Technical Report
Mobile Robots Reactive Navigation with Neural Networks

Alysson Ribeiro da Silva
Computer Science Graduate Program
Federal University of Minas Gerais
Belo Horizonte, Brazil

Abstract—Desastres naturais, como tsunamis e enchentes são fonte recorrente de danos sociais, econômicos, e matam milhões de pessoas todos os anos. Para tentar mitigar os problemas inerentes aos desastres naturais, robôs vem sendo utilizados em missões de exploração e resgate desde os ataques em 11 de Setembro ao World Trade Center com as mais variadas técnicas de aprendizado e tomada de decisões. Neste contexto, a proposta desta investigação é propor, implementar, e implantar um sistema para controle de robôs com redes neurais profundas em um ambiente simulado no Robot Operating System para avaliar seu desempenho ao navegar em ambientes desconhecidos. Duas abordagens de evolução e aprendizado são abordadas, sendo elas a Neuroevolução e o Backpropagation. Os resultados obtidos apontam potencial das soluções avaliadas assim como rebutes do aprendizado utilizando Backpropagation.

Index Terms—Neural Networks, Navigation, Mobile Robots, Neuroevolution

INTRODUÇÃO

Desastres naturais, como tsunamis e enchentes, são fonte recorrente de danos sociais e econômicos ao redor do mundo, por consequência da alta taxa de mortalidade [7], [12], custos com reconstrução, e reparos na infraestrutura das regiões afetadas. Como exemplo, os tsunamis e enchentes [36] estão dentre os tipos mais devastadores de desastres, conseguem cobrir grandes quantidades de terras em curtos periodos de tempo e causar danos substanciais [9]. Em 28 de Setembro de 2018, na baia de Palu, Indonesia, uma tsunami provocou a morte de mais de duas mil pessoas. Já em 2004, uma tsunami em Banda Aceh causou aproximadamente cento e sessenta mil vítimas e proporcionou um prejuízo pós-desastre de aproximadamente seis bilhões de dólares [20]. Diferentes abordagens vem sendo tomadas para tentar mitigar as consequências dos desastres naturais [1], o que inclui o uso de robôs inteligentes em missões de busca e resgate [18], [19].

Vários problemas são inerentes a este contexto e estão relacionados as incertezas provenientes dos ambientes em questão que podem reduzir a qualidade dos algoritmos e técnicas de navegação. Consequentemente, a proposta desta investigação é propor, implementar, e implantar um sistema para controle de robôs com redes neurais profundas em um ambiente simulado no Robot Operating System. Duas abordagens de evolução e aprendizado são abordadas, sendo elas a Neuroevolução e o Backpropagation, com o objetivo de permitir a evolução de soluções que fomentem a exploração e navegação em um ambiente desconhecido. Ao fim deste documento é apresentado um apêndice que contém algumas informações sobre a estrutura do projeto em anexo.

1 REVISÃO BIBLIOGRÁFICA

Robôs estão sendo utilizados em missões de resgate desde os ataques em 11 de Setembro ao World Trade Center [6]. Desse dia em diante, a comunidade acadêmica começou a dar mais atenção a aspectos que possam ajudar a mitigar danos, como comportamento inteligente e evacuação eficiente, sistemas que atuam como guias, e também abordagens relacionadas à navegação autônoma e semi-autônoma [19]. Há um grande esforço no desenvolvimento de sistemas de navegação autônomos inteligentes que baseiam-se em Deep Reinforcement Learning, [10], [27], [3], [22], [26], principalmente por consequência da habilidade de tais soluções em adaptar a mudanças no ambiente e de serem capazes de encontrar vítimas de forma eficiente.

Há também abordagens que baseiam-se na simulação de um Processo de Decisão de Markov Parcialmente Observável (PDMPO) e incorporação de aprendizado temporal em redes adaptativas inspiradas na Fuzzy Adaptive Resonanse Theory (ART). Tais métodos foram inicialmente propostos por [4], [16], [15], estendidas no modelo ART Map [5], e posteriormente no Adaptive Resonance Associative Maps [29], [30], foram utilizados para aprendizado não supervisionado para controle de agentes em ambientes incertos

e dinâmicos [17], [34], [32]. Algoritmos Genéticos também podem ser utilizados de forma a melhorar os parâmetros das mesmas ou de forma híbrida como abordado por [31].

Além das técnicas que baseiam-se em aprendizado temporal por redes neurais, existem também as abordagens Neuroevolutivas, onde o principal objetivo é criar ou adaptar a topologia ou os pesos de uma rede neural utilizando algoritmos evolucionários [28], [11], [23]. Tais métodos são utilizados para gerar controladores para agentes em diversos contextos como jogos [14], [33], [24], [25], [23] e robôs autônomos [8], [2]. Por exemplo, Caceres et al. [2] propõe a utilização de uma técnica denominada Neuroevolution of Augmented Topologies (NEAT) para controlar robôs móveis. Diferentemente, Galassi et al. [13] utiliza a neuroevolução de forma online em um enxame de robôs, permitindo evoluir e atuar ao mesmo tempo, mitigando o problema de ter um objetivo fixo.

