

1 2 9 0



UNIVERSIDADE
COIMBRA

Rolling in the Hill Evolutionary Edition
Fundamentos de Inteligência Artificial

Licenciatura em Engenharia Informática
2022/2023

11 de agosto de 2023

Autores

João Moreira

 joaomoreira@student.dei.uc.pt

 PL1

 2020230563

Tomás Pinto

 tomaspinto@student.dei.uc.pt

 PL1

 2020224069

Índice

1	Introdução	3
2	Meta 1 - Modelação e desenvolvimento do Algoritmo Genetico	3
2.1	Recombinação	3
2.2	Mutação	4
2.3	Seleção de pais	4
2.4	Seleção de sobreviventes (elitismo)	5
2.5	Aptidão	5
2.6	Parametrização	5
3	Experimentação e análise - GapRoad	5
3.1	Métodos de Avaliação	5
3.2	Primeira Iteração	6
3.3	Segunda Iteração	8
3.4	Terceira Iteração	10
3.5	Resultado final	14
4	Experimentação e análise - HillRoad	14
4.1	Primeira Iteração	14
4.2	Segunda Iteração	15
4.3	Resultado final	16
5	Experimentação e análise - ObstacleRoad	17
5.1	Primeira Iteração	17
5.2	Segunda Iteração	18
5.3	Resultado final	19
6	Conclusão	20

1. Introdução

Este projeto tem como finalidade o desenvolvimento de um algoritmo evolucionário capaz de projetar um veículo motorizado. O objetivo principal é garantir que o veículo possa percorrer o trajeto inserido o mais longe possível, tornando-se um desafio para o algoritmo projetar um veículo eficiente e capaz de cruzar a meta.

Para atingir esse objetivo, foi desenvolvido um simulador virtual, em *Unity*, capaz de avaliar o desempenho do veículo motorizado em diversos cenários, incluindo trajetos com buracos e quedas.

2. Meta 1 - Modelação e desenvolvimento do Algoritmo Genetico

Neste capítulo explicamos todas as etapas envolvidas neste trabalho para o desenvolvimento do Algoritmo Genetico, tais como:

- Recombinação
- Mutação
- Seleção de pais
- Seleção de sobreviventes (elitismo)
- Aptidão
- Parametrização

2.1. Recombinação

A etapa de recombinação, é o momento onde se recombina os genes para a formação de uma nova geração. Para este fim, foi implementado o operador de recombinação uniforme.

Na mutação uniforme, cada gene é selecionado de um dos seus progenitores aleatoriamente[3], ou seja, cada indivíduo tem a mesma probabilidade de herdar um determinado gene do cromossoma do seu pai ou da sua mãe.

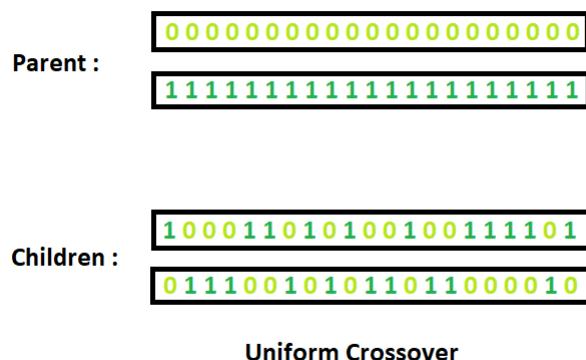


Figura 1: Esquema de recombinação uniforme.

Esta é uma técnica muito simples de implementar, porém tem algumas limitações: Por vezes, o indivíduo pode herdar um mau gene de um progenitor e o melhor gene pode ser perdido. No entanto, se tivermos dois pais bons, tipicamente teremos alta probabilidade de ter um bom filho e no caso de ele não ser bom, será eliminado na próxima geração (seleção natural).

2.2. Mutação

A mutação é o operador genético que permite obter diversidade genética de uma população de indivíduos, análogo à mutação biológica. Este método é utilizado para melhorar a diversidade das nossas populações e equilibrar a proporção das mutações e da herança para garantir a convergência do nosso algoritmos evolucionário [2].

Após a recombinação genérica será aplicado um processo de mutação. Para tal, implementamos um algoritmo de mutação gaussiana.

Neste algoritmo cada gene dos cromossomas de um indivíduo têm probabilidade igual de sofrer uma mutação, ao qual o novo gene é calculado na base de uma função gaussiana [1].

2.3. Seleção de pais

A etapa de seleção de pais é onde escolhemos os progenitores que irão gerar os indivíduos das próximas gerações. A escolha dos pais é um ponto crucial na convergência do nosso algoritmo evolucionário, pois bons pais têm tendência a gerar filhos melhores e mais adequados ao meio.

Nesta etapa utilizamos um algoritmo de seleção proporcional à aptidão dos indivíduos: Seleção de Roleta.

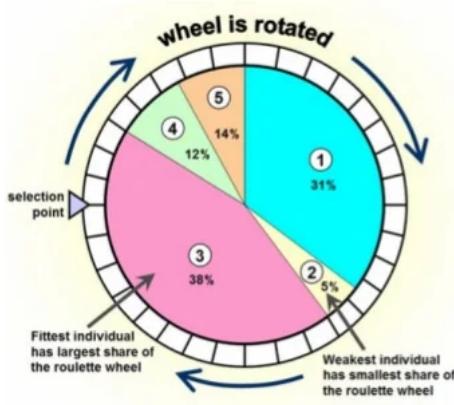


Figura 2: Seleção de pais por roleta.

Neste algoritmo seguimos uma abordagem de roleta russa. Construímos uma roleta onde cada fatia é proporcional à aptidão de cada indivíduo. Por fim, giramos a roleta aleatoriamente e um indivíduo é selecionado [4].

Obviamente, quanto maior for a aptidão do indivíduo maior é a probabilidade de ser selecionado. No entanto, não deixa de ser uma boa heurística, pois combinamos aleatoriedade com a sobrevivência do indivíduo mais forte.

2.4. Seleção de sobreviventes (elitismo)

O elitismo é o método de cópia dos cromossomas de uma *offspring* antes dos métodos de mutação e recombinação. Basicamente, mantemos os melhores indivíduos de uma geração na próxima, sem que sofram alterações genéticas.

Sendo assim, organizamos a população atual por ordem decrescente de aptidão e selecionamos o número de agentes especificados na Parametrização.

2.5. Aptidão

A aptidão é a métrica que permite avaliar se um indivíduo possui características genéticas que o tornam capaz de desempenhar bem a tarefa proposta.

Dessa forma, a função de aptidão, que mede a qualidade de cada indivíduo da população, é fundamental para guiar o processo de seleção, mutação e recombinação que caracteriza um algoritmo evolucionário. Ao selecionar os indivíduos mais aptos para a reprodução, é possível gerar novos indivíduos com características genéticas que favoreçam a solução do problema em questão.

Por este motivo, a função de aptidão deve definir e refletir adequadamente a capacidade dos veículos em desempenhar sua tarefa.

Para esta meta, a nossa função de aptidão corresponde ao valor da distância máxima percorrida pelo carro. Não será a função de aptidão final, nós iremos ajustar a função de acordo com a nossa necessidade e veremos a importância desta etapa.

2.6. Parametrização

No desenvolvimento desta etapa, tivemos o cuidado de fazer o código o mais genérico e parametrizável possível para que facilmente se pudesse alterar os valores ao longo das experiências.

