambiente.

Cada solução foi avaliada dez vezes, e seu fitness foi a média dos fitness das avaliações. Para cada um dos AGs, foram realizadas 25 simulações variando em cada uma delas o valor da

semente aleatória. Cada execução demorou cerca de 40 horas na plataforma computacional utilizada. De modo a reduzir o tempo total, diversas execuções ocorreram em paralelo.

#### a. Resultados

Para comparar a evolução dos algoritmos foi calculada a média do fitness do melhor indivíduo de cada geração para cada um dos quatro algoritmos estudados. Os resultados estão representados na figura 6.



**Figura 6:** Média do fitness do melhor indivíduo de cada execução de cada um dos três AGs.

Do mesmo modo da fase 3, 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.

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

**Tabela 2:** 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 significativa 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.

## 7. COMPARAÇÃO E ANÁLISE DOS RESULTADOS OBTIDOS NAS FASES 3 E 5.

No gráfico da figura 5 é possível observar que os AGs se comportam de maneira similar em relação a mudança no ambiente, havendo uma queda brusca ou um aumento repentino do fitness de acordo com a mudança.

Com exceção do Algoritmo Genético com ambiente estático, no qual não ocorreu mudanças no ambiente durante a fase de treinamento, todos os AGs produziram rapidamente soluções adaptadas ao novo ambiente. O AG com imigrantes aleatórios se destaca por ter obtido o maior fitness entre os AGs na maioria dos ambientes de simulação.

Em relação a solução obtida nos ambientes de robustez, os algoritmos conseguiram resultados bem similares, porém, em média, os resultados produzidos pelo AG com imigrantes aleatórios foram superiores (Tabela 1).

Para confirmar essa superioridade do AG com imigrantes aleatórios, foi realizado o teste de Wilcoxon Signed-Rank para verificar a significância estatística dos resultados obtidos. Os resultados do AG foram superiores em 11 dos 15 casos, sendo estatisticamente significativo em 6 (Tabela 1).

Já nos experimentos realizados na fase 5, 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 treinamento (exceto para o AG no Ambiente Estático).

Com essa mudança na função de fitness, não ocorreram alterações bruscas no valor de fitness, que ficou com uma taxa crescimento muito pequena ao decorrer das gerações (figura 6). É possível observar também que o AG no ambiente estático obteve melhores fitness durante o treinamento. Já em relação aos AGs que foram treinados em diferentes ambientes

com obstáculos, o AG com imigrantes aleatórios obteve um resultado superior durante o treinamento.

Em relação a solução obtida nos ambientes de robustez, nenhum dos algoritmos conseguiram se destacar em todos os ambientes, porém, em média, os resultados produzidos pelo AG com imigrantes aleatórios foram superiores (Tabela 2). Os melhores resultados da estratégia por Imigrantes Aleatórios são explicados pela maior diversidade da população proporcionada durante o processo de otimização. Tal diversidade é, portanto, importante em problemas de otimização robusta.

Para confirmar essa superioridade, novamente foi realizado o teste de Wilcoxon Signed-Rank para verificar a significância estatística dos resultados obtidos. Os resultados do AG foram superiores em 12 dos 15 casos, sendo significativo superior em 1 caso e inferior em 1 caso (Tabela 2).

Na Tabela 3, são comparados estatisticamente os resultados obtidos pelos experimentos realizado na fase 3 e da fase 5, utilizando-se o teste de Wilcoxon Signed-Rank com um nível de significância de 5%.

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 estratégia por otimização evolutiva dinâmica é melhor ou pior que a estratégia por otimização evolutiva robusta.**

Como é possível observar na Tabela 3, os resultados obtidos pelo primeiro experimento foram superiores em 5 dos 15 casos, sendo apenas 1 desses casos estatisticamente significativo. Já nos outros 10 casos, os resultados do segundo experimento foram superiores, sendo em 5 deles estatisticamente significativo.

Nota-se que, para o AG com Imigrantes Aleatórios, a segunda estratégia é melhor que a primeira nos ambientes 2, 3 e 5. Na Figura 10 (Apêndice A) são apresentadas algumas soluções produzidas pelo AG com Imigrantes Aleatórios no segundo experimento. Verifica-se que os ambientes 1, 3 e 5 são 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

## 8. TESTE NO ROBÔ REAL

O robô real é suscetível a falhas [TINÓS & CARVALHO, 2006]. Como exemplo, podemos citar: alteração na iluminação, nível de carga da bateria que influencia os sensores e atuadores; falhas de hardware, como por exemplo falha em um sensor. Além das falhas, o robô não é totalmente preciso, percorrendo distância aproximadas, realizando curvas com ângulos aproximados e a leitura dos sensores é realizado a uma distância aproximada.