Dentre os métodos citados, cada um tem as suas vantagens e desvantagens. Por exemplo, o aprendizado por reforço é capaz de desenvolver comportamentos emergentes úteis sem a interferência de um especialista, seja ele incorporado com auxílio de redes profundas ou redes adaptativas. Em contrapartida, a Neuroevolução, como citado por Risi [23], possui diversas vantagens e tende a produzir melhores resultados para resolver diversos problemas. Por outro lado, técnicas de aprendizado como o Backpropagation são essenciais na maioria dos tipos de aprendizado que envolvem redes neurais, especialmente em conjunto a utilização de redes profundas. Mesmo sem o auxílio de uma técnica de aprendizado temporal por reforço, podem ser capazes de gerar comportamentos viáveis devido a sua capacidade de aproximação de hiperplanos.

2 Fundamentação Teórica

Nesta seção é inicialmente apresentada uma contextualização sobre navegação em ambientes incertos e dinâmicos. Posteriormente é apresentada a modelagem do problema através de Processos de Decisão de Markov Parcialmente Observáveis (PDMPO) e qual a função das redes neurais neste contexto.

2.1 Navegação em Ambientes Incertos e Dinâmicos

O problema de navegação em um ambiente desconhecido e dinâmico pode ser modelado em duas etapas, sendo uma delas referente a aquisição da informação e a outra referente a otimização e tomada de decisões. A primeira etapa é responsável por permitir que o robô adquira informações relevantes de um ambiente a ser explorado. Consequentemente, a mesma é caracterizada pela extração ou aprendizado de features e construção de representações, como as Evidence Grids [21] utilizadas no contexto do Simultaneous Exploration and Mapping (SLAM) [35], que sirvam como suporte ao processo de tomada de decisões. Já a segunda etapa consiste em permitir que o robô tome decisões de forma a utilizar seus atuadores para navegar e interagir com o ambiente.

A falta de informação sobre o ambiente e até mesmo sobre as consequências de se executar determinada ação, perante a incerteza de um ambiente pós-desastre, sugere a utilização de representações e técnicas para aprendizado não supervisionado através da experimentação, logo, a otimização e tomada de decisões de um robô pode ser assistida por um Processo de Decisão de Markov Parcialmente Observavel (PDMPO) e Aprendizado por Diferença Temporal.

2.2 Processo de Decisão de Markov Parcialmente Observável

Tipicamente a etapa de otimização e tomada de decisões de um robô pode ser assistida por um PDMPO. O mesmo é um modelo numérico probabilístico que permite representar estados, ações, e associar as mesmas a recompensas de forma a permitir gerar um plano universal de ações que engloba todos os estados de um espaço de soluções possíveis. Por exemplo, para que um robô navegue até uma vítima ou explore o ambiente de forma eficiente, gastando menos tempo, é preciso que o mesmo execute uma sequência de ações, considerando incertezas que podem interferir no processo com uma determinada probabilidade, levando o mesmo a estados desejados ou indesejados.

O PDMPO é descrito como uma tupla, lista ordenada de elementos, da forma $(S, A, T, R, \Omega, O, \gamma)$, onde S é o conjunto de estados, A é o conjunto de ações, T é o conjunto de probabilidades que descreve a chance de transição entre estados pertencentes a S, $R : S \times A- > \Re$ é a função que calcula as recompensas ao executar ações, Ω é o conjunto de observações, O é a probabilidade condicional, e γ sendo um fator de desconto multiplicativo.

Da perspectiva de um PDMPO, o principal objetivo é maximizar a recompensa cumulativa total decidindo quais ações são mais importantes em determinado estado. A recompensa total do agente, descrita pela Equação 1, é então igual a soma de todas as recompensas intermediárias obtidas durante suas decisões, onde t representa os passos no tempo, $gamma$ é utilizado como desconto para a recompensa recebida, e r_t representa a recompensa recebida ao executar uma ação no tempo t. Uma vez que $gamma$ é elevado a t, ações mais próximas ao objetivo final tem

mais valor do que ações tomadas no início da interação do agente com o ambiente.

$$E = \sum_{t} \gamma^t r_t \quad (1)$$

As transições em um POMDP podem ocorrer com probabilidade $T(s'|s, a)$, onde s é o estado atual, a sendo a ação executada pelo agente, e s' o novo estado obtido ao executar a ação a. Nesse contexto, o objetivo é então permitir que as redes neurais sejam capazes de aprender a abstrair e generalizar cada estado $s_i \in S$ e fazer predições de um conjunto de ações $a_i \in A$ que maximizem o desempenho do mesmo.

3 MÉTODO

3.1 Neuroevolução

Nesta investigação, a Neuroevolução é realizada com auxílio de redes neurais profundas e um algoritmo genético. Cada indivíduo $i \in I$ do algoritmo genético está associado a uma rede neural $p \in P$ e um robô $r \in R$, onde seus pesos são convertidos para matrizes. Por sua vez, cada matriz está associada a uma camada da rede.

O comportamento de cada rede neural profunda é construído ao longo das iterações do algoritmo genético através da troca de genes entre os indivíduos. Como descrito pelo Algoritmo 1, a Neuroevolução irá solicitar que todos os robôs entrem em modo de avaliação com a função `check_evaluation()`. Uma vez em modo de avaliação, cada robô irá interagir com o ambiente selecionando uma ação a proveniente da predição de sua rede neural dada uma observação do contexto do estado atual s.

```
void Neuroevolution()
  eval = check_evaluation()

  if eval:
    for robot in ROBOTS:
      fitness[robot.id] = robot.fit
      publish_stop_robot(robot)
      publish_reset_robot(robot)

    GeneticEngine.step()

    all_robots_evaluated = false

    for robot in ROBOTS:
      publish_evaluate(robot)
```

Algoritmo 1 - Peseudo código para a Neuroevolução.

