UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Davi Becker da Silva

Impacto da duração de episódios de avaliação em robótica evolutiva usando técnicas de nicho

Florianópolis
2023

Davi Becker da Silva

Impacto da duração de episódios de avaliação em robótica evolutiva usando técnicas de nicho

Trabalho de Conclusão de Curso do Curso de Graduação em Ciência da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Ciência da Computação.
Orientador: Prof. Dr. Jônata Tyska Carvalho

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

da Silva, Davi Becker
Impacto da duração de episódios de avaliação em robótica evolutiva usando técnicas de nicho / Davi Becker da Silva ; orientador, Jônata Tyska Carvalho, 2023.
76 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2023.

Inclui referências.

1. Ciências da Computação. 2. Robótica Adaptativa. Evolução de Nicho. Tempo de Avaliação.. I. Carvalho, Jônata Tyska. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Davi Becker da Silva

Impacto da duração de episódios de avaliação em robótica evolutiva usando técnicas de nicho

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Ciência da Computação" e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação.

Florianópolis, 22 de maio de 2023.

Banca Examinadora:

Prof. Dr. Jônata Tyska Carvalho
Orientador

Prof. Dr. Mauro Roisenberg
Avaliador
Instituição UFSC

Prof. Dr. Pedro Belin Castellucci
Avaliador
Instituição UFSC

Este trabalho é dedicado aos meus colegas e à minha família.

AGRADECIMENTOS

Gostaria de expressar meus agradecimentos a todos que estiveram presente nessa jornada. Todos exerceram funções que foram de extrema importância para o início, meio e fim deste trabalho e graduação.

Aos meus pais, Aline, Jorge Davi e Marcel, pelos incentivos diários, mesmo estando longe. Por toda a dedicação para que seu filho pudesse estudar em uma das melhores faculdades do Brasil. Por todo o amor e carinho que fizeram com que eu me tornasse a pessoa que sou hoje. E por fim, por confiarem em mim e nas minhas escolhas, agradeço do fundo do meu coração, amo muito todos vocês.

A minha namorada Bianca, por apoiar escolhas e atitudes que fizeram com que eu chegasse a esse momento. Pelo amor e compreensão em todos os momentos, tristes ou felizes, durante toda a graduação e o TCC. Por estar junto à mim, algumas vezes não fisicamente, para me dar força e sabedoria. Te amo.

Ao meu orientador, Jonata Tyska pela ajuda, motivação, ensinamento e confiança no desenvolvimento deste trabalho e acreditar que tudo isso seria possível.

Aos amigos da graduação, com vocês a graduação ficou mais leve e alegre, rimos das tragédias e comemoramos as conquistas. Saibam que este trabalho não seria possível sem a presença de vocês.

RESUMO

A robótica móvel é uma das área da robótica que se concentra no desenvolvimento e aplicação de robôs capazes de se locomoverem e operar em diferentes ambientes. Junto com a robótica adaptativa, projetam robôs autônomos que possuem a capacidade de se adaptarem, sendo a robótica evolutiva um dos métodos. A robótica evolutiva é uma metodologia que utiliza computação evolutiva para desenvolver robôs capazes de obterem comportamentos com pouca ou nenhuma interação humana. Se utiliza de algoritmos evolutivos para permitir que robôs desenvolvam ações através de um processo evolutivo, assim, permitindo a criação de indivíduos adaptáveis e flexíveis, capazes de lidar com ambientes complexos e em constante mudança. A geração de robôs autônomos por meio da robótica evolutiva enfrenta uma série de desafios relacionados à complexidade da solução, tempo computacional, adaptabilidade, além de muitas variáveis que influenciam no processo evolutivo, como escolha do algoritmo evolutivo, o ambiente, entre outros. Na robótica evolutiva, o ambiente é o espaço físico, ou simulado, em que o robô opera para desenvolver seu comportamento, contendo diferentes configurações, como obstáculos, paredes, objetos. Com isso, a variação ambiental ao longo do tempo introduz mudanças que ajudam na capacidade dos robôs de evoluírem seus comportamentos. A técnica de nichos utiliza o conceito de diferentes configurações de ambientes em forma de nichos, a fim de usufruir as vantagens da variação ambiental. Outro fator que influencia na efetividade é a duração da interação do indivíduo com o ambiente, conhecido como episódio de avaliação. O objetivo do episódio de avaliação é desenvolver o comportamento do robô e avaliar seu resultado em um valor chamado *fitness*. Este trabalho tem como objetivo a criação de algoritmos que utilizam a variação ambiental através de nichos e a variação na duração do episódio de avaliação. Assim, foram criados algoritmos que utilizam os dois conceitos apresentados, o primeiro com condições ambientais iniciais fixas em cada nicho e o segundo em que as condições ambientais iniciais de cada nicho variam ao decorrer da evolução, sendo comparados a algoritmos evolutivos. Ao colocar os resultados em gráficos *boxplot*, pode ser observado que o segundo algoritmo apresentou uma melhor mediana em 2 dos 3 ambientes de simulação selecionados, Hopper, Ant e HalfCheetah . No ambiente de simulação Hopper, apresentou um aumento de 5% e no ambiente HalfCheetah, apresentou uma melhora de 12%.

Palavras-chave: Robótica Adaptativa. Evolução de Nicho. Tempo de Avaliação.

ABSTRACT

Mobile robotics is one of the areas of robotics that focuses on the development and application of robots capable of moving and operating in different environments. Together with adaptive robotics, design autonomous robots that have the ability to adapt, with evolutionary robotics being one of the methods. Evolutionary robotics is a methodology that uses evolutionary computing to develop robots capable of obtaining behaviors with little or no human interaction. Using algorithms evolutionary processes to allow robots to develop actions through an evolutionary process, thus, allowing the creation of adaptable and flexible individuals, capable of dealing with complex and constantly changing environments. The generation of autonomous robots through of evolutionary robotics faces a series of challenges related to the complexity of solution, computational time, adaptability, in addition to many variables that influence in the evolutionary process, such as choosing the evolutionary algorithm, the environment, among others. In evolutionary robotics, the environment is the physical, or simulated, space in which the robot operates to develop your behavior, containing different configurations such as obstacles, walls, objects. As a result, environmental variation over time introduces changes that help in the ability of robots to evolve their behaviors. The niche technique uses the concept of different configurations of environments in the form of niches, in order to enjoy the advantages of environmental variation. Another factor that influences effectiveness is the duration of the individual's interaction with the environment, known as the assessment episode. The objective of the evaluation episode is to develop the robot's behavior and evaluate its result in a value called fitness. This work aims to create algorithms that utilize environmental variation across niches and variation in duration of the evaluation episode. Thus, algorithms were created that use both concepts presented, the first with fixed initial environmental conditions in each niche and the second in which the initial environmental conditions of each niche vary throughout evolution, being compared to evolutionary algorithms. By plotting the results in boxplot graphs, It can be seen that the second algorithm presented a better median in 2 of the 3 selected simulation environments, Hopper, Ant and HalfCheetah. In the environment of Hopper simulation, presented an increase of 5% and in the HalfCheetah environment, presented a 12% improvement.

Keywords: Adaptive Robotic. Niche Evolution. Evaluate Time.

LISTA DE FIGURAS

Figura 1 – Gene, Cromossomo e População 21
Figura 2 – Mutação 21
Figura 3 – Cross-over 21
Figura 4 – Ciclo algoritmos evolutivos 22
Figura 5 – Processo de otimização do NES contendo apenas 2 parâmetros e uma função *fitness* (vermelho = bom, azul = ruim). Em cada iteração é mostrado o valor atual do parâmetro (em branco), a população de possíveis soluções (em preto), e o gradiente estimado (flecha branca). Os parâmetros são movidos na direção da seta até uma solução melhor. 24
Figura 6 – Arquitetura Neurônios 25
Figura 7 – NeuroEvolution 27
Figura 8 – Robótica Evolutiva 28
Figura 9 – Diferentes agentes utilizados nos ambientes de simulação Hopper, Ant, Walker2D e Humanoid em sequência. 30
Figura 10 – Agentes dos ambientes Hopper, Halfcheetah, Ant em sequência 38
Figura 11 – Performance dos melhores agentes evoluídos nos ambientes selecionados. Cada *box* representa o valor da *fitness* dos agentes após a pós evolução. Suas condições de nichos e duração de episódios variam de acordo com os dados apresentados posteriormente. 41
Figura 12 – Performance dos agentes evoluídos em quatro diferentes grupos no ambiente Hopper, onde cada grupo apresenta 5 diferentes algoritmos evolutivos. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. A Figura (a), (b), (c) e (d) apresentam os algoritmos com 2, 3, 5 e 10 nichos e suas equivalências 42
Figura 13 – Performance dos agentes evoluídos no ambiente Hopper no grupo contendo 3 nichos. Os algoritmos presentes são os já utilizados, apenas sendo adicionado o algoritmo RandSeed-NE. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. 43

Figura 14 – Performance dos agentes evoluídos no ambiente Hopper no grupo contendo 3 nichos, variando a quantidade de episódios em 1, 5 e 10. Os algoritmos escolhidos foram o (a)FixedSeed-NE, (b)RandSeed-NE e (c)OpenAi-Es-Ne, todos sendo comparados ao melhor resultado de LA-Es. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. 44
Figura 15 – Performance das replicações do algoritmo FixedSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Hopper. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução. 44
Figura 16 – Performance das replicações do algoritmo RandSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Hopper. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução. 45
Figura 17 – Performance dos agentes evoluídos em quatro diferentes grupos no ambiente Ant, onde cada grupo apresenta 5 diferentes algoritmos evolutivos. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. A Figura (a), (b), (c) e (d) apresentam os algoritmos com 2, 3, 5 e 10 nichos e suas equivalências 46
Figura 18 – Performance dos agente evoluídos no ambiente Ant no grupo contendo 3 nichos. Os algoritmos presente são os já utilizados, apenas sendo adicionado o algoritmo RandSeed-NE. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. 47

Figura 19 – Performance dos agentes evoluídos no ambiente Ant no grupo contendo 3 nichos, variando a quantidade de episódios em 1, 5 e 10. Os algoritmos escolhidos foram o (a)FixedSeed-NE, (b)RandSeed-NE e (c)OpenAi-Es-Ne, todos sendo comparados ao melhor resultado de LA-Es. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. 48
Figura 20 – Performance das replicações do algoritmo FixedSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Ant. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução. 48
Figura 21 – Performance das replicações do algoritmo RandSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Ant. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução. 49
Figura 22 – Performance dos agentes evoluídos em quatro diferentes grupos no ambiente HalfCheetah, onde cada grupo apresenta 5 diferentes algoritmos evolutivos. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. A Figura (a), (b), (c) e (d) apresentam os algoritmos com 2, 3, 5 e 10 nichos e suas equivalências 50
Figura 23 – Performance dos agentes evoluídos no ambiente HalfCheetah no grupo contendo 3 nichos. Os algoritmos presentes são os já utilizados, apenas sendo adicionado o algoritmo RandSeed-NE. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. 51

Figura 24 – Performance dos agentes evoluídos no ambiente HalfCheetah no grupo contendo 3 nichos, variando a quantidade de episódios em 1, 5 e 10. Os algoritmos escolhidos foram o (a)FixedSeed-NE, (b)RandSeed-NE e (c)OpenAi-Es-Ne, todos sendo comparados ao melhor resultado de LA-Es. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. 52
Figura 25 – Performance das replicações do algoritmo FixedSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente HalfCheetah. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução. 53
Figura 26 – Performance das replicações do algoritmo RandSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente HalfCheetah. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução. 53

LISTA DE TABELAS

Tabela 1 – Relação das condições iniciais nos algoritmos 37
Tabela 2 – Relação da duração dos episódio em cada algoritmo 39
Tabela 3 – Relação da duração dos episódio nos algoritmos 39
Tabela 4 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Hopper. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. *O algoritmo RandSeed-NE apresentou significância estatística em relação ao OpenAi-Es-Ne e ao LA-Es. 45
Tabela 5 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Ant. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. *O algoritmo LA-Es apresentou significância estatística em relação ao FixedSeed-NE e ao RandSeed-NE. 49
Tabela 6 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente HalfCheetah. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. . . 54
Tabela 7 – Média, Mediana e Desvio Padrão dos algoritmos contendo 2 nichos no ambiente Hopper. 72
Tabela 8 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Hopper. 72
Tabela 9 – Média, Mediana e Desvio Padrão dos algoritmos contendo 5 nichos no ambiente Hopper. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE. 72
Tabela 10 – Média, Mediana e Desvio Padrão dos algoritmos contendo 10 nichos no ambiente Hopper. *Os algoritmos LA-Es, LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE. 73
Tabela 11 – Média, Mediana e Desvio Padrão dos algoritmos contendo 2 nichos no ambiente Ant. *O algoritmo LA-SP apresentou significância estatística em relação ao OpenAi-Es-NE. 73
Tabela 12 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Ant. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE. 73

Tabela 13 – Média, Mediana e Desvio Padrão dos algoritmos contendo 5 nichos no ambiente Ant. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística sendo LA-SP em relação a todos e FixedSeed-NE-SP em relação ao OpenAi-Es-Ne e ao FixedSeed-NE. 74
Tabela 14 – Média, Mediana e Desvio Padrão dos algoritmos contendo 10 nichos no ambiente Ant. *Os algoritmos LA-Es, LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE. 74
Tabela 15 – Média, Mediana e Desvio Padrão dos algoritmos contendo 2 nichos no ambiente HalfCheetah. *O algoritmo LA-SP apresentou significância estatística em relação ao OpenAi-Es-NE. 74
Tabela 16 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente HalfCheetah. 75
Tabela 17 – Média, Mediana e Desvio Padrão dos algoritmos contendo 5 nichos no ambiente HalfCheetah. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE. 75
Tabela 18 – Média, Mediana e Desvio Padrão dos algoritmos contendo 10 nichos no ambiente HalfCheetah. *Os algoritmos LA-Es, LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE. 75

SUMÁRIO

1 INTRODUÇÃO 16
1.1 OBJETIVOS 17
1.1.1 Objetivo Geral 17
1.1.2 Objetivos Específicos 17
1.2 ORGANIZAÇÃO DO TRABALHO 18
2 FUNDAMENTAÇÃO TEÓRICA 19
2.1 ROBÔS AUTÔNOMOS 19
2.2 ROBÓTICA ADAPTATIVA 20
2.3 ALGORITMOS EVOLUTIVOS 20
2.3.1 Estratégia Evolutiva 23
2.3.1.1 NES 24
2.4 REDES NEURAIS 25
2.4.1 Neuroevolution 26
2.5 ROBÓTICA EVOLUTIVA 27
3 TRABALHOS RELACIONADOS 29
3.1 ON THE IMPACT OF THE DURATION OF EVALUATION EPISODES ON THE EVOLUTION OF ADAPTIVE ROBOTS 29
3.2 FAVORING THE EVOLUTION OF ADAPTIVE ROBOTS THROUGH ENVIRONMENTAL DIFFERENTIATION 30
3.3 THE EFFECTIVENESS OF NICHING ON OPENAI-EVOLUTION STRATEGIES IN THE EVOLUTION OF ROBOTIC BEHAVIOR 33
3.3.1 Considerações finais 35
4 DESENVOLVIMENTO 36
4.1 METODOLOGIA 36
4.1.1 Algoritmos 36
4.1.2 Ambientes de Simulação 37
4.1.3 Protocolo experimental 38
4.2 RESULTADOS 40
4.2.1 Hopper 40
4.2.2 Ant 45
4.2.3 HalfCheetah 50
5 CONCLUSÃO 55
6 TRABALHOS FUTUROS 57
APÊNDICE A – ARTIGO SBC 59
APÊNDICE B – LISTA DE TABELAS 72
B.1 HOOPER 72
B.2 ANT 73

B.3 HALFCHEETAH 74
C – REFERENCIAS 76

1 INTRODUÇÃO

A medida em que a tecnologia continua a se desenvolver rapidamente, robôs têm se tornado cada vez mais presentes no nosso dia-a-dia. Podem ser encontrados dentro de fábricas e outras áreas da indústria, exercendo papéis perigosos ou difíceis para seres humanos. Além disso, também estão começando a aparecer mais em ambientes domiciliares e hospitalares, um exemplo são os robôs aspiradores de pó, que são capazes de operar automaticamente por horas.

Robôs autônomos são robôs que possuem um determinado nível de autonomia para exercer certa função. Para isso, recebem dados sobre o ambiente, podendo ser através de câmeras, sensores, etc, analisam e exercem um comportamento predeterminado, como no exemplo anterior, a ação de limpar.

Alguns fatores devem ser levados em consideração em relação a criação de um robô: morfologia, sensores, motores, controles, etc (BRUNO SICILIANO, 2016). Nolfi (2021) define um robô sendo um sistema artificial que contém: (i) um corpo físico que inclui sensores, um cérebro, etc, (ii) é situado em um ambiente físico e, eventualmente, em um ambiente social, que inclui outros robôs e/ou humanos e (iii) exibe um comportamento que é relacionado a uma função. Um dos maiores desafios encontrados é a relação entre todos esses fatores e como uni-los para que o robô possa exercer o comportamento desejado, com isso, deve-se observar cada aspecto separadamente ou como um todo. Uma das maneiras de resolver esse problema é utilizar algoritmos que imitam a evolução dos seres vivos para desenvolver um robô. Com isso, surgiu a robótica evolutiva.

Segundo Nolfi (NOLFI; BONGARD *et al.*, 2016) (NOLFI, 2021) robótica evolutiva é uma técnica para a criação de robôs autônomos, inspirada pelo princípio de seleção natural, feita por Charles Darwin. A ideia por trás da robótica evolutiva funciona com uma geração inicial de candidatos, gerados de forma aleatória, que vão evoluindo com foco em um resultado previamente selecionado. Para que tudo isso seja possível, essa técnica utiliza algoritmos evolutivos para desenvolver, modificar e encontrar o melhor robô para a solução desejada.

A ideia por trás dos algoritmos evolutivos vem de meados de 1990, quando pesquisadores idealizaram resoluções de problemas ao tentarem imitar a capacidade intelectual de um indivíduo ou de uma população. Segundo Bäck (1996), algoritmos evolutivos modelam o processo de aprendizado coletivo dentro de uma população de indivíduos, cada qual representando uma potencial solução. A população é inicializada por um método referente ao algoritmo, e evolui em direção a uma área que possa conter melhores resultados dentro do conjunto solução, por processos de recombinação, mutação e seleção. Assim como na natureza, o ambiente contribui na evolução de uma população, sendo um agente que influencia no comportamento dos indivíduos.

Os ambientes são os meios em que os indivíduos serão treinados e avaliados medi-

ante a sua interação. Cada ambiente possui condições iniciais, como obstáculos, objetos e restrições diferentes. A relação entre indivíduo e ambiente é avaliada comparando o comportamento performado em relação ao comportamento desejado. Essa avaliação dita quais indivíduos irão permanecer e/ou quais adaptações deverão ser feitas no robô, se assemelhando com a plasticidade fenotípica. Plasticidade fenotípica é um termo da biologia que pode ser definida como a "habilidad" que consiste na capacidade dos organismos de alterarem sua morfologia ou fisiologia de acordo com o meio (PIGLIUCCI; MURREN; SCHLICHTING, 2006).

Indivíduos treinados em diferentes ambientes mostram comportamentos mais completos e complexos (AUERBACH; BONGARD, 2014), com isso, podemos utilizar variações ambientais para criar os melhores indivíduos em diferentes condições, deixando-os mais genéricos. Assim surgiu a ideia de dividir uma população em diferentes áreas, conhecidas como ilhas ou nichos, cada uma com ambientações diferentes. O intuito é que cada conjunto de indivíduos seja avaliado dentro do seu ambiente, nicho, e, posteriormente, possa popular outros nichos, caso seja melhor que os lá presentes (CARVALHO; NOLFI, 2017).

Além da configuração do ambiente, o tempo em que cada indivíduo permanece interagindo também pode alterar a forma que ele evolui. Esse período de interação é conhecido como episódio de avaliação e, mesmo sendo de grande importância, ainda é pouco estudado. Porém, já se sabe que a duração do episódio de avaliação pode ter efeitos drásticos no comportamento e na performance de um robô (ROSA *et al.*, 2022).

Neste trabalho houve a junção de duas das características que influenciam em uma evolução, variação ambiental e duração do episódio de avaliação. Essas adaptações foram utilizadas para a criação de dois algoritmos: o primeiro com condições ambientais iniciais fixas em cada nicho e o segundo em que as condições ambientais iniciais de cada nicho variam ao decorrer da evolução.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Investigar efetividade da combinação da diferenciação ambiental usando nichos com a variação da duração dos episódios de avaliação em algoritmos evolutivos aplicados a robótica evolutiva.

1.1.2 Objetivos Específicos

- Modificar algoritmos evolutivos com evolução de nicho para que cada nicho tenha um tempo de evolução específico.
- Escolha e agrupamento de algoritmos para comparação.
- Seleção de ambientes de simulação para testar os novos algoritmos evolutivos.

