LunarLander Evolution

Fundamentos de Inteligência Artificial
18 de abril de 2025

Luana Carolina Cunha Reis, nº 2022220606
Cristiano David Nascimento Domingues Santos, nº 2022219696
Carlos Manuel Gomes da Fonseca Augusto Pereira, nº 2022232042

ÍNDICE

1. Introdução

1.1. Contexto do Problema ("Lunar Lander" e Neuroevolução)

O "Lunar Lander" é um ambiente de simulação 2D utilizado para desenvolver e testar técnicas de controlo autónomo. O objetivo consiste em aterrar uma nave espacial numa plataforma específica da superfície lunar, considerando restrições críticas:

- **Posição:** Aterragem entre duas bandeiras (coordenadas (0,0)).
- **Velocidade vertical:** Inferior a 0.2 m/s para evitar danos estruturais.
- **Orientação:** Ângulo máximo de ±20° em relação à vertical.
- **Contacto com o solo:** Ambas as pernas devem tocar simultaneamente no solo.

A nave é controlada por uma rede neuronal cujos pesos são evoluídos através de um Algoritmo Evolucionário (AE). Esta abordagem, denominada de neuroevolução, combina redes neuronais (para tomada de decisões contínuas) com algoritmos genéticos (para otimização dos parâmetros). O espaço de observação inclui 8 variáveis de estado (posição, velocidade, orientação, etc.) e o espaço de ação 2 valores contínuos (controlo do motor principal e controlo dos motores secundários).

1.2. Objetivos do Trabalho

Este trabalho pretende:

1. Modelar e implementar um AE para evoluir redes neuronais capazes de aterrar a nave em ambientes sem vento e com vento.
2. Analisar criticamente o impacto de componentes do AE:
    - **Operadores genéticos** (*crossover*, mutação).
    - **Mecanismos de seleção** (*tournament*, *rank*).
    - **Parâmetros evolutivos** ("PROB_MUTATION", "PROB_CROSSOVER", "ELITE_SIZE").
3. Avaliar experimentalmente o desempenho através de métricas como:
    - ***Fitness* médio:** Calculado pela função objetivo (combinação de distância horizontal à plataforma, posição vertical, velocidade horizontal/vertical e orientação).

- **Taxa de sucesso:** Percentagem de aterragens válidas em 1000 episódios de teste.

4. Comparar diferentes configurações para identificar *trade-offs* entre exploração e robustez computacional.

2. Metodologia

2.1. Arquitetura do Algoritmo Evolucionário

O algoritmo implementado segue uma estrutura clássica de AE (*steady-state*), com as seguintes fases:

1. **Inicialização:** Geração de uma população de "POPULATION_SIZE" indivíduos, cada um representado por um vetor de pesos (genótipo), amostrado uniformemente em [-1, 1].
2. **Avaliação:** Cálculo do *fitness* via simulação do "Lunar Lander" (função simulate()), paralelizada por "NUM_PROCESSES" processos.
3. **Seleção de Progenitores:** Torneio para escolher progenitores para reprodução.
4. **Variação Genética:**
   - *Crossover*: Ponto único (*single-point*) com probabilidade "PROB_CROSSOVER".
   - Mutação: *Uniform* com probabilidade "PROB_MUTATION".
5. **Seleção de Sobreviventes:** Elitismo puro, preservando os "ELITE_SIZE" (0 ou 1) melhores indivíduos.
6. **Iteração:** Repetição das fases 2–5 por "NUMBER_OF_GENERATIONS" gerações.

2.2. Representação do Indivíduo (Rede Neural)

Cada agente é uma rede neural *feedforward* (que processa informações numa única direção, da entrada para a saída) com arquitetura SHAPE = (8, 12, 2):

- **Entrada:** 8 neurónios (observações do ambiente, Tabela 1 do enunciado).
- **Camada Oculta:** 12 neurónios com função de ativação *tanh*.
- **Saída:** 2 neurónios (controlo dos motores), sem função de ativação.

- **Genótipo:** Vetor de "GENOTYPE_SIZE" (sem *biases*) = 120 pesos (8×12 + 12×2).

2.3. Função Objetivo e Critérios de Sucesso

A função "objective_function(observation)" calcula o valor do *fitness* (qualidade de um indivíduo) com base no estado da nave. O valor do *fitness* é construído a partir de penalizações relacionadas com:

- **Altura vertical (y):**
  Penalização proporcional a |y| - quanto mais alto estiver o módulo, pior o *fitness*.
- **Distância horizontal à plataforma de aterragem (x):**
  Penalização linear para |x| ≤ 0.2 e penalização dobrada (|x| × 2) para |x| > 0.2.
- **Orientação do módulo (θ):**
  Penalização proporcional a |θ|/100. Se |θ| > 20°, a penalização é multiplicada por 1.5.
- **Velocidade vertical (vy):**
  Penalização linear para descidas suaves (vy > -0.2) e penalização reforçada (×1.5) para descidas mais rápidas (vy ≤ -0.2).
- **Velocidade horizontal (vx):**
  Penalização linear com base no valor absoluto de vx.

