

# Survival of the Softest: Evolving Adaptive Soft Robots

Ana Carolina Morais<sup>1</sup>[2021222056] and Fernanda Fernandes<sup>2</sup>[2021216620]

<sup>1</sup> Departamento de Física, Universidade de Coimbra

<sup>2</sup> Departamento de Engenharia Informática, Universidade de Coimbra

**Abstract.** Este trabalho aborda o desenvolvimento de algoritmos evolutivos para projetar robôs flexíveis (soft robots) utilizando o ambiente de simulação EvoGym. Implementamos três abordagens distintas: (1) evolução da estrutura com um controlador pré-definido, (2) evolução do controlador para uma estrutura fixa, e (3) co-evolução simultânea de estrutura e controlador. Para a primeira e segunda tarefa, desenvolvemos dois algoritmos para cada. As experiências das várias tarefas foram realizados em diferentes ambientes, com análise estatística para comparar o desempenho das abordagens. Os resultados demonstram que na primeira tarefa o GA robusto superou significativamente a busca aleatória em 5 dos 6 cenários testados. Na segunda tarefa, dedicada à evolução de controladores, os algoritmos CMA-ES e DE apresentaram desempenhos estatisticamente semelhantes. Na coevolução, a abordagem híbrida GA+CMA-ES conseguiu convergir para soluções viáveis, embora com algumas limitações. Este estudo contribui para a compreensão da interação entre morfologia e controle em robôs flexíveis e ilustra a eficácia dos algoritmos evolutivos na otimização dessas estruturas complexas.

**Keywords:** Computação Evolutiva · Algoritmos Evolutivos · EvoGym · Otimização

## 1 Introdução

A robótica flexível (soft robotics) é uma abordagem inovadora que utiliza materiais deformáveis para criar robôs adaptativos, ideais para interações seguras com humanos e ambientes dinâmicos. Ao contrário dos robôs rígidos, os robôs flexíveis oferecem maior versatilidade graças à sua capacidade de deformação.

O projeto destes robôs apresenta desafios complexos tanto na conceção da estrutura como nas estratégias de controlo. A computação evolutiva surge como uma solução promissora, permitindo a evolução conjunta da morfologia e do comportamento robótico, inspirando-se na evolução biológica.

Neste trabalho, utilizamos o ambiente de simulação EvoGym [1], que fornece uma plataforma voxelizada 2D para a criação e teste de robôs flexíveis. O objetivo é aplicar algoritmos evolutivos para desenvolver robôs que se desloquem eficientemente, através de uma abordagem em três fases: evolução da estrutura com controlador fixo, evolução do controlador com estrutura fixa, e coevolução de ambos.

A organização deste documento segue a estrutura: secção 2 descreve o problema; secção 3, a metodologia; secção 4, a configuração experimental; secção 5, os resultados; e secção 6, as conclusões e trabalho futuro.

## 2 Descrição do Problema

Este trabalho aborda o desenvolvimento de robôs flexíveis em ambientes diversos utilizando a plataforma EvoGym, um simulador físico 2D baseado em voxels.

### 2.1 Tipos de Voxels

O EvoGym disponibiliza quatro tipos de voxels:

|                              |   |
|------------------------------|---|
| <b>Sólido (Rigid)</b>        | Não deformável, proporciona estrutura e estabilidade.     |
| <b>Flexível (Soft)</b>       | Deformável sob força, adiciona flexibilidade.             |
| <b>Atuadores Verticais</b>   | Expandem/contraem verticalmente para movimentos de salto. |
| <b>Atuadores Horizontais</b> | Expandem/contraem horizontalmente para propulsão lateral. |

### 2.2 Ambientes

Os ambientes de teste incluem:

|                             |                                 |
|-----------------------------|---------------------------------|
| <b>Walker-v0</b>            | Locomoção em terreno plano      |
| <b>BridgeWalker-v0</b>      | Navegação em solo deformável    |
| <b>DownStepper-v0</b>       | Descida de degraus              |
| <b>ObstacleTraverser-v0</b> | Navegação em terreno irregular  |
| <b>GapJumper-v0</b>         | Transposição de lacunas         |
| <b>CaveCrawler-v0</b>       | Navegação em espaços confinados |

### 2.3 Objetivo

Este trabalho visa desenvolver algoritmos evolutivos para otimizar robôs que navegam eficientemente do ponto inicial à meta. O projeto desenvolve-se em três fases:

|               |  |
|---------------|--|
| <b>Fase 1</b> | Evolução da Estrutura com Controlador Fixo: Comparação entre Algoritmo Genético e busca aleatória. |
| <b>Fase 2</b> | Evolução do Controlador para Estrutura Fixa: Aplicação de CMA-ES e Evolução Diferencial (DE).      |
| <b>Fase 3</b> | Co-evolução Integrada: Abordagem híbrida combinando GA e CMA-ES.                                   |

### 3 Metodologia

#### 3.1 Abordagem Geral

Esta seção apresenta uma visão geral dos componentes fundamentais utilizados nos algoritmos evolutivos aplicados à otimização de robôs flexíveis no EvoGym. A escolha dos algoritmos foi motivada pelas características de cada subproblema: a evolução da estrutura foi tratada com um GA robusto e busca aleatória, que favorecem a diversidade em espaços discretos; a evolução dos controladores utilizou CMA-ES e DE, mais adequados à otimização em espaços contínuos e de alta dimensionalidade; por fim, a co-evolução combinou GA e CMA-ES para lidar com a heterogeneidade entre estrutura e controle. A Tabela 1 resume os principais componentes de cada abordagem.