- Rodar os algoritmos com diferentes quantidade de nichos e tempo de evolução em cada um dos ambientes selecionados.
- Comparar os resultados com outros algoritmos evolutivos evolutivos.

1.2 ORGANIZAÇÃO DO TRABALHO

A seguir está descrito como o trabalho estará organizado. O segundo capítulo apresentará diversos conceitos obrigatórios para o entendimento completo do trabalho. No terceiro capítulo serão apresentados outros trabalhos análogos a esse estudo. O capítulo de número quatro apresentará os resultados prévios. O quinto capítulo apresentará conclusões mediante os resultados obtidos neste trabalho, e, por fim, o sexto capítulo apresentará possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos para o entendimento deste trabalho.

2.1 ROBÔS AUTÔNOMOS

Robôs autônomos são aqueles que realizam tarefas, em ambientes desestruturados, com certo nível de independência humana. Existem vários tipos de tarefas realizadas por robôs autônomos, que vão desde limpezas de um local, até exploração de ambientes que não são adequados para os seres humanos.

Robôs são constituídos de componentes que interagem para exibir um comportamento específico. Stefano (NOLFI, 2021) define robôs como um sistema constituído por um corpo que inclui atuadores, sensores e um cérebro, é situado em um ambiente, físico ou simulado, e exibe um comportamento.

Cada aspecto do corpo de um robô é definido pelo ambiente onde ele estará e pela função que exercerá. Um robô, o qual tem como função segurar um objeto, necessita conter braços, por exemplo. Robôs que precisam observar o ambiente devem conter câmeras, e etc.

São os sensores que permitem com que os robôs obtenham as informações dos ambiente que estão presentes, como por exemplo a distância até uma parede, a frequência da luz, medição da temperatura, etc. Existem dois tipos de sensores: proprioceptivo e exteroceptivo. Sensores proprioceptivos medem os valores internos de um robô, como a velocidade da roda, quantidade de bateria, posição de uma perna, etc. Já os sensores exteroceptivos medem os valores externos como câmeras, microfones, etc. Sensores podem ser combinados com atuadores ou outras partes para extrairem mais informações.

O cérebro, também conhecido como *controller* ou *policy*, regula os atuadores, partes responsáveis pelos movimentos, de acordo com os estados e informações enviadas pelos sensores. O comportamento desenvolvido pelo robô é o resultado da interação entre ele e o ambiente, orquestrado pelo cérebro (NOLFI, 2021). No princípio, o cérebro dos robôs era feito de fios e transistores, porém, atualmente o cérebro é feito por um sistema computacional que conecta os sensores e os atuadores.

Braitenberg foi um médico e psiquiatra que acreditava que criar agentes artificiais capazes de exibir um comportamento natural contribuiria para entender melhor a inteligência natural. Estes agentes foram posteriormente chamados de veículos de Braitenberg (BRAITENBERG, 1986). Esses veículos incluíam sensores que captam propriedades do ambiente e motores de movimentar os veículos. O cérebro era constituído por fios que ligavam os sensores aos motores, diretamente ou indiretamente, através de mediadores. As topologias das conexões dos veículos foram inspiradas em um sistema nervoso como simetria, conexões contrárias, excitação e inibição.

Os veículos de Braitenberg variam pela quantidade de sensores, quantidade de

motores e como funcionam as conexões. Eles demonstram como um robô pode exercer vários comportamentos que dependem crucialmente das características do local onde ele está. Além disso, mostram como o número e o tipo de comportamentos produzidos por um robô não depende unicamente de um cérebro, mas também de um ambiente.

2.2 ROBÓTICA ADAPTATIVA

Segundo Nolfi (NOLFI, 2022), o termo robótica adaptativa se refere a métodos que permitem o design de um robô capaz de desenvolver suas habilidades autonomamente, através de um processo de aprendizado ou processo evolutivo. Requerem mínima ação humana para produzir o comportamento desenvolvido pelo robô e as regras de controles que o produzem sejam descobertas, tudo através de um processo adaptativo autônomo com base em uma recompensa ou uma função fitness, a qual mede se o comportamento desenvolvido pelo robô foi eficaz.

Tem seu foco em aprendizado ponta-a-ponta (*end-to-end*), quando sistemas recebem diretamente o estado dos sensores do robô e determinam diretamente a ação que será exercida por tal, sem envolver nenhum tipo de pré-processamento.

Seus primeiros experimentos foram realizados em 1994 (NOLFI; FLOREANO *et al.*, 1994), porém, apenas utilizando robôs pouco complexos e realizando tarefas simples. Robôs capazes de exercer atividades mais complexas foram desenvolvidos mais recentemente (TAN *et al.*, 2018) (ANDRYCHOWICZ *et al.*, 2020).

Grandes avanços foram feitos nesta área quando utilizados algoritmos de adaptação mais poderosos, além do uso de simuladores que barateiam e aumentam a curva de aprendizado em menos tempo, e processos de exploração com maior capacidade.

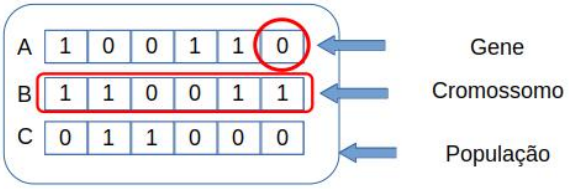
2.3 ALGORITMOS EVOLUTIVOS

Algoritmos evolutivos são métodos artificiais que buscam reproduzir dinâmicas similares à evolução natural, muito embora não seja um de seus objetivos reproduzi-las fidedignamente. Buscam na evolução natural a bio-inspiração para desenvolver estratégias que permitam encontrar soluções satisfatórias, ou até ótimas, para um determinado conjunto de problemas. Utilizam processos comumente encontrados na natureza como reprodução, mutação, recombinação e seleção.

Nesses algoritmos, um conjunto de possíveis soluções a ser testado para um determinado problema é chamado de população ou candidatos. Dentro dessa população, cada solução é chamada de indivíduo, genótipo ou cromossomo. Dentro dessas soluções há genes, que são parâmetros que constituem uma parte da solução, como mostra a Figura 1.

Cada indivíduo recebe uma nota de acordo com uma avaliação, conhecida como *fitness*. Essa nota é quem dita quais indivíduos devem permanecer e quais devem ser eliminados. Após as primeiras possíveis soluções serem avaliadas, são escolhidas as melhores

Figura 1 – Gene, Cromossomo e População



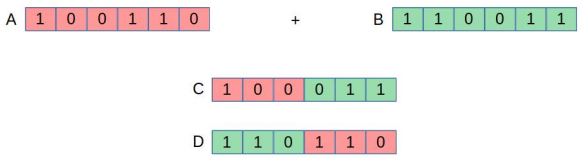
Fonte: Adaptado de (MUNONYE, 2018)

para serem parentes da próxima geração. Há duas maneiras de gerar novos candidatos, a mutação e a recombinação. A mutação acontece quando um dos parentes é selecionado e alterado os valores do seu gene, como mostra a Figura 2. A recombinação, também conhecida como *cross-over*, acontece quando é selecionada uma divisão entre os dois parentes e combinado partes dos seus genótipos, Figura 3, sempre buscando um maior *fitness*.

Figura 2 – Mutação

Fonte: Adaptado de (MUNONYE, 2018)

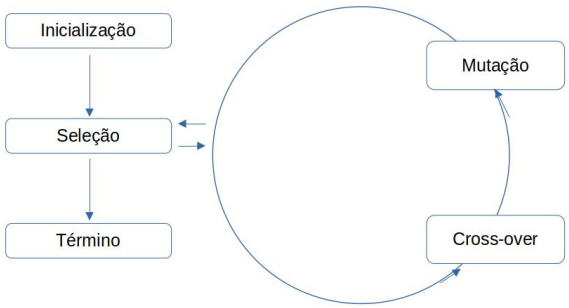
Figura 3 – Cross-over



Fonte: Adaptado de (MUNONYE, 2018)

Após um número de iterações, o processo é terminado quando atinge um critério predefinido, Figura 4. Esse critério pode ser um valor de *fitness*, um comportamento alcançado ou um limite no número de gerações. Ao chegar nesta etapa, a melhor solução é encontrada.

Figura 4 – Ciclo algoritmos evolutivos



Fonte: Adaptado de (MUNONYE, 2018)

2.3.1 Estratégia Evolutiva

Estratégia evolutiva, conhecida como *Evolution strategy* (ES), é uma técnica de otimização global *black-box* que pertence à classe dos algoritmos evolutivos. Seu início foi em meados de 1960, na Universidade Técnica de Berlim (TUB), quando P. Bienert, I. Rechenberg e H.-P.Schwefel necessitavam otimizar problemas relacionados à hidrodinâmica, mas não conseguiam por conta da sua complexidade (BÄCK, 1996). Esse tipo de algoritmo se diferencia dos demais em relação a como a população é representada e como a mutação e recombinação são efetuadas, além de poder ser aplicado em todos os campos de otimização incluindo contínuo, discreto, entre outras (BEYER, 2007).

Um termo muito utilizado, e de grande importância, é o *search space*. *Search space* é um conjunto de soluções possíveis de um problema de otimização que satisfazem suas restrições. Esse conjunto solução é uma estrutura de dados que pode ser finita mas não necessariamente com tamanho fixo. Um exemplo é o *search space* de números reais com N-dimensões, ou um *search space* dos números inteiros, ou *search space* dos binários, ou até mesmo uma mistura de espaços e subespaços (BEYER; SCHWEFEL, 2002).

Existem duas formas padrão desse algoritmo (μ/ρ,λ)-ES e (μ/ρ+λ)-ES, onde μ denota a quantidade de pais, ρ denota a quantidade de pais envolvidos na criação de novos indivíduos, *offspring*, e λ o número de *offspring*. Os novos pais são selecionados deterministicamente ou dentro dos *offspring*, na primeira forma do algoritmo (*comma-selection*) onde ($\mu < \lambda$), ou entre os antigos pais e os *offspring*, na segunda forma do algoritmo (*plus-selection*).

Cada indivíduo da população é constituído por *a:=(y, s, F(y))*, onde y é o vetor de parâmetros a serem otimizados, s é um conjunto de estratégias e F(y) o *fitness* referente ao *search space*.

O algoritmo conceitual das duas formas segue o seguinte padrão (BEYER, 2007):

1. Inicializa a população pai $P\mu = \{a1,...,a2\}$
2. Gera λ *offspring* $\tilde{a}$ formando a população de *offspring* $P\lambda = \{\tilde{a}1,...,\tilde{a}2\}$
   a) Seleciona (aleatoriamente) ρ pais de $P\mu$
   b) Recombina os ρ pais ã selecionados para formar uma recombinação de indivíduos r
   c) Faz uma mutação no parâmetro s da recombinação r
   d) Faz uma mutação no parâmetro y da recombinação r usando o parâmetro modificado s
3. Seleciona uma nova população de pais, de acordo com qual forma vai ser implementada (μ/ρ,λ)-ES ou (μ/ρ+λ)-ES

4. Volta para o passo de geração de offspring até que o critério para terminar seja encontrado.

2.3.1.1 NES

Natural Evolution Strategies (NES) é uma família de estratégias evolucionárias que, ao invés de modificar as soluções apenas adicionando uma perturbação, utiliza uma distribuição de probabilidade para gerar as mutações. Utiliza-se de gradientes, vetores que apontam na direção de um outro resultado para atualizar sua solução (WIERSTRA *et al.*, 2011).

A ideia principal por trás do NES é a utilização de *search gradients* para atualizar a distribuição de probabilidade. *Search gradients* são definidos como amostras do vetor dos *fitness* esperados e a distribuição de probabilidade pode ser qualquer distribuição, sendo necessário a possibilidade da sua derivação, como a Gaussiana ou a de Cauchy.

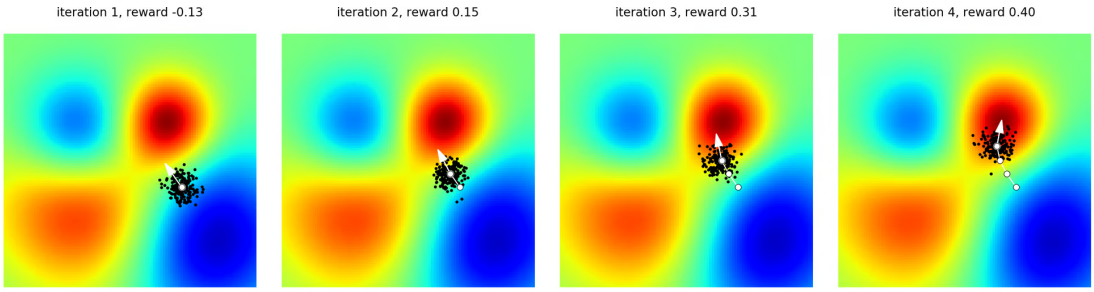
Os *fitness* esperados são calculados utilizando a fórmula (WIERSTRA *et al.*, 2011):

$$\nabla_\theta J(\theta) = \frac{1}{\lambda}\sum_{k=1}^{\lambda} f(\mathbf{z}_k)\nabla_\theta log\pi(\mathbf{z}_k|\theta),$$

Onde θ representa os parâmetros da distribuição de probabilidade $\pi(\mathbf{z}_k|\theta)$, $f(\mathbf{z})$ representa a função *fitness* das possíveis soluções $\mathbf{z}$ e λ representa o tamanho da população. O vetor da estimativa do *fitness* é quem mostra a direção que deve ser seguida dentro do espaço da distribuição de probabilidade. Com isso, a distribuição de probabilidade é atualizada a cada iteração por um gradiente ascendente:

$$\theta \leftarrow \theta\eta\nabla_\theta J(\theta).$$

Figura 5 – Processo de otimização do NES contendo apenas 2 parâmetros e uma função *fitness* (vermelho = bom, azul = ruim). Em cada iteração é mostrado o valor atual do parâmetro (em branco), a população de possíveis soluções (em preto), e o gradiente estimado (flecha branca). Os parâmetros são movidos na direção da seta até uma solução melhor.



Fonte: (KARPATHY *et al.*, 2017)

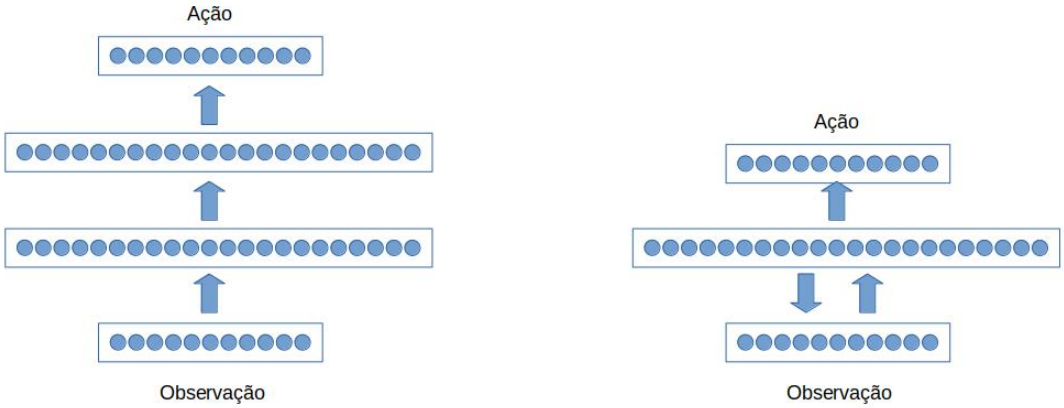
2.4 REDES NEURAIS

Redes neurais, também conhecidas como rede neural artificial (ANN), são modelos computacionais inspirados em um sistema nervoso, onde são capazes de realizar o aprendizado de máquina, bem como o reconhecimento de padrão.

Redes neurais artificiais (NOLFI, 2021) são constituídas por unidades computacionais simples chamadas de neurônios, interconectados por ligações com peso, chamadas conexões. Neurônios produzem como *output* um valor real, que é o resultado de uma função, onde as conexões com outros neurônios servem como *input*, ou em caso de neurônios sensoriais, os *input* são sensores. O valor do *input* recebido pela conexão depende do *output* do neurônio com que a conexão se origina, neurônio pré sináptico, e o peso da conexão, que é relativa a um número real. As conexões podem aumentar ou diminuir o valor de saída do neurônio pré sináptico, e são referenciadas como conexões excitatórias e inibitórias, respectivamente.

A arquitetura de uma rede neural é estruturada de acordo com a maneira que os neurônios são conectados. Geralmente os neurônios são organizados em níveis ou camadas: o nível sensorial, um ou mais níveis internos e o nível de motores. Em uma arquitetura *feedforward*, cada neurônio de uma camada é conectado com os outros neurônios da próxima camada. A informação possui um fluxo unidimensional que vai dos neurônios sensoriais até os neurônios motores. Outra arquitetura é a rede neural recorrente, nessa proposta os neurônios podem se conectar com neurônios da mesma camada ou até da camada anterior. Esses modelos de arquiteturas podem ser vistos na Figura 6.

Figura 6 – Arquitetura Neurônios



Fonte: Adaptado de (NOLFI, 2022)

2.4.1 Neuroevolution

Neuroevolução, ou *neuroevolution*, é uma técnica que aplica algoritmos evolutivos para construir redes neurais artificiais, tendo como inspiração a evolução natural do sistema nervoso (LEHMAN, J.; MIIKKULAINEN, R., 2013). Diferente dos demais métodos, a neuroevolução permite aprendizados sem um foco explícito, apenas com *feedbacks* e uma rede neural estruturada qualquer. É muito utilizada nos campos de robótica evolutiva e *artificial life*.

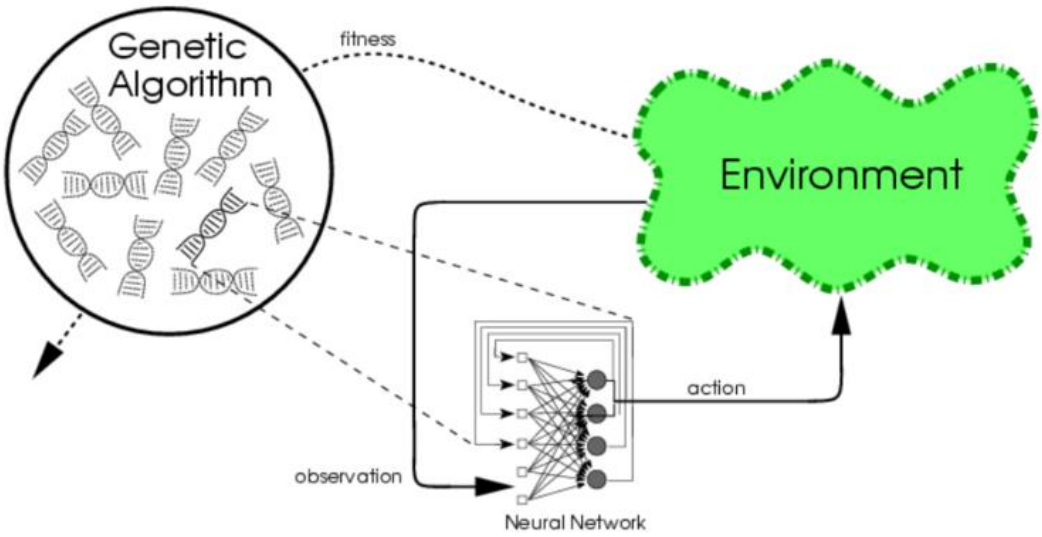
Um dos grandes objetivos da *neuroevolution* é o de evoluir complexas redes neurais capazes de desenvolver um comportamento inteligente. Como resultado disso, pode ser utilizado tanto para estudar como um comportamento inteligente evolui na natureza, quanto para fazer com que redes neurais desempenhem uma tarefa específica.

Nessa técnica que envolve redes neurais e algoritmos evolutivos, há uma diferença na hora de avaliar os genótipos, que é a adição de um novo passo. Cada genótipo contém os pesos das conexões de uma rede neural, chamada de fenótipo. Cada fenótipo contém entradas e saídas que, no caso da robótica, podem ser dados de sensores e dados de motores respectivamente.

Os fenótipos são utilizados pelos robôs. Utilizam a rede neural para exercer o comportamento e retornam seus *fitness*, assim segue o algoritmo evolutivo. A evolução que acontece no fenótipo é relacionada tanto a topologia, quantidade de neurônios e quais suas conexões, tanto em relação ao peso de cada conexão, a fim de obter um resultado desejado.

Utilizando os robôs de Braitenberg como base, podemos substituir convencionalmente os fios que conectam os sensores e os motores por uma rede neural artificial com neurônios sensoriais e neurônios motores e, se necessários, neurônios internos.

Figura 7 – NeuroEvolution



Fonte: (LEHMAN, Joel; MIIKKULAINEN, Risto, 2013)

2.5 ROBÓTICA EVOLUTIVA

Robótica evolutiva (ER) é um dos campos da robótica que procura criar robôs autônomos com maior robustez e adaptabilidade (DONCIEUX *et al.*, 2015). Essa subdivisão aplica conceitos de seleção, variação e princípios hereditários de evolução natural.