Para tentar diminuir esses problemas, os testes foram realizados no Laboratório de Sistemas Computacionais Complexos do DCM com as luzes acesas e com as baterias do robô em sua carga máxima.

A solução utilizada para o teste foi uma das soluções produzidas pelo algoritmo genético com imigrantes aleatórios da fase 5. O AG com Imigrantes Aleatórios foi escolhido pois este apresentou os melhores resultados nas simulações considerando-se a robustez das soluções. O tempo de execução de cada uma das soluções no robô real demorou cerca de 20 minutos.

Foram realizadas 4 simulações na arena sem obstáculo alterando a posição inicial do robô. Os valores de fitness obtidos nos experimentos foram 0,7, 0,74, 0,73 e 0,69. Quando comparado com os valores obtidos na simulação (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>*

---

## **SEÇÃO E – DESCRIÇÃO E AVALIAÇÃO DO APOIO INSTITUCIONAL**

---

A Universidade de São Paulo de Ribeirão Preto tem dado o apoio necessário para que este projeto seja desenvolvido conforme desejado. Seja pela infraestrutura com bibliotecas, salas de estudos, laboratórios e profissionais das áreas de conhecimento envolvidas no projeto. Este projeto está sendo realizado no Laboratório de Sistemas Computacionais Complexos do Departamento de Computação e Matemática (DCM). O Laboratório de Sistema Computacionais Complexos é de responsabilidade do orientador deste projeto (Prof. Dr. Renato Tinós) em conjunto com outros dois professores.

---

## SEÇÃO F – CONCLUSÃO

---

Todos AGs estudados conseguiram encontrar leis de controle robustas às mudanças no ambiente, obtendo soluções que não colidiram em nenhum dos ambientes utilizados na fase de teste de robustez. Em relação as 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, as soluções evoluídas devem ser testadas em um robô real. Também, outras estratégias robustas descritas em [FU et al., 2015] devem ser investigadas.

No apêndice A deste relatório, se encontram todos os ambientes utilizados nos experimentos, e algumas ilustrações dos trajetos percorridos pelas soluções encontradas pelos algoritmos.