Estes são os parâmetros que o nosso algoritmo tem em conta:

- Probabilidade de Mutação
- Tamanho de Elitismo
- Probabilidade de *Crossover*
- Número de Gerações

3. Experimentação e análise - GapRoad

3.1. Métodos de Avaliação

De forma a avaliar os resultados, extraímos dos dados a média e desvio padrão ao longo das gerações, isto porque a média permite-nos perceber o crescimento/decrescimento da aptidão dos indivíduos e o desvio padrão permite perceber a variação da aptidão entre gerações. Também teremos em conta o número de gerações que contenham pelo menos um indivíduo a terminar.

Sendo assim, procuramos experiências que contenham funções de média monótonas crescentes e usaremos o número de gerações que terminam com critério de desempate.

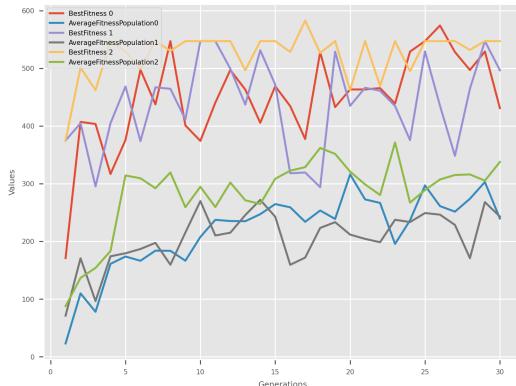
3.2. Primeira Iteração

Inicialmente, sentimos a necessidade de saber quais são os melhores parâmetros para o nosso algoritmo evolucionário. Sendo assim, corremos a Tabela 1 de experiências 3 vezes, com uma função de aptidão simples, pelo que, é diretamente proporcional à distância percorrida:

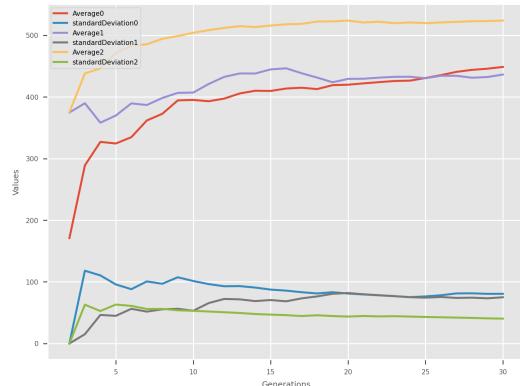
$$Fitness_i = MaxDistance_i$$

	Mutação	Elitismo	Crossover	Número Gerações
Experiência 1	0.05			
Experiência 2	0.2	0		
Experiência 3	0.05		0.9	
Experiência 4	0.2	2		30

Tabela 1: Tabela de experiências

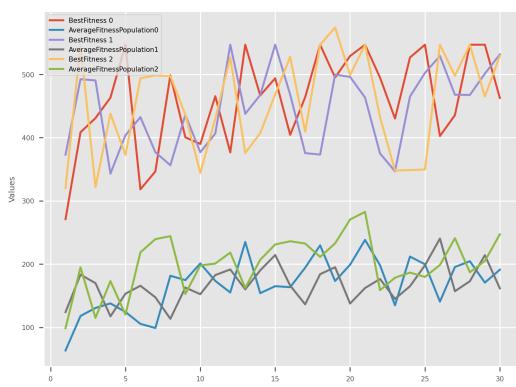


(a) BestFitness vs AverageFitness.

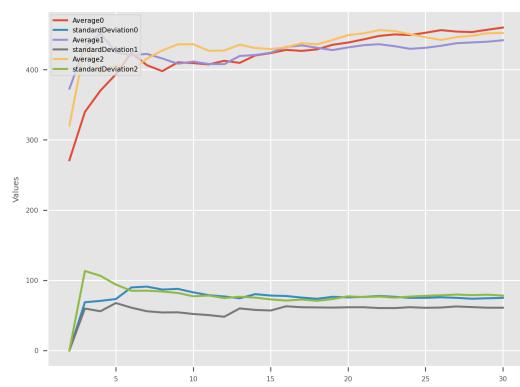


(b) Average vs StandardDeviation.

Figura 3: Experiência 1.

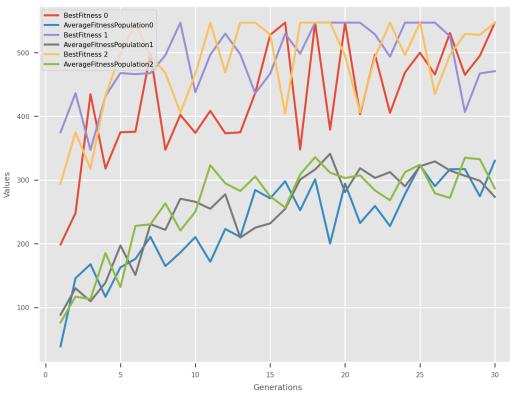


(a) BestFitness vs AverageFitness.

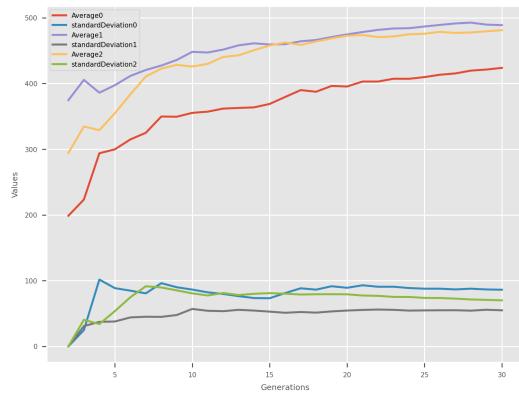


(b) Average vs StandardDeviation.

Figura 4: Experiências 2.

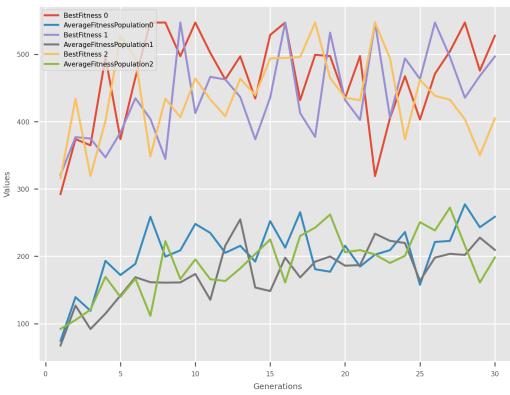


(a) BestFitness vs AverageFitness.

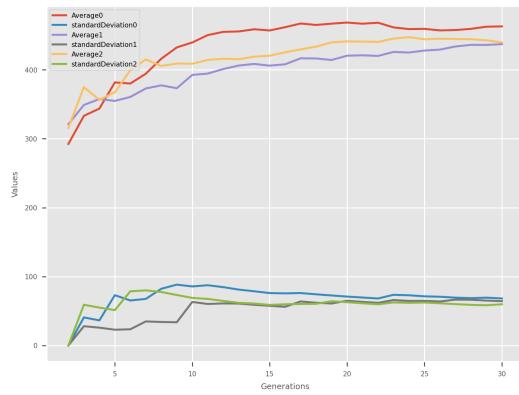


(b) Average vs StandardDeviation.

Figura 5: Experiência 3.



(a) BestFitness vs AverageFitness.



(b) Average vs StandardDeviation.

Figura 6: Experiência 4.

Com os resultados obtidos reparamos que experiências com maior valor de mutação, não apresentam forte evolução, como as experiências que têm baixo valor de mutação. Tal como vemos na experiência 2, quando um caro termina com uma boa avaliação de aptidão, a geração seguinte tem uma péssima avaliação. Este comportamento mantém-se ao longo das experiências e não existe progressão. No entanto, na experiência 4 temos uma ligeira progressão. Em contra partida, a sua progressão é muito lenta em relação à experiência 3, cuja difere apenas no valor de mutação. Isto tudo acontece porque quando um indivíduo tem uma boa prestação e é selecionado como progenitor da próxima geração, o seu descendente perde as suas boas características, devido à aleatoriedade do algoritmo.