ER utiliza algoritmos evolutivos para desenvolver comportamentos, ou também a morfologia robótica, a fim de obter um comportamento desejado pelo projetista. Para obter o *fitness* de cada indivíduo, é necessário colocá-lo em um ambiente, seja ele real ou virtual, com condições iniciais predeterminadas e deixá-lo interagir. O que será avaliado é o comportamento do robô mediante as condições encontradas no ambiente. O valor resultante é relacionado ao quão bem o robô performou seu objetivo. Esse valor, então, retorna ao algoritmo evolutivo para seguir o ciclo, como mostrado na Figura 8.

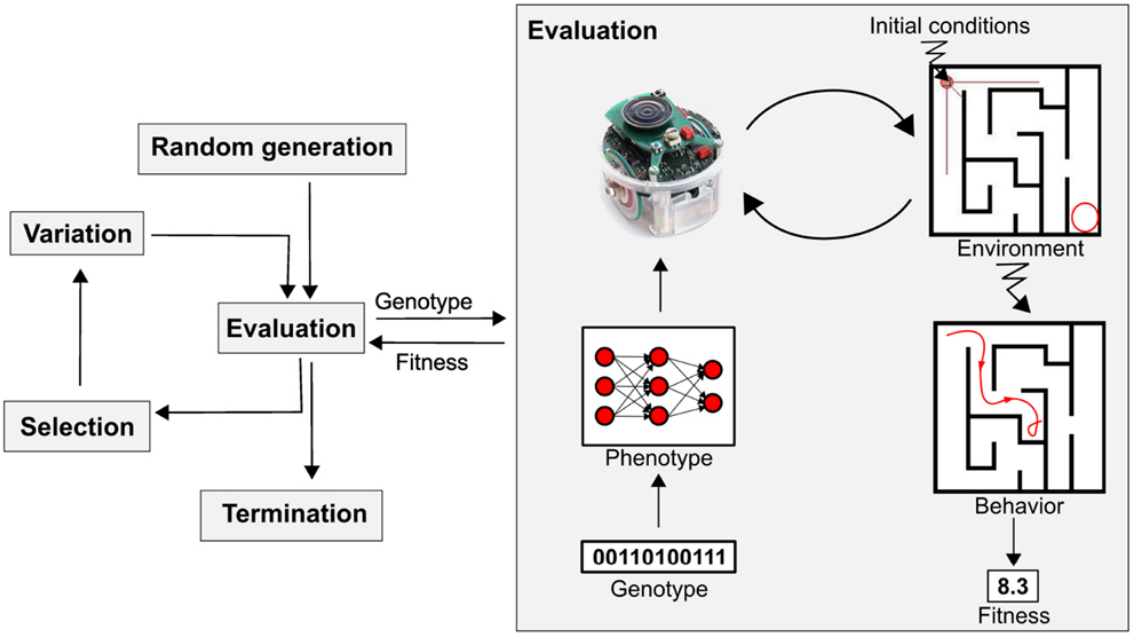
Em particular, robótica evolutiva contém uma grande ênfase em corporeidade, experiência e na interação de cérebro, corpo e ambiente, o qual é crucial para o desenvolvimento de uma inteligência, de um comportamento adaptável e de um processo de cognição (SILVA; CORREIA; CHRISTENSEN, 2016).

Nolfi (NOLFI, 2021) exemplifica um método que pode ser utilizado em etapas:

1. O projetista escolhe uma tarefa e um sistema robótico para o problema. Por exemplo, o projetista quer evoluir uma perna robótica para ter a habilidade de andar em um terreno irregular.

2. O projetista faz o projeto e implementa uma função *fitness* para avaliar o desempenho do robô.

Figura 8 – Robótica Evolutiva



Fonte: (DONCIEUX *et al.*, 2015)

3. O projetista evolui os pesos da conexão do cérebro do robô dentro de uma simulação utilizando um algoritmo evolutivo, como o já mostrado anteriormente.

4. O projetista coloca o cérebro mais evoluído dentro de um robô físico e verifica se a solução evoluída dentro da simulação funciona da mesma maneira na realidade.

3 TRABALHOS RELACIONADOS

Nesta seção serão abordados alguns dos trabalhos relacionados a este, contendo resumos e pontos relevantes destes artigos. Os três artigos apresentados servirão como base do trabalho, seguindo uma ordem lógica de como os trabalhos influenciaram a atual proposta. O primeiro artigo apresentado demonstra como a variação do episódio de duração durante o processo evolutivo impacta no desempenho dos comportamentos evoluídos. O segundo artigo apresenta conceitos necessários para o entendimento, implementação e impacto da variação ambiental. O terceiro artigo apresenta uma aplicação da variação ambiental, através de nichos, em um algoritmo evolutivo já existente. Neste artigo, o novo algoritmo foi testado e comparado ao algoritmo base.

3.1 ON THE IMPACT OF THE DURATION OF EVALUATION EPISODES ON THE EVOLUTION OF ADAPTIVE ROBOTS

Este artigo tem o intuito de demonstrar a relação entre o tempo de duração de um episódio de avaliação e o resultado do *fitness* de um robô. O valor do *fitness* de um robô não depende somente do projetista, mas sim de outras variáveis, como propriedades de um robô, as características do ambiente, o tempo de duração da relação entre um robô e o ambiente e uma possível terminação precoce.

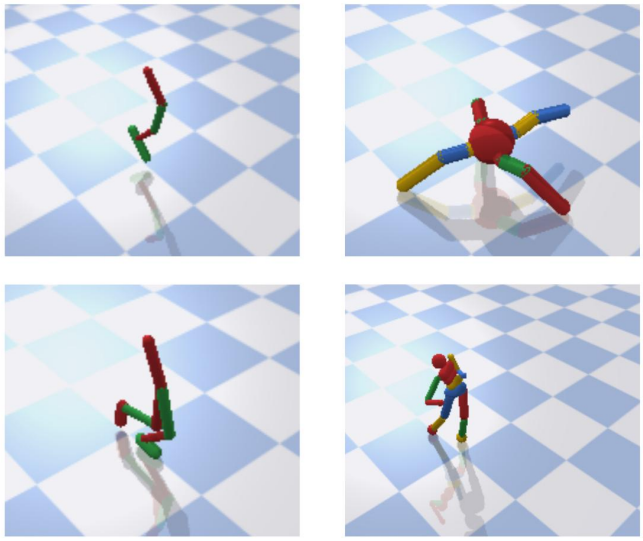
A terminação precoce vem do fato de que não é necessário continuar avaliando indivíduos que não podem melhorar posteriormente a um evento. Um exemplo seria um robô que necessita equilibrar um objeto, mas após um certo ângulo ele não conseguiria mais se recuperar. Ao terminar mais cedo o episódio de avaliação, o tempo de duração de uma avaliação diminui, fazendo com que o custo de computação e o tempo total da simulação também diminua.

A relação com a estrutura do robô e as condições iniciais de um ambiente são amplamente estudadas, contrário a terminação precoce e tempo de duração de um episódio, onde não há grandes pesquisas sobre. Há outro artigo que também estuda este impacto, porém, em relação ao tempo mínimo necessário para obter um certo comportamento, não há melhor relação para se obter um melhor *fitness*.

Este estudo utiliza, como algoritmos evolutivo o OpenAi-Es (SALIMANS *et al.*, 2017), algoritmo baseado em NES, o simulador Evorobotpy2 e quatro variações de ambientes da ferramenta PyBullet. Cada ambiente possui um agente diferente, mas necessitam produzir o mesmo comportamento, dirigir-se a uma localização o mais rápido possível. Os agentes são o Hopper, formado por 4 segmentos que simbolizam um perna que anda pulando, Ant, formado por 4 pernas conectados a um corpo, Walker2D, formado por 2 pernas conectadas a um torso e Humanoid, formado por duas pernas, dois braços, um

torso e uma cabeça.

Figura 9 – Diferentes agentes utilizados nos ambientes de simulação Hopper, Ant, Walker2D e Humanoid em sequência.



Fonte: (ROSA *et al.*, 2022)

O tempo de duração dos episódios são representados por um valor de *steps* e foram divididos da seguinte forma: número padrão com valor de 1000 *steps*, um número fixo, o qual foi repetido com 100, 200, 300, 400 e 500, dois números fixos sendo um a primeira metade da evolução com valores 100, 200, 300, 400 e 500, e a outra metade com 1000, um número incremental que inicializa em 100 e cresce de 100 em 100 a cada 10% da avaliação, uma condição específica onde o número é fixo em 1000 e a função *fitness* é otimizada, e por último um valor aleatório com intervalo de [50-1000] com distribuição uniforme.

O resultado demonstra que o tempo de duração dos episódios está diretamente ligado ao valor da *fitness* de um indivíduo. Foi observado que a variação do tempo, ou começando com valores menores e aumentando a medida que o processo evolutivo acontece ou metade do processo com um valor menor e a outra metade com um valor maior, obteve melhores resultados do que os outros padrões utilizados.

3.2 FAVORING THE EVOLUTION OF ADAPTIVE ROBOTS THROUGH ENVIRONMENTAL DIFFERENTIATION

Neste artigo foi discutido como a variação ambiental facilita a evolução de melhores possíveis soluções para o problema da navegação *double-pole*. Além disso, é proposto um novo algoritmo evolutivo baseado em nichos.

Da mesma maneira que mutações geram novidades genéticas, a variação do ambiente também gera, porém com maior importância. Isso pode ser observado a partir de

alguns fatores como: o ambiente afeta imediatamente um grande número de indivíduos, o ambiente gera modificações imediatas e persistentes, entre outros.

Nichos são subdivisões de um ambiente com características diferentes. Anterior a este trabalho, pesquisas foram efetuadas sobre este assunto, mas apenas para preservar a diversidade entre a população. Porém, neste artigo, foi adicionado outra etapa, a que o melhor indivíduo de cada nicho pode conquistar outros nichos. Isso se dá ao fato de que o melhor indivíduo gerado pelo algoritmo tem que ser o melhor em vários tipos de ambiente.

O ambiente *double-pole* é utilizado pelo uso extenso em evolução de agentes. Consiste em controlar um veículo que contêm dois polos anexados por uma articulação de dobradiça, conectadas em cima do carro para manter um equilíbrio, onde o agente não tem conhecimento da velocidade do veículo e nem dos polos. As condições iniciais diferem pela posição inicial do veículo e de cada polo.

O experimento foi separado em duas partes: a primeira foi que 20 agentes foram avaliados em condições iguais de ambiente, ou seja, as condições iniciais eram iguais, e outros 20 agentes foram avaliados colocados em 20 diferentes nichos com condições iniciais diferentes.

Os experimentos do mesmo ambiente utilizaram estratégia evolutiva (μ+1)-ES em que μ parentes geram 1 *offspring*, como mostra o algoritmo de ambiente homogêneo. Já os com condições diferentes utilizaram o novo algoritmo por eles criado, como mostra o algoritmo heterogêneo, utilizando (1+1).

Algorithm 1: Homogeneoues environments

```
Randomly generate N genotypes;
Randomly generate 1 niche;
for p = 1 to p = μ do
    Put pth genotype in niche 1;
    Evaluate pth genotype;
end
while nEval ≤ MAX EVALUATIONS do
    for p = 1 to p = μ do
        o ← Duplicate and Mutate pth genotype;
        Evaluate o;
    end
    Select the μ genotypes with highest fitness;
end
```

O resultado obtido foi que agentes evoluídos em ambientes heterogêneos obtiveram melhores valores do que aqueles evoluídos em ambientes homogêneos, sendo a diferença entre eles estatisticamente relevante. Foi relatado também que o algoritmo criado para usar em ambientes heterogêneos possui performance melhor do que outros métodos.

Algorithm 2: Heterogeneous environments

```
Randomly generate N genotypes;
Randomly generate N niches;
for n = 1 to n = N do
    Put nth genotype in niche n;
    Evaluate nth genotype;
end
while nEval ≤ MAX EVALUATIONS do
    for 50 iterations do
        for n = 1 to n = N do
            i ← best genotype from niche n;
            o ← duplicate and mutate i;
            Evaluate o in niche n;
            if fitness o ≥ fitness i then
                Replace i with o in niche n;
            end
        end
    end
    fitMatrix ← zeros(N,N);
    for n = 1 to n = N do
        bo ← the best offspring from niche n;
        for m = 1 to m = N do
            if niche m ≠ current niche of bo then
                Evaluate bo in niche m;
                fitMatrix_{n,m} ← fitness bo in niche m;
            end
            else
                fitMatrix_{n,m} ← 0;
            end
        end
    end
    for m = 1 to m = N do
        maxFit ← max(fitMatrix_{all,m});
        j ← index of max(fitMatrix_{all,m});
        i ← best genotype from niche m;
        if maxFit ≥ fitness i then
            o ← duplicate jth genotype;
            Replace i with o in niche m;
            fitMatrix_{j,all} ← 0;
        end
    end
end
```

3.3 THE EFFECTIVENESS OF NICHING ON OPENAI-EVOLUTION STRATEGIES IN THE EVOLUTION OF ROBOTIC BEHAVIOR

Neste artigo foi aplicado o conceito de nicho, e suas variações, no algoritmo criado pela OpenAi, o OpenAi-ES, resultando no OpenAi-Es-NE.

O artigo demonstra os nichos como um conjunto de condições iniciais geradas aleatoriamente, onde o agente será avaliado. Para isso, é criado uma matriz para representá-los, sendo a coluna a quantidade de nichos e as linhas a quantidade de avaliações de um indivíduo. Cada valor da matriz é uma *seed*, um valor que representa um conjunto de condições iniciais dos parâmetros do problema selecionado, neste caso, o *double-pole*.

É criado também um vetor com tamanho da quantidade de pesos da rede neural, chamado de centróide.

A evolução funciona avaliando os centróides dentro de seus nichos e, se necessários, trocando de lugar. Essa troca acontece quando um centroide de outro nicho obtém valores melhores do que o centróide do próprio nicho. As trocas são feitas quando uma certa quantidade de gerações foram avaliadas, fazendo com que, se houver a troca, os centróides que exerceram a troca já obtiveram valores de *fitness* consideráveis.

Algorithm 3: OpenAI-ES-NE

```
Randomly generate N niches;
Generate N centroids;
while nGens ≤ MAX_GENERATIONS do
    for 50 iterations do
        for n = 1 to n = N do
            Evaluate nth centroid in niche n;
            Optimize nth centroid;
        end
    end
    fitMatrix ← zeros(N,N);
    for n = 1 to n = N do
        cn ← nth centroid;
        for m = 1 to m = N do
            if niche n ≠ niche m then
                Evaluate cn in niche m;
                fitMatrix_{n,m} ← fitness cn in niche m;
            end
            else
                fitMatrix_{n,m} ← 0;
            end
        end
    end
    for m = 1 to m = N do
        maxFit ← max(fitMatrix_{all,m});
        j ← index of max(fitMatrix_{all,m});
        i ← mth centroid;
        if maxFit ≥ fitness i then
            o ← jth centroid;
            Replace i with o in niche m;
            fitMatrix_{all,j} ← 0;
            fitMatrix_{m,all} ← 0;
        end
    end
end
```

Os resultados demonstram que a adição de nichos e a variação do ambiente alteram favoravelmente a performance do algoritmo, tendo um aumento na média dos resultados de aproximadamente 9%, considerando o problema do *double-pole*.

3.3.1 Considerações finais

Como mostram os resultados obtidos em cada artigo, a variação no tempo de duração de um episódio e a variação ambiental utilizando nichos aumentam a eficácia na obtenção de um valor de *fitness* melhor, quando comparado a algoritmos que não utilizam nenhuma dessas adaptações. Com isso, os artigos referenciados serão utilizados como base para criar um novo tipo de algoritmo, onde será utilizada a variação da duração dos episódios presente no primeiro artigo, a varição das condições ambientais presente no segundo artigo e a utilização de nichos presente no terceiro artigo. Além disso, a fim de comparação, os ambientes de simulação selecionados, foram os que obtiveram melhores resultados no segundo artigo.

4 DESENVOLVIMENTO

Neste capítulo será apresentado a metodologia escolhida para este trabalho e os resultados obtidos.

4.1 METODOLOGIA

Neste trabalho houve a junção da variação da duração do episódio de avaliação com a diferenciação ambiental através dos nichos em um único algoritmo, o qual possuirá duas versões, a fim de evoluir os pesos da rede neural de robôs em ambientes de simulação selecionados. Os novos algoritmos são baseados no OpenAi-Es-Ne (BIANCHINI; MACHADO; CARVALHO, 2023), que já possui a diferenciação através dos nichos, porém, para adicionar a variação da duração, foi utilizada uma fórmula que divide proporcionalmente o tempo do episódio de avaliação. A diferença entre os novos algoritmos é relacionada às condições iniciais de cada nicho, onde um possui as condições iniciais fixas, FixedSeed-NE, e o outro varia essas condições ao decorrer da evolução, RandSeed-NE, a fim de aumentar as condições ambientais ao decorrer da evolução.

4.1.1 Algoritmos

A fim de comparação, foram escolhidos dois algoritmos que possuem bons resultados e usam como base um algoritmo vastamente utilizado. Os algoritmos de comparação são o OpenAi-Es-Ne (BIANCHINI; MACHADO; CARVALHO, 2023), algoritmo baseado no OpenAi-Es (SALIMANS *et al.*, 2017), que possui a variação de nichos,mas não possui a variação da duração dos episódios. O outro algoritmo (ROSA *et al.*, 2022) baseado também em OpenAi-Es, não possui nichos, mas possui a variação no tempo de avaliação, que iremos chamar de LA-Es. Além desses algoritmos, também será utilizada uma variação de um dos algoritmos deste trabalho, FixedSeed-NE, e do LA-Es, onde o poder computacional será aumentado.

Cada algoritmo possui uma duração total de evolução igual, no valor de 20 milhões de *steps*, já os que possuem poder aumentado, a duração será multiplicada pela relação da duração com o número de nicho. Cada um deles apresenta configurações diferentes em relação ao ambiente e ao tempo de um episódio de avaliação. Em relação ao ambiente, o algoritmo LA-Es utiliza uma única condição inicial do ambiente, que se altera durante toda a avaliação, o FixedSeed-NE, RandSeed-NE e o OpenAi-Es-Ne possuem condições iniciais diferentes em cada nicho, porém, o RandSeed-NE, assim como o LA-Es, varia essas condições.

Em relação a duração de um episódio de avaliação, os algoritmos apresentam ou um valor fixo de 1000, ou um valor variável com máximo de 1000. O algoritmo OpenAi-Es-Ne possui o número fixo, mas os algoritmos FixedSeed-NE, RandSeed-NE e o LA-Es possuem

Algoritmos	**Condições iniciais**
OpenAiEs-Ne	Fixa
FixedSeed-NE	Fixa
RandSeed-NE	Variada
LA-Es	Variada

Tabela 1 – Relação das condições iniciais nos algoritmos

uma variação que pode, ou variar ao longo do tempo de avaliação, LA-Es, ou variar de acordo com cada nicho, FixedSeed-NE e RandSeed-NE. O algoritmo LA-Es possui uma variação que começa com um valor inicial e aumenta de acordo com uma porcentagem do tempo total de avaliação até chegar ao valor máximo. Já os algoritmos propostos, FixedSeed-NE e RandSeed-NE, utilizam-se de uma fórmula para dividir o tempo entre os nichos, que é definido por:

$$MaxSteps = (\xi/\mu).(\theta + 1) \tag{1}$$

Onde ξ representa a duração máxima de um episódio, μ representa a quantidade de nichos e θ o *index* atual do nicho. Assim, a duração dos episódios é dividida proporcionalmente, além de cada nicho conter um valor diferente.

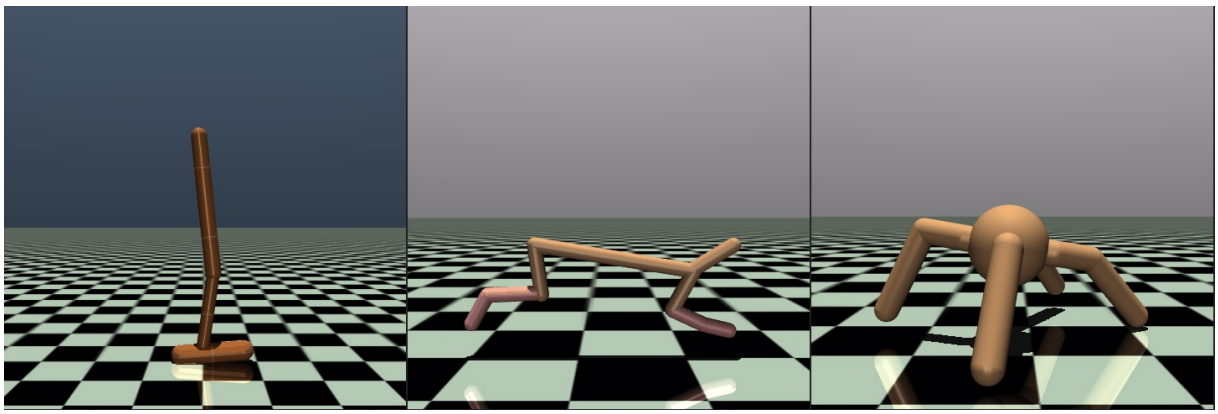
Para a criação de novos candidatos, todos os algoritmos utilizam estratégia evolutiva (1+1)-ES durante uma iteração de 44. No final das 44 iterações, 1 parente gera 1 *offspring* que pode substituir seu parente caso o valor de *fitness* obtido for maior ou igual do que seu parente. Em relação ao número de gerações, a quantidade final em cada algoritmo varia de acordo com o quão bem os indivíduos evoluem.