**Table 1.** Componentes principais dos algoritmos evolutivos utilizados.

| Componente            | Descrição por algoritmo   |
|-----------------------|---|
| Representação         | <b>GA / GA Aleatório:</b> matriz $5 \times 5$ de inteiros representando a estrutura do robô.<br><b>CMA-ES / DE:</b> vetor de números reais representando os pesos e vieses da rede neural do controlador.   |
| Fitness               | <b>Geral:</b> recompensa total fornecida pelo ambiente EvoGym.<br><b>Customizada:</b> combinação de métricas como distância, velocidade média, estabilidade, consistência temporal, complexidade e consumo energético, apenas aplicado em GA.   |
| Operadores Evolutivos | <b>GA:</b> crossover uniforme (estruturas), mutação com promoção de diversidade ( <i>mutate_diverse</i> ).<br><b>GA Aleatório:</b> mutação simples via troca aleatória de voxels.<br><b>CMA-ES:</b> amostragem de soluções via distribuição multivariada adaptativa.<br><b>DE:</b> combinação vetorial de candidatos com estratégia rand/1/bin. |
| Seleção               | <b>GA:</b> torneio com elitismo.<br><b>CMA-ES:</b> seleção baseada na média ponderada das melhores amostras.<br><b>DE:</b> seleção determinística entre vetor mutado e candidato atual.   |

Os algoritmos foram parametrizados conforme descrito na Seção 4, e aplicados em três fases distintas: (1) evolução da estrutura, (2) evolução do controlador e (3) co-evolução estrutura-controlador.

### 3.2 Evolução da Estrutura com Controlador Pré-definido

Desenvolvemos dois algoritmos (GA robusto e busca aleatória) para evoluir estruturas robóticas 5x5 nas tarefas Walker-v0 e BridgeWalker-v0.

| Representação e Controle |  |
|--------------------------|--|
| <b>Codificação</b>       | da 0: Vazio  |
| <b>Estrutura</b>         | 1: Voxel Sólido<br>2: Voxel Flexível<br>3: Atuador Vertical<br>4: Atuador Horizontal   |
| <b>Controladores</b>     | <b>Gait Alternado:</b> Expansão/contração cíclica simulando locomoção com pernas<br><b>Onda Sinusoidal:</b> Função sinusoidal de atuação simulando movimento serpantino<br><b>Movimento de Salto:</b> Sincronização de atuadores para gerar impulsos verticais |

### 3.3 Comparação entre Algoritmo Genético Robusto e Algoritmo de Busca Aleatória

Ambos os algoritmos seguem a mesma lógica evolutiva geral, incluindo elitismo e crossover uniforme. No entanto, diferem significativamente na função de fitness utilizada, nos operadores de mutação e na forma como exploram o espaço de soluções.

| Aspecto              | Algoritmo Genético Robusto  | Algoritmo de Busca Aleatória   |
|----------------------|---|--|
| Base comum           | População inicial aleatória, elitismo, crossover uniforme   | Idêntico   |
| Mutação              | Diversificada e adaptativa: atua sobre blocos, atuadores e parâmetros com diferentes intensidades ao longo das gerações | Simples e ponto-a-ponto: atua aleatoriamente sobre a estrutura sem adaptação |
| Seleção              | Torneio entre indivíduos  | Torneio entre indivíduos   |
| Função de fitness    | Composta: considera desempenho, robustez e conectividade da estrutura   | Simples: baseada apenas no desempenho final                                  |
| Exploração do espaço | Dirigida: favorece soluções robustas e progressivamente mais adaptadas  | Aleatória: explora o espaço sem direção clara                                |
| Complexidade         | Elevada: maior controlo, adaptabilidade e custo computacional   | Baixa: mais leve e fácil de implementar                                      |
| Objetivo             | Otimizar desempenho mantendo robustez estrutural e diversidade  | Testar rapidamente diversas soluções com mutações básicas                    |

**Table 2.** Resumo comparativo entre os algoritmos implementados.

### 3.4 Evolução do Controlador para uma Estrutura Pré-especificada

Para a evolução do controlador com uma estrutura pré-definida, implementamos e comparamos dois algoritmos de otimização: CMA-ES (Covariance Matrix Adaptation Evolution Strategy) e DE (Differential Evolution). Ambos foram aplicados às tarefas DownStepper-v0 e ObstacleTraverser-v0, utilizando a estrutura fixa especificada no enunciado.