Se observarmos as experiências em que o elitismo está presente, existe progressão desde o inicio ao fim da experiência. Como vemos na experiência 1, não existe elitismo, e assim pelo facto de não guardarmos as boas soluções, não temos como as melhorar e não apresentamos evolução. Na experiência 3, levamos elitismo, guardamos as boas soluções e podemos melhorá-las através de mutações e de recombinações entre progenitores. Como podemos ver esta é a melhor das soluções, isto porque sem elitismo, o algoritmo pode ter dificuldade em escapar de mínimos locais sub-ótimos. Se as melhores soluções não forem protegidas e mantidas, a busca pode ficar presa em regiões do espaço de solução que não são ótimas, resultando em soluções de menor qualidade.

Assim, podemos concluir que:

- Alta mutação destrói boas soluções;
- Baixa mutação refina gradualmente as soluções e permite a procura local de soluções ótimas;
- Elitismo preserva boas soluções e permite a convergência mais rápido.

Sendo assim, pelas razões apresentadas iremos fixar os parâmetros da experiência 3 e explorar uma nova função de aptidão.

3.3. Segunda Iteração

Corremos diversas vezes a experiência 3 da iteração 1 e uma função de aptidão interessante que obtemos foi

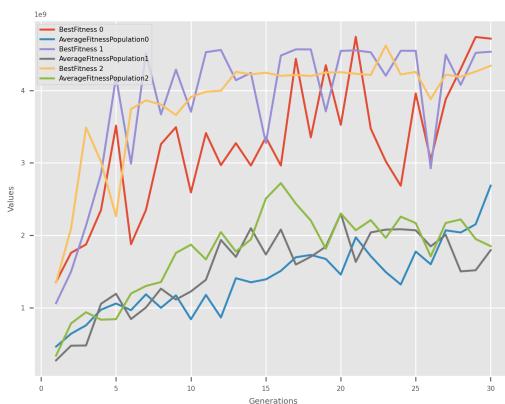
$$Fitness_i = (2 * MaxDistance_i + 10 * NumberOfWheels_i + 1.5 * MaxVelocity + CarMass)^3$$

Verificamos que os carros que tinham rodas grandes na sua base e muita velocidade, tinham mais capacidade para atingir a meta. Sendo assim, adicionamos o número de rodas e a massa do carro de forma a valorizar tais características. Isto porque o tamanho das rodas está diretamente relacionado com o peso do carro.

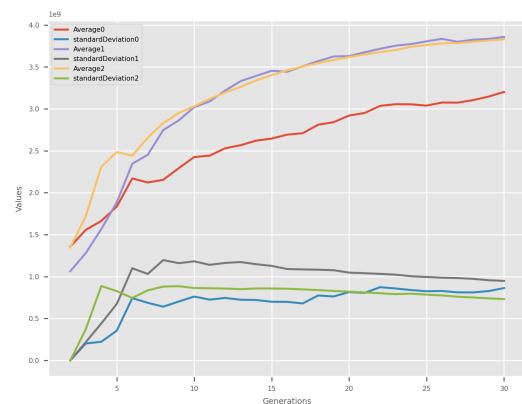
Elevamos tudo ao cubo, para que haja uma distinção maior entre indivíduos que tenham valores de aptidão parecidos, aumentando assim a pressão seletiva na roleta.

	Mutação	Elitismo	Crossover	Número Gerações
Experiência 1	0.05	2	0.4	30
Experiência 2			0.5	
Experiência 3			0.6	
Experiência 4			0.7	
Experiência 5			0.8	

Tabela 2: Tabela de experiências

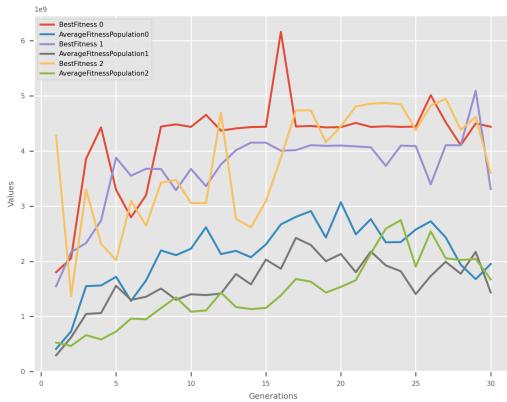


(a) BestFitness vs AverageFitness.

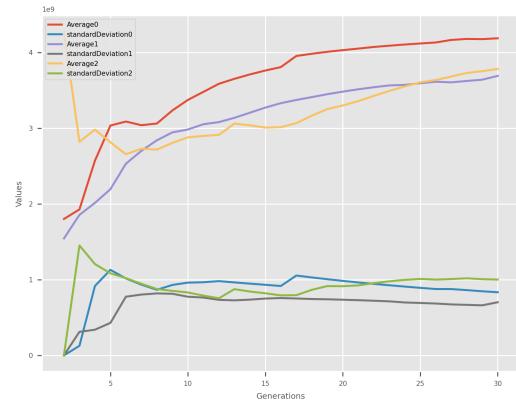


(b) Average vs StandardDeviation.

Figura 7: Experiência 1.

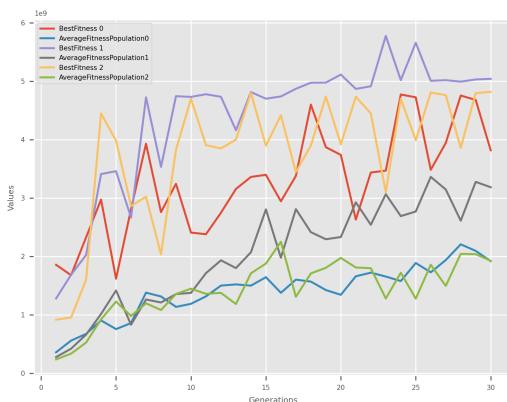


(a) BestFitness vs AverageFitness.

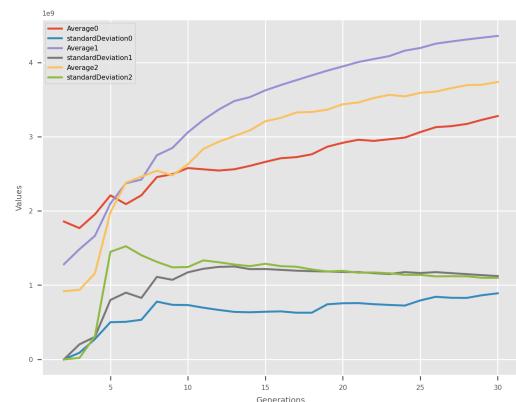


(b) Average vs StandardDeviation.

Figura 8: Experiência 2.

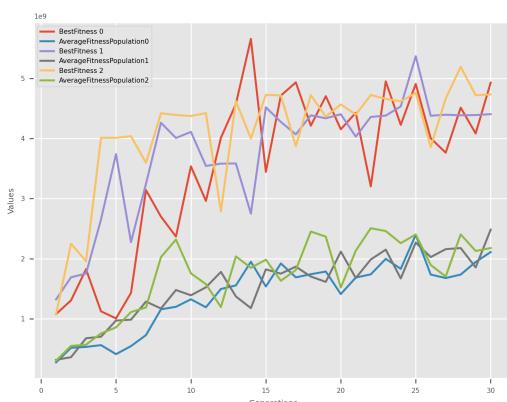


(a) BestFitness vs AverageFitness.