Fórmula geral aproximada do fitness:

$$fitness = -(|y| + P_x + P_{vy} + P_\theta + |v_x|)$$

onde:

- $P_x = |x| \; se \; |x| \leq 0.2, \; caso \; contrário \; P_x = 2 \cdot |x|$
- $P_{vy} = |vy| \; se \; vy > -0.2, \; caso \; contrário \; P_{vy} = 1.5 \cdot |vy|$
- $P_\theta = \frac{|\theta|}{100} \; se \; |\theta| \leq 20, \; caso \; contrário \; \frac{1.5 \cdot |\theta|}{100}$

Critério de sucesso ("check_successful_landing()")
A aterragem é considerada bem sucedida se:

1. Ambas as pernas estão em contacto com o solo.
2. Distância horizontal à plataforma: $|x| \leq 0.2$
3. Velocidade vertical segura: $vy \geq -0.2\, m/s$
4. Orientação adequada: $\theta \leq 20º$

2.4. Componentes do AE:

2.4.1. Seleção de Progenitores

Método	Implementação no código	Vantagens	Desvantagens
Torneio (k=10)	"tournament_selection(population, k)"	Simples, eficaz, funciona bem mesmo com *fitness* negativos	Pode perder diversidade genética se k for muito grande
Rank-Based Selection (nr_ranks=5)	"rank_based_selection(population, nr_ranks)"	Evita problemas com *fitness* negativos ou muito desbalanceados	Pode ser mais lento e menos preciso se a população for grande
Fitness Proportionate Selection	Não implementado	Intuitivo, favorece indivíduos com melhor desempenho	Não funciona bem com valores negativos; sensível a diferenças extremas de *fitness*
Random Selection	Não implementado	Mantém diversidade, extremamente simples	Ignora completamente o desempenho dos indivíduos

2.4.2. Operadores de Crossover

Método	Implementação no código	Vantagens	Desvantagens
Uniform Crossover	"uniform_crossover(p1, p2)"	Garante alta variabilidade genética; simples de implementar	Pode quebrar estruturas boas (epistasia); perda de herança de blocos
One-point Crossover	"one_point_crossover(p1, p2)"	Preserva sequências de genes contíguos; rápido	Menor variabilidade do que *uniform*; pode gerar soluções menos diversas
Two-point Crossover	Não implementado	Preserva ainda mais blocos de genes que o *one-point*; bom equilíbrio	Mais complexo de implementar; risco de convergência prematura se mal usado
Whole Arithmetic Recombination	Não implementado	Garante suavidade nas soluções contínuas; útil para otimização de parâmetros	Pode causar convergência lenta; menos eficaz para variáveis discretas

2.4.3. Operadores de Mutação

Método	Implementação no código	Vantagens	Desvantagens
Uniform Mutation	"uniform_mutation(p)"	Simples de implementar; introduz boa diversidade.	Pode introduzir alterações muito grandes → instabilidade na solução.
Gaussian Mutation	"gaussian_mutation(p)"	Mais controlada; ideal para afinações locais.	Pode ser demasiado conservadora se o desvio padrão for pequeno.
Bit Flip Mutation	Não implementado	Boa para genótipos binários (ex: codificação de decisões discretas).	Inadequado para representações contínuas (como vetores de *floats*).
Swap Mutation	Não implementado	Útil para problemas de ordenação (ex: TSP).	Irrelevante para vetores de controlo contínuo (como no "Lunar Lander").

2.4.4. Mecanismo de Seleção de Sobreviventes

Elitismo Puro ("survival_selection()"):

Esta função garante que os melhores indivíduos (elite) da geração anterior sejam reavaliados e preservados na nova população. Primeiro, a descendência é ordenada por *fitness* decrescente. Em seguida, os melhores indivíduos da população atual (elite) são reavaliados e adicionados à nova população, substituindo parte da descendência. Por fim, a nova população é novamente ordenada por *fitness*, assegurando que os indivíduos mais aptos sejam mantidos para a próxima geração.

3. Experimentação

3.1. Configuração Experimental

O processo experimental foi estruturado para avaliar sistematicamente o impacto dos parâmetros do AE no desempenho do agente. As principais componentes da configuração incluem:

Ambiente Técnico:

- Python 3.12 com bibliotecas gymnasium[box2d] e numpy.
- Paralelização: NUM_PROCESSES = os.cpu_count() para avaliação concorrente.

Variáveis Independentes:

- Taxa de mutação (“PROB_MUTATION”): 0.008 vs. 0.05
- Taxa de *crossover* (“PROB_CROSSOVER”): 0.5 vs 0.9
- Tamanho da *elite* (“ELITE_SIZE”): 0 vs. 1

Parâmetros Fixos:

Parâmetro	Valor	Justificação
“SHAPE”	(8, 12, 2)	Define a arquitetura da rede neuronal: 8 entradas (observações), 12 neurónios ocultos, 2 saídas (ações). Este valor equilibra a capacidade de representação com a complexidade computacional, sendo suficiente para o controlo do “Lunar Lander” sem ser excessivo.
“ntests”	1000	Número de simulações para validação de um indivíduo. Garante uma avaliação estatisticamente significativa do desempenho do agente, reduzindo o impacto de variação estocástica.
“seeds”	30	Lista de 30 *seeds* (sementes) para inicializar o gerador de números aleatórios nas experiências. Isto permite reprodutibilidade e avaliação robusta contra variação aleatória (para efeitos de experimentação foram são usadas apenas 5 destas 30).
“STD_DEV”	0.1	Desvio padrão da mutação gaussiana. Controla a intensidade das mutações, equilibrando exploração e estabilidade.