**Representação do Controlador** O controlador foi implementado como uma rede neural feedforward que mapeia dados sensoriais para valores de atuação dos voxels ativos. A arquitetura da rede consiste em:

- Camada de entrada: recebe informações sensoriais do robô e do ambiente
- Uma ou mais camadas ocultas com ativação ReLU
- Camada de saída: gera valores de atuação para os voxels ativos, com ativação tanh para limitar os valores entre -1 e 1

| Aspetto              | Algoritmo CMA-ES  | Algoritmo DE  |
|----------------------|---|---|
| Base comum           | População inicial aleatória, evolução de parâmetros contínuos   | Idêntico  |
| Mutação              | Adaptativa: utiliza matriz de covariância que se ajusta automaticamente baseada nos resultados de fitness | Simples e baseada em diferenças: combina vetores de indivíduos existentes usando fator de escala fixo |
| Seleção              | Implícita na geração de novos indivíduos através da distribuição adaptada                                 | Seleção gulosa: filho substitui pai apenas se tiver fitness igual ou superior                         |
| Função de fitness    | Invertida para minimização, avaliada em paralelo  | Direta, avaliada sequencialmente  |
| Exploração do espaço | Dirigida: concentra a busca em regiões promissoras através da adaptação da matriz de covariância          | Moderada: explora diferenças entre indivíduos existentes com operadores fixos                         |
| Complexidade         | Elevada: requer ajuste da matriz de covariância e maior custo computacional                               | Baixa: implementação simples com poucos parâmetros fixos  |
| Objetivo             | Convergir eficientemente em problemas não-lineares complexos  | Proporcionar busca robusta com implementação direta e eficiente                                       |

**Table 3.** Resumo comparativo entre os algoritmos implementados.

### 3.5 Co-evolução de Estrutura e Controlador

Adotámos uma abordagem híbrida para otimizar simultaneamente a estrutura (via Algoritmo Genético) e o controlador (via CMA-ES) dos robôs. Cada indivíduo é composto por:

- **Estrutura:** matriz  $5 \times 5$  com voxels de diferentes tipos (vazio, sólido, flexível, atuadores);
- **Controlador:** vetor de pesos de uma rede neural com arquitetura fixa.

Durante a avaliação, os pesos do controlador são ajustados dinamicamente para garantir compatibilidade com a estrutura, já que esta afeta as dimensões de entrada e saída do ambiente.

A evolução ocorre de forma sincronizada, com avaliação cruzada entre populações. O fitness de cada indivíduo é definido pela média de desempenho com os  $k$  melhores parceiros. A cada geração:

- Estruturas evoluem via GA (seleção por torneio, crossover uniforme, mutação com verificação de conectividade);
- Controladores evoluem via CMA-ES, que atualiza os pesos com base no fitness invertido.

A cada  $n$  gerações, avaliam-se todas as combinações; nas restantes, uma amostra parcial é usada. Uma matriz acumulada de fitness evita avaliações redundantes e melhora a eficiência.

## 4 Configuração Experimental

Para avaliar os algoritmos em diferentes cenários, realizámos 5 execuções independentes com diferentes *seeds* aleatórias para garantir robustez estatística. A cada geração, foram armazenados os dados da população e os valores de fitness correspondentes para posterior análise.

### 4.1 Parâmetros – Tarefa 1

Os parâmetros utilizados na Tarefa 1 foram definidos a partir de um *grid search* explorando combinações entre TOURNAMENT\_SIZE, CROSSOVER\_RATE e MUTATION\_RATE. Embora as execuções finais não tenham sido repetidas 5 vezes, os valores abaixo correspondem à melhor configuração observada:

| Parâmetro       | Valor |
|-----------------|-------|
| MUTATION_RATE   | 0.5   |
| CROSSOVER_RATE  | 0.5   |
| POPULATION_SIZE | 50    |
| TOURNAMENT_SIZE | 3     |
| NUM_GENERATIONS | 250   |
| STEPS           | 500   |

**Table 4.** Parâmetros utilizados na Tarefa 1 (pós grid search).

#### 4.2 Parâmetros – Tarefa 2

Os parâmetros para a Tarefa 2 foram definidos manualmente com base em testes exploratórios, focando na estabilidade do comportamento dos robôs e na convergência dos algoritmos.

| Parâmetro           | Valor |
|---------------------|-------|
| SIGMA_INIT (CMA-ES) | 0.2   |
| MUTATION_RATE (DE)  | 0.5   |
| CROSSOVER_RATE (DE) | 0.7   |
| POPULATION_SIZE     | 50    |
| NUM_GENERATIONS     | 100   |
| STEPS               | 500   |

**Table 5.** Parâmetros utilizados na Tarefa 3 (pós testes).

#### 4.3 Parâmetros – Tarefa 3

Os parâmetros utilizados na Tarefa 3 foram herdados dos melhores resultados da Tarefa 2, ajustando apenas a população e número de gerações devido ao maior custo computacional da co-evolução.

| Parâmetro           | Valor |
|---------------------|-------|
| MUTATION_RATE       | 0.5   |
| CROSSOVER_RATE      | 0.7   |
| TOURNAMENT_SIZE     | 3     |
| SIGMA_INIT (CMA-ES) | 0.2   |
| NUM_PARTNERS        | 3     |
| NUM_GENERATIONS     | 50    |
| POPULATION_SIZE     | 20    |
| STEPS               | 500   |

**Table 6.** Parâmetros utilizados na Tarefa 3 (co-evolução).

#### 4.4 Métricas de Avaliação

Para comparar implementações algorítmicas com parâmetros idênticos, utilizamos:

- **Fitness Final:** Valor da melhor solução encontrada.
- **Eficiência:** Gerações necessárias para atingir determinados níveis de fitness.
- **Diversidade:** Variabilidade genética medida por distância entre genótipos.