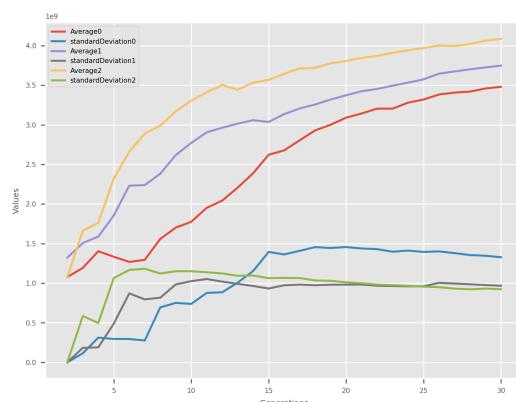


(b) Average vs StandardDeviation.

Figura 9: Experiências 3.



(a) BestFitness vs AverageFitness.



(b) Average vs StandardDeviation.

Figura 10: Experiência 4.

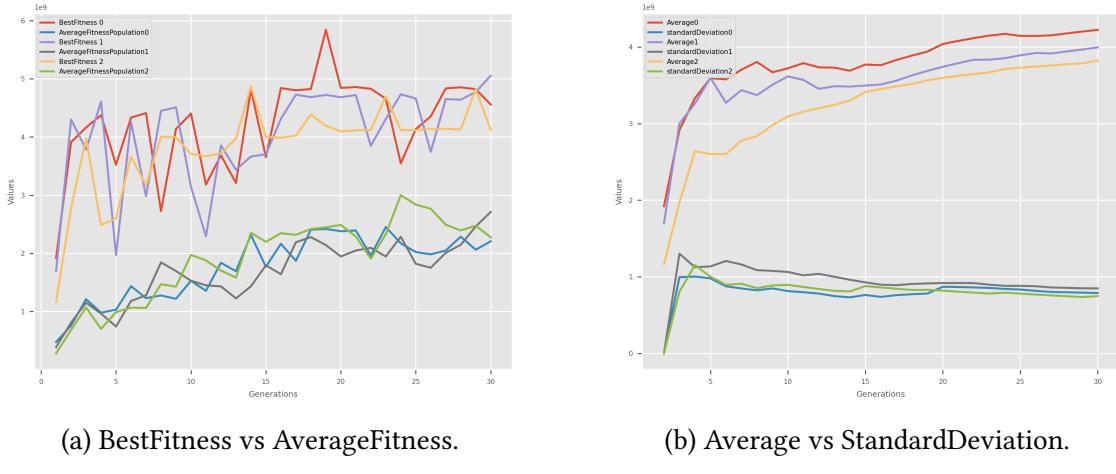


Figura 11: Experiência 5.

Nesta iteração pretendia-se ver o comportamento do nosso algoritmo à medida que a probabilidade de *crossover* varia.

Do ponto de vista teórico, com uma alta probabilidade de *crossover*, há uma maior chance de combinar informações genéticas de diferentes indivíduos em cada geração. Isso permite uma rápida disseminação de características benéficas pela população, o que acelera a convergência para soluções ótimas. Existe também uma maior probabilidade de diferentes partes genéticas interajam com mais frequência, o que facilita a exploração de diferentes combinações de características genéticas e aumenta a diversidade genética.

Em contra partida, o facto de convergirmos rapidamente para uma solução pode fazer com que algumas características interessantes de explorar sejam perdidas no cruzamento dos progenitores.

Se observarmos os resultados das experiências com atenção, conseguimos ver que as experiências com valores mais altos de *crossover* apresentam uma curva de convergência maior do que experiências com *crossover* baixo, como por exemplo: a experiência 1 e 5. A experiência 5 rapidamente entra no momento de estabilidade.

No decorrer das experiências, observamos que se a convergência for lenta, os indivíduos têm tendência a manter as suas características e ficam menos suscetíveis a ruídos. Sendo assim, se aumentarmos o número de gerações e permitirmos uma convergência lenta talvez iremos obter uma solução mais perto do ótimo.

3.4. Terceira Iteração

Nas iterações anteriores, nós especulamos um pouco de como seria o melhor carro para completar o cenário e exprimimos na nossa função de aptidão. Isso poderá ter sido um erro e desta vez iremos tentar com que seja o algoritmo a dizer qual é o melhor *shape* para o percurso. Sendo assim, definimos que a função de aptidão é diretamente proporcional à distância percorrida:

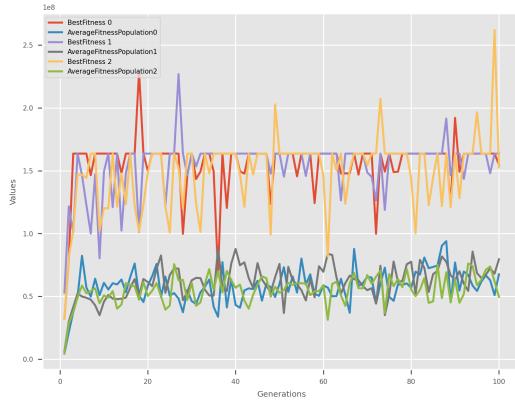
$$Fitness_i = (MaxDistance_i)^3$$

Desta vez, o objetivo é fazer com que o algoritmo converja para a solução ótima lentamente, de forma a que aproveite o máximo de todas as soluções que possam ser ótimas ou que contenham características que possam forjar a solução ótima.

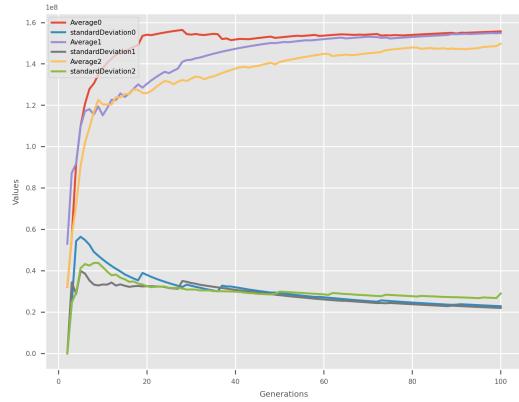
Sendo assim, iremos aumentar o número de gerações e diminuir os valores de *crossover* e mutação às nossas experiências.

	Mutação	Elitismo	Crossover	Número Gerações	
Experiência 1	0.01	2	0.2	100	
Experiência 2	0.02				
Experiência 3	0.05				
Experiência 4	0.07				
Experiência 5	0.02		0.3		
Experiência 6	0.05				
Experiência 7	0.07				

Tabela 3: Tabela de experiências

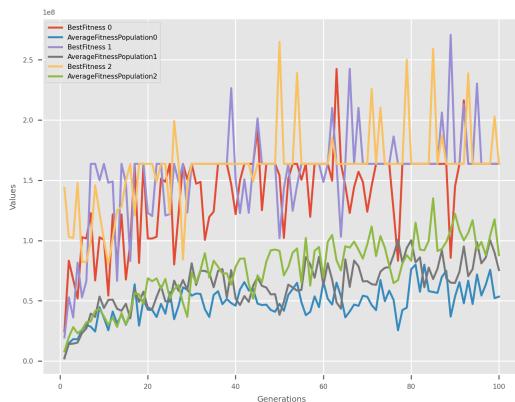


(a) BestFitness vs AverageFitness.