Após cada robô finalizar sua avaliação, o Algoritmo 1 armazena o valor da função objetivo, através da função `robot.fit`, solicitará os mesmos para reinicializarem seus estados, e solicitará ao algoritmo genético que efetue um `step` com a função `GeneticEngine.step()`. Um `step` do algoritmo genético efetua as operações de seleção, cruzamento, e mutação, até que uma nova população seja gerada. Esse processo é repedido até que um número máximo de gerações é alcançado. Ao finalizar todas as avaliações, o algoritmo avisa a todos os robôs que os mesmos devem entrar em modo de avaliação novamente para dar início a uma nova geração. Esse último processo é realizado pela função `publish_evaluate(robot)`.

3.2 Backpropagation

Diferentemente da Neuroevolução, o algoritmo de treinamento do backpropagation é realizado apenas com um robô e de forma online. Como descrito pelo Algoritmo 2, o robô interage constantemente com o ambiente e capta informações sobre seus arredores através da função `get_obs(sensors)`. Após capturar as informações necessários sobre seus arredores, o robô faz duas predições. A primeira predição efetuada pelo comando `e = spc_get(obs)` é proveniente de um `forward pass` na rede neural e ela representa qual ação o robô irá executar. Já a segunda predição, é proveniente de um comportamento especialista que é capaz de desviar de obstáculos e é obtida através do comando `prediction = pred_train(obs, e)`.

```
void TrainRobot()
    obs = get_obs(sensors)
    e = spc_get(obs)
    prediction = pred_train(obs, e)
    action = select(prediction)
    robot_com.publish(action)
```

Algoritmo 2 - Peseudo código o backpropagation online.

Em conjunto a segunda predição, o robô também efetua o backpropagation em sua rede neural através da função `pred_train(obs, e)` para reduzir o erro obtido. Uma vez que o algoritmo é feito de forma online, o erro e o aprendizado ocorre a cada iteração do robô com o ambiente. Além disso, são utilizadas diretivas de segurança para evitar acidentes durante o processo.

3.3 Individuo

Como ilustrado pela Figura 1, cada indivíduo é modelado como um vetor contínuo $v_i \in I$ de números reais dividido em regiões, onde cada região está associada a uma camada da rede neural. Em especial, na Figura 1 é ilustrada uma rede neural e parte de um indivíduo, onde os genes em vermelho estão associados a segunda camada e os genes em azul estão associados a primeira camada da rede. Os mesmos foram modelados

com codificação real para permitir o mapeamento direto entre gene e peso da rede neural. A separação em regiões associadas a camadas da rede neural permite preservar o contexto ao efetuar troca de genes.

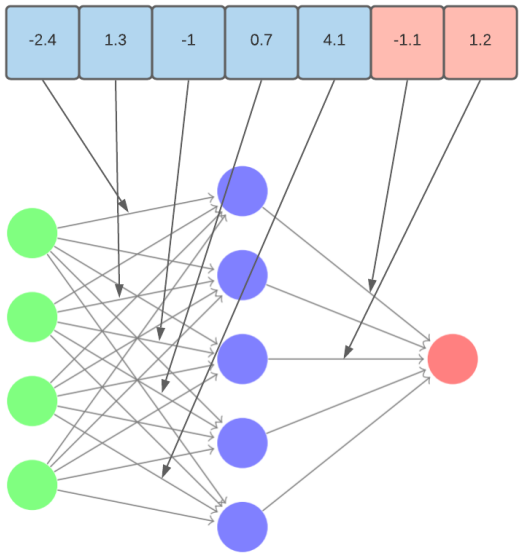

Figura 1: Representação de um indivíduo, onde os genes são mapeados diretamente em pesos da rede neural.

Para converter um indivíduo em pesos da rede, o Algoritmo 3 é utilizado. O mesmo recebe duas variáveis de entrada, sendo elas o `V` e o `topo`. O vetor de entrada `V` contém um indivíduo v_i. Já a variável `topo`, contém a especificação da topologia da rede neural. A descrição da topologia é utilizada para auxiliar no processo de decomposição do indivíduo em matrizes que representam os pesos da rede.

```
List Decode(V, topo)
  weights = []
  start = 0
  end   = 0

  for l in topo.layers:
    start = end
    end   = end + l.size
    nn_w = V[start:end]
    mat = w.reshape(l.l, l.c)
    weights.append(mat)

  return weights
```

Algoritmo 3 - Decodificação de um indivíduo para matrizes da rede neural.

Cada robô é avaliado de acordo com seu tempo de vida e distância percorrida. Esses aspectos são calculados pela função objetivo descrita através da Equação 2,

$$f(i) = l + 2^{\sum \Delta s_i} \tag{2}$$

onde l representa o tempo de vida total de um robô desde o início de sua simulação e Δs_i representa a distância percorrida pelo robô associado ao indivíduo i entre as iterações de tempo t e $t+1$. Essa função objetivo visa permitir que os robôs evitem executar ações perigosas que podem colocar sua existência em risco e também a maximizar a distância total percorrida.

A seleção é conduzida por competição entre k indivíduos, onde o melhor é selecionado para cruzamento. Já o cruzamento entre dois indivíduos $A = \{a_1, ..., a_n\}$ e $B = \{b_1, ..., b_n\}$, é realizado de forma uniforme para cada gene. Dado dois genes $a_i \in A$ e $b_i \in B$, o gene c_i resultante do cruzamento pertence ao intervalo $[a_i - \alpha(b_i - a_i), b_i + \alpha(b_i - a_i)]$ caso $a_i \leq b_i$ para $\alpha \in [0, +\inf]$. Esse formato de cruzamento é comumente chamado de Blend Crossover e o seu objetivo é explorar ou extrapolar o intervalo numérico de uma codificação real. Nesta investigação, também optou-se por mutação de um ponto e o uso de elitismo, onde o melhor indivíduo é propagado para iterações futuras.