4.1.2 Ambientes de Simulação

Para testar essas diferenças, foram escolhidos ambientes de simulação diferentes. Os ambientes são compostos por um robô, sua rede neural e um ambiente propriamente dito. Cada ambiente de simulação foi selecionado pelos bons valores apresentados em trabalhos e pesquisas anteriores, que usam alguns dos algoritmos utilizados como comparação neste trabalho. Tais ambientes são o Ant e Hopper, além de um ambiente não utilizado anteriormente, o HalfCheetah. Todos possuem uma rede neural do tipo *feed-foward* com 3 camadas e 50 neurônios internos, sendo sensores na primeira cada e a última camada é relacionada ao movimento do robô, nesses casos, a locomoção, controlando os atuadores.

Um dos ambientes é o HalfCheetah, baseado em Wawrzyński (2009), constituído por um robô bidimensional contendo 9 conexões, que simulam partes rígidas, e 8 juntas, que simulam a junção entre as conexões, local onde é aplicado torque para que cumpra o objetivo. Seu objetivo é correr em uma direção, neste caso à direita, o mais rápido

Figura 10 – Agentes dos ambientes Hopper, Halfcheetah, Ant em sequência



Fonte: Desenvolvido pelo autor

possível. Seu *fitness* é calculado pela diferença entre a distância percorrida dividido pela duração dessa ação, e um valor de penalidade caso suas ações sejam consideradas muito demoradas. Seu torso e cabeça são fixos, não podendo ser aplicado torque em suas juntas, sobrando apenas 6, as quais são das pernas, joelhos e pés. Sua rede neural é composta por 26 neurônios na primeira camada e 6 neurônios na última camada.

O ambiente Ant é baseado em Schulman *et al.* (2018), constituído por um robô tridimensional possuindo 1 torso e 4 pernas que contêm 2 juntas cada, uma conecta a perna ao torso e outra a junção das partes que constituem a perna. Seu objetivo é coordenar as pernas para andar para frente, neste caso também à direita, aplicando torque nas 8 juntas. Sua função *fitness* é calculada pela soma de duas recompensas e subtração de uma penalidade. Suas recompensas são de acordo com cada passo dado e pela distância percorrida dividido pela duração dessa ação. A penalidade é calculada caso aconteça de uma força externa ser aplicada. Sua rede neural é composta por 28 neurônios na primeira camada e 8 neurônios na última camada.

O último ambiente é o Hopper. Ambiente baseado em Durrant-Whyte, Roy e Abbeel (2012), constituído por um robô bidimensional que contém torso, perna, separada em coxa e canela e pé. Seu objetivo é coordenar suas partes com intuito de se mover em uma direção, aplicando torque em suas 3 juntas que conectam suas 4 partes. Sua função *fitness* é calculada seguindo o mesmo cálculo do Ant. Sua rede neural é composta por 15 neurônios na primeira camada e 3 neurônios na última camada.

4.1.3 Protocolo experimental

Para investigar os efeitos da variação da duração dos episódios em relação à presença dos nichos, as experiências foram separadas em 4 grupos, onde cada grupo contém uma avaliação de cada algoritmo. Os algoritmos que utilizam nichos apresentam 2, 3, 5 e 10 nichos e 50 gerações para se obter a troca de nichos. Já o algoritmo LA-Es, que não utiliza nicho, utiliza uma duração de episódios equivalente. Quando comparado aos algoritmos

Episódios por Algoritmo	**2 Nichos**	**3 Nichos**	**5 Nichos**	**10 Nichos**
OpenAiEs-Ne	1000	1000	1000	1000
FixedSeed-NE	500 por nicho	333 por nicho	200 por nicho	100 por nicho
RandSeed-NE	500 por nicho	333 por nicho	200 por nicho	100 por nicho
LA-Es	500 + 500 a cada 50%	333 + 333 a cada 34%	200 + 200 a cada 20%	100 + 100 a cada 10%

Tabela 2 – Relação da duração dos episódio em cada algoritmo

com 2 nichos, LA-Es inicializa utilizando o valor de 500 e muda para 1000 quando chega em 50% da população. Quando comparado aos de 3 nichos, começa com 333 e aumenta 333 a cada 34%. Quando aos de 5 nichos, inicializa com 200 e aumenta 200 a cada 20%. Já quando comparado com os de 10, se inicializa em 100 e aumenta 100 a cada 10%. E, por fim, o algoritmo OpenAI-Es-NE utiliza um número fixo que dura toda a avaliação no valor de 1000.

Mediante as condições iniciais apresentadas, outras comparações devem ser feitas. Como cada algoritmo possui um tempo total de duração igual, os que apresentam nichos possuem um *budget* computacional menor, já que é dividido entre todos os nichos. Para obter uma igualdade neste tópico, os experimentos FixedSeed-NE e LA-Es também devem apresentar suas versões com o poder computacional equivalente, onde o *budget* atual será multiplicado proporcionalmente pelo número de nichos da comparação. Essas versões serão chamadas de FixedSeed-NE-SP e LA-SP.

Algoritmos	**2 Nichos**	**3 Nichos**	**5 Nichos**	**10 Nichos**
FixedSeed-NE-SP	500 por nicho	333 por nicho	200 por nicho	100 por nicho
LA-SP	500 + 500 a cada 50%	333 + 333 a cada 34%	200 + 200 a cada 20%	100 + 100 a cada 10%

Tabela 3 – Relação da duração dos episódio nos algoritmos

Outro estudo que deve ser feito é em relação a diferentes condições iniciais experienciadas pelos robôs ao decorrer da evolução e sua relação com o valor de *fitness* final. Para isso, os algoritmos OpenAi-Es-Ne, FixedSeed-NE e RandSeed-NE, serão avaliados aumentando a quantidade de episódios durante a evolução, fazendo com que o RandSeed-NE possa experienciar uma maior quantidade de condições iniciais diferentes, assim, comparando os resultados obtidos dos três algoritmos com os melhores resultados do LA-Es. Para o cálculo final de *fitness*, é feito uma média onde o valor de cada episódio será somado e depois divido pela quantidade de episódios utilizados, porém, para se obter equivalência, o *budget* computacional é aumentado de acordo com o número de episódios

utilizado pelo algoritmo, dando mais "tempo" aos robôs. As condições escolhidas para o estudo foram de utilizar 3 nichos e 1, 5 e 10 episódios.

Para os estudos, todos os experimentos necessários rodaram seus algoritmos 10 vezes em cada ambiente, isso significa que cada algoritmo utilizou 10 condições iniciais de ambientes diferentes. Após as avaliações, a melhor solução é pós-avaliada. A pós-avaliação funciona iniciando as condições do agente, como postura, direção das conexões ou direção das juntas, de forma aleatória e aplicando a rede neural evoluída em um ambiente com condições iniciais aleatórias, as quais diferem das condições do ambiente utilizadas para treinar a rede neural. Essa interação gera outro valor de *fitness*, que é utilizado para se comparar a eficiência de cada resultado. A pós-avaliação deve ser feita utilizando sempre o mesmo algoritmo para se manter um padrão, neste trabalho, é utilizado 1000 episódios de pós-avaliação e OpenAi-Es-Ne como algoritmo, a fim de se obter um valor mais preciso.

Para um estudo mais analíticos, após a coleta de todos os dados, valores de *fitness*, medianas e médias, foram feitos testes de significância estatística para que possamos comprovar que os resultados obtidos tem relevância estatística. Estes testes foram feitos primeiramente em todos os resultados obtidos nos grupos e, posteriormente, nos melhores resultados obtidos nas diferentes condições de episódios, todos utilizando a correção de Bonferroni.

4.2 RESULTADOS

Esta seção relaciona os experimentos efetuados com cada ambiente previamente selecionado e seus resultados apresentados. Todos os resultados obtidos foram plotados em gráficos utilizando *boxplot*, onde cada *box* representa os valores dentro dos quartis, as linhas horizontais pretas representam a mediana e os pontos brancos representam a média. As marcações acima e abaixo das *box*, no final das linhas verticais pretas, representam o melhor e pior valor respectivamente. Cada *box* representa o resultado obtido durante 10 replicações de cada experimento correspondente.

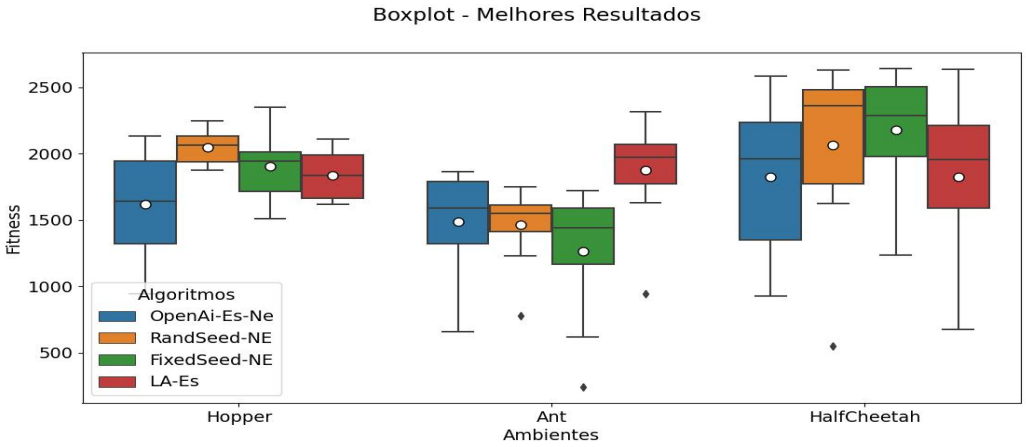
A imagem a seguir mostra os melhores resultados obtidos por todos os algoritmos bases em todos os ambientes de simulação selecionados, a fim de entender, de forma geral, os resultados obtidos.

A partir desta imagem, serão apresentados de forma mais aprofundada os resultados obtidos de cada algoritmo mediante as condições selecionadas e os ambientes em que efetuaram a simulação.

4.2.1 Hopper

O ambiente Hopper foi utilizado para investigar a variação da duração dos episódios, o aumento do *budget* computacional e o aumento da variação das condições iniciais. A Figura 12 mostra os valores de *fitness* obtidos em cada grupo especificado anterior-

Figura 11 – Performance dos melhores agentes evoluídos nos ambientes selecionados. Cada *box* representa o valor da *fitness* dos agentes após a pós evolução. Suas condições de nichos e duração de episódios variam de acordo com os dados apresentados posteriormente.



Fonte: Desenvolvido pelo autor

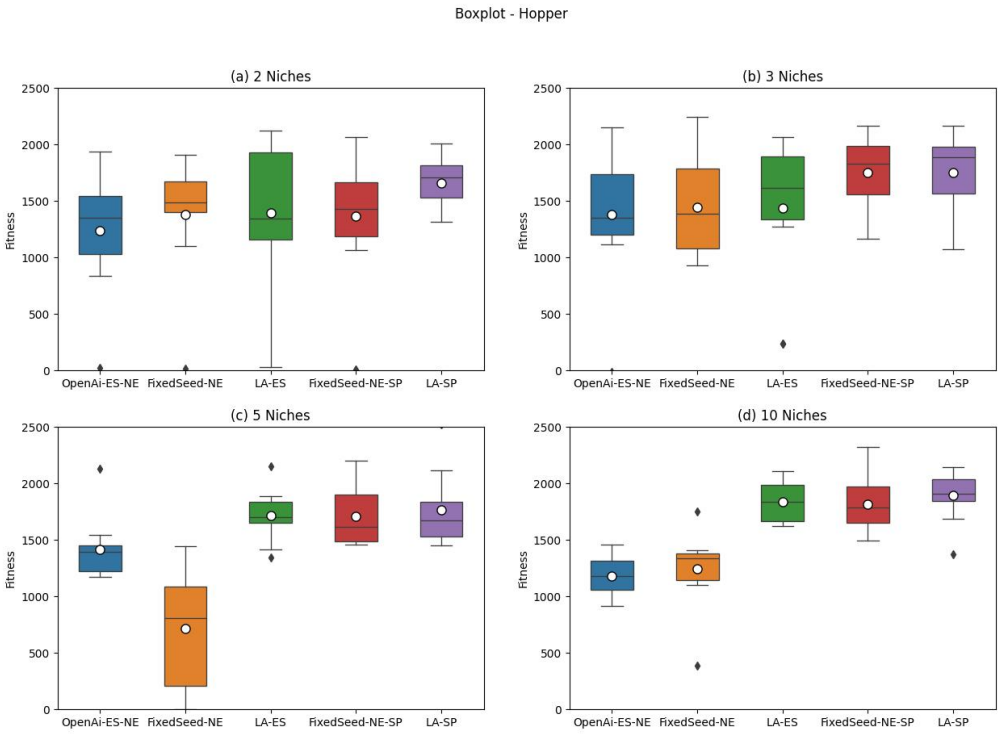
mente, que contém, além dos algoritmos iniciais, suas versões com poder computacional aumentado.

Como pode ser observado na imagem, primeiro comparando os algoritmos que utilizam nichos, o que apresenta um valor fixo da duração do episódio, OpenAi-Es-Ne, apresenta resultados piores comparado ao que contém a variação para cada nicho, FixedSeed-NE, sendo apenas superior com 5 nichos. Já em relação aos algoritmos que utilizam a variação, o algoritmo LA-Es obteve valores consideravelmente melhores a cada aumento no número de nichos, quando comparado ao FixedSeed-NE, não sendo melhor apenas com 2 nichos, onde o FixedSeed-NE obteve os melhores resultados. Na visão geral, o algoritmo LA-Es obteve o melhor desempenho de todos, obtendo, quase sempre, o maior valor entre os máximos e mínimos, além de apresentar melhores medianas. Em relação ao poder computacional, a Figura 12 mostra que este quesito é um gargalo, sendo possível ser aumentado para se obter melhores resultados. Pode-se observar que com o *budget* aumentado, os nichos conseguem se desenvolver melhor e apresentar valores melhores. Além disso, o aumento do poder computacional também demonstrou que, mesmo com um maior *budget*, o algoritmo LA-Es ainda apresenta melhores resultados.

Mediante a estes resultados, o grupo que contém a comparação de 3 nichos apresentou bons resultados de FixedSeed-NE e sua variação com *budget* aumentado e, por isso, foi adicionado nesse grupo o algoritmo RandSeed-NE. O resultado, então, é mostrado na Figura 13.

O gráfico mostra que o algoritmo RandSeed-NE apresenta o segundo melhor resultado mediante as configurações de 3 nichos, perdendo apenas para o LA-Es. Os valores apresentados contêm uma variação menor, podendo ser observado pelo tamanho da *box*, sendo assim, mesmo não apresentando a melhor mediana, possui valores mais constan-

Figura 12 – Performance dos agentes evoluídos em quatro diferentes grupos no ambiente Hopper, onde cada grupo apresenta 5 diferentes algoritmos evolutivos. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. A Figura (a), (b), (c) e (d) apresentam os algoritmos com 2, 3, 5 e 10 nichos e suas equivalências



Fonte: Desenvolvido pelo autor

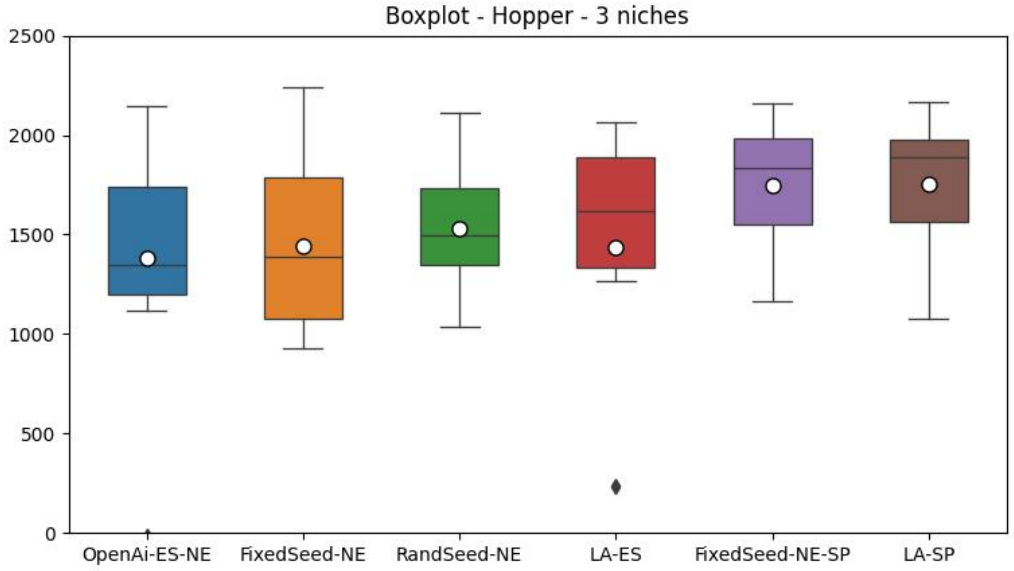
tes que os outros, mostrando que, mesmo com *budget* computacional menor, a variação ambiental demonstra bons resultados.

Com isso, há o estudo do aumento da variação das condições iniciais, no ambiente Hopper, os resultados dos diferentes experimentos podem ser observados na Figura 14.

Analisando o gráfico, os três algoritmos apresentam comportamentos diferentes à medida em que o número de episódios vai aumentando. O algoritmo FixedSeed-NE apresenta seu melhor resultado com 5 episódios, onde apresenta a maior mediana dos três, possuindo também o maior valor de *fitness* e o maior mínimo de *fitness*. Quando comparado ao melhor resultado de LA-Es, os experimentos com 5 e 10 episódios mostram melhores medianas e valores.

Já o algoritmo RandSeed-NE apresenta o melhor comportamento dos três, onde a cada aumento de episódio, apresenta valores cada vez melhores. Quando utilizado com 5 e 10 episódios, principalmente com 10, apresenta medianas melhores do que a do algoritmo LA-Es. Além disso, apresenta uma diminuição na variedade dos valores de *fitness*, tendo

Figura 13 – Performance dos agentes evoluídos no ambiente Hopper no grupo contendo 3 nichos. Os algoritmos presentes são os já utilizados, apenas sendo adicionado o algoritmo RandSeed-NE. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente.



Fonte: Desenvolvido pelo autor

resultados mais parecidos.

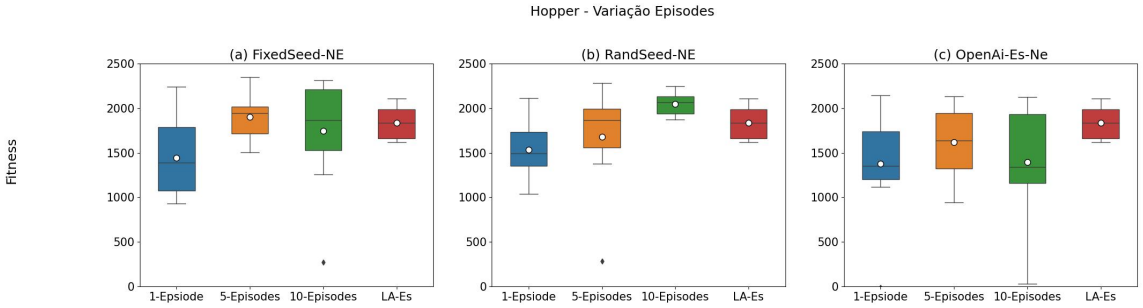
E por fim o OpenAi-Es-Ne não apresenta ótimos resultados, tendo seu melhor desempenho com 5 episódios, mas mesmo assim sendo pior do que o LA-Es. Seu gráfico mostra valores com uma maior variância do que os outros algoritmos da comparação, especialmente quando há 10 nichos.

As Figuras 15 e 16 apresentam os valores obtidos ao decorrer das evoluções dos algoritmos FixedSeed-NE e RandSeed-NE. Cada figura apresenta as evoluções utilizando 1, 5 e 10 episódios.

Na Figura 15 podemos observar que o valor de *fitness* aumenta mais cedo à medida em que a quantidade de episódios aumenta, tendo seu melhor valor com 5 episódios. As curvas de *fitness* também mantêm esse mesmo padrão, onde sua curva começa a estabilizar com valores cada vez maiores. A figura apresenta o motivo do algoritmo ter resultados melhores com 5 episódios, já que apresentam evoluções mais constantes e mais parelhas, obtendo assim, uma melhor mediana.