---

## SEÇÃO G – REFERÊNCIAS BIBLIOGRÁFICAS

---

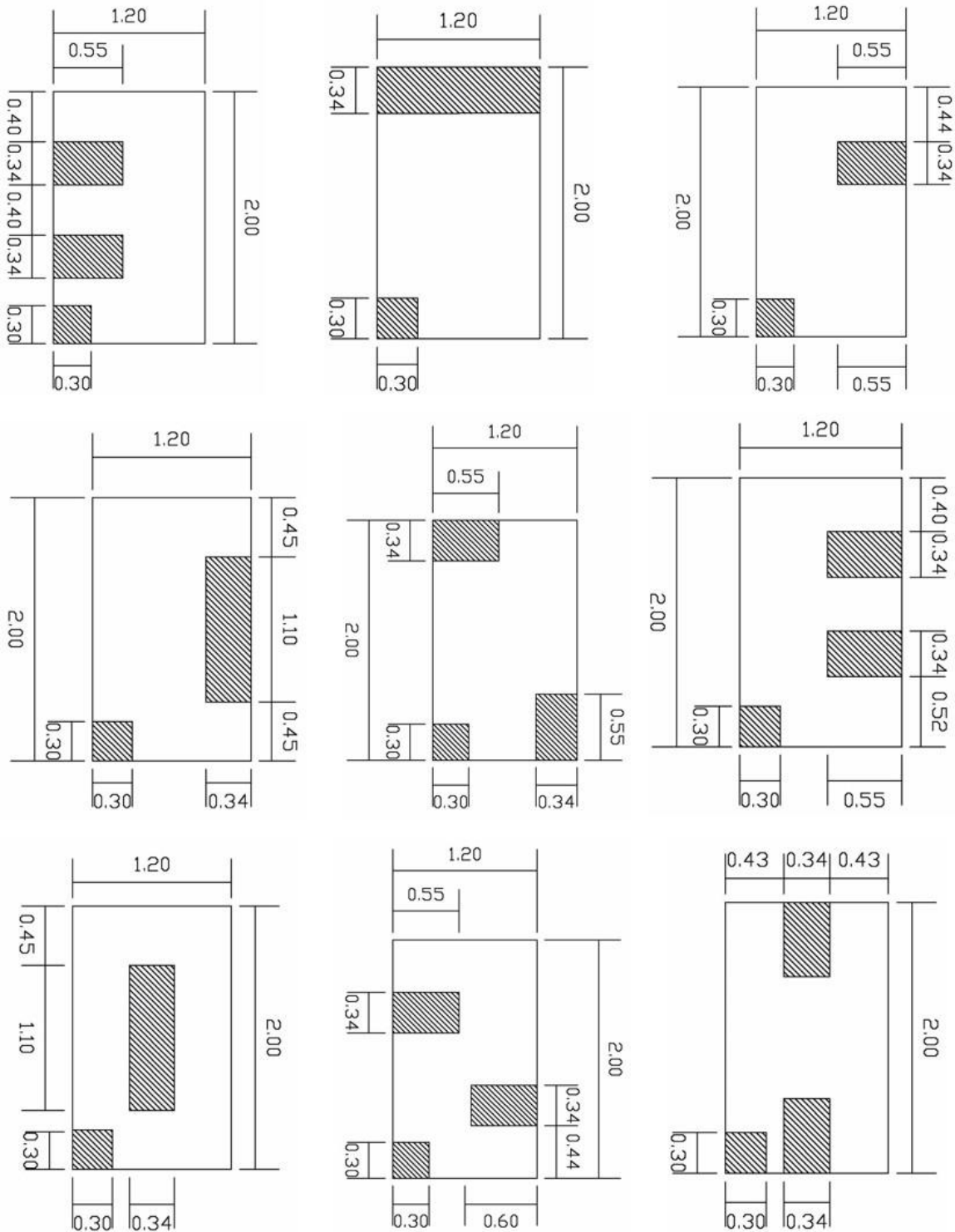
- BEYER, H. G. & SENDHOFF, B. (2007). “Robust optimization—a comprehensive survey”, *Computer methods in applied mechanics and engineering*, 196(33): 3190-3218.
- BILLARD, A.; IJSPEERT, A. J. & MARTINOLI, A. (2000). “A multi-robot system for adaptive exploration of a fast changing environment: probabilistic modeling and experimental study”, *Connection Science*, 11(3/4): 357-377.
- BRANKE, J. (2002). “Evolutionary Optimization in Dynamic Environments”, Kluwer Academic Publishers.
- COBB, H. G. & GREFENSTETTE, J. J. (1993). “Genetic algorithms for tracking changing environments”, In S. Forrest (ed.), 5th Int. Conf. on Genetic Algorithms, Morgan Kaufmann, p. 523-530.
- EIBEN, A. E., & SMITH, J. (2015). “From evolutionary computation to the evolution of things”, *Nature*, 521(7553): 476-482.
- FU, H.; SENDHOFF, B.; TANG, K. & YAO, X. (2015). “Robust optimization over time: Problem difficulties and benchmark problems”, *IEEE Transactions on Evolutionary Computation*, 19(5): 731-745.
- JAEGER, H. & HAAS, H. (2004). “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication”, *Science*, 304(5667): 78–80.
- JIN, Y & BRANKE, J. (2005). “Evolutionary optimization in uncertain environments - a survey”, *IEEE Transactions on Evolutionary Computation*, 9(3): 303-317.
- LINDEN, R (2012). “Algoritmos Genéticos”, Ciência Moderna, 3.ed.
- MANTAS, L. (2012). “A Practical Guide to Applying Echo State Networks”, Jacobs University Bremen, Campus Ring 1. [Online] Disponível na internet via URL: <http://minds.jacobs-university.de/sites/default/files/uploads/papers/PracticalESN.pdf>. Arquivo consultado em 17 de Janeiro de 2018.
- MEYER, J. A. (1998). “Evolutionary approaches to neural control in mobile robots”, In the Proc. of the 1998 IEEE Conf. on Systems, Man, and Cybernetic, San Diego, USA.
- MITCHELL, M. (1996). “An introduction to genetic algorithms”, MIT Press.
- NOLFI, S. & FLOREANO, D. (2000). “Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines”, MIT Press/Bradford Books.
- OLIVEIRA, Á. V. F. M. de, (2015). “Estratégia de Navegação com Planejamento Dinâmico e Algoritmo Genético Aplicado a Robôs Terrestres”, Centro de Tecnologia da Universidade Federal do Rio Grande do Norte.