3.2. Combinações de Parâmetros Testadas

Foram realizadas 8 experiências principais, cada uma replicada 5 vezes:

Experiência	“POPULATION_SIZE”	“NUMBER_OF_GENERATIONS”	“PROB_CROSSOVER”	”PROB_MUTATION”	“ELITE_SIZE”
1	100	100	0.5	0.008	0
2	100	100	0.5	0.05	0
3	100	100	0.9	0.008	0
4	100	100	0.9	0.05	0
5	100	100	0.5	0.008	1
6	100	100	0.5	0.05	1
7	100	100	0.9	0.008	1
8	100	100	0.9	0.05	1

Critérios de Seleção: Variabilidade na pressão seletiva, *Trade-off* exploração/explotação, Custos computacionais

3.3. Métricas de Avaliação (*Fitness*, Taxa de Sucesso, Diversidade)

3.3.1. *Fitness* Médio

Calculado como:

$$Fitness_{médio} = \frac{1}{ntests} \sum_{i=1}^{ntests} fit$$

- Faixa Ótima: Valores próximos de 0 (penalizações mínimas)

3.3.2. Taxa de Sucesso

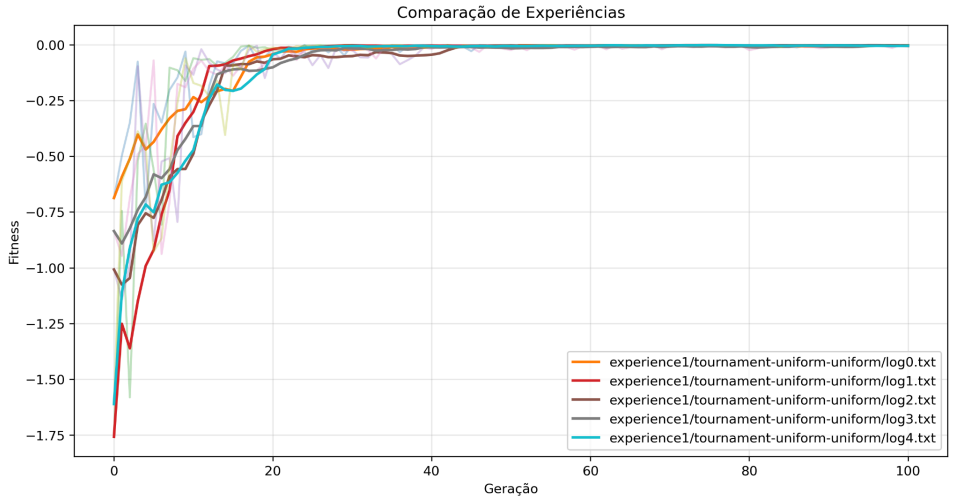
$$Taxa_{sucesso} = \frac{success}{ntests} * 100$$

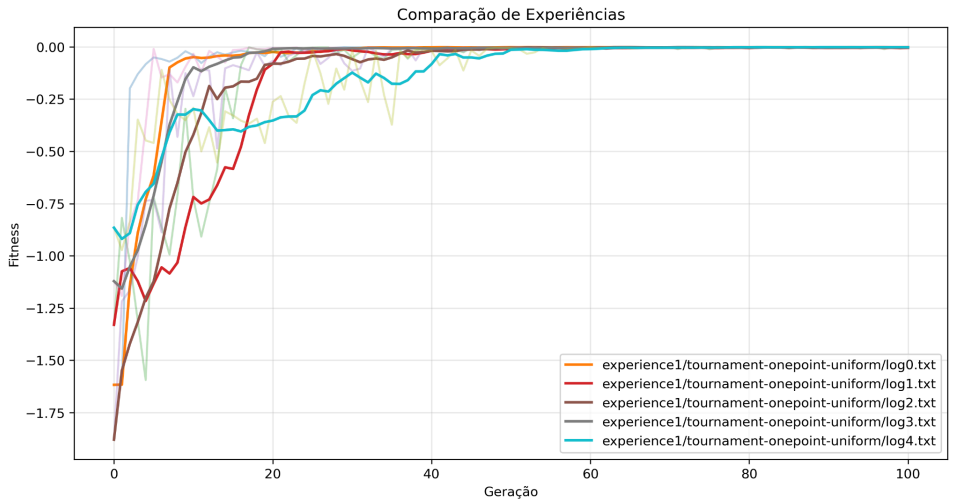
- *Threshold* Aceitável: >75%
- Critério de Validação: Verificado pela função "check_successful_landing()"