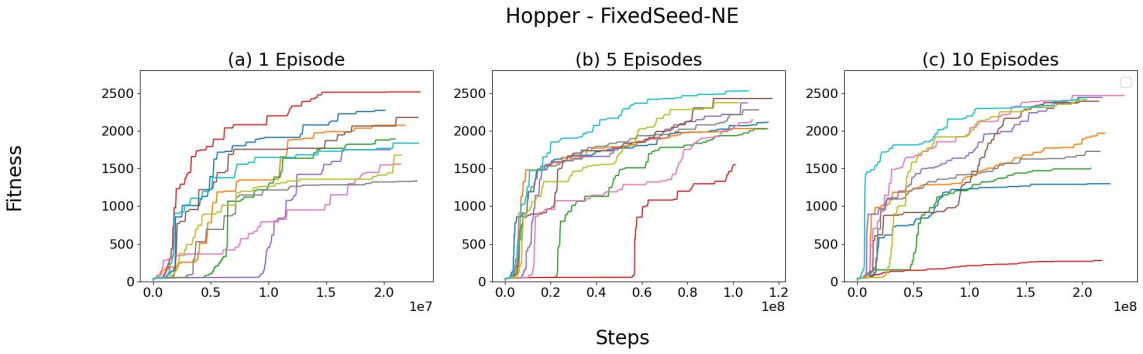
Já na Figura 16, a diferença dos valores e das curvas de *fitness* ficam mais nítidas. Diferente da Figura 15, o RandSeed-NE apresenta valores melhores com 10 episódios, informação que já era constatada na Figura 14. Diferente do algoritmo anterior, este demonstra a ótima evolução nas configurações de 10 episódios, onde desde o início da

Figura 14 – Performance dos agentes evoluídos no ambiente Hopper no grupo contendo 3 nichos, variando a quantidade de episódios em 1, 5 e 10. Os algoritmos escolhidos foram o (a)FixedSeed-NE, (b)RandSeed-NE e (c)OpenAi-Es-Ne, todos sendo comparados ao melhor resultado de LA-Es. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente.



Fonte: Desenvolvido pelo autor

Figura 15 – Performance das replicações do algoritmo FixedSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Hopper. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução.



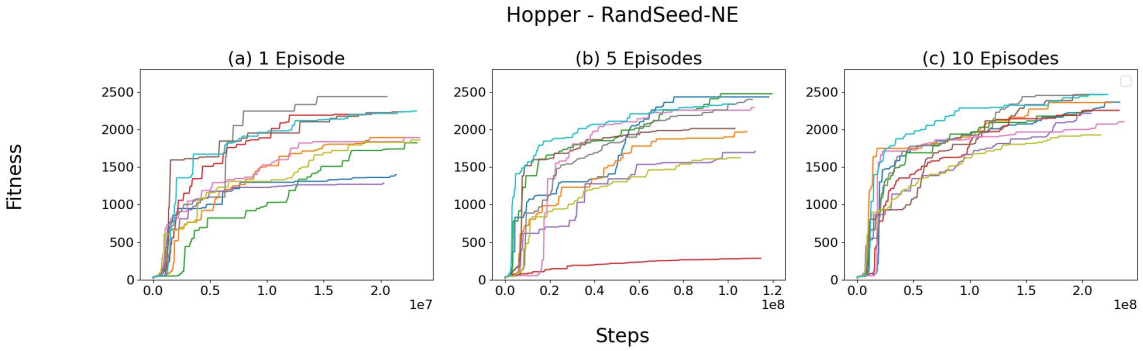
Fonte: Desenvolvido pelo autor

evolução, os resultados tendem a se tornarem bons valores de *fitness*.

Para melhor visualização, a partir dos melhores resultados, os dados estatísticos obtidos foram colocados em uma tabela que consta com mediana, média e desvio padrão dos algoritmos OpenAI-Es-Ne, FixedSeed-NE, RandSeed-NE e LA-Es.

Como pode ser observado na Tabela 4, o melhor resultado do algoritmo FixedSeed-NE apresenta mediana e média melhores do que o melhor resultado do LA-Es, com a mediana tendo um aumento de 6% e a média de 3,6%. Já o algoritmo que apresenta a maior variação nas condições iniciais, RandSeed-NE, apresenta a maior mediana e média de toda a comparação, tendo, na mediana, um aumento de 6% em relação ao FixedSeed-NE e de 12% em relação ao LA-ES, já na média, apresenta um aumento de 7,3% para o

Figura 16 – Performance das replicações do algoritmo RandSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Hopper. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução.



Fonte: Desenvolvido pelo autor

Algoritmo	Mediana	Média
OpenAi-Es-Ne	1638.36	1616.01 ± 388.72
FixedSeed-NE	1946.13	1904.57 ± 237.88
*RandSeed-NE	2064.29	2045.59 ± 115.72
LA-Es	1837.38	1837.45 ± 175.12

Tabela 4 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Hopper. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. *O algoritmo RandSeed-NE apresentou significância estatística em relação ao OpenAi-Es-Ne e ao LA-Es.

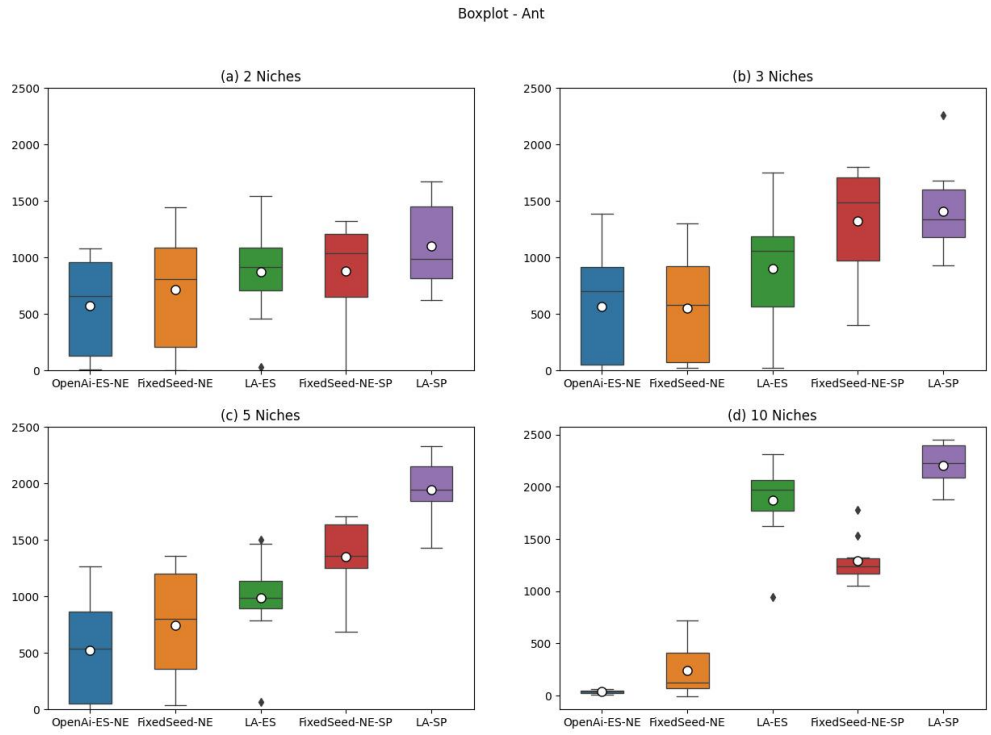
FIxedSeed-NE e 10,9% para o LA-Es.

4.2.2 Ant

Para o Ambiente Ant, foram selecionadas as mesmas comparações do ambiente Hopper, a variação da duração dos episódios, o aumento do *budget* computacional e o aumento na variação das condições iniciais. Primeiro foi feita a comparação com algoritmos bases e com poder computacional aumentado em todos os grupos, 2, 3, 5 e 10, como mostra a Figura 17.

Segundo a imagem, quando comparamos os algoritmos que possuem nichos, novamente o FixedSeed-NE apresenta melhores valores que o OpenAi-Es-Ne. Quando há 3 nichos, o algoritmo OpenAi-Es-Ne possui uma melhor mediana, mas possui valores menores. Quando comparamos os algoritmos que possuem a variação da duração do episódio, o algoritmo LA-Es sempre possui valores melhores, sendo os valores resultantes ou

Figura 17 – Performance dos agentes evoluídos em quatro diferentes grupos no ambiente Ant, onde cada grupo apresenta 5 diferentes algoritmos evolutivos. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. A Figura (a), (b), (c) e (d) apresentam os algoritmos com 2, 3, 5 e 10 nichos e suas equivalências



Fonte: Desenvolvido pelo autor

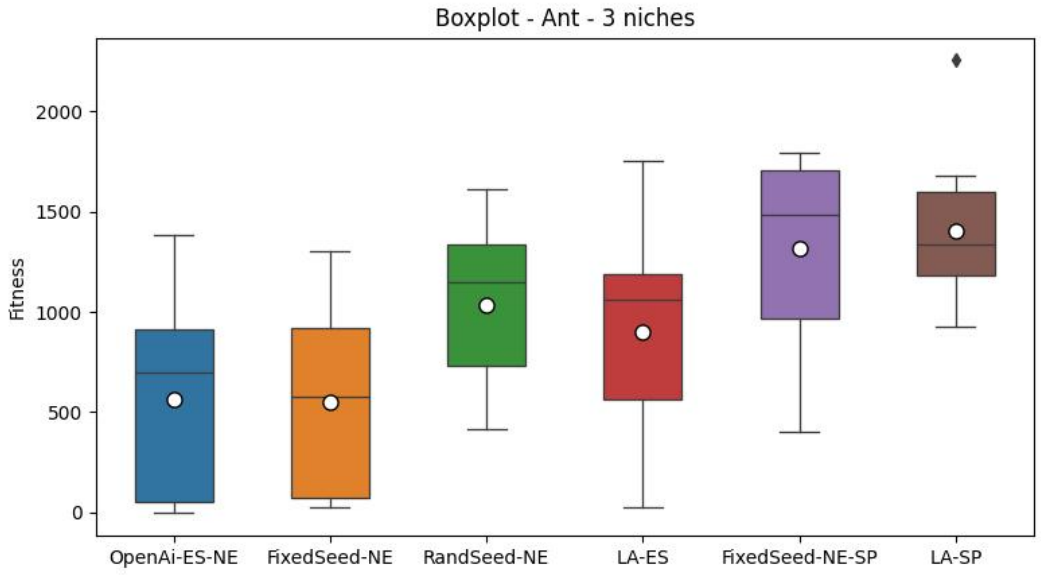
a mediana. Em uma comparação geral, o algoritmo que não possui nicho, mas possui a variação, apresenta os melhores valores, principalmente na comparação de 10 nichos. Com o aumento do poder computacional, o algoritmo FixedSeed-SP aumenta significativamente, comparado ao FixedSeed-NE, que não possui o aumento do *budget*, conseguindo ter uma mediana maior que ao do LA-SP nos casos de 2 e 3 nichos. Mostrando que, caso os nichos apresentem o *budget* computacional proporcional a outros algoritmos, eles apresentam melhores resultados nesta configuração. Entretanto, quando aumenta para 5 e 10 nichos, a diferença se inverte e o algoritmo LA-SP se sobressai.

Pode ser observado que, para o ambiente Ant, os algoritmos possuem uma variação muito grande no valor obtido na pós-evolução. Essa condição começa a mudar no grupo com 5 nichos, onde LA-Es, FixedSeed-NE-SP e LA-SP possuem mais valores concentrados em uma região. Já no grupo com 10 nichos, os valores são bem concentrados, fazendo com que as *boxs* fiquem menores.

A fim de comparação, o algoritmo RandSeed-NE foi utilizado seguindo as configu-

rações do grupo que contém 3 nichos. Seus resultados de pós-avaliação foram reunidos e comparados aos obtidos anteriormente pelos outros algoritmos, como consta na Figura 18.

Figura 18 – Performance dos agente evoluídos no ambiente Ant no grupo contendo 3 nichos. Os algoritmos presente são os já utilizados, apenas sendo adicionado o algoritmo RandSeed-NE. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente.



Fonte: Desenvolvido pelo autor

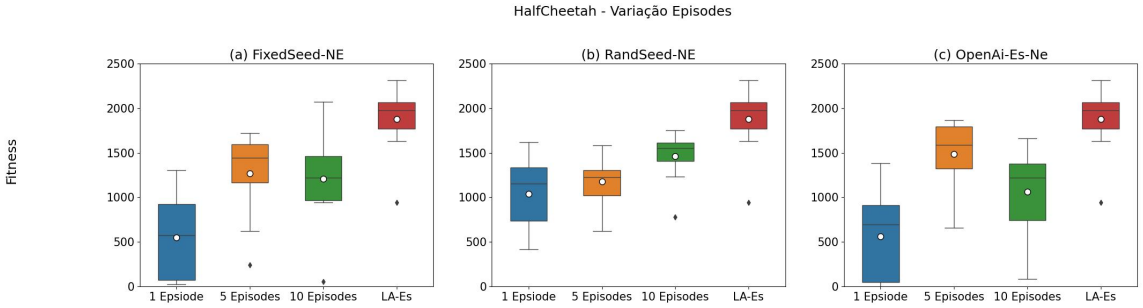
Foi constatado que, para o ambiente Ant, com 3 nichos, o algoritmo RandSeed-NE apresenta ótimos resultados, tendo bons valores quando comparado ao OpenAI-Es-Ne, ao FixedSeed-NE e até mesmo o LA-Es. Sua variação dentro dos valores de *fitness* é baixa, contendo o maior valor dos mínimos apresentados pelos algoritmos, perdendo apenas para o LA-SP.

Com os bons resultados de RandSeed-NE e FixedSeed-NE-SP com 3 nichos, foi também analisado sobre o aumento na quantidade de episódios de avaliação, resultando em uma maior variação ambiental. Os valores pós-avaliados estão presentes na Figura 19.

Primeiramente, pode ser observado que não há uma diferença considerável quando comparamos FixedSeed-NE e OpenAi-Es-NE, isso demonstra que, para o ambiente Ant com 3 nichos, a variação da duração de episódios sozinha não altera significativamente o valor do *fitness* final, mesmo aumentando o número de episódios.

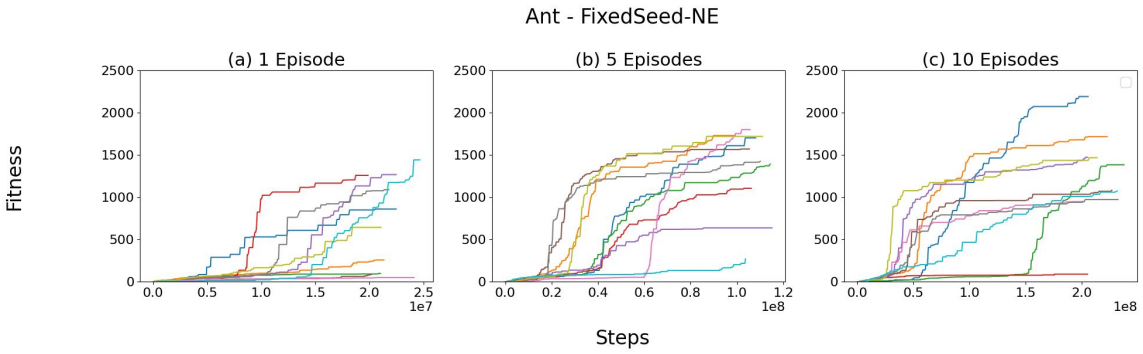
Quando observamos os algoritmos separadamente, o FixedSeed-Ne apresenta seus melhores resultados quando utilizado com 5 episódios, assim como foi mostrado no ambiente Hopper na Figura 14. Já o algoritmo RandSeed-NE apresenta seus melhores resultados com 10 episódios, onde sua variação é baixa, tendo sua *box* compacto. Em relação

Figura 19 – Performance dos agentes evoluídos no ambiente Ant no grupo contendo 3 nichos, variando a quantidade de episódios em 1, 5 e 10. Os algoritmos escolhidos foram o (a)FixedSeed-NE, (b)RandSeed-NE e (c)OpenAi-Es-Ne, todos sendo comparados ao melhor resultado de LA-Es. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente.



Fonte: Desenvolvido pelo autor

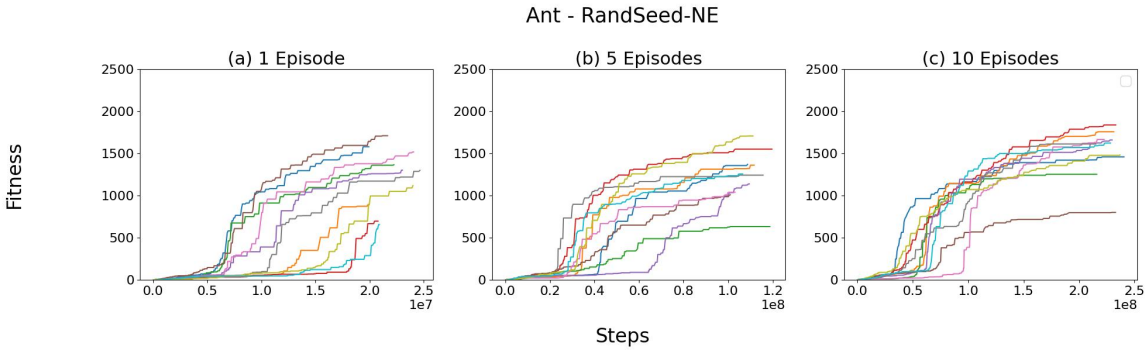
Figura 20 – Performance das replicações do algoritmo FixedSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Ant. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução.



Fonte: Desenvolvido pelo autor

a sua mediana, apresenta o melhor valor entre os três, 1, 5 e 10. E, por fim, o algoritmo OpenAi-Es-Ne apresenta características semelhantes ao algoritmo FixedSeed-NE, como já mencionado anteriormente. Apesar dos bons valores em todos os algoritmos, nenhum apresentou valores melhores que os obtidos a partir do algoritmo LA-ES.

Figura 21 – Performance das replicações do algoritmo RandSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente Ant. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução.



Fonte: Desenvolvido pelo autor

Ao analisarmos mais profundamente como os algoritmos se comportam durante a evolução e como a variação no número de episódios influencia, pode ser observado que, no FixedSeed-NE, apenas com 5 episódios de avaliação o algoritmo possui um bom começo e mantém o *fitness* crescendo à medida que o robô vai evoluindo. Quando observamos o RandSeed-NE, todos apresentam bom começo e mantêm uma crescente em relação ao valor de *fitness*, porém, com 10 episódios, mostra uma uniformidade entre todas as vezes em que o algoritmo foi rodado, sempre alcançando bons resultados.

Para uma melhor análise, os dados de mediana, média e desvio padrão foram coletados e colocados em uma tabela. Para comparação, os algoritmos presentes na tabela são o OpenAi-Es-Ne, FixedSeed-NE, RandSeed-NE e LA-Es.

Algoritmo	**Mediana**	**Média**
OpenAiEs-Ne	1587.67	1486.04 ± 376.07
FixedSeed-NE	1440.70	1265.94 ± 457.13
RandSeed-NE	1551.32	1460.94 ± 269.85
*LA-Es	1972.75	1875.98 ± 370.82

Tabela 5 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Ant. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. *O algoritmo LA-Es apresentou significância estatística em relação ao FixedSeed-NE e ao RandSeed-NE.

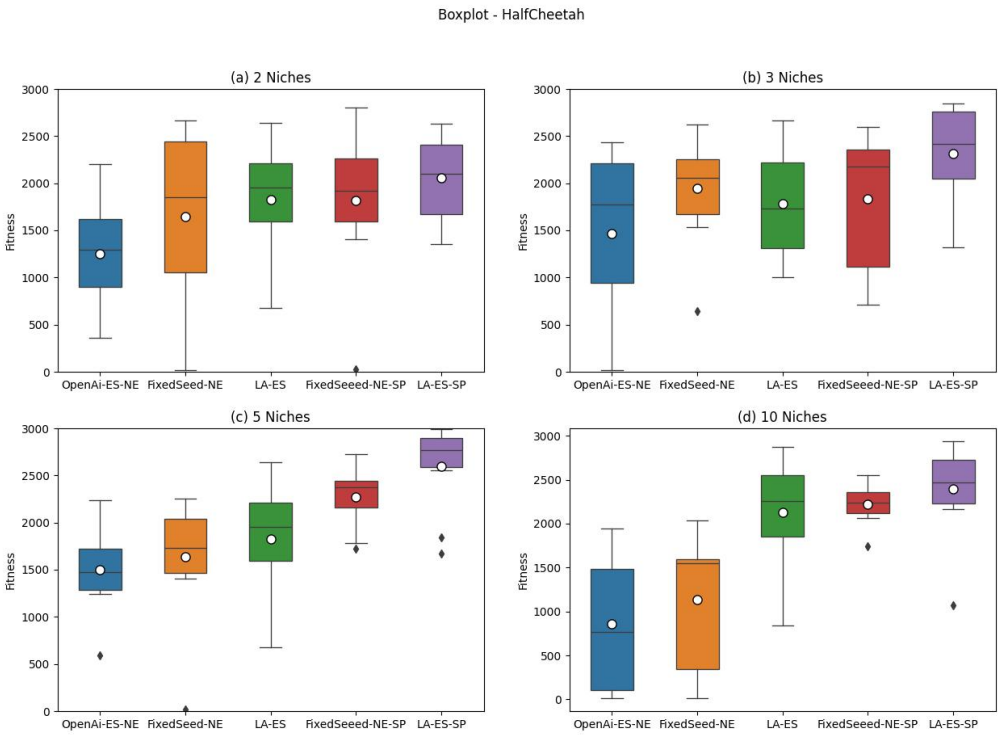
Com a ajuda da tabela, pode ser observado que o algoritmo LA-Es apresenta uma grande diferença entre o segundo colocado tanto de mediana, quanto de média. Em termos de porcentagem, a mediana do LA-Es apresenta uma diferença de 24% e a média apresenta uma diferença de 26%. Quando comparado aos novos algoritmos, essa diferença

só aumenta, sendo a mediana 36% superior ao FixedSeed-NE e 27% ao RandSeed-NE, e a média 48% superior ao FixedSeed-NE e 28% ao RandSeed-NE.