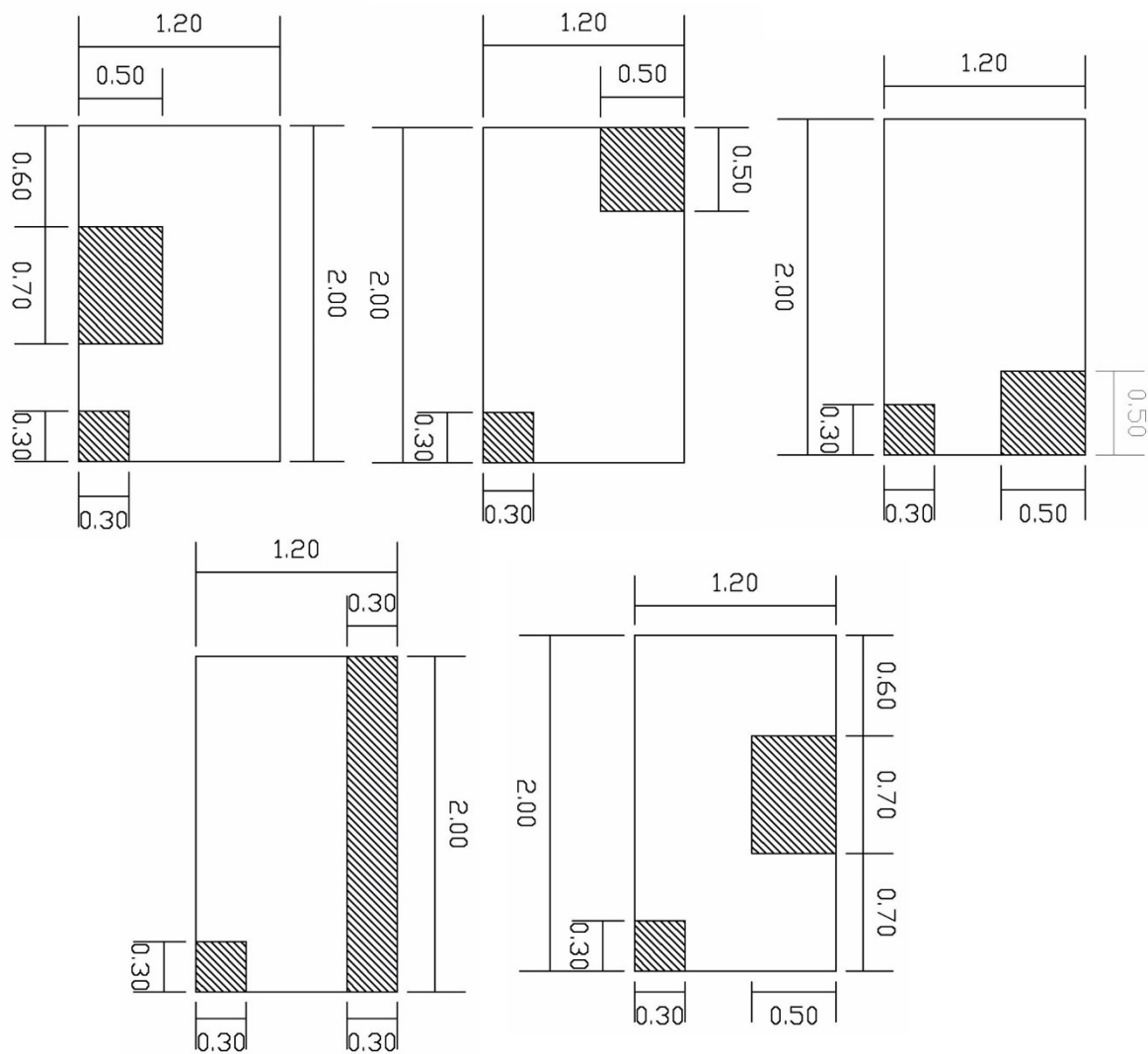
- ROMERO, R. A. F.; BREVE, F.; JUNIOR, J. R. B, (2006) “Echo State Network”, Universidade de São Paulo – ICMC. [Online] Disponível na internet via URL: <https://www.fabriciobreve.com/trabalhos/esn.pdf>. Arquivo consultado em 17 de Janeiro de 2018.
- SIEGWART, R. & NOURBAKHSI, I. R., (2011). “Introduction to Autonomous Mobile Robots”.
- SILVA, L. E. V.; HUMMEL, A. D.; TERRA, M. H. & TINÓS, R (2008). “Técnicas de Manutenção de Diversidade em Algoritmos Genéticos Aplicados a Robôs Móveis em Ambientes Dinâmicos”, Revista Eletrônica de Iniciação Científica (REIC) da Sociedade Brasileira de Computação, 8(4).
- TINÓS, R. & 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. (2012). “*Computação Evolutiva em Ambientes Dinâmicos*”, Tese de Livre Docência, FFFCLRP, Universidade de São Paulo.
- TINÓS, R., & 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 behavior?”, *Behavioral and Brain Sciences*, 24(6): 1033-1050.
- WHITLEY, D.; TINÓS, R. & CHICANO, F., (2015). “Optimal Neuron Selection: NK Echo State Networks for Reinforcement Learning”, In arXiv preprint arXiv: 1505.01887.
- ZUBEN, F. J. V., (2013) “Redes Neurais Recorrentes”, DCA/FEEC/Unicamp. [Online] Disponível na internet via URL: [ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353\\_1s13/topico5\\_1s2013.pdf](ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_1s13/topico5_1s2013.pdf). Arquivo consultado em 17 de janeiro de 2018.