#### 4.5 Análise Estatística

Aplicamos os seguintes testes estatísticos para comparar os algoritmos:

- **Teste de Normalidade:** Para verificar se os dados seguem uma distribuição normal (teste de Shapiro-Wilk);
- **Testes t independentes ou de Mann-Whitney:** Para comparações entre dois grupos independentes;
- **Testes t dependentes ou de Wilcoxon:** Para comparações entre duas condições relacionadas;
- **ANOVA ou Kruskal-Wallis:** Para comparações entre três ou mais grupos.

### 5 Resultados e Discussão

#### 5.1 Evolução da Estrutura com Controlador Pré-definido

**Resultados de Fitness** A Tabela 7 apresenta os resultados de fitness para os algoritmos desenvolvidos, aplicados aos ambientes Walker-v0 e BridgeWalker-v0 com diferentes controladores.

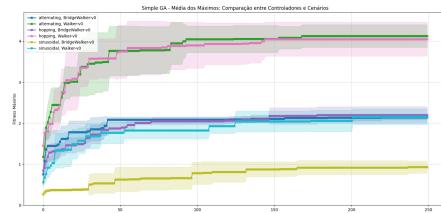
**Table 7.** Melhores Resultados de Fitness e Reward

| Métricas                  | Walker-v0       |              |                 | BridgeWalker-v0 |              |              |
|---------------------------|-----------------|--------------|-----------------|-----------------|--------------|--------------|
|                           | alt             | sin          | hop             | alt             | sin          | hop          |
| <b>GA robusto:Fitness</b> | 9.82<br>(42)    | 9.70<br>(45) | 9.71<br>(45)    | 9.45<br>(43)    | 9.53<br>(43) | 9.48<br>(42) |
| <b>Reward</b>             | 4.45            | 0.77         | 0.75            | 0.25            | 0.08         | -3.00        |
| <b>GA robusto:Reward</b>  | 5.01<br>(46)    | 1.67<br>(46) | 3.84<br>(42,43) | 1.51<br>(44)    | 0.20<br>(45) | 1.99<br>(46) |
| <b>Fitness</b>            | 9.74            | 9.66         | 9.69            | 9.24            | 9.47         | 9.19         |
| <b>Simple_GA:Fitness</b>  | 4.36<br>(44,45) | 2.45<br>(43) | 4.74<br>(42)    | 2.22<br>(46)    | 1.05<br>(46) | 2.56<br>(44) |

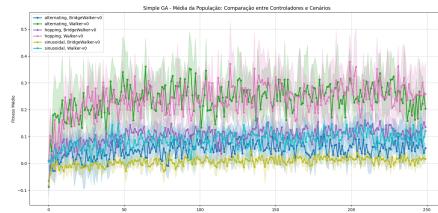
**Evolução da Fitness** As Figuras 1, 2, 3 e 4 mostra a evolução da fitness média (com desvio padrão) ao longo das gerações para os dois algoritmos nas tarefas Walker-v0 e BridgeWalker-v0.

#### 5.2 Análise Estatística

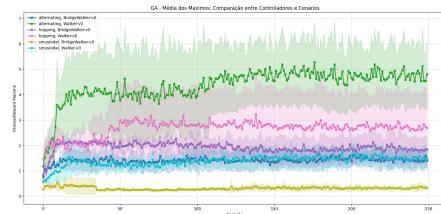
Os testes estatísticos confirmaram diferenças significativas em diversos aspectos dos experimentos. A Tabela 8 apresenta os p-valores do teste *t* dependente para comparação entre ambientes, revelando desempenho significativamente superior



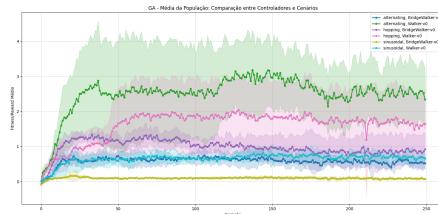
**Fig. 1.** Simple GA - Média dos Máximos: Evolução da fitness ao longo das gerações.



**Fig. 2.** Simple GA - Média da População: Evolução da fitness ao longo das gerações.



**Fig. 3.** GA Robusto - Média dos Máximos: Evolução da fitness ao longo das gerações.



**Fig. 4.** GA Robusto - Média da População: Evolução da fitness ao longo das gerações.

**Table 8.** Comparação entre ambientes Walker-v0 e BridgeWalker-v0 (p-valores dos testes t dependentes).

| Controlador GA _ robusto | Simple _ GA    |
|--------------------------|----------------|
| alternating              | <b>0,0183*</b> |
| hopping                  | 0,3028         |
| sinusoidal               | <b>0,0003*</b> |

\* Diferença estatisticamente significativa ( $p < 0,05$ )

do *Walker-v0* em relação ao *BridgeWalker-v0* nos controladores *alternating* e *sinusoidal* ( $p < 0,05$ ). Para o controlador *hopping*, não foram observadas diferenças estatisticamente significativas entre os ambientes.

A análise entre diferentes controladores demonstrou diferenças significativas em ambos ambientes. No *BridgeWalker-v0*, foram detectadas diferenças significativas entre controladores tanto para GA (ANOVA,  $p = 0,0057$ ) quanto para Simple\_GA (ANOVA,  $p = 0,0001$ ). No *Walker-v0*, o mesmo foi observado para o GA (ANOVA,  $p = 0,0285$ ), enquanto não houve diferença significativa entre controladores para o Simple\_GA (ANOVA,  $p = 0,0535$ ).