4.2.3 HalfCheetah

Os primeiros experimentos feitos no ambiente HalfCheetah foram utilizadas as configurações estabelecidas para cada grupo contendo todos os cinco algoritmos. Os resultados obtidos foram plotados em gráficos, resultando na Figura 22.

Figura 22 – Performance dos agentes evoluídos em quatro diferentes grupos no ambiente HalfCheetah, onde cada grupo apresenta 5 diferentes algoritmos evolutivos. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente. A Figura (a), (b), (c) e (d) apresentam os algoritmos com 2, 3, 5 e 10 nichos e suas equivalências



Fonte: Desenvolvido pelo autor

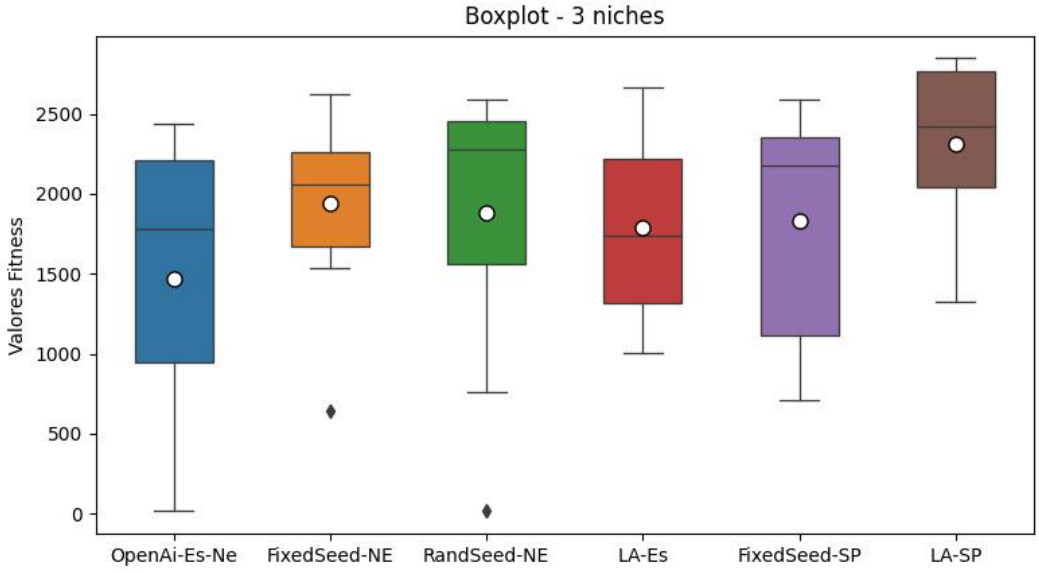
Comparando primeiro os algoritmos que apresentam nichos, o algoritmo que não possui a variação da duração, OpenAi-Es-NE, apresenta medianas consideravelmente menores em relação ao que possui a variação, FixedSeed-NE, podendo ser observado em todos os grupos. Já quando comparamos os dois algoritmos que apresentam a variação da duração de episódio, o algoritmo que não possui nichos, LA-Es, apresenta valores melhores à medida que a quantidade de nichos aumenta em relação ao que possui nichos, FixedSeed-NE, podendo ser observado nos grupos de 5 e 10 nichos. O melhor resultado apresentado

pelo algoritmo FixedSeed-NE foi utilizando 3 nichos, onde apresentou mediana melhor do que os outros algoritmos da comparação. De um modo geral, os valores apresentados pelos algoritmos não possuem uma variação muito grande, sendo o grupo com 10 nichos o único que apresenta mais de um algoritmo com grande variação.

Em relação ao *budget* computacional, os que contêm 2 nichos não apresentaram melhores resultados da *fitness* quando comparado às sua versões com menos poder computacional, porém, ao aumentar a quantidade de nichos, os algoritmos com maior poder computacional apresentam cada vez mais resultados que superam suas versões mais básicas, como mostra na Figura 23, mostrando a influência do *budget* a medida que a quantidade de nichos aumenta. Quando observamos apenas os dois algoritmos com maior *budget*, FixedSeed-NE-SP e LA-Es-SP, as suas diferenças se mantêm constantes, sendo o grupo com 5 nichos o que apresenta maior discrepância entre os resultados, além de ser o grupo onde o LA-SP apresenta o melhor resultado, tendo sua mediana perto de 3000.

A partir desses resultados, foi selecionado o grupo com 3 nichos, o qual apresenta o melhor resultado do FixedSeed-NE, sendo superior ao LA-Es, para conter o RandSeed-NE. O resultado do algoritmo, junto com os anteriores, está presente na Figura 23.

Figura 23 – Performance dos agentes evoluídos no ambiente HalfCheetah no grupo contendo 3 nichos. Os algoritmos presentes são os já utilizados, apenas sendo adicionado o algoritmo RandSeed-NE. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente.



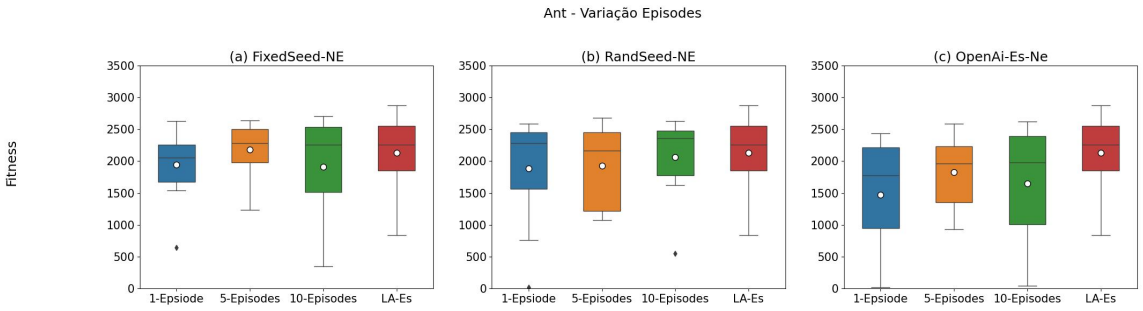
Fonte: Desenvolvido pelo autor

O resultado obtido demonstra que o algoritmo RandSeed-NE apresenta bons resultados perante os demais. Quando o comparamos com os que possuem *budget* igual,

apresenta a mediana superior que todos os outros, até mesmo da sua variação sem a troca das condições iniciais, FixedSeed-NE. Há também a possibilidade da comparação com os que apresentam *budget* computacional superior, tendo uma mediana superior ao FixedSeed-NE-SP, mas com valor menor em relação ao LA-SP.

Outro estudo a ser realizado é relacionado à variação das condições iniciais do ambiente. Com estes bons resultados, foi escolhido o grupo que contém 3 nichos. Os algoritmos FixedSeed-NE, RandSeed-NE e OpenAi-Es-NE foram avaliados com as diferentes quantidades de episódios e apresentaram os resultados como no gráfico da Figura 24.

Figura 24 – Performance dos agentes evoluídos no ambiente HalfCheetah no grupo contendo 3 nichos, variando a quantidade de episódios em 1, 5 e 10. Os algoritmos escolhidos foram o (a)FixedSeed-NE, (b)RandSeed-NE e (c)OpenAi-Es-Ne, todos sendo comparados ao melhor resultado de LA-Es. Dados obtidos pós-evoluindo o melhor candidato 1000 vezes, no qual o estado inicial do robô era variado. As *box* representam os valores dos *fitness* finais, além da média e mediana. Cada *box* é resultado do *fitness* obtido durante 10 replicações de cada algoritmo correspondente.

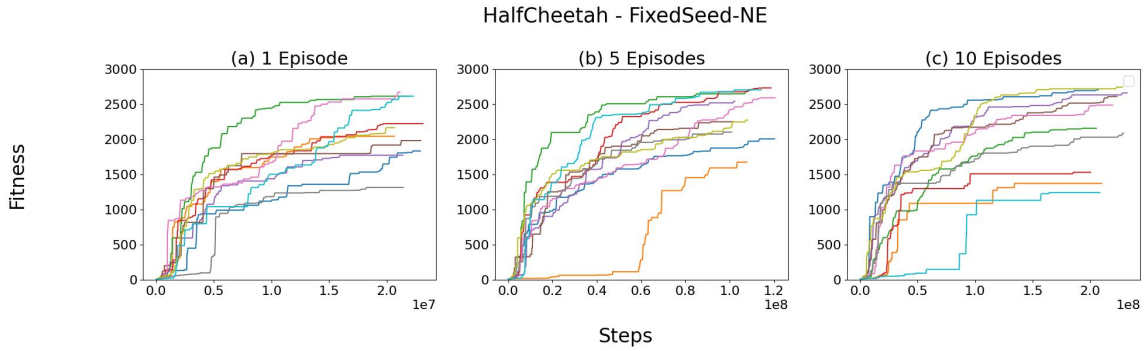


Fonte: Desenvolvido pelo autor

Como no ambiente Ant, apenas um algoritmo apresentou valores melhores do que o LA-Es, o RandSeed-NE. Esse algoritmo já apresentava valores superiores, mantendo uma mediana superior com 10 episódios, porém, com 5 episódios, houve uma piora. Em relação aos outros algoritmos, apresentaram comportamentos semelhantes, apenas se diferenciando na dispersão dos valores de fitness, onde o FixedSeed-NE resulta em *boxs* mais compactos e o OpenAi-Es-NE apresenta *boxs* mais dispersos. Os dois não conseguiram obter medianas que ultrapassassem o do algoritmo LA-Es. Para analisar melhor o comportamento do valor de *fitness* ao decorrer da evolução, os 10 experimentos de cada algoritmo foram plotados e estão nas Figuras 25 e 26.

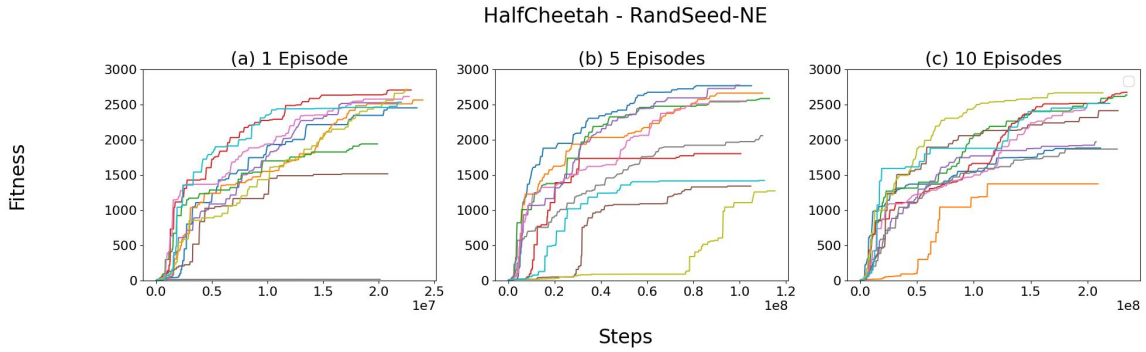
No algoritmo FixedSeed-NE, para 1 episódio, os experimentos apresentam uma elevação no valor da *fitness* inicialmente pequena, fazendo com que o *fitness* de cada experimento atinja valores menores. Para 5 episódios, o aumento inicial no valor da *fitness* acontece em conjunto e dura mais do que em relação a 1 episódio apenas. E por fim, com 10 episódios é onde acontece o maior aumento inicial, porém, os experimentos não melhoram da mesma maneira, fazendo com que os valores finais fiquem mais espalhados.

Figura 25 – Performance das replicações do algoritmo FixedSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente HalfCheetah. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução.



Fonte: Desenvolvido pelo autor

Figura 26 – Performance das replicações do algoritmo RandSeed-NE tendo (a)1, (b)5 e (c)10 episódios de avaliação e 3 nichos no ambiente HalfCheetah. Cada um dos três gráficos contém 10 linhas onde cada linha representa uma replicação. As performances são relacionadas aos valores de *fitness* obtidos ao decorrer da evolução.



Fonte: Desenvolvido pelo autor

Já em relação ao algoritmo RandSeed-NE, com 1 episódio ele já apresenta bons valores, tendo uma evolução uniforme entre as experiências, fazendo com que a mediana seja mais alta e sua *box* seja mais compacto. Quando observamos os experimentos que utilizaram 5 episódios, pode ser concluido que as evoluções das *fitness* foram mais espalhadas, porém com grande evolução inicial, fazendo com que a *box* fique mais dispersa, mas com bons valores e mediana. Por último, os experimentos que utilizaram 10 episódios apresentaram uma boa uniformidade na evolução da *fitness* e bons resultados,ficando mais parelhos, tendo a melhor mediana de todas e a menor varição também.

A fim de se comparar as médias e medianas, os respectivos valores foram retirados da Figura 24 e colocados na Tabela 6.

Como pôde ser observado na Figura 23, o algoritmo RandSeed-NE apresenta a maior mediana de todos, com valor de 2362,03, sendo 3,3% melhor que a segunda melhor

Algoritmo	**Mediana**	**Média**
OpenAiEs-Ne	1959.24	1824.25 ± 544.67
FixedSeed-NE	2283.81	2177.39 ± 417.74
RandSeed-NE	2362.03	2066.10 ± 614.64
LA-Es	2253.67	2132.60 ± 633.49

Tabela 6 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente HalfCheetah. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio.

mediana, FixedSeed-NE e 4,6% melhor do que LA-Es. Quando comparamos a média, o algoritmo que apresenta melhores resultados é o FixedSeed-NE, obtendo valor de 2177.39, sendo 2,05% melhor que o segundo colocado, LA-Es e 5,1% melhor que RandSeed-NE.

5 CONCLUSÃO

O presente trabalho se propôs a desenvolver novos algoritmos evolutivos que unem a variação na duração de um episódio e variação ambiental utilizando nichos, a fim de evoluir a rede neural de um robô. Ademais, buscou-se entender melhor quais aspectos mais influenciavam na hora da evolução e quais alterações poderiam ser feitas para uma melhora.

Ao longo deste trabalho, foram aprofundados conceitos fundamentais de robótica evolutiva, permitindo uma investigação otimizada dos conceitos de algoritmos evolutivos e redes neurais, com o objetivo de apresentar melhorias e mudanças na evolução do comportamento de robôs. Os resultados alcançados neste estudo atingiram o objetivo de encontrar respostas e soluções para os desafios enfrentados durante o desenvolvimento e implementação, mesmo considerando a natureza inovadora da robótica evolutiva.

Primeiramente foi apresentado um agrupamento contendo um primeiro algoritmo proposto e outros algoritmos evolutivos, sendo eles o OpenAi-Es-Ne e o La-Es, os quais apresentam características parecidas com o proposto. O primeiro algoritmo da comparação, OpenAi-Es-Ne, apresenta a variação ambiental através de nichos e o segundo algoritmo da comparação, La-Es, apresenta a variação na duração do episódio de avaliação. Dentro de cada grupo, foram selecionadas condições diferentes para cada algoritmo, a fim de performá-los variando quesitos específicos como número de nichos e a duração dos episódios de avaliação, e até, o conjunto dos dois.

Com as configurações prévias, foi então necessário aplicá-los em ambientes de simulação já conceituados. Houve um estudo mais aprofundado para entender como realmente os ambientes funcionavam, seus propósitos, suas condições, além de seu resultado para as redes neurais. O objetivo geral desses ambientes era comum, a locomoção, variando apenas como a locomoção era executada e incentivada.

Neste trabalho, outras condições também foram estudadas para entender se limitavam ou melhoravam o primeiro algoritmo criado. Essas condições foram o aumento do *budget* computacional, que eleva o poder computacional na hora da evolução, e a constante variação nas condições iniciais do ambiente de simulação, que, junto à variação já existente nos nichos, aumenta ainda mais as condições experienciadas pelos robôs nos ambientes, obtendo assim uma melhora na performance realizada.

Com o bom desempenho da constante variação nas condições iniciais, foi então criado o segundo algoritmo, contendo variação na duração de um episódio, variação ambiental e o aumento na variação ambiental através da variação nas condições iniciais ao decorrer da evolução. Este novo algoritmo apresentou melhores resultados quando comparado à sua primeira versão, fazendo com que outros novos experimentos pudessem ser realizados. Esses novos experimentos foram em relação a aumentar ainda mais as diferentes condições experienciadas pelos robôs, aumentando o número de vezes que interagiam com

o ambiente de simulação. Como as condições mudavam constantemente antes de cada iteração, quanto maior o número de episódios, maior era o número de diferentes condições em que o robô entrava em contato. Isso foi de extrema necessidade, já que ajudou a entender ainda mais a relação entre um bom desempenho de um robô com as diferentes condições experienciadas.

Por fim, percebeu-se que a união da variação ambiental e a variação na duração de episódios influenciam diretamente em um bom algoritmo evolutivo. Utilizando essas condições, houve um aumento considerável na evolução da rede neural dos robôs em dois dos três ambientes utilizados. Comparado ao melhor resultado do melhor algoritmo, La-Es, no ambiente Hopper o algoritmo RandSeed-NE apresentou uma melhora de 12% na mediana e no ambiente HalfCheetah, apresentou uma melhora de aproximadamente 5% na mediana. Já no ambiente Ant, o algoritmo apresentou uma piora de 22%. Mesmo com esses resultados, ainda é necessária uma constante evolução no campo da robótica evolutiva, buscando o aprimoramento e entendimento para que cada vez mais, através de pesquisas acadêmicas, possamos aplicá-lo na sociedade para um futuro mais tecnológico.

6 TRABALHOS FUTUROS

Ao longo deste trabalho, foram identificadas possíveis mudanças na proposta. Por isso, considerando manter o projeto em um nível viável, essas ideias foram documentadas e detalhadas abaixo para serem implementadas em trabalhos futuros.

Primeiramente, um dos projetos que pode ser feito é relacionado à adição de outros tipos de ambientes de simulação, tanto na quantidade de robôs que utilizam locomoção, como com outros tipos robôs. Esse aumento na quantidade faz com que outros possíveis estudos possam ser realizados, a fim de entender se os algoritmos propostos possam ser melhores em outras modalidades de robôs. Além desse, outro estudo a ser feito é aumentar o *budget* computacional no algoritmo RandSeed-NE para que se obtenha uma maior equivalência dos nichos em relação aos outros algoritmos. Por fim, também pode ser analisado mais profundamente o melhor robô em cada geração. Isso poderá ajudar a entender como melhorar ainda mais os algoritmos propostos e, caso necessário, alterá-los.

Apêndices

APÊNDICE A – ARTIGO SBC

Impacto da duração de episódios de avaliação em robótica evolutiva usando técnicas de nicho

Davi B. da Silva, Jônata T. Carvalho

[1]Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
CEP: 88040-370 – Florianópolis – SC – Brasil

davi.becker@hotmail.com

Abstract. *With the advancement of technology, robots are increasingly present in our daily lives. The development of robots presents great difficulties, one of which is the creation of behavior. A robot's behavior can be developed using algorithms that are bio-inspired by human evolution, called evolutionary algorithms. Within evolutionary algorithms, robots develop their neural networks in order to perform a certain behavior. Two aspects are important for development, the environment in which the robot will interact and the time in which the robot interacts with the environment. Therefore, in this work there is a combination of the increase in the number of environments in niches and the variation in the duration of the evaluation episode.*
Keywords*: Adaptive Robotic. Niche Evolution. Evaluate Time.*

Resumo. *Com o avanço da tecnologia, robôs estão cada vez mais presentes no nosso dia-a-dia. O desenvolvimento dos robôs apresentam grandes dificuldades, sendo uma delas a criação de um comportamento. O comportamento de um robô pode ser desenvolvido utilizando algoritmos que tem a bio-inspiração na evolução humana, chamados de algoritmos evolutivos. Dentro dos algoritmos evolutivos, os robôs desenvolvem suas redes neurais a fim de desempenharem um certo comportamento. Dois aspectos são importantes para o desenvolvimento, o ambiente em que o robô irá interagir e o tempo em que o robô interage com o ambiente. Com isso, neste trabalho há a união do aumento do número de ambientes em nichos e a variação do tempo de duração do episódio de avaliação.*
Palavras-chave*: Robótica Adaptativa. Evolução de Nicho. Tempo de Avaliação.*

1. Introdução

A medida em que a tecnologia continua a se desenvolver rapidamente, robôs têm se tornado cada vez mais presentes no nosso dia-a-dia. Podem ser encontrados dentro de fábricas e outras áreas da indústria, exercendo papéis perigosos ou difíceis para seres humanos. Além disso, também estão começando a aparecer mais em ambientes domiciliares e hospitalares, um exemplo são os robôs aspiradores de pó, que são capazes de operar automaticamente por horas.