3.4 Topologia da Rede

A rede neural profunda utilizada para controlar os robôs, ambos para a Neuroevolução e para o Backpropagation, é descrita pela Figura 2. Nesta investigação, foi optado pela utilização de 3 camadas escondidas, $n = 6$ entradas, e $c = 3$ saídas. A camada de entrada recebe dados provenientes dos sensores do robô e cada camada escondida possui m neurônios, onde $m = n+2$. Todas as ativações são obtidas através da Rectified Linear Unit (ReLu) com exceção da última camada que é ativada através da função sigmoide. Primariamente, essa escolha foi feita de forma a evitar o problema do `Vanishing Gradient` devido ao Backpropagation ser avaliado com a mesma topologia de rede da Neuroevolução.

Em sua última camada, os neurônios são ativados pela função Softmax, descrita pela Equação 3,

$$f(S) = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{3}$$

onde, S é o vetor de ativação da última camada. A normalização através da função Softmax é efetuada para transformar a saída em uma distribuição de probabilidades utilizada no processo de tomada de decisão dos robôs a cada iteração.

O erro, por sua vez, foi definido como o Categorical Cross-Entropy que é descrito pela Equação 4,

$$loss = -\sum_i^C t_i log(f(S)_i) \tag{4}$$

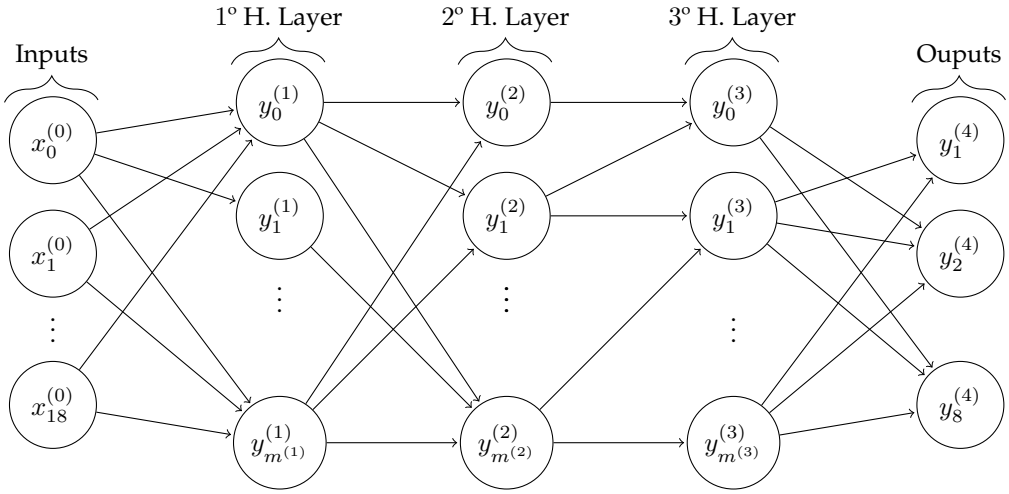

Figura 2: Topologia da rede profunda perceptron com três camadas escondidas que controla os robôs.

onde $f(S)_i$ representa a predição da rede que é saída da camada Softmax e $T = t_1, ..., t_i, ..., t_n$ representa uma predição correta.

3.5 Robô Pioneer 3-At

O robô utilizado para avaliação da proposta é o Pioneer 3-At ilustrado na Figura 3. O mesmo foi equipado com um sensor laser Hokuyo com 60 feixes, taxa de atualização de $10hz$, e distância máxima de 30 metros.

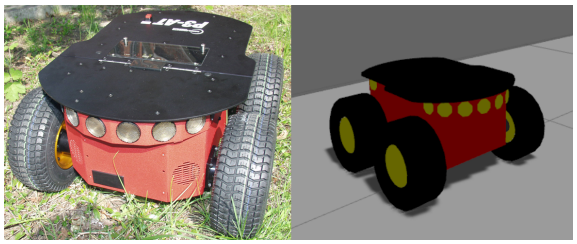

Figura 3: A esquerda é ilustrado o robô Pioneer 3-At real e a direita o mesmo robô simulado utilizando o simulador Gazebo integrado ao Robot Operating System (ROS).

As entradas da rede neural profunda são provenientes dos feixes laser do robô. Uma vez que o espaço de busca é muito extenso, foi feita uma redução de dimensionalidade do vetor de entrada. O mesmo foi reduzido para n, onde o mesmo representa o tamanho de entrada para a topologia proposta. Como ilustrado pela Figura 4, o robô pode decidir executar uma dentre três ações a cada iteração que são provenientes da saída da rede. A primeira ação, permite ao robô a girar 5 graus para a esquerda, já a segunda ação visa permitir o mesmo a girar 5 graus para a direita, e a última permite que o robô se locomova a uma velocidade arbitrária para frente.