A Tabela 10 mostra a variação entre diferentes *seeds*, indicando maior estabilidade no Simple\_GA comparado ao GA\_robusto.

**Table 9.** Comparação entre algoritmos GA\_robusto e Simple\_GA.

| Cenário e Controlador        | p-valor        | Resultado                   |
|------------------------------|----------------|-----------------------------|
| BridgeWalker-v0, alternating | <b>0,0043*</b> | GA_robusto superior         |
| Walker-v0, alternating       | <b>0,0080*</b> | GA_robusto superior         |
| BridgeWalker-v0, hopping     | <b>0,0255*</b> | GA_robusto superior         |
| Walker-v0, hopping           | 0,0628         | Sem diferença significativa |
| BridgeWalker-v0, sinusoidal  | <b>0,0364*</b> | GA_robusto superior         |
| Walker-v0, sinusoidal        | <b>0,0010*</b> | GA_robusto superior         |

\* Diferença estatisticamente significativa ( $p < 0,05$ )

**Table 10.** Análise da variação entre seeds (p-valores dos testes Kruskal-Wallis/ANOVA).

| Controlador | Walker-v0      |           | BridgeWalker-v0 |           |
|-------------|----------------|-----------|-----------------|-----------|
|             | GA_robusto     | Simple_GA | GA_robusto      | Simple_GA |
| alternating | <b>0,0000*</b> | 0,8773    | <b>0,0009*</b>  | 0,8897    |
| hopping     | <b>0,0000*</b> | 0,1130    | <b>0,0000*</b>  | 0,1077    |
| sinusoidal  | <b>0,0037*</b> | 0,9944    | <b>0,0016*</b>  | 0,1624    |

\* Diferença estatisticamente significativa ( $p < 0,05$ )

**Análise de Diversidade Genética** Para avaliar a diversidade genética das populações ao longo das gerações, utilizamos a distância de Hamming [2] média entre os indivíduos. A distância de Hamming entre dois indivíduos  $x$  e  $y$  de comprimento  $n$  é definida como:

$$d_H(x, y) = \sum_{i=1}^n \delta(x_i, y_i) \quad (1)$$

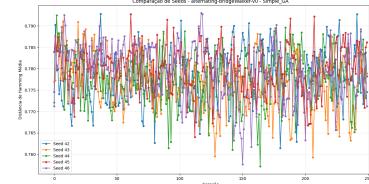
onde  $\delta(x_i, y_i) = 1$  se  $x_i \neq y_i$  e  $\delta(x_i, y_i) = 0$  se  $x_i = y_i$ .

A Figura 5 e Figura 6 apresentam a diversidade de Hamming média para diferentes *seeds* em ambos os algoritmos no caso do *alternating-BridgeWalker-v0*, e a Figura 7 e Figura 8 mostra uma comparação da diversidade entre diferentes controladores e cenários para cada um dos algoritmos.

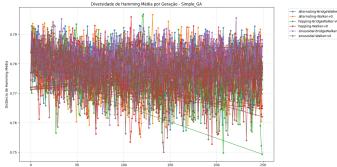
Com base nos gráficos apresentados, a análise da diversidade genética medida pela distância de Hamming revela dois comportamentos algorítmicos distintos: enquanto o Simple\_GA manteve alta diversidade (0,76-0,79) sem clara convergência ao longo das 250 gerações, indicando exploração persistente do espaço de busca, o GA\_robusto demonstrou uma queda significativa inicial da diversidade seguida de estabilização (0,2-0,4), evidenciando um balanço mais eficaz entre exploração inicial e intensificação subsequente, o que sugere maior eficiência na convergência para soluções ótimas, particularmente observável na variante hopping-Walker-v0 que alcançou os menores valores de diversidade.



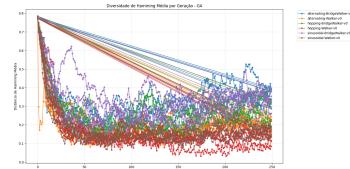
**Fig. 5.** Distância de Hamming média por geração para diferentes seeds no algoritmo `alternating-BridgeWalker-v0` com `Ga_robusto`.



**Fig. 6.** Distância de Hamming média por geração para diferentes seeds no algoritmo `alternating-BridgeWalker-v0` com `Simple_GA`.



**Fig. 7.** Comparação da diversidade de Hamming média por geração para o `Simple_GA`. As linhas representam tendências lineares aproximadas da evolução da diversidade ao longo das gerações.