Segundo Nolfi [Nolfi et al. 2016] [Nolfi 2021] robótica evolutiva é uma técnica para a criação de robôs autônomos, inspirada pelo princípio de seleção natural, feita por

Charles Darwin. A ideia por trás da robótica evolutiva funciona com uma geração inicial de candidatos, gerados de forma aleatória, que vão evoluindo com foco em um resultado previamente selecionado. Para que tudo isso seja possível, essa técnica utiliza algoritmos evolutivos para desenvolver, modificar e encontrar o melhor robô para a solução desejada.

Segundo bäck1996evolutionary, algoritmos evolutivos modelam o processo de aprendizado coletivo dentro de uma população de indivíduos, cada qual representando uma potencial solução. A população é inicializada por um método referente ao algoritmo, e evolui em direção a uma área que possa conter melhores resultados dentro do conjunto solução, por processos de recombinação, mutação e seleção. Assim como na natureza, o ambiente contribui na evolução de uma população, sendo um agente que influencia no comportamento dos indivíduos.

Os ambientes são os meios em que os indivíduos serão treinados e avaliados mediante a sua interação. Cada ambiente possui condições iniciais, como obstáculos, objetos e restrições diferentes. A relação entre indivíduo e ambiente é avaliada comparando o comportamento performado em relação ao comportamento desejado. Assim surgiu a ideia de criar diferentes ambientes, chamados de nichos, para que o robô possa experienciar vários ambientes.

Além da configuração do ambiente, o tempo em que cada indivíduo permanece interagindo também pode alterar a forma que ele evolui. Esse período de interação é conhecido como episódio de avaliação.

Neste trabalho houve a junção de duas das características que influenciam em uma evolução, variação ambiental e duração do episódio de avaliação.

2. Fundamentação teórica e trabalhos relacionados

2.1. Fundamentação teórica

A seguir, serão abordados os principais conceitos, que serão de extrema importância para entendimento do presente trabalho

2.1.1. Algoritmos Evolutivos

Algoritmos evolutivos são métodos artificiais que buscam reproduzir dinâmicas similares à evolução natural, muito embora não seja um de seus objetivos reproduzi-las fidedignamente. Buscam na evolução natural a bio-inspiração para desenvolver estratégias que permitam encontrar soluções satisfatórias, ou até ótimas, para um determinado conjunto de problemas. Utilizam processos comumente encontrados na natureza como reprodução, mutação, recombinação e seleção.

Nesses algoritmos, um conjunto de possíveis soluções a ser testado para um determinado problema é chamado de população ou candidatos. Dentro dessa população, cada solução é chamada de indivíduo, genótipo ou cromossomo. Dentro dessas soluções há genes, que são parâmetros que constituem uma parte da solução. Cada indivíduo recebe uma nota de acordo com uma avaliação, conhecida como *fitness*. Essa nota é quem dita quais indivíduos devem permanecer e quais devem ser eliminados.

Após as primeiras possíveis soluções serem avaliadas, são escolhidas as melhores para serem parentes da próxima geração. Há duas maneiras de gerar novos candidatos, a mutação e a recombinação. A mutação acontece quando um dos parentes é selecionado e alterado os valores do seu gene. A recombinação, também conhecida como *cross-over*, acontece quando é selecionada uma divisão entre os dois parentes e combinado partes dos seus genótipos sempre buscando um maior *fitness*.

2.1.2. Estratégia Evolutiva

Estratégia evolutiva, conhecida como *Evolution strategy* (ES), é uma técnica de otimização global *black-box* que pertence à classe dos algoritmos evolutivos. Esse tipo de algoritmo se diferencia dos demais em relação a como a população é representada e como a mutação e recombinação são efetuadas, além de poder ser aplicado em todos os campos de otimização incluindo contínuo, discreto, entre outras [Beyer 2007].

Um termo muito utilizado, e de grande importância, é o *search space*. *Search space* é um conjunto de soluções possíveis de um problema de otimização que satisfazem suas restrições. Esse conjunto solução é uma estrutura de dados que pode ser finita mas não necessariamente com tamanho fixo. Um exemplo é o *search space* de números reais com N-dimensões, ou um *search space* dos números inteiros, ou *search space* dos binários, ou até mesmo uma mistura de espaços e subespaços [Beyer and Schwefel 2002].

Existem duas formas padrão desse algoritmo (μ/ρ,λ)-ES e (μ/ρ+λ)-ES, onde μ denota a quantidade de pais, ρ denota a quantidade de pais envolvidos na criação de novos indivíduos, *offspring*, e λ o número de *offspring*. Os novos pais são selecionados deterministicamente ou dentro dos *offspring*, na primeira forma do algoritmo (*comma-selection*) onde (μ ¡ λ), ou entre os antigos pais e os *offspring*, na segunda forma do algoritmo (*plus-selection*).

Cada indivíduo da população é constituído por *a:=(y, s, F(y))*, onde y é o vetor de parâmetros a serem otimizados, s é um conjunto de estratégias e F(y) o *fitness* referente ao *search space*.

2.1.3. Redes Neurais

Redes neurais, também conhecidas como rede neural artificial (ANN), são modelos computacionais inspirados em um sistema nervoso, onde são capazes de realizar o aprendizado de máquina, bem como o reconhecimento de padrão.

Redes neurais artificiais [Nolfi 2021] são constituídas por unidades computacionais simples chamadas de neurônios, interconectados por ligações com peso, chamadas conexões. Neurônios produzem como *output* um valor real, que é o resultado de uma função, onde as conexões com outros neurônios servem como *input*, ou em caso de neurônios sensoriais, os *input* são sensores. O valor do *input* recebido pela conexão depende do *output* do neurônio com que a conexão se origina, neurônio pré sináptico, e o peso da conexão, que é relativa a um número real. As conexões podem aumentar ou diminuir o valor de saída do neurônio pré sináptico, e são referenciadas como conexões excitatórias e inibitórias, respectivamente.

A arquitetura de uma rede neural é estruturada de acordo com a maneira que os neurônios são conectados. Geralmente os neurônios são organizados em níveis ou camadas: o nível sensorial, um ou mais níveis internos e o nível de motores. Em uma arquitetura *feedforward*, cada neurônio de uma camada é conectado com os outros neurônios da próxima camada. A informação possui um fluxo unidimensional que vai dos neurônios sensoriais até os neurônios motores. Outra arquitetura é a rede neural recorrente, nessa proposta os neurônios podem se conectar com neurônios da mesma camada ou até da camada anterior.

2.1.4. Neuroevolution

Neuroevolução, ou *neuroevolution*, é uma técnica que aplica algoritmos evolutivos para construir redes neurais artificiais, tendo como inspiração a evolução natural do sistema nervoso [Lehman and Miikkulainen 2013]. Diferente dos demais métodos, a neuroevolução permite aprendizados sem um foco explícito, apenas com *feedbacks* e uma rede neural estruturada qualquer. É muito utilizada nos campos de robótica evolutiva e *artificial life*.

Um dos grandes objetivos da *neuroevolution* é o de evoluir complexas redes neurais capazes de desenvolver um comportamento inteligente. Como resultado disso, pode ser utilizado tanto para estudar como um comportamento inteligente evolui na natureza, quanto para fazer com que redes neurais desempenhem uma tarefa específica.

Nessa técnica que envolve redes neurais e algoritmos evolutivos, há uma diferença na hora de avaliar os genótipos, que é a adição de um novo passo. Cada genótipo contém os pesos das conexões de uma rede neural, chamada de fenótipo. Cada fenótipo contém entradas e saídas que, no caso da robótica, podem ser dados de sensores e dados de motores respectivamente.

Os fenótipos são utilizados pelos robôs. Utilizam a rede neural para exercer o comportamento e retornam seus *fitness*, assim segue o algoritmo evolutivo. A evolução que acontece no fenótipo é relacionada tanto a topologia, quantidade de neurônios e quais suas conexões, tanto em relação ao peso de cada conexão, a fim de obter um resultado desejado.

2.1.5. Robótica Evolutiva

Robótica evolutiva (ER) é um dos campos da robótica que procura criar robôs autônomos com maior robustez e adaptabilidade [Doncieux et al. 2015]. Essa subdivisão aplica conceitos de seleção, variação e princípios hereditários de evolução natural.

ER utiliza algoritmos evolutivos para desenvolver comportamentos, ou também a morfologia robótica, a fim de obter um comportamento desejado pelo projetista. Para obter o *fitness* de cada indivíduo, é necessário colocá-lo em um ambiente, seja ele real ou virtual, com condições iniciais predeterminadas e deixá-lo interagir. O que será avaliado é o comportamento do robô mediante as condições encontradas no ambiente. O valor resultante é relacionado ao quão bem o robô performou seu objetivo. Esse valor, então, retorna ao algoritmo evolutivo para seguir o ciclo.

2.2. Trabalhos relacionados

A seguir, serão abordados alguns dos trabalhos relacionados a este, contendo resumos e pontos relevantes destes artigos.

2.2.1. On the impact of the duration of evaluation episodes on the evolution of adaptive robots

Este artigo tem o intuito de demonstrar a relação entre o tempo de duração de um episódio de avaliação e o resultado do *fitness* de um robô.

Este estudo utiliza, como algoritmos evolutivo o OpenAi-Es [Salimans et al. 2017], o simulador Evorobotpy2 e quatro variações de ambientes da ferramenta PyBullet. Cada ambiente possui um agente diferente, mas necessitam produzir o mesmo comportamento, dirigir-se a uma localização o mais rápido possível. Os agentes são o Hopper, formado por 4 segmentos que simbolizam um perna que anda pulando, Ant, formado por 4 pernas conectados a um corpo, Walker2D, formado por 2 pernas conectadas a um torso e Humanoid, formado por duas pernas, dois braços, um torso e uma cabeça.

O tempo de duração dos episódios são representados por um valor de *steps* e foram divididos da seguinte forma: número padrão com valor de 1000 *steps*, um número fixo, o qual foi repetido com 100, 200, 300, 400 e 500, dois números fixos sendo um a primeira metade da evolução com valores 100, 200, 300, 400 e 500, e a outra metade com 1000, um número incremental que inicializa em 100 e cresce de 100 em 100 a cada 10% da avaliação, uma condição específica onde o número é fixo em 1000 e a função *fitness* é otimizada, e por último um valor aleatório com intervalo de [50-1000] com distribuição uniforme.

O resultado demonstra que o tempo de duração dos episódios está diretamente ligado ao valor da *fitness* de um indivíduo. Foi observado que a variação do tempo, ou começando com valores menores e aumentando a medida que o processo evolutivo acontece ou metade do processo com um valor menor e a outra metade com um valor maior, obteve melhores resultados do que os outros padrões utilizados.

2.2.2. Favoring the Evolution of Adaptive Robots Through Environmental Differentiation

Neste artigo foi discutido como a variação ambiental facilita a evolução de melhores possíveis soluções para o problema da navegação *double-pole*. Além disso, é proposto um novo algoritmo evolutivo baseado em nichos.

Da mesma maneira que mutações geram novidades genéticas, a variação do ambiente também gera, porém com maior importância. Isso pode ser observado a partir de alguns fatores como: o ambiente afeta imediatamente um grande número de indivíduos, o ambiente gera modificações imediatas e persistentes, entre outros.

O experimento foi separado em duas partes: a primeira foi que 20 agentes foram avaliados em condições iguais de ambiente, ou seja, as condições iniciais eram iguais,

e outros 20 agentes foram avaliados colocados em 20 diferentes nichos com condições iniciais diferentes.

Os experimentos do mesmo ambiente utilizaram estratégia evolutiva (μ+1)-ES em que μ parentes geram 1 *offspring*, como mostra o algoritmo de ambiente homogêneo. Já os com condições diferentes utilizaram o novo algoritmo por eles criado, como mostra o algoritmo heterogêneo, utilizando (1+1).

O resultado obtido foi que agentes evoluídos em ambientes heterogêneos obtiveram melhores valores do que aqueles evoluídos em ambientes homogêneos, sendo a diferença entre eles estatisticamente relevante. Foi relatado também que o algoritmo criado para usar em ambientes heterogêneos possui performance melhor do que outros métodos.

2.2.3. The Effectiveness of Niching on OpenAI-Evolution Strategies in the Evolution of Robotic Behavior

Neste artigo foi aplicado o conceito de nicho, e suas variações, no algoritmo criado pela OpenAi, o OpenAi-ES, resultando no OpenAi-Es-NE.

O artigo demonstra os nichos como um conjunto de condições iniciais geradas aleatoriamente, onde o agente será avaliado. Para isso, é criado uma matriz para representá-los, sendo a coluna a quantidade de nichos e as linhas a quantidade de avaliações de um indivíduo. Cada valor da matriz é uma *seed*, um valor que representa um conjunto de condições iniciais dos parâmetros do problema selecionado, neste caso, o *double-pole*.

É criado também um vetor com tamanho da quantidade de pesos da rede neural, chamado de centróide.

A evolução funciona avaliando os centróides dentro de seus nichos e, se necessários, trocando de lugar. Essa troca acontece quando um centroide de outro nicho obtém valores melhores do que o centróide do próprio nicho. As trocas são feitas quando uma certa quantidade de gerações foram avaliadas, fazendo com que, se houver a troca, os centróides que exerceram a troca já obtiveram valores de *fitness* consideráveis.

Os resultados demonstram que a adição de nichos e a variação do ambiente alteram favoravelmente a performance do algoritmo, tendo um aumento na média dos resultados de aproximadamente 9%, considerando o problema do *double-pole*.

3. Desenvolvimento

Neste trabalho houve a junção da variação da duração do episódio de avaliação com a diferenciação ambiental através dos nichos em um único algoritmo, o qual possuirá duas versões, a fim de evoluir os pesos da rede neural de robôs em ambientes de simulação selecionados. Os novos algoritmos são baseados no OpenAi-Es-Ne [Bianchini et al. 2023], que já possui a diferenciação através dos nichos, porém, para adicionar a variação da duração, foi utilizada uma fórmula que divide proporcionalmente o tempo do episódio de avaliação. A diferença entre os novos algoritmos é relacionada às condições iniciais de cada nicho, onde um possui as condições iniciais fixas, FixedSeed-NE, e o outro varia essas condições ao decorrer da evolução, RandSeed-NE, a fim de aumentar as condições ambientais ao decorrer da evolução.

Episódios por Algoritmo	2 Nichos	3 Nichos	5 Nichos	10 Nichos
OpenAiEs-Ne	1000	1000	1000	1000
FixedSeed-NE	500 por nicho	333 por nicho	200 por nicho	100 por nicho
RandSeed-NE	500 por nicho	333 por nicho	200 por nicho	100 por nicho
LA-Es	500 + 500 a cada 50%	333 + 333 a cada 34%	200 + 200 a cada 20%	100 + 100 a cada 10%

Tabela 1. Relação da duração dos episódio em cada algoritmo

A fim de comparação, foram escolhidos dois algoritmos que possuem bons resultados e usam como base um algoritmo vastamente utilizado. Os algoritmos de comparação são o OpenAi-Es-Ne [Bianchini et al. 2023], algoritmo baseado no OpenAi-Es [Salimans et al. 2017], que possui a variação de nichos,mas não possui a variação da duração dos episódios. O outro algoritmo [Rosa et al. 2022] baseado também em OpenAi-Es, não possui nichos, mas possui a variação no tempo de avaliação, que iremos chamar de LA-Es. Além desses algoritmos, também será utilizada uma variação de um dos algoritmos deste trabalho, FixedSeed-NE, e do LA-Es, onde o poder computacional será aumentado.

Em relação a duração de um episódio de avaliação, os algoritmos apresentam ou um valor fixo de 1000, ou um valor variável com máximo de 1000. O algoritmo OpenAi-Es-Ne possui o número fixo, mas os algoritmos FixedSeed-NE, RandSeed-NE e o LA-Es possuem uma variação que pode, ou variar ao longo do tempo de avaliação, LA-Es, ou variar de acordo com cada nicho, FixedSeed-NE e RandSeed-NE. O algoritmo LA-Es possui uma variação que começa com um valor inicial e aumenta de acordo com uma porcentagem do tempo total de avaliação até chegar ao valor máximo. Já os algoritmos propostos, FixedSeed-NE e RandSeed-NE, utilizam-se de uma fórmula para dividir o tempo entre os nichos, que é definido por:

$$MaxSteps = (\xi/\mu).(\theta + 1) \tag{1}$$

Onde ξ representa a duração máxima de um episódio, μ representa a quantidade de nichos e θ o *index* atual do nicho. Assim, a duração dos episódios é dividida proporcionalmente, além de cada nicho conter um valor diferente.

Para investigar os efeitos da variação da duração dos episódios em relação à presença dos nichos, as experiências foram separadas em 4 grupos, onde cada grupo contém uma avaliação de cada algoritmo. Os algoritmos que utilizam nichos apresentam 2, 3, 5 e 10 nichos e 50 gerações para se obter a troca de nichos. Já o algoritmo LA-Es, que não utiliza nicho, utiliza uma duração de episódios equivalente. Quando comparado aos algoritmos com 2 nichos, LA-Es inicializa utilizando o valor de 500 e muda para 1000 quando chega em 50% da população. Quando comparado aos de 3 nichos, começa com 333 e aumenta 333 a cada 34%. Quando aos de 5 nichos, inicializa com 200 e aumenta 200 a cada 20%. Já quando comparado com os de 10, se inicializa em 100 e aumenta 100 a cada 10%. E, por fim, o algoritmo OpenAI-Es-NE utiliza um número fixo que dura toda a avaliação no valor de 1000.

Mediante as condições iniciais apresentadas, outras comparações devem ser feitas. Como cada algoritmo possui um tempo total de duração igual, os que apresentam nichos possuem um *budget* computacional menor, já que é dividido entre todos os nichos. Para obter uma igualdade neste tópico, os experimentos FixedSeed-NE e LA-Es também devem apresentar suas versões com o poder computacional equivalente, onde o *budget* atual será multiplicado proporcionalmente pelo número de nichos da comparação. Essas versões serão chamadas de FixedSeed-NE-SP e LA-SP.

Algoritmos	**2 Nichos**	**3 Nichos**	**5 Nichos**	**10 Nichos**
FixedSeed-NE-SP	500 por nicho	333 por nicho	200 por nicho	100 por nicho
LA-SP	500 + 500 a cada 50%	333 + 333 a cada 34%	200 + 200 a cada 20%	100 + 100 a cada 10%

Tabela 2. Relação da duração dos episódio nos algoritmos

Outro estudo que deve ser feito é em relação a diferentes condições iniciais experienciadas pelos robôs ao decorrer da evolução e sua relação com o valor de *fitness* final. Para isso, os algoritmos OpenAi-Es-Ne, FixedSeed-NE e RandSeed-NE, serão avaliados aumentando a quantidade de episódios durante a evolução, fazendo com que o RandSeed-NE possa experienciar uma maior quantidade de condições iniciais diferentes, assim, comparando os resultados obtidos dos três algoritmos com os melhores resultados do LA-Es. Para o cálculo final de *fitness*, é feito uma média onde o valor de cada episódio será somado e depois divido pela quantidade de episódios utilizados, porém, para se obter equivalência, o *budget* computacional é aumentado de acordo com o número de episódios utilizado pelo algoritmo, dando mais "tempo" aos robôs. As condições escolhidas para o estudo foram de utilizar 3 nichos e 1, 5 e 10 episódios.

Para os estudos, todos os experimentos necessários rodaram seus algoritmos 10 vezes em cada ambiente, isso significa que cada algoritmo utilizou 10 condições iniciais de ambientes diferentes. Após as avaliações, a melhor solução é pós-avaliada. A pós-avaliação funciona iniciando as condições do agente, como postura, direção das conexões ou direção das juntas, de forma aleatória e aplicando a rede neural evoluída em um ambiente com condições iniciais aleatórias, as quais diferem das condições do ambiente utilizadas para treinar a rede neural. Essa interação gera outro valor de *fitness*, que é utilizado para se comparar a eficiência de cada resultado. A pós-avaliação deve ser feita utilizando sempre o mesmo algoritmo para se manter um padrão, neste trabalho, é utilizado 1000 episódios de pós-avaliação e OpenAi-Es-Ne como algoritmo, a fim de se obter um valor mais preciso.