A integração com a arquitetura do ROS é feita através de vários robôs, onde cada um

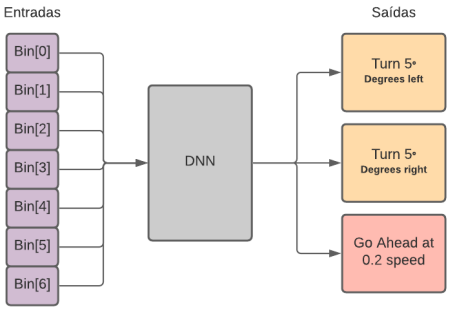

Figura 4: Entradas e saídas da rede profunda para a Neuroevolução e Backpropagation.

contém um comportamento capaz de gerenciar uma rede neural profunda. Como exemplificado pela Figura 5, é proposto a utilização de um controlador centralizado denominado `NeuroEvolution Controller` que é responsável pela gestão do Algoritmo 1 da Neuroevolução e da execução dos passos do algoritmo genético. Diferentemente, para a utilização do Backpropagation, apenas um Neural Controller é utilizado em um robô.

A Arquitetura proposta permite alto grau de paralelismo para simulações, porém existem uma troca relativa ao custo de espaço de armazenamento devido a vasta quantidade de parâmetros e `overhead` na comunicação entre os robôs pela rede.

4 Configuração Experimental

Para a realização dos testes foi utilizado o ROS Noetic no Ubuntu 20.04 e um Ryzen 7 com 32GB de memória. O treinamento do Backpropagation foi feito de forma online em uma RTX2070. Já o armazenamento dos pesos das redes neurais

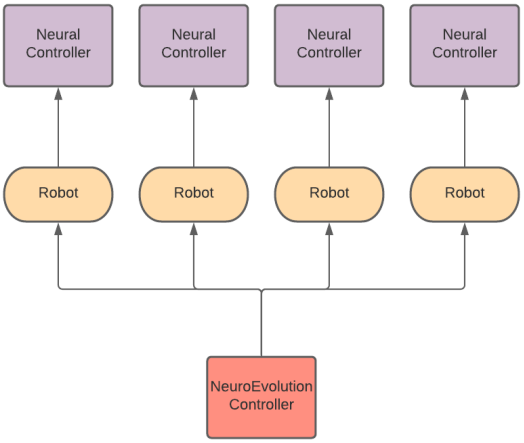

Figura 5: Arquitetura de integração do ROS com o sistema de Neuroevolução.

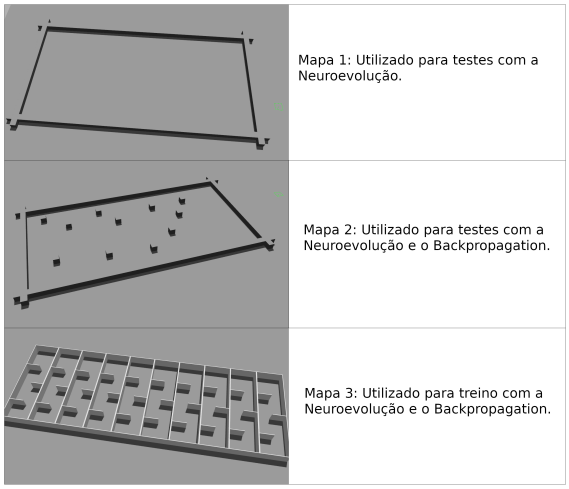

Figura 6: Mapas criados e utilizados para a avaliação das propostas.

profundas da Neuroevolução foram feitos na memória RAM e o forward-pass foi efetuado diretamente na CPU para evitar gargalos e problemas relacionados. Para a simulação foi utilizado o simulador Gazebo 11.3 em conjunto a um modelo do robô Pioneer 3-At modificado.

A metodologia de avaliação consiste em três etapas: Validação; Verificação do comportamento da Neuroevolução; e comparação preliminar da Neuroevolução com o Backpropagation. Durante a validação, é feita a verificação da convergência do algoritmo genético utilizando a codificação proposta. Já a Neuroevolução é avaliada em três objetivos simples. Sendo eles, aprender a andar para frente, encontrar um objetivo no canto de um mapa de teste, e tentar desviar de obstáculos. Por fim, a comparação entre Neuroevolução e Backpropagation é feita treinando-se os dois algoritmos em um mapa especial e avaliando a performance dos dois em um mapa desconhecido.

Para os comportamentos da Neuroevolução, foi implementada uma rede Feedforward utilizando Tensorflow sem bibliotecas adicionais, por facilidade de integração entre o sistema e a conversão entre um indivíduo do algoritmo genético para tensores das redes profundas. Diferentemente, o Backpropagation foi implementado e incorporado dentro do comportamento do ROS com auxílio da biblioteca Keras. A utilização da mesma foi feita com o intuito de permitir fácil integração da fita de gradientes do Tensorflow para o cálculo do Categorical Cross-Entropy.

5 Mapas Utilizados

Três mapas, ilustrados pela Figura 6, foram utilizados para validar e avaliar o comportamento das técnicas.

O mapa 1 é utilizado como sandbox para avaliar o desempenho da Neuroevolução ao tentar aprender um comportamento simples. Já o segundo mapa é utilizado para avaliação do desempenho dos robôs em um local desconhecido. Dessa forma, o mesmo nunca é utilizado durante o treino. Por fim, o mapa 3 é utilizado para o treinamento de ambas as técnicas abordadas e também para averiguar o comportamento da Neuroevolução com o que diz respeito a ser capaz de desviar de obstáculos. É importante notar que no terceiro mapa, os robôs foram separados em câmaras de treinamento para evitar distúrbios provenientes com a possível interação entre os diversos robôs.