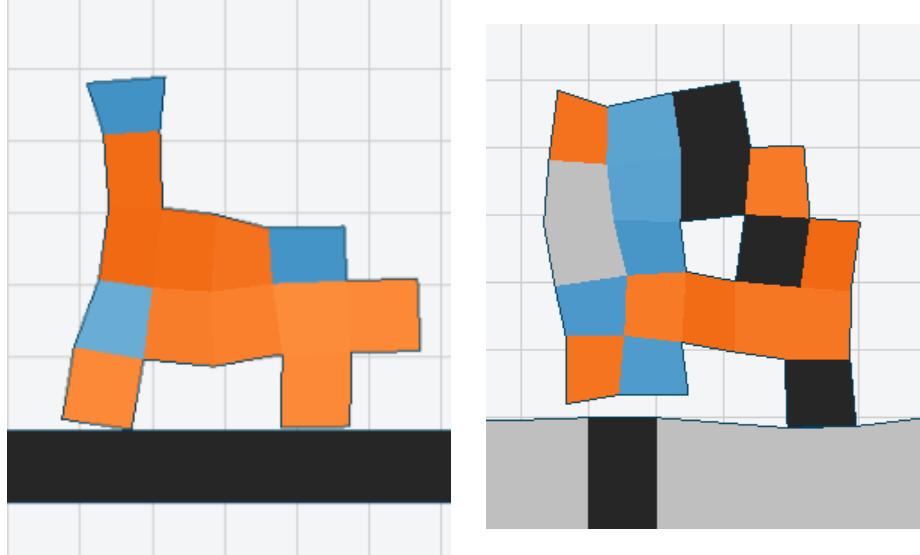
**Fig. 8.** Comparação da diversidade de Hamming média por geração para o `GA_robusto`. As linhas representam tendências lineares aproximadas da evolução da diversidade ao longo das gerações.

**Melhores Estruturas Evoluídas** A Figura 9 mostra a melhor estrutura evoluída para cada cenário.

**Discussão** Os resultados indicam que o `GA_robusto` superou o `Simple_GA` em 5 de 6 cenários, devido à sua função de fitness multidimensional e mutação adaptativa, que equilibram exploração e refinamento. No entanto, os valores elevados de fitness nem sempre resultaram em maior reward, refletindo a complexidade da avaliação. O desempenho variou conforme o controlador e o ambiente, com o padrão hopping mostrando melhor generalização. A diversidade genética revelou estratégias distintas de convergência entre os algoritmos, e o `GA_robusto` mostrou-se mais sensível às condições iniciais. Apesar das vantagens, há limitações em termos de variabilidade, custo computacional e subotimização do reward.

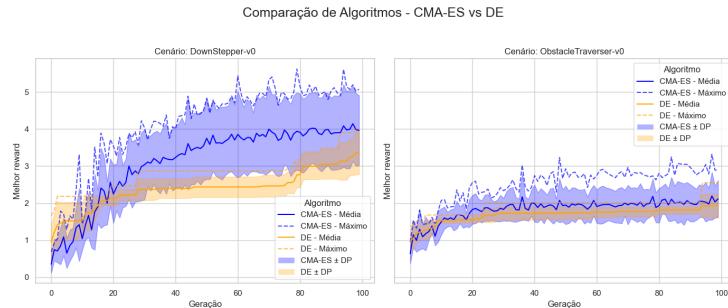
### 5.3 Evolução do Controlador para uma Estrutura Pré-especificada

**Resultados de Fitness** A Tabela 11 apresenta os resultados de fitness para os dois algoritmos desenvolvidos (CMA-ES e DE), aplicados às tarefas `DownStepper-v0` e `ObstacleTraverser-v0`, para 5 execuções independentes.



**Fig. 9.** Melhores estruturas evoluídas: Walker-V0 (alternating-GA\_robusto) à esquerda e BridgeWalker-v0 (Simple GA, Hopping Motion) à direita.

**Evolução da Fitness** A Figura 10 mostra a evolução da fitness média (com desvio padrão) ao longo das gerações para os dois algoritmos nas tarefas DownStepper-v0 e ObstacleTraverser-v0.



**Fig. 10.** Evolução da fitness média ( $\pm$  desvio padrão) e máxima ao longo das gerações para a evolução do controlador.

**Análise Estatística** A Tabela 12 apresenta os resultados da análise estatística para evolução do controlador e a Tabela 13 a análise estatística para a comparação entre os algoritmos CMA-ES e DE para os cenários propostos.

**Table 11.** Resultados de fitness para evolução do controlador.

| Seed          | DownStepper-v0 |      | ObstacleTraverser-v0 |      |
|---------------|----------------|------|----------------------|------|
|               | CMA-ES         | DE   | CMA-ES               | DE   |
| 1             | 5.62           | 4.00 | 1.87                 | 1.67 |
| 2             | 2.74           | 3.90 | 1.79                 | 2.55 |
| 3             | 5.44           | 3.16 | 2.53                 | 1.78 |
| 4             | 5.07           | 2.37 | 3.32                 | 1.84 |
| 5             | 3.49           | 3.38 | 3.03                 | 1.78 |
| Média         | 3.12           | 2.39 | 1.84                 | 1.68 |
| Desvio Padrão | 1.34           | 0.59 | 0.50                 | 0.25 |

\*Média ± desvio padrão referentes aos melhores indivíduos no total das 5 seeds

**Table 12.** Resultados da análise estatística para evolução do controlador.

| Tarefa                                 | Algoritmo | p-value  | Conclusão               | Algoritmo | p-value  | Conclusão               |
|--|-----------|----------|-------------------------|-----------|----------|-------------------------|
| DownStepper-v0 entre seeds             | CMA-ES    | 7.16e-09 | Diferença significativa | DE        | 9.38e-23 | Diferença significativa |
| ObstacleTraverser-v0 entre seeds       | CMA-ES    | 0.0169   | Diferença significativa | DE        | 6.18e-19 | Diferença significativa |
| DownStepper-v0 vs ObstacleTraverser-v0 | CMA-ES    | 0.0230   | DownStepper-v0          | DE        | 0.0159   | DownStepper-v0          |