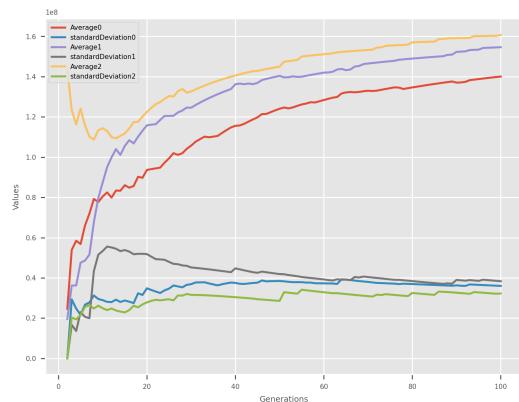


(b) Average vs StandardDeviation.

Figura 12: Experiência 1.

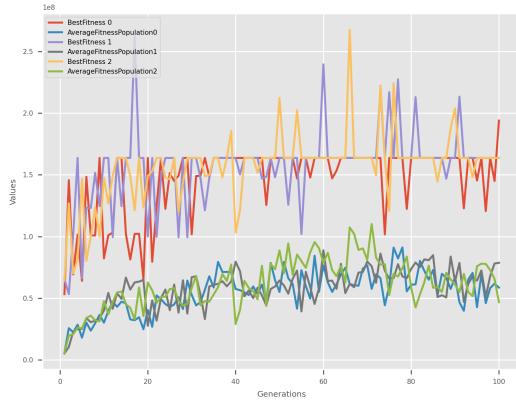


(a) BestFitness vs AverageFitness.

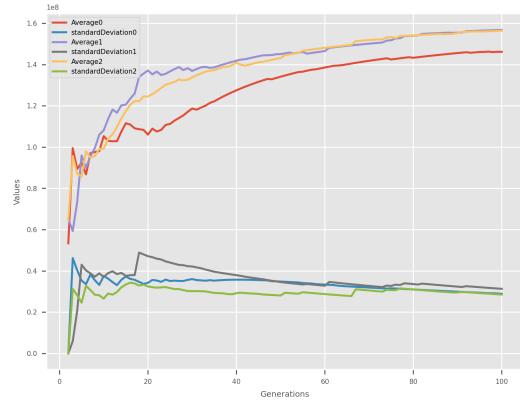


(b) Average vs StandardDeviation.

Figura 13: Experiência 2.

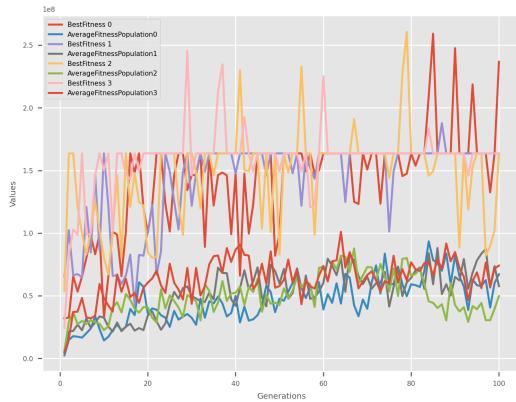


(a) BestFitness vs AverageFitness.

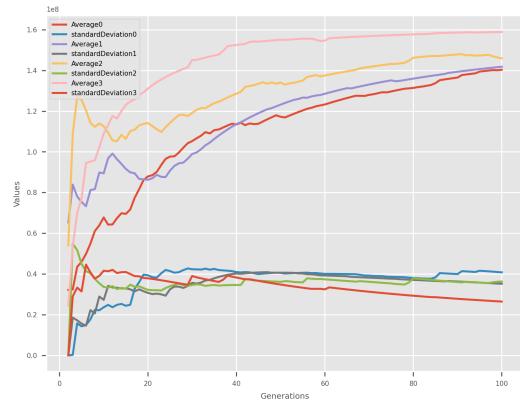


(b) Average vs StandardDeviation.

Figura 14: Experiências 3.

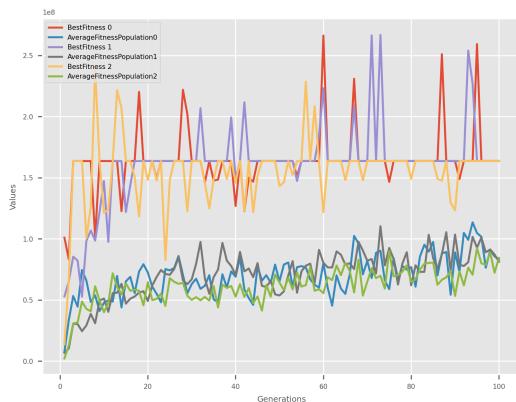


(a) BestFitness vs AverageFitness.

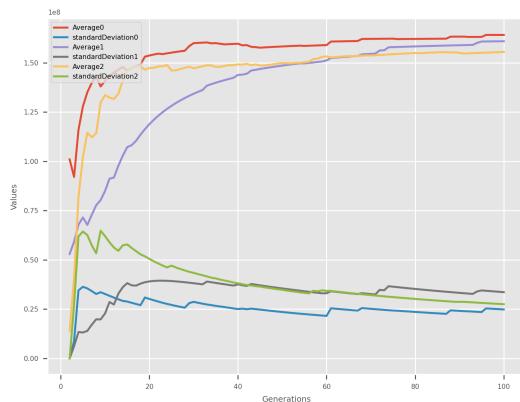


(b) Average vs StandardDeviation.

Figura 15: Experiência 4.



(a) BestFitness vs AverageFitness.



(b) Average vs StandardDeviation.

Figura 16: Experiência 5.

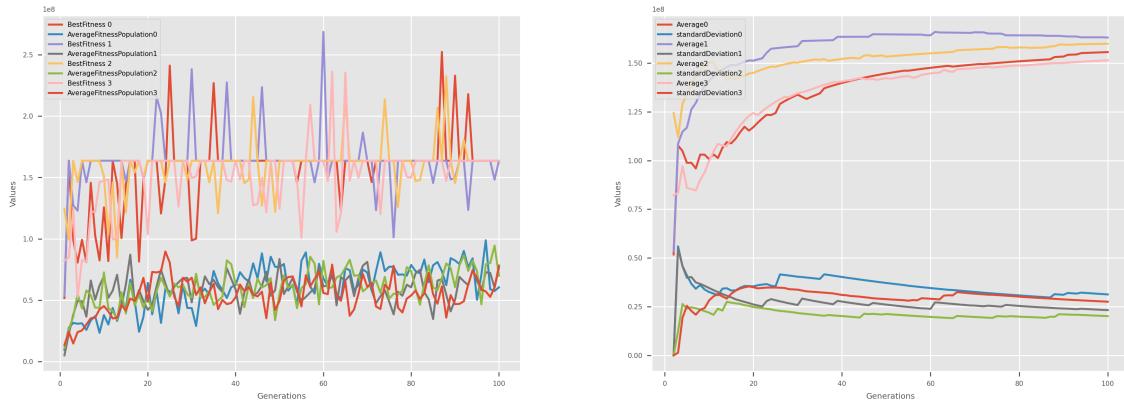


Figura 17: Experiência 6.