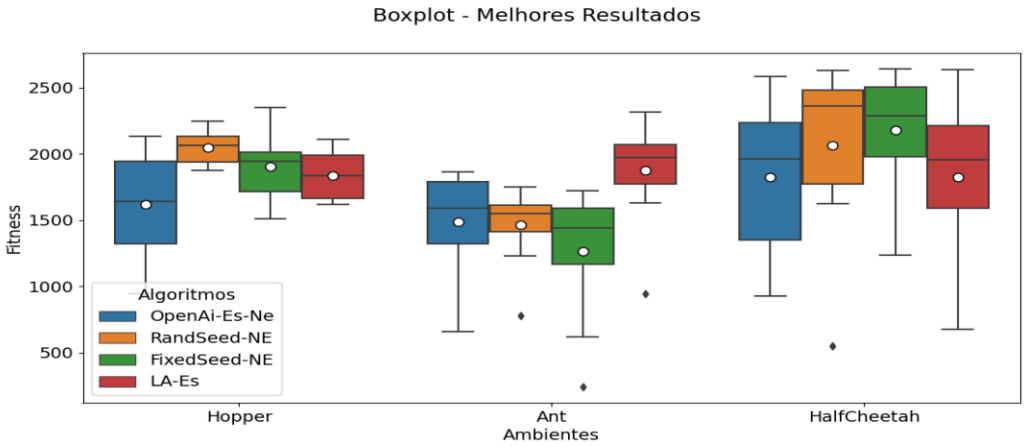
Para testar essas diferenças, foram escolhidos ambientes de simulação diferentes. Os ambientes são compostos por um robô, sua rede neural e um ambiente propriamente dito. Cada ambiente de simulação foi selecionado pelos bons valores apresentados em trabalhos e pesquisas anteriores, que usam alguns dos algoritmos utilizados como comparação neste trabalho. Tais ambientes são o Ant e Hopper, além de um ambiente não utilizado anteriormente, o HalfCheetah. Todos possuem uma rede neural do tipo *feed-foward* com 3 camadas e 50 neurônios internos, sendo sensores na primeira cada e a última camada é relacionada ao movimento do robô, nesses casos, a locomoção, controlando os atuadores.

4. Resultados e Discussões

Esta seção relaciona os experimentos efetuados com cada ambiente previamente selecionado e seus resultados apresentados. Todos os resultados obtidos foram plotados em gráficos utilizando *boxplot*, onde cada *box* representa os valores dentro dos quartis, as linhas horizontais pretas representam a mediana e os pontos brancos representam a média. As marcações acima e abaixo das *box*, no final das linhas verticais pretas, representam o melhor e pior valor respectivamente. Cada *box* representa o resultado obtido durante 10 replicações de cada experimento correspondente.

A imagem a seguir mostra os melhores resultados obtidos por todos os algoritmos bases em todos os ambientes de simulação selecionados, a fim de entender, de forma geral, os resultados obtidos.

Figura 1. Performance dos melhores agentes evoluídos nos ambientes selecionados. Cada *box* representa o valor da *fitness* dos agentes após a pós evolução. Suas condições de nichos e duração de episódios variam de acordo com os dados apresentados posteriormente.



Fonte: Desenvolvido pelo autor

Para o ambiente Hopper, os algoritmos apresentam seus melhores resultados no grupo que contém 3 nichos, obtendo resultados superiores ao LA-Es. O algoritmo RandSeed-NE apresenta seus melhores resultados com 10 episódios, o algoritmo FixedSeed-NE apresenta seus melhores resultados com 5 episódios e o algoritmo OpenAi-Es-Ne apresenta também com 5 episódios. Quando comparados os LA-Es, apresentam os resultados a seguir.

Para o ambiente Ant, os algoritmos apresentam seus melhores resultados no grupo que contém 3 nichos, obtendo resultados inferiores ao LA-Es. O algoritmo RandSeed-NE apresenta seus melhores resultados com 10 episódios, o algoritmo FixedSeed-NE apresenta seus melhores resultados com 5 episódios e o algoritmo OpenAi-Es-Ne apresenta também com 5 episódios. Quando comparados os LA-Es, apresentam os resultados a seguir.

Para o ambiente HalfCheetah, os algoritmos apresentam seus melhores resultados no grupo que contém 3 nichos, obtendo resultados superiores ao LA-Es. O algoritmo RandSeed-NE apresenta seus melhores resultados com 10 episódios, o algoritmo

Algoritmo	Mediana	Média
OpenAi-Es-Ne	1638.36	1616.01 ± 388.72
FixedSeed-NE	1946.13	1904.57 ± 237.88
*RandSeed-NE	2064.29	2045.59 ± 115.72
LA-Es	1837.38	1837.45 ± 175.12

Tabela 3. Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Hopper. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. *O algoritmo RandSeed-NE apresentou significância estatística em relação ao OpenAi-Es-Ne e ao LA-Es.

Algoritmo	Mediana	Média
OpenAiEs-Ne	1587.67	1486.04 ± 376.07
FixedSeed-NE	1440.70	1265.94 ± 457.13
RandSeed-NE	1551.32	1460.94 ± 269.85
*LA-Es	1972.75	1875.98 ± 370.82

Tabela 4. Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Ant. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio. *O algoritmo LA-Es apresentou significância estatística em relação ao FixedSeed-NE e ao RandSeed-NE.

FixedSeed-NE apresenta seus melhores resultados com 5 episódios e o algoritmo OpenAi-Es-Ne apresenta também com 5 episódios. Quando comparados os LA-Es, apresentam os resultados a seguir.

Algoritmo	Mediana	Média
OpenAiEs-Ne	1959.24	1824.25 ± 544.67
FixedSeed-NE	2283.81	2177.39 ± 417.74
RandSeed-NE	2362.03	2066.10 ± 614.64
LA-Es	2253.67	2132.60 ± 633.49

Tabela 5. Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente HalfCheetah. Os dados dos algoritmos OpenAi-Es-Ne e FixedSeed-NE foram utilizando 5 episódios, o algoritmo RandSeed-NE utilizando 10 episódios, e o algoritmo LA-ES utilizando 1 episódio.

Em relação ao *budget* computacional, pôde ser observado que influencia diretamente nos resultados obtidos, porém, outros algoritmos aproveitam de uma forma melhor esse aumento, como o LA-SP, utilizado neste trabalho.

5. Conclusão

Primeiramente foi apresentado um agrupamento contendo um primeiro algoritmo proposto e outros algoritmos evolutivos, sendo eles o OpenAi-Es-Ne e o La-Es, os quais apresentam características parecidas com o proposto. O primeiro algoritmo da comparação,

OpenAi-Es-Ne, apresenta a variação ambiental através de nichos e o segundo algoritmo da comparação, La-Es, apresenta a variação na duração do episódio de avaliação. Dentro de cada grupo, foram selecionadas condições diferentes para cada algoritmo, a fim de performá-los variando quesitos específicos como número de nichos e a duração dos episódios de avaliação, e até, o conjunto dos dois.

Com as configurações prévias, foi então necessário aplicá-los em ambientes de simulação já conceituados. Houve um estudo mais aprofundado para entender como realmente os ambientes funcionavam, seus propósitos, suas condições, além de seu resultado para as redes neurais. O objetivo geral desses ambientes era comum, a locomoção, variando apenas como a locomoção era executada e incentivada.

Neste trabalho, outras condições também foram estudadas para entender se limitavam ou melhoravam o primeiro algoritmo criado. Essas condições foram o aumento do *budget* computacional, que eleva o poder computacional na hora da evolução, e a constante variação nas condições iniciais do ambiente de simulação, que, junto à variação já existente nos nichos, aumenta ainda mais as condições experienciadas pelos robôs nos ambientes, obtendo assim uma melhora na performance realizada.

Com o bom desempenho da constante variação nas condições iniciais, foi então criado o segundo algoritmo, contendo variação na duração de um episódio, variação ambiental e o aumento na variação ambiental através da variação nas condições iniciais ao decorrer da evolução. Este novo algoritmo apresentou melhores resultados quando comparado à sua primeira versão, fazendo com que outros novos experimentos pudessem ser realizados. Esses novos experimentos foram em relação a aumentar ainda mais as diferentes condições experienciadas pelos robôs, aumentando o número de vezes que interagiam com o ambiente de simulação. Como as condições mudavam constantemente antes de cada iteração, quanto maior o número de episódios, maior era o número de diferentes condições em que o robô entrava em contato. Isso foi de extrema necessidade, já que ajudou a entender ainda mais a relação entre um bom desempenho de um robô com as diferentes condições experienciadas.

Por fim, percebeu-se que a união da variação ambiental e a variação na duração de episódios influenciam diretamente em um bom algoritmo evolutivo. Utilizando essas condições, houve um aumento considerável na evolução da rede neural dos robôs em dois dos três ambientes utilizados. Comparado ao melhor resultado do melhor algoritmo, La-Es, no ambiente Hopper o algoritmo RandSeed-NE apresentou uma melhora de 12% na mediana e no ambiente HalfCheetah, apresentou uma melhora de aproximadamente 5% na mediana. Já no ambiente Ant, o algoritmo apresentou uma piora de 22%. Mesmo com esses resultados, ainda é necessária uma constante evolução no campo da robótica evolutiva, buscando o aprimoramento e entendimento para que cada vez mais, através de pesquisas acadêmicas, possamos aplicá-lo na sociedade para um futuro mais tecnológico.

Referências

[Beyer 2007] Beyer, H. (2007). Evolution strategies. *Scholarpedia*, 2(8):1965. revision #199317.

[Beyer and Schwefel 2002] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies - a comprehensive introduction. *Natural Computing*, 1:3–52.

[Bianchini et al. 2023] Bianchini, A. H., Machado, B. S., and Carvalho, J. T. (2023). The effectiveness of niching on openai-evolution strategies in the evolution of robotic behavior. *GECCO '23 Companion*, 1:4.

[Doncieux et al. 2015] Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G. (2015). Evolutionary robotics: What, why, and where to. *Frontiers in Robotics and AI*, 2.

[Lehman and Miikkulainen 2013] Lehman, J. and Miikkulainen, R. (2013). Neuroevolution. *Scholarpedia*, 8(6):30977. revision #137053.

[Nolfi 2021] Nolfi, S. (2021). *Behavioral and Cognitive Robotics An adaptive perspective*. Institute of Cognitive Sciences and Technologies, National Research Council, Roma, Italy.

[Nolfi et al. 2016] Nolfi, S., Bongard, J., Husbands, P., and Floreano, D. (2016). Evolutionary robotics. In *Springer handbook of robotics*, pages 2035–2068. Springer.

[Rosa et al. 2022] Rosa, L. G., Homem, V. H., Nolfi, S., and Carvalho, J. T. (2022). On the impact of the duration of evaluation episodes on the evolution of adaptive robots. In Rudolph, G., Kononova, A. V., Aguirre, H., Kerschke, P., Ochoa, G., and Tušar, T., editors, *Parallel Problem Solving from Nature – PPSN XVII*, pages 520–529, Cham. Springer International Publishing.

[Salimans et al. 2017] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning.

APÊNDICE B – LISTA DE TABELAS

Tabelas de comparação de mediana, média e desvio padrão de todos os grupos dos ambientes Hopper, Ant e HalfCheetah.

B.1 HOOPER

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1353.54	1232.73 ± 503.41
FixedSeed-NE	1484.52	1378.62 ± 500.17
LA-Es	1343.15	1396.25 ± 598.52
FixedSeed-NE-SP	1426.91	1364.52 ± 548.94
LA-SP	1707.12	1657.90 ± 214.48

Tabela 7 – Média, Mediana e Desvio Padrão dos algoritmos contendo 2 nichos no ambiente Hopper.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1350.645	1378.64 ± 577.11
FixedSeed-NE	1386.63	1443.42 ± 431.45
LA-Es	1617.83	1433.81 ± 640.52
FixedSeed-NE-SP	1832.13	1748.46 ± 323.21
LA-SP	1889.22	1753.03 ± 315.57

Tabela 8 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Hopper.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1394.17	1416.98 ± 268.23
FixedSeed-NE	805.98	714.508 ± 505.80
LA-Es	1702.62	1712.69 ± 219.13
*FixedSeed-NE-SP	1614.71	1706.875 ± 254.56
*LA-SP	1674.91	1767.40 ± 318.02

Tabela 9 – Média, Mediana e Desvio Padrão dos algoritmos contendo 5 nichos no ambiente Hopper. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1181.67	1180.32 ± 170.54
FixedSeed-NE	1334.38	1242.80 ± 334.67
*LA-Es	1837.38	1837.45 ± 175.12
*FixedSeed-NE-SP	1788.48	1816.06 ± 242.99
*LA-SP	1908.50	1889.57 ± 216.20

Tabela 10 – Média, Mediana e Desvio Padrão dos algoritmos contendo 10 nichos no ambiente Hopper. *Os algoritmos LA-Es, LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE.

B.2 ANT

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	657.47	568.15 ± 413.11
FixedSeed-NE	805.98	714.51 ± 505.80
LA-Es	912.94	871.56 ± 399.65
FixedSeed-NE-SP	1034.43	876.55 ± 433.42
*LA-SP	986.55	1103.01 ± 348.62

Tabela 11 – Média, Mediana e Desvio Padrão dos algoritmos contendo 2 nichos no ambiente Ant. *O algoritmo LA-SP apresentou significância estatística em relação ao OpenAi-Es-NE.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	697.03	564.57 ± 473.96
FixedSeed-NE	576.65	551.36 ± 447.29
LA-Es	1059.13	897.50 ± 490.75
*FixedSeed-NE-SP	1484.59	1320.14 ± 449.34
*LA-SP	1336.67	1407.85 ± 361.54

Tabela 12 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente Ant. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	533.19	521.29 ± 465.81
FixedSeed-NE	800.65	739.83 ± 497.77
LA-Es	985.99	984.68 ± 378.39
*FixedSeed-NE-SP	1355.14	1346.79 ± 311.21
*LA-SP	1945.26	1944.20 ± 256.55

Tabela 13 – Média, Mediana e Desvio Padrão dos algoritmos contendo 5 nichos no ambiente Ant. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística sendo LA-SP em relação a todos e FixedSeed-NE-SP em relação ao OpenAi-Es-Ne e ao FixedSeed-NE.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	40.85	37.43 ± 16.59
FixedSeed-NE	124.97	240.76 ± 242.16
*LA-Es	1972.75	1875.98 ± 370.82
*FixedSeed-NE-SP	1239.83	1293.51 ± 204.71
*LA-SP	2231.16	2207.01 ± 202.46

Tabela 14 – Média, Mediana e Desvio Padrão dos algoritmos contendo 10 nichos no ambiente Ant. *Os algoritmos LA-Es, LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE.

B.3 HALFCHEETAH

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1294.78	1253.90 ± 526.20
FixedSeed-NE	1851.67	1647.29 ± 918.26
LA-Es	1953.98	1826.18 ± 591.38
FixedSeed-NE-SP	1924.20	1817.66 ± 721.16
*LA-SP	2100.45	2056.96 ± 444.24

Tabela 15 – Média, Mediana e Desvio Padrão dos algoritmos contendo 2 nichos no ambiente HalfCheetah. *O algoritmo LA-SP apresentou significância estatística em relação ao OpenAi-Es-NE.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1777.03	1468.80 ± 866.39
FixedSeed-NE	2056.92	1944.12 ± 557.96
LA-Es	1734.46	1786.22 ± 580.60
FixedSeed-NE-SP	2177.08	1833.50 ± 727.52
LA-SP	2419.01	2313.44 ± 493.74

Tabela 16 – Média, Mediana e Desvio Padrão dos algoritmos contendo 3 nichos no ambiente HalfCheetah.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	1478.09	1498.59 ± 437.00
FixedSeed-NE	1735.18	1635.29 ± 614.08
LA-Es	1953.98	1826.18 ± 591.38
*FixedSeed-NE-SP	2371.34	2271.46 ± 300.80
*LA-SP	2767.58	2598.85 ± 439.84

Tabela 17 – Média, Mediana e Desvio Padrão dos algoritmos contendo 5 nichos no ambiente HalfCheetah. *Os algoritmos LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE.

Algoritmo	**Mediana**	**Média**
OpenAi-Es-Ne	765.94	854.92 ± 730.38
FixedSeed-NE	1547.72	1134.78 ± 750.73
*LA-Es	2253.67	2132.60 ± 633.49
*FixedSeed-NE-SP	2239.67	2222.02 ± 212.66
*LA-SP	2466.62	2395.07 ± 510.26

Tabela 18 – Média, Mediana e Desvio Padrão dos algoritmos contendo 10 nichos no ambiente HalfCheetah. *Os algoritmos LA-Es, LA-SP e FixedSeed-NE-SP apresentaram significância estatística em relação ao FixedSeed-NE e ao OpenAi-Es-NE.

C REFERENCIAS

1. Auerbach, J. E., & Bongard, J. C. (2014). Environmental Influence on the Evolution of Morphological Complexity in Machines. PLOS Computational Biology, 10(1), 1-17. doi: 10.1371/journal.pcbi.1003399. Disponível em: https://doi.org/10.1371/journal.pcbi.1003399.

2. Braitenberg, V. (1986). Vehicles: Experiments in Synthetic Psychology. MIT Press. ISBN: 9780262521123. Disponível em: https://books.google.com.br/books?id=7KkUAT_q_sQC.

3. Lehman, J., & Miikkulainen, R. (2013). Neuroevolution. Scholarpedia, 8(6), 30977. DOI: 10.4249/scholarpedia.30977. (Revision #137053).

4. Silva, F., Correia, L., & Christensen, A. Lyhne. (2016). Evolutionary Robotics. Scholarpedia, 11(7), 33333. DOI: 10.4249/scholarpedia.33333. (Revision #168217).

5. Doncieux, S., Bredeche, N., Mouret, J.-B., & Eiben, A. E. (2015). Evolutionary Robotics: What, Why, and Where to. Frontiers in Robotics and AI, 2. DOI: 10.3389/frobt.2015.00004. ISSN: 2296-9144. Disponíveel em: https://www.frontiersin.org/articles/10.3389/frobt.2015.00004

6. Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. The International Journal of Robotics Research, 39(1), 3-20. SAGE Publications Sage Reino Unido: Londres, Inglaterra.

7. Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., Vanhoucke, V. (2018). Sim-to-real: Learning agile locomotion for quadruped robots. arXiv preprint arXiv:1804.10332.

8. Nolfi, S., Floreano, D., Miglino, O., Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In Artificial life iv: Proceedings of the fourth international workshop on the synthesis and simulation of living systems (pp. 190-197). MIT press.

9. Nolfi, S. (2022). Progress and challenges in adaptive robotics. Frontiers in Robotics and AI, 9, 1020462. doi: 10.3389/frobt.2022.1020462. Disponível em https://www.frontiersin.org/articles/10.3389/frobt.2022.1020462

10. Beyer, H. (2007). Evolution strategies. Scholarpedia, 2(8), 1965. Disponível em https://doi: 10.4249/scholarpedia.1965.

11. Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies - A comprehensive introduction. Natural Computing, 1(1), 3-52. Disponível em https://doi: 10.1023/A:1015059928466.

12. Bianchini, A. H., Machado, B. S., Carvalho, J. T. (2023). The Effectiveness of Niching on OpenAI-Evolution Strategies in the Evolution of Robotic Behavior. GECCO '23 Companion, 1, 4. Disponível em https://doi: 10.1145/3583133.3596341.

13. Rosa, L. G., Homem, V. H., Nolfi, S., Carvalho, J. T. (2022). On the Impact of the Duration of Evaluation Episodes on the Evolution of Adaptive Robots. In G. Rudolph, A. V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa, T. Tušar (Eds.), Parallel Problem Solving from Nature – PPSN XVII (pp. 520-529). Cham: Springer International Publishing. ISBN: 978-3-031-14714-2.

14. Nolfi, S., Bongard, J., Husbands, P., Floreano, D. (2016). Evolutionary Robotics. In Springer Handbook of Robotics (pp. 2035-2068). Springer.

15. Bäck, T. (1996). Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press.

16. Pigliucci, M., Murren, C. J., & Schlichting, C. D. (2006). Phenotypic plasticity and evolution by genetic assimilation. Journal of Experimental Biology, 209(12), 2362-2367. Disponível em https://doi:10.1242/jeb.02070.

17. Siciliano, B., & Khatib, O. (2016). Springer Handbook of Robotics (2nd ed.). Springer Cham, Nova Iorque, Nova Iorque.

18. Nolfi, S. (2021). Behavioral and Cognitive Robotics: An Adaptive Perspective. Institute of Cognitive Sciences and Technologies, National Research Council, Roma, Italia.

19. Abril, P. S., & Plant, R. (2016). The patent holder's dilemma: Buy, sell, or troll? Communications of the ACM, 2(27), Junho. doi: 978-3-319-32552-1. Disponível em https://doi.org/10.1007/978-3-319-32552-1

20. Karpathy, A., Salimans, T., Ho, J., Chen, P., Sutskever, I., Brockman, G., & Sidor, S. (2013). Maio. Acessado em 26 de julho de 2023, disponível em <https://openai.com/research/evolution-strategies>.