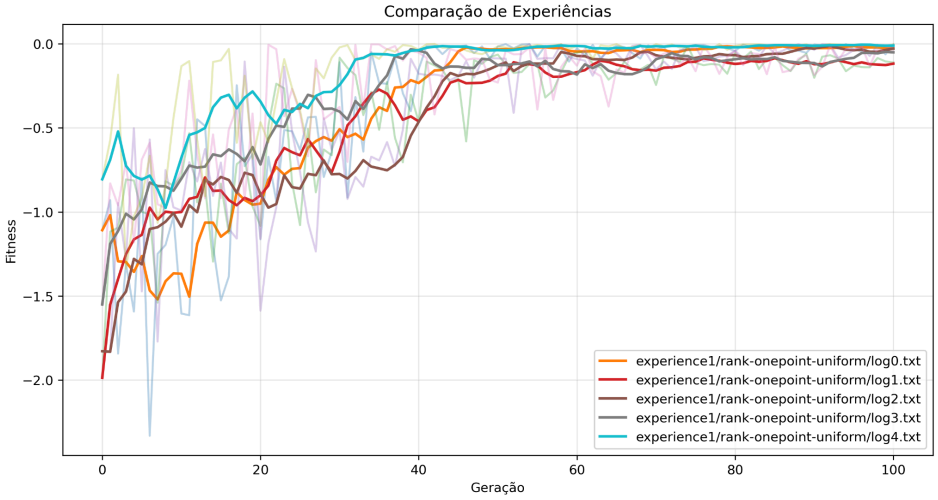
3.3.3. Curvas de Aprendizagem

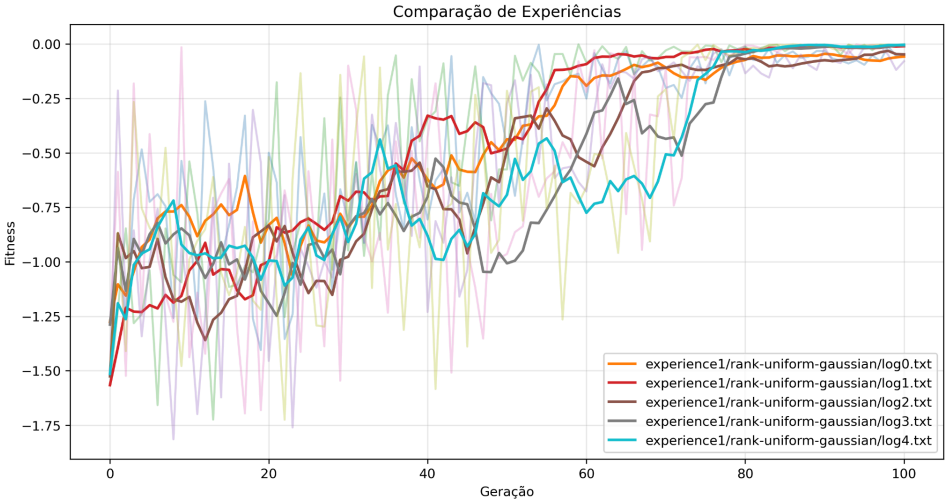
A evolução temporal do *fitness* foi analisada através de gráficos, com legenda do tipo 'seleção-crossover-mutação', gerados pelo código "plot_evolution.py" (todos os gráficos gerados estão incluídos com o código):

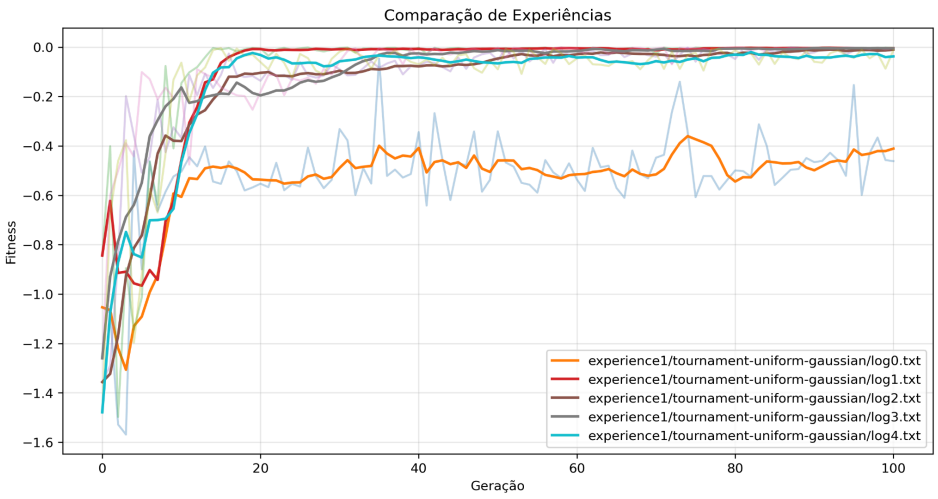
Para a experiência 1:

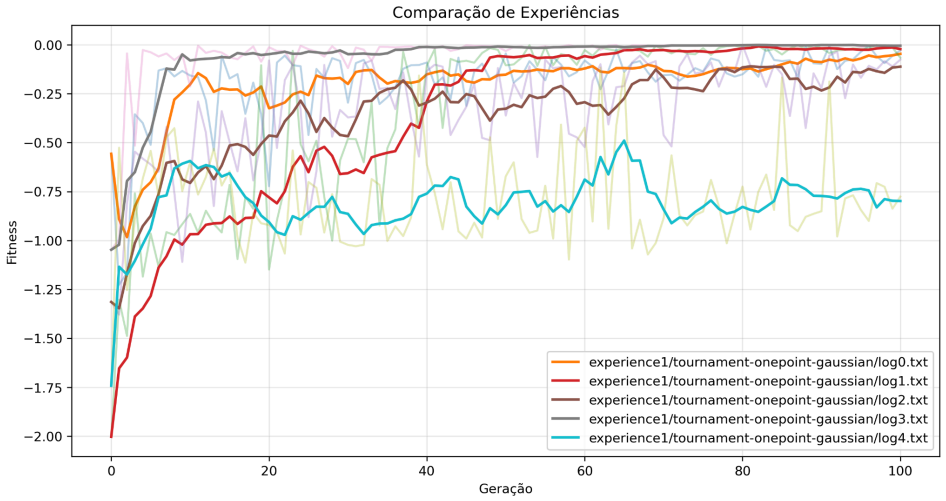


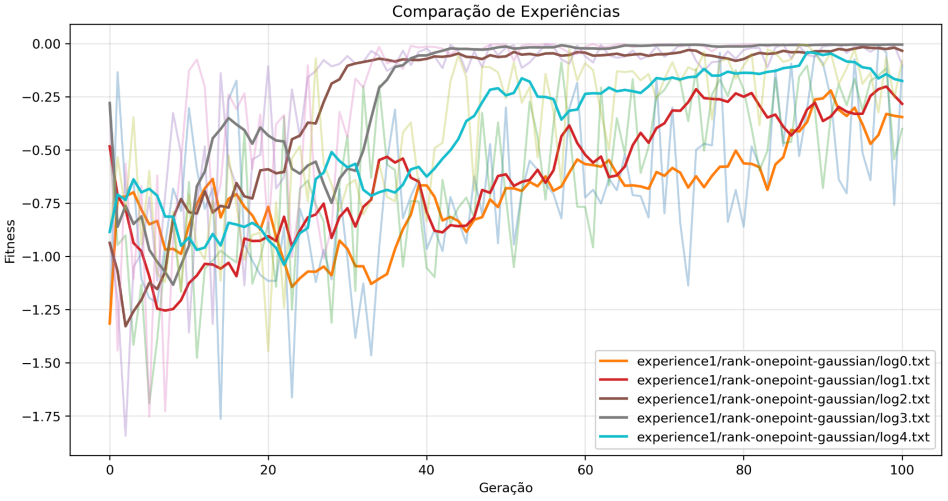


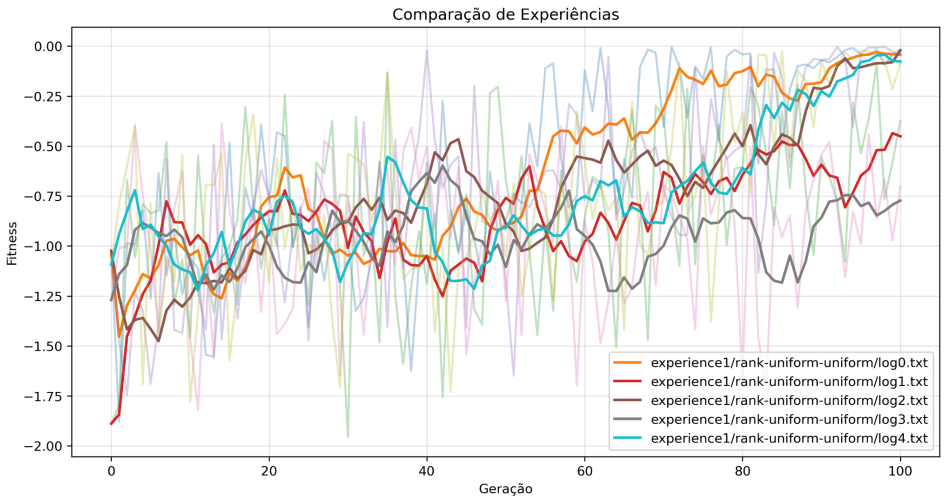

Comparação de Experiências
Fitness
Geração
experience1/rank-onepoint-uniform/log0.txt
experience1/rank-onepoint-uniform/log1.txt
experience1/rank-onepoint-uniform/log2.txt
experience1/rank-onepoint-uniform/log3.txt
experience1/rank-onepoint-uniform/log4.txt


Comparação de Experiências
Fitness
Geração
experience1/rank-uniform-gaussian/log0.txt
experience1/rank-uniform-gaussian/log1.txt
experience1/rank-uniform-gaussian/log2.txt
experience1/rank-uniform-gaussian/log3.txt
experience1/rank-uniform-gaussian/log4.txt


Comparação de Experiências
Fitness
Geração
experience1/tournament-uniform-gaussian/log0.txt
experience1/tournament-uniform-gaussian/log1.txt
experience1/tournament-uniform-gaussian/log2.txt
experience1/tournament-uniform-gaussian/log3.txt
experience1/tournament-uniform-gaussian/log4.txt


Comparação de Experiências
Fitness
Geração
experience1/tournament-onepoint-gaussian/log0.txt
experience1/tournament-onepoint-gaussian/log1.txt
experience1/tournament-onepoint-gaussian/log2.txt
experience1/tournament-onepoint-gaussian/log3.txt
experience1/tournament-onepoint-gaussian/log4.txt