**Discussão** A análise estatística Tabela 12 revela que tanto o CMA-ES quanto o DE apresentam diferenças significativas ( $p < 0,05$ ) quando comparados entre seeds e entre tarefas diferentes, indicando sensibilidade à seed inicial e ao tipo de tarefa. Contudo, quando comparados diretamente Tabela 13, não há diferenças estatisticamente significativas entre os algoritmos em nenhuma das tarefas ( $p = 0,1365$  para DownStepper-v0 e  $p = 0,0952$  para ObstacleTraverser-v0). Analisando os gráficos Figura 10, observam-se diferenças importantes no comportamento dos algoritmos: No cenário DownStepper-v0, o CMA-ES demonstra convergência mais rápida e alcança valores de recompensa consistentemente superiores ao DE, atingindo médias próximas a 4,0, enquanto o DE estabiliza em torno de 3,0. A área sombreada (desvio padrão) mostra maior variabilidade no CMA-ES, especialmente nas primeiras 40 gerações, sugerindo maior exploração do espaço de busca. Já no cenário ObstacleTraverser-v0, ambos os algoritmos apresentam desempenho mais similar, com curvas de aprendizado que convergem para valores próximos a 2,0. Neste cenário, a diferença entre os algoritmos é menos pronunciada, corroborando o p-value de 0,0952 que se aproxima, mas não atinge, o limiar de significância estatística. Os valores máximos (linhas tracejadas) mostram que o CMA-ES consegue encontrar soluções de maior qualidade em ambos os cenários, indicando melhor capacidade de exploração, mesmo que a média geral não seja estatisticamente diferente do DE.

**Table 13.** Comparação entre algoritmos CMA-ES e DE.

| Cenário                | p-value       | Conclusão                      |
|------------------------|---------------|--------------------------------|
| DownStepper-v0         | <b>0.1365</b> | Não há diferença significativa |
| ObstacleTransverser-v0 | <b>0.0952</b> | Não há diferença significativa |

#### 5.4 Co-evolução de Estrutura e Controlador

**Resultados de Fitness** A Tabela 14 apresenta os resultados de fitness para a abordagem híbrida GA+CMA-ES, aplicada às tarefas GapJumper-v0 e CaveCrawler-v0, para 5 execuções independentes.

**Table 14.** Resultados de fitness para co-evolução de estrutura e controlador.

| Seed          | GapJumper-v0 | CaveCrawler-v0 |
|---------------|--------------|----------------|
| 1             | 3,09         | 2,04           |
| 2             | 1,97         | 2,17           |
| 3             | 2,03         | 2,05           |
| 4             | 2,04         | 2,16           |
| 5             | 2,05         | 2,00           |
| Média         | 1,89         | 1,83           |
| Desvio Padrão | 0,15         | 0,22           |

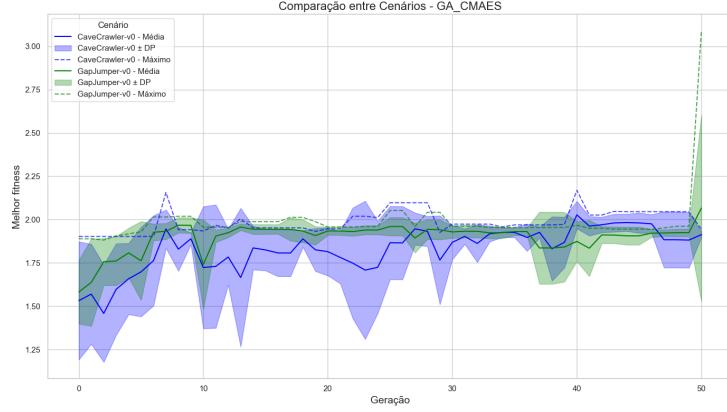
\*Média ± desvio padrão referentes aos melhores indivíduos no total das 5 seeds

**Análise Estatística** A Tabela 15 apresenta os resultados da análise estatística para coevolução.

**Table 15.** Resultados da análise estatística para evolução do controlador.

| Tarefa                         | Algoritmo   | p-value  | Conclusão                      |
|--------------------------------|-------------|----------|--------------------------------|
| CaveCrawler-v0 entre seeds     | GA + CMA-ES | 2.52e-06 | Diferença significativa        |
| GapJumper-v0 entre seeds       | GA + CMA-ES | 7.12e-06 | Diferença significativa        |
| CaveCrawler-v0 vs GapJumper-v0 | GA + CMA-ES | 0.6904   | Não há diferença significativa |

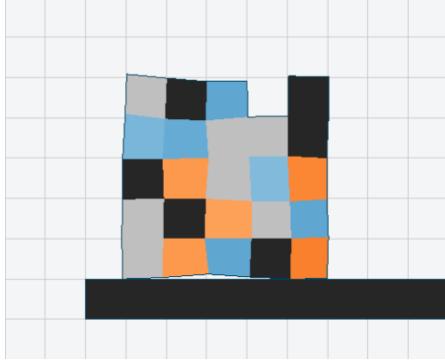
**Evolução da Fitness** A Figura 11 mostra a evolução da fitness média (com desvio padrão) ao longo das gerações para a abordagem híbrida nas tarefas GapJumper-v0 e CaveCrawler-v0.