## APÊNDICE A – AMBIENTES DE SIMULAÇÃO

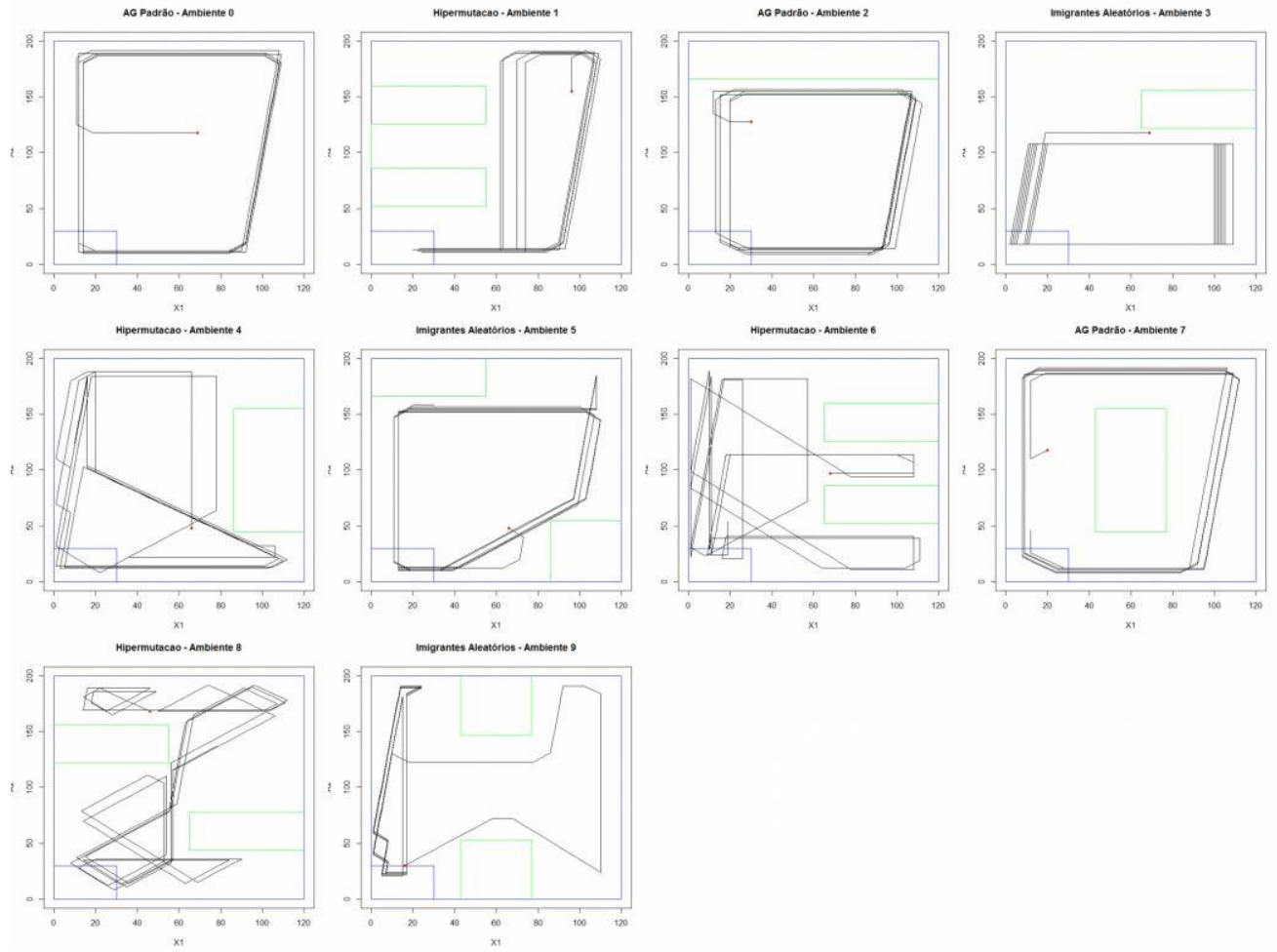
As imagens a seguir ilustram respectivamente os 10 ambientes utilizados na fase de treinamento, e os 5 ambientes utilizados para avaliar a robustez.