Comparação de Experiências
Fitness
Geração
experience1/rank-onepoint-gaussian/log0.txt
experience1/rank-onepoint-gaussian/log1.txt
experience1/rank-onepoint-gaussian/log2.txt
experience1/rank-onepoint-gaussian/log3.txt
experience1/rank-onepoint-gaussian/log4.txt


Comparação de Experiências
Fitness
Geração
experience1/rank-uniform-uniform/log0.txt
experience1/rank-uniform-uniform/log1.txt
experience1/rank-uniform-uniform/log2.txt
experience1/rank-uniform-uniform/log3.txt
experience1/rank-uniform-uniform/log4.txt

3.3.4. Tabelas com *Fitness* e Taxa de Sucesso Médias

Foram também geradas tabelas, para cada experiência, para cada combinação de tipo de seleção de progenitores, *crossover* e mutação e para cada tipo de execução, isto é, com ou sem vento (todas as tabelas geradas estão incluídas com o código).

Para a experiência 4 ("tournament-uniform-gaussian", sem vento):

Execução	Average Fitness	Success Rate (%)
"exec0"	-0.2750	82.50%
"exec1"	-0.0346	98.80%
"exec2"	-0.0634	87.70%
"exec3"	-0.2490	81.30%
"exec4"	-0.1347	83.10%
Média	-0.1513	86.68%

Para a experiência 7 ("tournament-uniform-uniform", com vento):

Execução	Average Fitness	Success Rate (%)
"exec0"	-0.9264	32.90%
"exec1"	-0.1828	85.00%
"exec2"	-1.3053	24.00%
"exec3"	-0.6285	37.30%
"exec4"	-0.2333	53.40%
Média	-0.6553	46.52%

4. Análise de Resultados

4.1. Evolução do Fitness ao Longo das Gerações

A evolução do *fitness* ao longo das gerações foi monitorizada para cada experiência, utilizando os *logs* gerados automaticamente pelo algoritmo. Em todas as execuções, observa-se uma tendência geral de aumento do *fitness* médio do melhor indivíduo, especialmente nas primeiras gerações, refletindo a adaptação progressiva dos agentes ao problema do "Lunar Lander". Esta melhoria inicial é normalmente mais acentuada, seguida de uma fase de

estabilização, indicando que a população converge para soluções próximas do ótimo local.

A análise gráfica das curvas de *fitness* para as combinações "tournament-uniform-uniform" e "tournament-onepoint-uniform" revela que a convergência é atingida entre 20 a 50 gerações (experiência 1). Além disso, a variação do *fitness* entre execuções com diferentes *seeds* evidencia a natureza estocástica do processo evolutivo, sendo fundamental a realização de múltiplas repetições para obter conclusões estatisticamente significativas.

A monitorização do fitness revelou padrões consistentes:

- Fase de exploração inicial (0-20 gerações): Aumento abrupto nos valores de *fitness*
- Fase de consolidação (20-50 gerações): Taxa de melhoria reduzida em relação à fase anterior
- Convergência (>50 gerações): Variação pouco relevantes entre gerações consecutivas

4.2. Comparação Entre Diferentes Configurações

A análise dos resultados evidencia que a escolha dos operadores genéticos e dos parâmetros do algoritmo tem um impacto direto e significativo no desempenho dos agentes evoluídos, tanto em ambientes sem vento como com vento.

Seleção de Progenitores:

A seleção por torneio destacou-se em praticamente todas as experiências, proporcionando maior pressão seletiva e favorecendo a rápida propagação de soluções de elevada qualidade. Esta abordagem revelou-se particularmente robusta, garantindo uma taxa de sucesso média superior a 89% na experiência 5, sem vento, e mantendo desempenhos competitivos também em cenários com vento (taxa de sucesso média de 68.32% na experiência 4). Foi este tipo de seleção que permitiu a taxa de sucesso máxima de 98.8% em ambientes sem vento (experiência 4) e a taxa máxima de 85% em ambientes com vento (experiência 7).

Operador de *Crossover*:

O *crossover* uniforme mostrou-se ligeiramente mais eficaz do que o *one-point* para a obtenção de taxas de sucesso isoladas superiores (98.8% com uniforme vs. 98% com *one-point* em ambientes sem vento e 85% com uniforme vs. 84.5% com *one-point* em ambientes com vento), especialmente quando combinado com seleção por torneio, tendo sido o tipo de *crossover* que originou a maior percentagem de sucesso. A sua capacidade de recombinar informação genética de forma mais diversificada contribuiu para uma maior exploração do espaço de soluções, refletindo-se em curvas de aprendizagem mais suaves e convergência mais consistente. No entanto, o crossover *one-point*, associado à seleção por torneio, para além de não ter ficado muito distante do tipo anterior (pois conseguiu obter as segundas melhores percentagens), conseguiu destacar-se em mais experiências nas execuções sem vento.

Operador de Mutação:

A mutação gaussiana foi determinante para a obtenção do melhor resultado num ambiente sem vento (em combinação com seleção por torneio e *crossover* uniforme obteve os 98.8% máximos). Este operador permitiu pequenas perturbações controladas nos genótipos, promovendo a exploração local sem comprometer soluções já otimizadas. Em contraste, a mutação uniforme favorece uma maior diversidade e tem menor estabilidade na convergência. Apesar disto, foi essencial para a obtenção da maior taxa de sucesso num ambiente com vento (em conjunto com seleção por torneio e *crossover* uniforme obteve os 85% máximos) Note-se que, sem vento, a mutação uniforme também não ficou muito atrás no resultado obtido (98%) e que, com vento, a mutação gaussiana obteve uma percentagem igualmente semelhante à sua máxima (84.5%).