6 Validação do Algoritmo Genético

O algoritmo genético foi validado para verificar a capacidade da modelagem de convergir em diversas superfícies de busca. As Equações 5 a 10 foram utilizadas como funções objetivo de teste para validar o algoritmo.

$$Rastrigin = An + \sum_{i}^{n}[x_i^2 - Acos(2_i)] \quad (5)$$

$$Beale = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \quad (6)$$

$$Booth = (x + 2y - 7)^2 + (2x + y - 5)^2 \quad (7)$$

$$Matyas = 0.26(x^2 + y^2) - 0.48xy \quad (8)$$

$$Levi = sin^2 3\pi x + (x-1)^2(1+sin^2 3\pi y) + (y-1)^2(1+sin^2 2\pi y) \quad (9)$$

$$Schaffer2 = 0.5\frac{sin^2(x^2-y^2)-0.5}{[1+0.001(x^2+y^2)]^2} \quad (10)$$

Os resultados da validação do algoritmo genético são apresentados na Tabela 2. São apresentadas estatísticas do maior, menor, média, e desvios inferiores e superiores para cada função objetivo de teste. Para cada teste foram feitas 100 execuções com 500 indivíduos e 1000 gerações. Todos os valores foram normalizados para o intervalo $[0, 1]$ pelo fato de o problema ser de minimização, onde 0 significa o pior valor de fitness e 1 significa que o ótimo global foi encontrado. Os demais parâmetros foram previamente selecionados como descrito na Tabela 1 param garantir diversidade, mutação, e cruzamentos que fomentem a troca de genes e geração de novas soluções.

Cruzamento	Mutação	Alpha	Troca de Gene
1.0	0.4	0.2	0.3

Table 1: Parâmetros previamente escolhidos para avaliação do genético com codificação real em diversas superfícies de busca.

Como ilustrado pela Tabela 2, o algoritmo convergiu para o ótimo global em todas os testes com desvios superiores e inferiores iguais ou próximos a 0. Isso demonstra a capacidade da modelagem dos indivíduos em conjunto ao cruzamento utilizado de efetuar uma busca no espaço de diversas superfícies e convergir ao ótimo global.

Por consequência dos resultados obtidos durante a validação do algoritmo genético demonstrarem seu poder de busca, o restante dos testes foram efetuados com os mesmos parâmetros escolhidos previamente e descritos na Tabela 1. O algoritmo de Neuroevolução foi executado somente 20 gerações por consequência de seu alto custo computacional. Além disso, a taxa de aprendizado para o Backpropagation foi fixada em 0.01.

7 Comportamentos da Neuroevolução

A Figura 7 Ilustra os resultados obtidos para o primeiro testes efetuado, que diz respeito ao aprendizado do comportamento de se deslocar para frente. Os robôs foram capazes de aprender a se deslocar de forma satisfatória após apenas 20 gerações.

Com relação a tentar encontrar objetivos no canto superior esquerdo do mapa, a Neuroevolução se saiu muito bem. A esquerda da

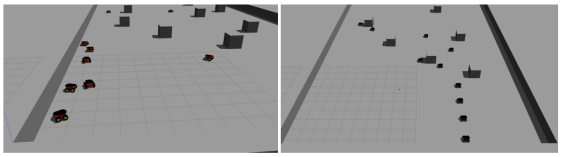

Figura 7: A esquerda os robôs durante a primeira geração da Neuroevolução tentando aprender a andar para frente. A esquerda os mesmos robôs ao fim do processo de otimização.

Figura 9 os robôs demonstraram comportamentos aleatórios. Porém após as 20 gerações, os mesmos convergiram e conseguiram encontrar um caminho em direção ao objetivo proposto.

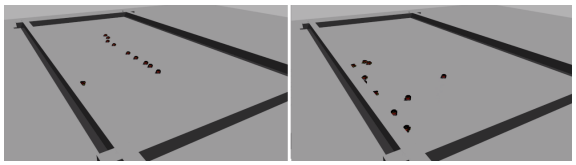

Figura 8: A esquerda os robôs durante a primeira geração da Neuroevolução tentando aprender a encontrar um objetivo no canto superior esquerdo do mapa. A esquerda os mesmos robôs ao fim do processo de otimização.

Na Figura **??**, pode-se observar que os robôs estão todos perdidos durante as primeiras gerações, porém após algum tempo os mesmos convergem para um caminho na diagonal direita de cada sala pois ela está livre. O comportamento foi obtido através da convergência prematura do algoritmo possivelmente devido ao tamanho desfavorável da tripulação.

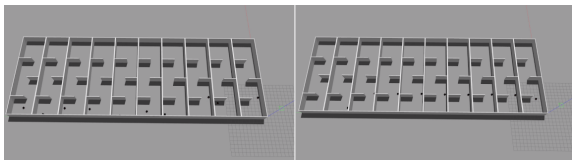

Figura 9: A esquerda os robôs durante a primeira geração da Neuroevolução tentando aprender a desviar de obstáculos. A esquerda os mesmos robôs ao fim do processo de otimização.