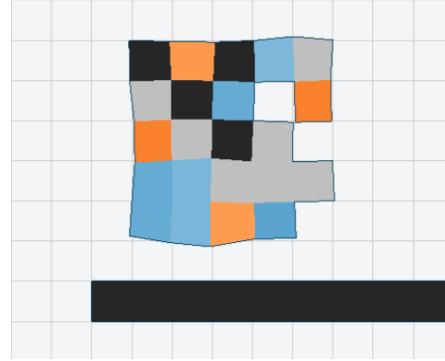
**Fig. 11.** Evolução da fitness média ( $\pm$  desvio padrão) ao longo das gerações para a co-evolução.

**Melhores Soluções Evoluídas** A Figura 14 mostra as melhores soluções (estrutura) evoluídas para cada cenário.

**Discussão** Analisando os resultados da Tabela 14, observa-se que a abordagem híbrida GA+CMA-ES alcançou médias de fitness similares para ambas as tarefas: 1,89 para GapJumper-v0 e 1,83 para CaveCrawler-v0, com desvios padrão de 0,15 e 0,22, respectivamente. No gráfico da Figura 11 observa-se uma tendência de estabilização entre as gerações 10 e 45, indicando que o algoritmo conseguiu convergir relativamente cedo. O CaveCrawler-v0 apresenta maior variabilidade (área sombreada azul mais ampla), confirmando o maior desvio padrão observado na Tabela 14. Nota-se um aumento acentuado da fitness nas últimas gerações (próximo à geração 50), particularmente para GapJumper-v0, sugerindo uma possível descoberta tardia de soluções promissoras. Este comportamento pode indicar que ciclos de evolução mais longos poderiam potencialmente resultar em soluções melhores, apesar que há demasiada estabilidade entre os valores encontrados, sugerindo falta de evolução em ambos os cenários. A análise estatística Tabela 15 revelou diferenças significativas entre as execuções dentro de cada tarefa ( $p$ -values  $< 0,0001$ ), mas não houve diferença estatisticamente significativa entre o desempenho das duas tarefas ( $p$ -value = 0,6904). Observa-se, através das Figuras 12 e 13 e tendo em conta as características de cada cenário, que: no primeiro caso, o controlador já evoluiu para algo mais coerente com o ambiente e move-se através de "saltos", embora caia no primeiro "buraco". Acreditamos que a evolução seguiu um bom caminho, podendo necessitar de mais gerações ou de ajustes nos parâmetros. Neste caso, consideramos também que a estrutura evoluída não facilita o movimento do controlador. No segundo caso, o controlador não foi o mais adequado, uma vez que, mesmo no melhor caso, a distância



**Fig. 12.** \*  
(a) GapJumper-v0



**Fig. 13.** \*  
(b) CaveCrawler-v0

**Fig. 14.** Melhores soluções evoluídas para as tarefas GapJumper-v0 (esquerda) e CaveCrawler-v0 (direita).

percorrida pelo robô foi mínima e apesar de não atingir o primeiro obstáculo, a sua estrutura é demasiado grande para o ultrapassar.

## 6 Conclusão

Este trabalho investigou o uso de algoritmos evolutivos para o desenvolvimento de robôs flexíveis no ambiente de simulação EvoGym, abordando três aspectos fundamentais: a evolução da estrutura, a evolução do controlador e a co-evolução integrada de ambos.

Na evolução da estrutura, o Algoritmo Genético robusto superou consistentemente a busca aleatória em cinco dos seis cenários testados. A função de fitness multidimensional e a mutação adaptativa do GA robusto proporcionaram um melhor equilíbrio entre exploração e refinamento do espaço de busca, resultando em estruturas mais eficientes. Observou-se, porém, que valores elevados de fitness nem sempre se traduziram em maior reward, evidenciando a complexidade da avaliação de desempenho neste domínio.

Na evolução do controlador, tanto o CMA-ES quanto o DE apresentaram desempenhos estatisticamente similares, embora o CMA-ES tenha demonstrado uma tendência de convergência mais rápida, especialmente na tarefa DownStepper-v0. A maior variabilidade observada nas execuções do CMA-ES sugere uma exploração mais ampla do espaço de soluções.

A abordagem híbrida GA+CMA-ES aplicada à co-evolução conseguiu convergir para soluções viáveis, mas os resultados indicam que ciclos evolutivos mais longos poderiam potencialmente produzir soluções de maior qualidade. A estabilização prematura da fitness sugere que ajustes nos parâmetros ou na estratégia de busca poderiam melhorar o desempenho do algoritmo.

As limitações identificadas incluem a alta sensibilidade às condições iniciais (seeds), o elevado custo computacional, especialmente na co-evolução, e a dificuldade em otimizar simultaneamente múltiplos objetivos conflitantes.

Para trabalhos futuros, sugerimos:

Desenvolvimento de algoritmos híbridos com alternância dinâmica de estratégias evolutivas Exploração de estruturas mais complexas e materiais diversos Implementação de técnicas de fitness multi-objetivo para otimizar simultaneamente velocidade, estabilidade e eficiência energética

Em síntese, este trabalho demonstra o potencial dos algoritmos evolutivos para o design de robôs flexíveis adaptativos, contribuindo para avanços na interseção entre computação evolutiva e robótica flexível. Os resultados obtidos abrem caminhos promissores para o desenvolvimento de sistemas robóticos mais versáteis e resilientes.

## References

1. <https://evolutiongym.github.io/>
2. <https://pt.wikipedia.org/wiki/Dist>