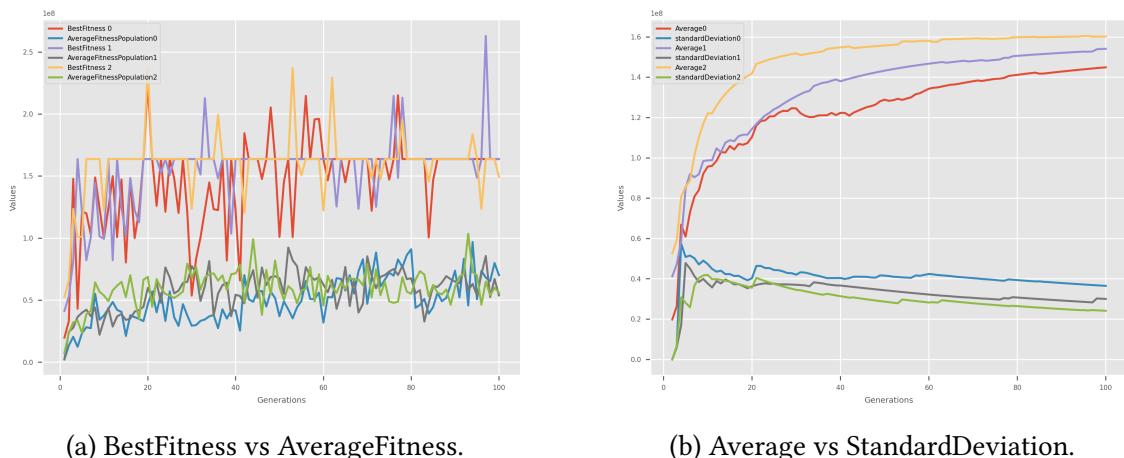


Figura 18: Experiência 7.

Podemos analisar assim que a nossa intuição estava certa, se progredirmos lentamente conseguimos convergir para uma solução ótima. Algumas experiências precisavam de mais algumas gerações, tais como: experiência 2, experiência 4 e experiência 7, mas elas já estavam todas na zona de convergência.

Ambas as experiências apresentam bons resultados, mas as que se destacam um pouco são as 3 e 6. Elas conseguem ter o seu *average fitness* da população a crescer em média. No entanto, na experiência 3 o desvio padrão em relação às gerações anteriores não tem grandes alterações.

Sendo assim, escolhemos a experiência 3 para levar-mos como solução final.

3.5. Resultado final

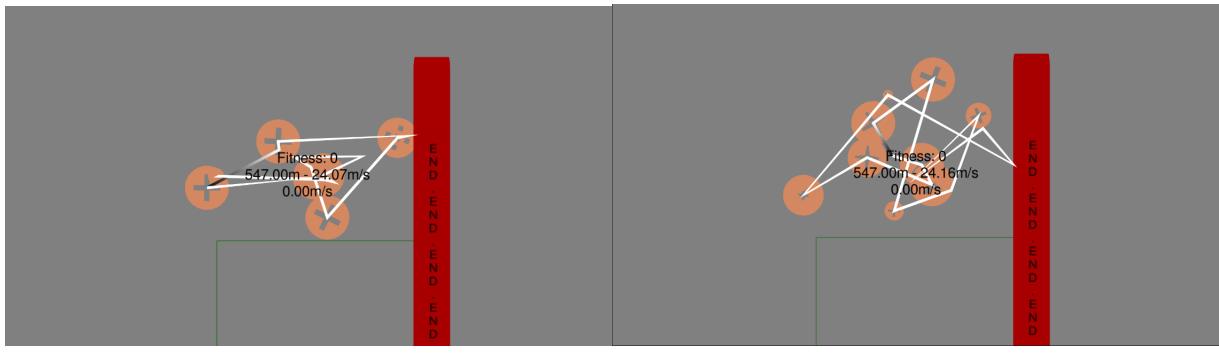


Figura 19: Experiência 7.

Como podemos ver, estes carros apresentam características distintas, mas são capazes de concluir o percurso com sucesso.

4. Experimentação e análise - HillRoad

4.1. Primeira Iteração

Após a análise e discussão das experiências realizadas no cenário *GapRoad*, começamos a experimentação no cenário *HillRoad*. Decidimos inicialmente fazer testes a partir de uma função de aptidão que obteve bons resultados no cenário anterior, sendo esta a seguinte:

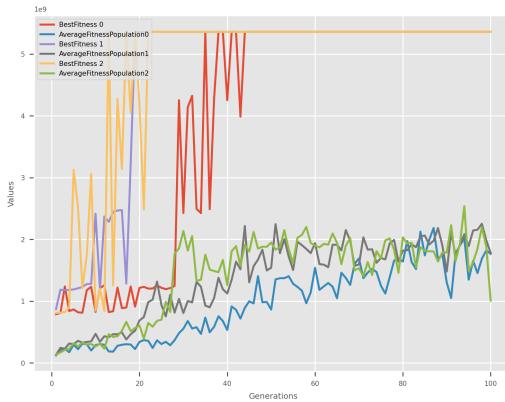
$$Fitness_i = (MaxDistance_i)^3$$

Mais uma vez, o objetivo é fazer com que o algoritmo converja para a solução ótima lentamente, de forma a que aproveite o máximo de todas as soluções que possam ser ótimas, explorando uma característica muito importante para o término do percurso (*MaxDistance*) e aplicamos pressão seletiva na roleta com a elevação ao cubo.

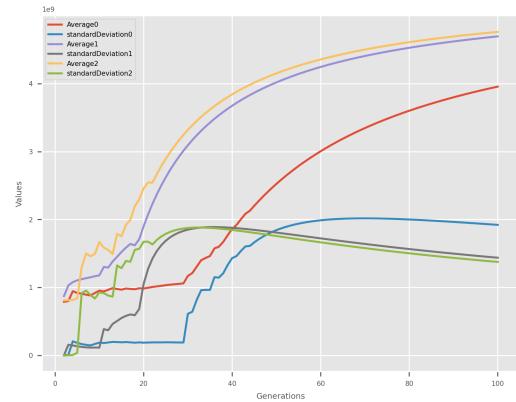
Em termos de parâmetros do algoritmo evolucionário utilizados para as experiências, foram utilizados os parâmetros iguais à experiência 3 do enunciado.

	Mutação	Elitismo	Crossover	Número Gerações
Experiênci	0.05	2	0.9	100

Tabela 4: Tabela de experiências



(a) BestFitness vs AverageFitness.



(b) Average vs StandardDeviation.

Figura 20: Experiência 1.

4.2. Segunda Iteração

Apesar de já ser possível realizar o objetivo de chegar ao fim do cenário na iteração anterior, os resultados mostravam-se tardios, apenas sendo possível existir um agente de uma geração a chegar à meta depois das 30 gerações.

Na iteração anterior reparamos que os carros que geraram as soluções ótimas apresentavam todos as mesmas características: eram baixos, compactos e rápidos. Para poder subir as montanhas os carros têm de ter velocidade, caso contrário ele não será capaz de subir. No entanto, após uma subida vem uma descida e os carros têm tendência a dar um salto. Para que não fiquem imobilizados, os carros precisam de ter um centro de massa estável, logo carros baixos e compactos terão melhor desempenho.

Após mais experimentos conseguimos chegar a uma função de aptidão que representa essa informação sobre os indivíduos.

$$Fitness_i = (2 * MaxDistance_i + 1.5 * MaxVelocity_i - CarMass_i)^3$$

Como já foi concluído que a distância máxima percorrida tem sempre um grande impacto, daí ser o atributo que priorizamos mais. Como o carro tem de ter uma velocidade significativa, atribuímos um bom peso à velocidade máxima para que tivesse influencia no valor final, cujo obtemos experimentalmente. Visto que a massa é diretamente proporcional ao tamanho do carro, queremos penalizar os carros pelo valor da massa que eles têm para que tendam a ficar pequenos e compactos.