8 Neuroevolução e Backpropagation em ambientes desconhecidos

Durante a competição, como Ilustrado pela Figura 10, a Neuroevolução se mostrou ineficiente para gerar uma solução pois convergiu prematuramente. Por outro lado, o Backpropagation convergiu para um ótimo global em tempo hábil. Ao andar pelo mapa, como Ilustrado na Figura 10, o mesmo demonstrou robustez ao navegar pelo ambiente desconhecido e também

	Máximo	Mínimo	Média	Desvio	Desvio Sup.	Desvio Inf.
Rastrigin	1.00	0.125	0.65	0.28	0.039	0.098
Beale	1.00	1.00	1.00	0.00	0.00	0.00
Booth	1.00	1.00	1.00	0.00	0.00	0.00
Matyas	1.00	0.99	1.00	0.00	0.00	0.00
Schaffer2	1.00	0.99	0.99	0.00	0.00	0.000166
Levi	1.00	1.00	1.00	0.00	0.00	0

Table 2: Fitness encontrado para cada função de teste considerada nesta investigação. Os valores estão normalizados entre 0 e 1, onde 0 significa a pior fitness e 1 significa o ótimo global.

poder de generalização o suficiente para lidar com obstáculos que são adicionados dinamicamente.

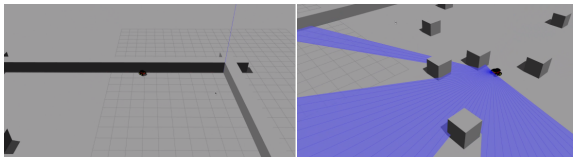

Figura 10: A esquerda os robôs durante a primeira geração da Neuroevolução tentando aprender a desviar de obstáculos. A esquerda os mesmos robôs ao fim do processo de otimização.

Com relação a convergência do algoritmo de Neuroevolução, pode-se observar através da Figura 11 a curva de convergência prematura do mesmo para o teste realizado. Essa curva prematura pode ter ocorrido por consequência da falta de diversidade ou parâmetros escolhidos para o algoritmo.

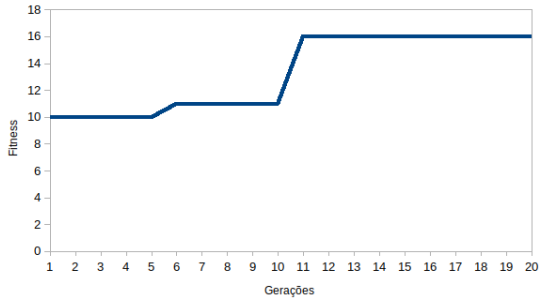

Figura 11: Gráfico da evolução da fitness do Neuroevolution ao longo das gerações.

Já com relação ao algoritmo de Backpropagation, o mesmo demonstrou um erro instável durante oi treinament como mostra a Figura 12. Isso ocorreu possivelmente por consequência de o aprendizado ser feito a cada iteração e não haver um custo fixo para todos os estados. Porém, isso não afetou a convergência do algoritmo e a obtenção de uma taxa de acerto de 100% se tornando robusto até a modificações no ambiente.

9 Conclusão

Nesta investigação é proposta a criação e integração de um ambiente de simulação que utiliza

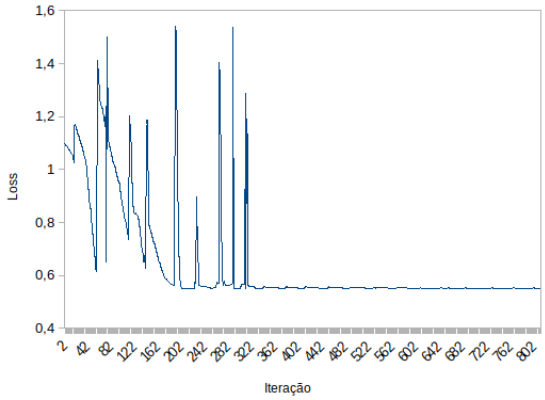

Figura 12: Gráfico da evolução da fitness do Neuroevolution ao longo das gerações.

o Robot Operating System e o simulador Gazebo. Além disso, foram preparados e integrados enxames de robôs do tipo Pioneer 3-At para navegação utilizando redes neurais profundas. Foram exploradas duas técnicas de evolução e aprendizado, a Neuroevolução através de um algoritmo genético com codificação real e o Backpropagation. Vários testes foram conduzidos para validar o algoritmo genético e alguns experimentos foram conduzidos para avaliar o desempenho de ambas as técnicas de evolução e aprendizado.

Os resultados apontam a eficácia do algoritmo genético com codificação real para busca e otimização em diversos espaços de busca em diversas funções de teste. Além disso, os experimentos conduzidos mostram indícios do potencial da técnica de Neuroevolução. Por fim, o Backpropagation é capaz de aproximar os estados experienciados pelo robô de forma robusta chegando a uma taxa de acerto de 100%. Tal comportamento permitiu ao robô controlado pela rede neural profunda a se deslocar pelo ambiente sem ajuda de especialistas sem provocar colisões.

Neste mini-projeto foram aprendidos conceitos e o algoritmo da Neuroevolução; integração de técnicas de aprendizado com deep learning em robôs móveis; codificação real para algoritmos genéticos; implementação do backpropagation; sobre arquiteturas de redes neurais profundas e suas diversas funções de ativação; finalidade das funções de erro; modelagem para solucionar os problemas inerentes a

Neuroevolução e robótica.

Para direções futuras, pretende-se expandir o sistema e as técnicas de aprendizado para a utilização de Reinforcement Learning e outras. Além disso, também pretende-se melhorar o ambiente de simulação com foco na criação ou incorporação de um benchmark padronizado para avaliação do desempenho dos robôs.

REFERÊNCIAS

[1] Progress from catastrophe. *Nature Geoscience*, 10(8):537–537, Aug 2017.