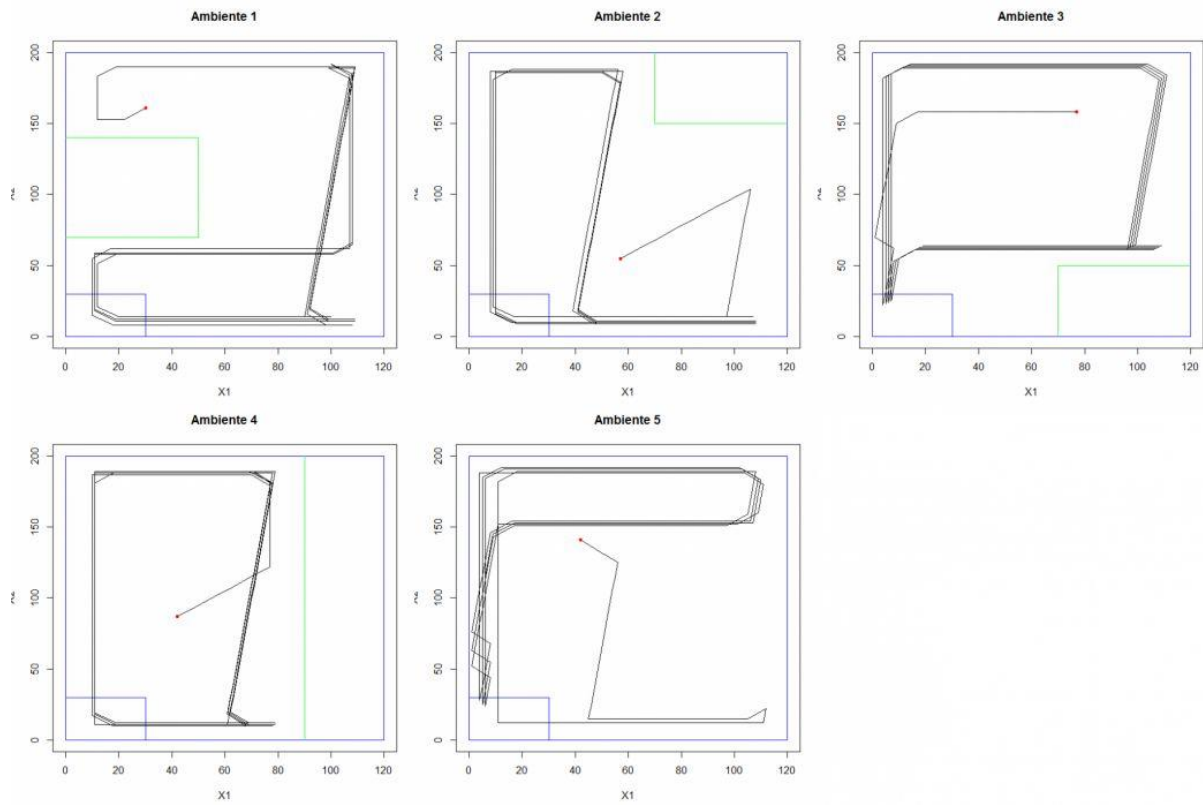
**Figura 7: Ambientes utilizados na fase de treinamento. Os ambientes foram numerados da esquerda para direita, de cima para baixo.**



**Figura 8: Ambientes utilizados para avaliar a robustez. Os ambientes foram numerados da esquerda para direita, de cima para baixo.**



**Figura 9: Soluções produzidas pelos AGs estudados na fase 5.**



**Figura 10: Soluções produzidas pelo AG com Imigrantes Aleatórios no experimento da fase 5.**