Foram utilizados os mesmos parâmetros que a interação anterior mas desta vez para 30 gerações:

	Mutação	Elitismo	Crossover	Número Gerações
Experiênciia	0.05	2	0.9	30

Tabela 5: Tabela de experiências

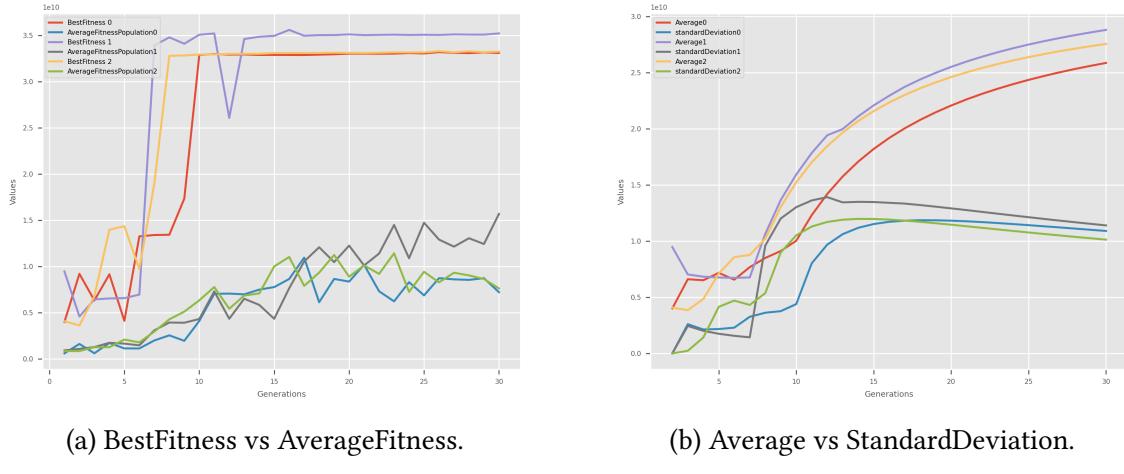


Figura 21: Experiência 1.

Neste tipo de problema, devido à sua simplicidade, dificilmente um carro termina por sorte. Sendo assim, podemos fazer com que a convergência do carro seja mais rápida, poupando assim, recursos computacionais.

Neste caso, verificamos por ambas as experiências realizadas, que quando um indivíduo termina, todos os seus descendentes acabam por conseguir terminar. Nas experiências desta iteração verificou-se o término de um agente entre a 5^a e a 10^a geração constantemente, sendo assim estes os melhores resultados que obtemos para este cenário *HillRoad*.

4.3. Resultadofinal

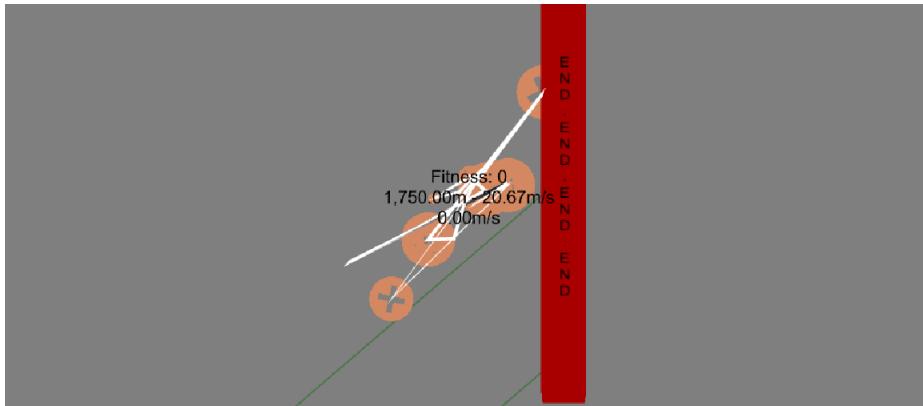


Figura 22: Agente que termina o cenário HillRoad

Como podemos ver a nossa intuição sobre as características do carro estava certa: carro baixo, compacto, ponto de equilíbrio no centro do carro e rápido fez com que facilmente chegássemos a uma solução capaz de terminal o percurso.

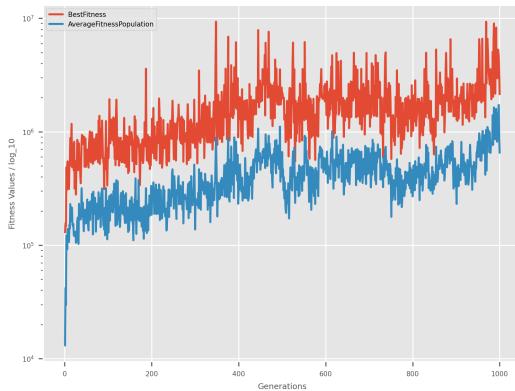
5. Experimentação e análise - ObstacleRoad

Depois de termos obtido bons resultados na experimentação do cenário *HillRoad*, decidimos começar a explorar o cenário *ObstacleRoad*.

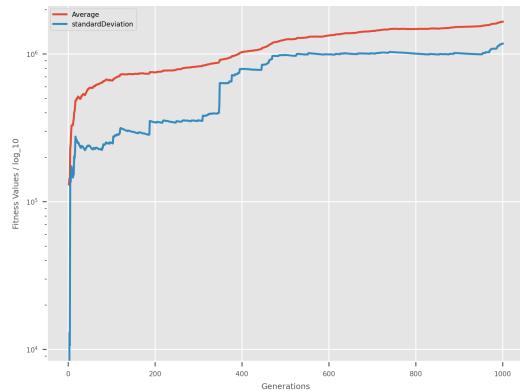
5.1. Primeira Iteração

Para uma primeira iteração, optamos por começar de uma forma parecida ao cenário anterior e explorar inteiramente a distância máxima que os agentes conseguem atingir, que achamos ser uma métrica sempre importante. Mais uma vez utilizamos os parâmetros do algoritmo evolucionário relativos à experiência 3 do enunciado.

$$fitness_i = (MaxDistance_i)^3$$

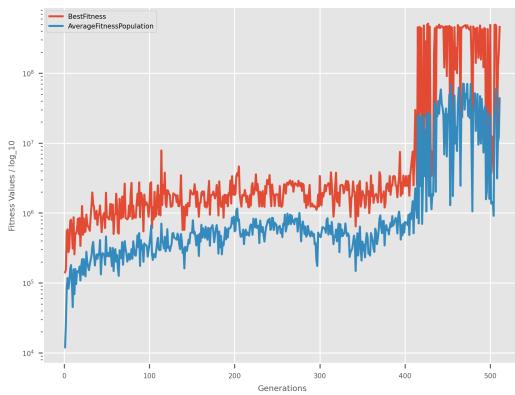


(a) BestFitness vs AverageFitness.

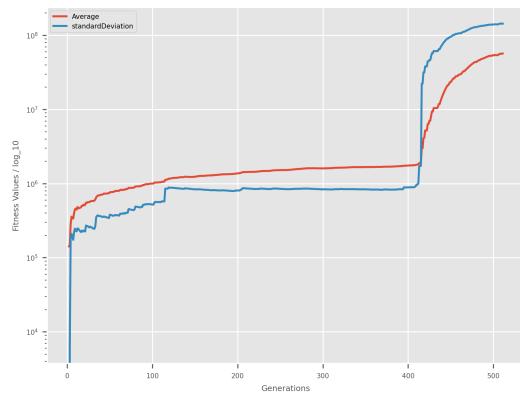


(b) Average vs StandardDeviation.

Figura 23: Experiência 1.