Parâmetros de Elitismo e População:

Os resultados experimentais revelaram que configurações sem elitismo (“ELITE_SIZE” = 0) obtiveram as melhores taxas de sucesso em ambos os ambientes:

- 98.8% sem vento (*tournament-uniform-gaussian*, Experiência 4)
- 98% sem vento (*tournament-onepoint-uniform*, Experiência 3)
- 98% sem vento (*tournament-uniform-gaussian*, Experiência 3)
- 84.5% com vento (*tournament-onepoint-gaussian*, Experiência 4)

A única exceção a isto foi:

- 85% com vento (tournament-uniform-uniform, Experiência 7)

onde “ELITE_SIZE” = 1.

Resumo dos efeitos observados:

- Combinações com seleção de progenitores por torneio, *crossover* uniforme e mutação guassiana (ou torneio, *one-point* e uniforme) atingiram os melhores resultados para ambientes sem vento.
- Combinações com seleção de progenitores por torneio, *crossover* uniforme e mutação uniforme (ou torneio, *one-point* e gaussiana) atingiram os melhores resultados para ambientes com vento.
- Em ambientes sem vento, a melhor combinação atingiu 98,8% de sucesso, enquanto em ambientes com vento o máximo foi de 85%.
- O desempenho médio decresceu em ambientes adversos, mas as melhores configurações mantiveram uma degradação controlada, evidenciando robustez.

Em suma, a escolha criteriosa dos operadores e dos parâmetros do algoritmo é determinante para o sucesso da abordagem neuroevolutiva, sendo fundamental ajustar estes elementos em função das características do ambiente e dos objetivos específicos.

4.3. Impacto dos parâmetros no desempenho

A análise empírica realizada permite identificar quais os parâmetros com maior influência no desempenho dos agentes neuroevoluídos. A comparação entre os resultados obtidos em ambientes sem vento e com vento revelou padrões consistentes:

Melhor parametrização para ambientes sem vento:

Parâmetro	Valor	Justificação
“POPULATION_SIZE”	100	Tamanho da população em cada geração.
“NUMBER_OF_GENERATIONS”	100	Número de gerações para evolução.
“PROB_CROSSOVER”	0.9	Probabilidade de aplicar crossover ao criar descendentes.
“PROB_MUTATION”	0.05	Probabilidade de mutação em cada gene.
“ELITE_SIZE”	0	Número de indivíduos de elite preservados a cada geração.

Melhor parametrização para ambientes com vento:

Parâmetro	Valor	Justificação
"POPULATION_SIZE"	100	Tamanho da população em cada geração.
"NUMBER_OF_GENERATIONS"	100	Número de gerações para evolução.
"PROB_CROSSOVER"	0.9	Probabilidade de aplicar crossover ao criar descendentes.
"PROB_MUTATION"	0.008	Probabilidade de mutação em cada gene.
"ELITE_SIZE"	1	Número de indivíduos de elite preservados a cada geração.

- **Probabilidade de Mutação ("PROB_MUTATION"):**
  - Sem vento: 0.05 foi mais eficaz, pois promove maior diversidade e evita convergência prematura.
  - Com vento: 0.008 resultou em soluções mais estáveis, reduzindo o risco de sobreexploração em ambientes ruidosos.
- **Probabilidade de Crossover ("PROB_CROSSOVER"):**
  - Um valor mais elevado (0.9) foi comum às melhores configurações, em ambos os ambientes, promovendo recombinação eficaz e convergência mais rápida.
- **Elitismo ("ELITE_SIZE"):**
  - Sem vento: As melhores execuções ocorreram com "ELITE_SIZE = 0", favorecendo renovação populacional e maior exploração.
  - Com vento: A introdução de "ELITE_SIZE = 1" ajudou a preservar indivíduos robustos, estabilizando o desempenho, no caso dos ambientes com vento.

Estes resultados reforçam a importância de ajustar os parâmetros consoante a natureza do ambiente. Em contextos mais adversos (com vento), uma menor taxa de mutação e elitismo controlado mostraram-se benéficos para manter a robustez.

4.4. Robustez do Agente (Variabilidade Entre Execuções)

A robustez do agente foi avaliada através da repetição de cada experiência com múltiplas *seeds*, conforme exigido no enunciado. Os principais indicadores de robustez são:

- **Desvio padrão do *fitness* final e da taxa de sucesso:**
  - Experiências com menor desvio padrão são consideradas mais robustas, pois produzem agentes com desempenho consistente, independentemente da inicialização.

- **Taxa de sucesso mínima entre execuções:**
  - Uma configuração robusta mantém a taxa de sucesso acima do limiar aceitável (>75%) em todas as execuções.
- **Análise qualitativa dos *logs*:**
  - Configurações com elevada diversidade e pressão seletiva equilibrada apresentam menor variabilidade nos resultados finais.