[2] C. Caceres, J. M. Rosario, and D. Amaya. Approach of kinematic control for a nonholonomic wheeled robot using artificial neural networks and genetic algorithms. In *2017 International Conference and Workshop on Bioinspired Intelligence (IWOBI)*, pages 1–6, 2017.

[3] G. A. Cardona, C. Bravo, W. Quesada, D. Ruiz, M. Obeng, X. Wu, and J. M. Calderon. Autonomous navigation for exploration of unknown environments and collision avoidance in mobile robots using reinforcement learning. In *2019 SoutheastCon*, pages 1–7, 2019.

[4] Gail A. Carpenter and Stephen Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, March 1988.

[5] Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Netw.*, 4(6):759–771, November 1991.

[6] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385, 2003.

[7] Jorge G. F. Crempien, Alejandro Urrutia, Roberto Benavente, and Rodrigo Cienfuegos. Effects of earthquake spatial slip correlation on variability of tsunami potential energy and intensities. *Scientific Reports*, 10(1):8399, May 2020.

[8] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, May 2015.

[9] Phil R. Cummins, Ignatius R. Pranantyo, Jonathan M. Pownall, Jonathan D. Griffin, Irwan Meilano, and Siyuan Zhao. Earthquakes and tsunamis caused by low-angle normal faulting in the banda sea, indonesia. *Nature Geoscience*, 13(4):312–318, Apr 2020.

[10] A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, and P. Valigi. Towards generalization in target-driven visual navigation by using deep reinforcement learning. *IEEE Transactions on Robotics*, pages 1–16, 2020.

[11] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, Mar 2008.

[12] Elizabeth Frankenberg, Cecep Sumantri, and Duncan Thomas. Effects of a natural disaster on mortality risks over the longer term. *Nature Sustainability*, May 2020.

[13] M. Galassi, N. Capodieci, G. Cabri, and L. Leonardi. Evolutionary strategies for novelty-based online neuroevolution in swarm robotics. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002026–002032, 2016.

[14] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *J. Mach. Learn. Res.*, 9:937–965, June 2008.

[15] Stephen Grossberg. *The Adaptive Brain*, volume II. Elsevier, 1988.

[16] Stephen Grossberg. *The Adaptive Brain*, volume I and II. Elsevier, 1989.

[17] Ah hwee Tan, Senior Member, Ning Lu, and Dan Xiao. Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. *IEEE Transactions on Neural Networks*, page 230244, 2008.

[18] S. Kuswadi, S. I. Adji, R. Sigit, M. N. Tamara, and M. Nuh. Disaster swarm robot development: On going project. In *2017 International Conference on Electrical Engineering and Informatics (ICELTICs)*, pages 45–50, 2017.

[19] F. Matsuno and S. Tadokoro. Rescue robots and systems in japan. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 12–20, 2004.

[20] Jamie W. McCaughey, Patrick Daly, Ibnu Mundir, Saiful Mahdi, and Anthony Patt. Socio-economic consequences of post-disaster reconstruction in hazard-exposed areas. *Nature Sustainability*, 1(1):38–43, Jan 2018.

[21] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. volume 2, pages 116 – 121, 04 1985.

[22] X. Qiu, K. Wan, and F. Li. Autonomous robot navigation in dynamic environment using deep reinforcement learning. In *2019 IEEE 2nd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, pages 338–342, 2019.

[23] S. Risi and J. Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):25–41, 2017.

[24] Jacob Schrum, Igor V. Karpov, and Risto Miikkulainen. Ut2: Human-like behavior via neuroevolution of combat behavior and replay of human traces. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG 2011)*, pages 329–336, Seoul, South Korea, September 2011. IEEE.

[25] Jacob Schrum, Igor V. Karpov, and Risto Miikkulainen. *Human-Like Combat Behaviour via Multiobjective Neuroevolution*, pages 119–150. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[26] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3221–3226, 2020.

[27] H. Shi, L. Shi, M. Xu, and K. Hwang. End-to-end navigation strategy with deep reinforcement learning for mobile robots. *IEEE Transactions on Industrial Informatics*, 16(4):2393–2402, 2020.

[28] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[29] A. H. Tan. Adaptive resonance associative map: a hierarchical art system for fast stable associative learning. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pages 860–865 vol.1, Jun 1992.

[30] Ah-Hwee Tan. Adaptive resonance associative map. *Neural Networks*, 8(3):437 – 446, 1995.

[31] S. C. Tan, J. Watada, Z. Ibrahim, and M. Khalid. Evolutionary fuzzy artmap neural networks for classification of semiconductor defects. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):933–950, May 2015.

[32] T. H. Teng and A. H. Tan. Fast reinforcement learning under uncertainties with self-organizing neural networks. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pages 51–58, Dec 2015.

[33] Phillip Verbancsics and Kenneth O. Stanley. Evolving static representations for task transfer. *J. Mach. Learn. Res.*, 11:1737–1769, August 2010.

[34] D. Wang and A. H. Tan. Creating autonomous adaptive agents in a real-time first-person shooter computer game. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(2):123–138, June 2015.

[35] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997.

[36] Dapeng Yu, Jie Yin, Robert L. Wilby, Stuart N. Lane, Jeroen C. J. H. Aerts, Ning Lin, Min Liu, Hongyong Yuan, Jianguo Chen, Christel Prudhomme, Mingfu Guan, Avinoam Baruch, Charlie W. D. Johnson, Xi Tang, Lizhong Yu, and Shiyuan Xu. Disruption of emergency response to vulnerable populations during floods. *Nature Sustainability*, May 2020.