5. Discussão

5.1. Escolhas de *Design* e *Trade-offs*

As decisões de *design* do algoritmo evolutivo foram guiadas por objetivos de desempenho, simplicidade e adaptabilidade ao ambiente. A experiência mostrou que diferentes escolhas afetam de forma significativa o comportamento do agente.

- **Arquitetura Simples e Direta:**
  - A escolha de uma rede neural *feedforward* com topologia fixa e sem *biases* foi intencional para simplificar o vetor genético e o espaço de pesquisa. Embora isso limite ligeiramente a expressividade da política, permitiu uma evolução mais eficiente.
- **Operadores Genéticos Eficazes:**
  - A escolha para a implementação dos operadores genéticos teve sempre em conta o objetivo de maximizar a taxa de sucesso e de garantir bons valores de *fitness*.
- **Parâmetros Adaptativos:**
  A melhor combinação variou consoante o tipo de ambiente.

Estas decisões evidenciam o *trade-off* entre exploração e robustez:

1) Ambientes previsíveis beneficiam de mais diversidade e substituição frequente.
2) Ambientes instáveis exigem maior preservação de bons indivíduos e mutações mais discretas.

5.2. Limitações e Desafios Encontrados

Durante o desenvolvimento, foram enfrentadas várias limitações:

- **Variabilidade Estocástica**: O ambiente "Lunar Lander", sobretudo com vento, introduz elevada aleatoriedade, tornando difícil isolar o impacto de cada parâmetro sem múltiplas execuções.

- **Custo Computacional**: A simulação de episódios para avaliar o *fitness* é dispendiosa, exigindo paralelização para viabilizar as experiências.

- **Dificuldade de Sintonização de Parâmetros**: Pequenas alterações em parâmetros (como "PROB_MUTATION") podem levar a grandes flutuações de desempenho. A pesquisa por combinações ótimas exigiu tentativa e erro.

- **Limitações na Arquitetura da Rede Neural**: A rede utilizada é relativamente pequena e sem *biases*, o que limita a complexidade dos comportamentos que pode aprender.

Apesar destas limitações, os resultados globais foram satisfatórios, com alta taxa de sucesso em múltiplas configurações.

6. Conclusões e Trabalho Futuro

O desenvolvimento e análise de um algoritmo evolutivo para o controlo do "Lunar Lander" permitiu demonstrar a eficácia da neuroevolução na aprendizagem de comportamentos complexos em ambientes dinâmicos e estocásticos. O agente, representado por uma rede neuronal de arquitetura fixa, foi capaz de aprender a aterrar a nave de forma consistente, atingindo taxas de sucesso elevadas, de acordo com os critérios definidos no enunciado.

A experimentação sistemática evidenciou que:

- A diversidade genética é fundamental para evitar a convergência prematura e garantir a robustez dos resultados. Parâmetros como o tamanho da população, a taxa de mutação e o elitismo têm impacto direto na evolução do *fitness* e na taxa de sucesso.
- A seleção por torneio revelou-se eficaz, equilibrando pressão seletiva e diversidade, enquanto operadores simples de *crossover* e mutação (*single-point* e uniforme) foram suficientes para atingir desempenhos satisfatórios.
- A avaliação paralela foi crucial para viabilizar múltiplas execuções e garantir análise estatística significativa, dada a variabilidade do ambiente e do processo evolutivo.

No geral, o Algoritmo Evolucionário mostrou-se flexível e adaptável, sendo capaz de encontrar soluções viáveis mesmo com diferentes parametrizações e *seeds* iniciais. A abordagem revelou-se robusta para vários ambientes, cumprindo os objetivos.

7. Referências

[1] Ali Karazmoodeh. (2023, 30 de dezembro). *Crossovers in genetic algorithms*. LinkedIn. https://www.linkedin.com/pulse/crossovers-genetic-algorithms-ali-karazmoodeh-tthjf

[2] Ali Karazmoodeh. (2023, 31 de dezembro). *Mutations in genetic algorithms*. LinkedIn. https://www.linkedin.com/pulse/mutations-genetic-algorithms-ali-karazmoodeh-u94pf

[3] *Crossover (evolutionary algorithm)*. (2025, 15 de abril). Wikipedia. https://en.wikipedia.org/wiki/Crossover_(evolutionary_algorithm)

[4] *Crossover in Genetic Algorithm*. (2023, 10 de março). GeeksforGeeks. https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

[5] *Genetic Algorithms - Crossover*. (s.d.). Tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm

[6] *Genetic Algorithms - Mutation*. (s.d.).Tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm

[7] *Genetic Algorithms - Parent Selection*. Tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm

[8] *Mutation Algorithms for Real-Valued Parameters (GA)*. (2024, 11 de março). GeeksforGeeks. https://www.geeksforgeeks.org/mutation-algorithms-for-real-valued-parameters-ga/

[9] *Mutation in Genetic Algorithm*. (2024, 23 de agosto). Naukri Code360. https://www.naukri.com/code360/library/mutation-in-genetic-algorithm

[10] *Neuroevolution*. (2025, 2 de janeiro). Wikipedia. https://en.wikipedia.org/wiki/Neuroevolution

[11] *Selection (evolutionary algorithm)*. (2025, 15 de abril). Wikipedia. https://en.wikipedia.org/wiki/Selection_(evolutionary_algorithm)