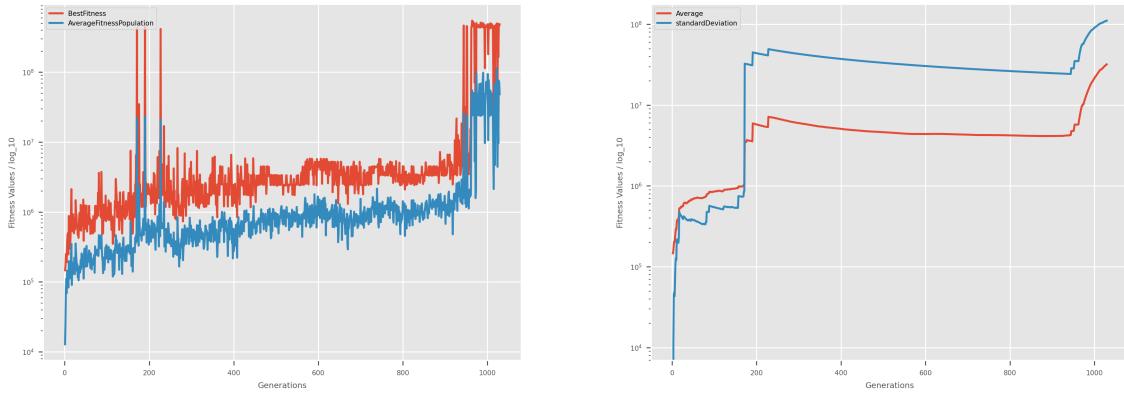


(a) BestFitness vs AverageFitness.



(b) Average vs StandardDeviation.

Figura 24: Experiência 2 com escala logarítmica.



(a) BestFitness vs AverageFitness.

(b) Average vs StandardDeviation.

Figura 25: Experiência 3 com escala logarítmica.

Nas 3 experiências realizadas, ao longo de 1000 gerações, foi possível obter carros a chegar ao fim em 2 delas (experiências 2 e 3), não tendo verificado nenhum carro a chegar à meta na primeira experiência. Verificamos também que a partir do momento que um carro conseguia chegar ao fim prosseguia-se uma grande afluência de gerações em que pelo menos um agente conseguia cumprir o objetivo.

Através dos resultados obtidos percebemos uma evolução muito lenta na distância máxima que os agentes conseguiam atingir ao longo das gerações, tornando-se evidente que através desta expressão não seria possível obter carros a chegar à meta nas 30 gerações (ou perto) como aconteceu nos cenários anteriores. Concluímos que seria necessário explorar outras funções de aptidão baseadas em mais informações do ambiente de modo a possivelmente obter melhores resultados.

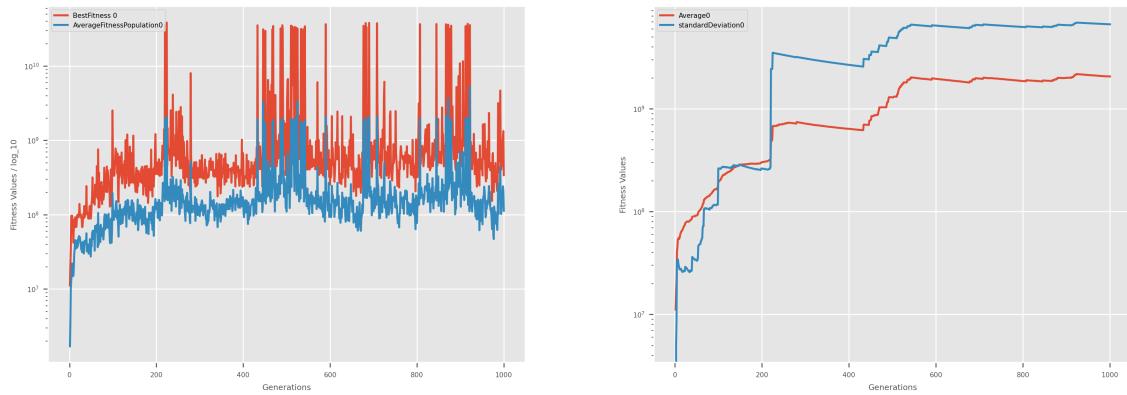
5.2. Segunda Iteração

Pelas experiências que obtiveram agentes a chegar ao fim do percurso, pudemos observar algumas características comuns que os tornavam capazes para cumprir o objetivo. Os carros normalmente eram baixos, com grande velocidade e apresentavam muitas rodas maioritariamente na parte da frente que lhes permitiam subir constantemente os blocos, possuindo menos no centro e na parte de traz, mas ainda assim estas mostraram-se importantes para o equilíbrio do carro.

A partir destas informações exploramos várias funções de aptidão e de entre as mesmas a que demonstrou melhores resultados foi:

$$\text{fitness}_i = (4 * \text{MaxDistance}_i + 2 * \text{MaxVelocity}_i + \text{NumberOfWheels}_i)^3$$

Nesta função continuamos a priorizar distância máxima atingida pelos agentes, mas desta vez atribuímos peso também à velocidade máxima atingida e ao número de rodas que os mesmos apresentam (quantas mais melhor), de acordo com as ideias de carro perfeito discutidas anteriormente.



(a) BestFitness vs AverageFitness.

(b) Average vs StandardDeviation.

Figura 26: Experiência 1.

Na tentativa de obter uma função de aptidão melhor, fizemos diversos testes com diversas variações nas informações do ambiente mas que ainda assim não se mostraram capazes de melhorar os resultados relativamente aos já atingidos nas iterações anteriores.

Listamos algumas das funções de aptidão testadas. Utilizando estas funções, ao longo de milhares de gerações, não foi possível obter agentes ótimos para o cenário e a evolução mostrou-se muito pequena ou mesmo inexistente, logo não as consideramos relevantes.

FitnessFunction
$(MaxDistance + MaxVelocity + CarMass)^3$
$(MaxDistance/200 + NumberOfWheels/15 + MaxVelocity/10 + IsRoadComplete + CarMass/600)^3$
$(MaxDistance + MaxVelocity * 2 + CarMass + (5/NumberOfWheels) * 50)^3$
$(MaxDistance + MaxVelocity * 4 + (20000/CarMass))^3$
$(MaxDistance + 600 - CarMass/3 + MaxVelocity * 3)^3$
$MaxDistance + NumberOfWheels * 10 + MaxVelocity + CarMass$

Tabela 6: Exemplos de funções que não obtiveram bons resultados

Apesar das várias tentativas não conseguimos encontrar uma função de aptidão ideal para atingir o final do percurso nas 30 gerações. As nossas melhores soluções mostraram um progresso lento até eventualmente atingirem o objetivo.

Conseguimos no entanto, apresentar alguns agentes que conseguiram terminar o cenário.

5.3. Resultado final



Figura 27: Agentes que terminam o cenário ObstacleRoad

Tal como discutido anteriormente, as características comuns aos indivíduos ótimos passavam por apresentar boa velocidade, grande número de rodas principalmente na parte da frente, com algumas a servir de equilíbrio na parte de traz, e um formato baixo e compacto.

6. Conclusão

Globalmente, considera-se que os objetivos propostos para este projeto foram, de um modo geral, alcançados. Apesar do fato de não termos conseguido encontrar uma função de aptidão que consistentemente resolvesse o cenário *ObstacleRoad*, este projeto permitiu-nos entender o mecanismo por trás dos algoritmos evolucionários, a influência que várias heurísticas (mutação, crossover, elitismo, ...) têm no seu desempenho.

Entendemos também a importância das funções de aptidão para a resolução de um problema, a atribuição de diferentes pesos para diferentes características e a sua influência na evolução dos agentes, bem como a natureza estocástica destes métodos.

Através da aplicação destes algoritmos, é possível resolver uma ampla gama de problemas. Permitiu explorar soluções em um espaço de busca vasto e multidimensional, para além de se adaptarem a mudanças no ambiente e evoluírem gradualmente em direção a soluções ótimas.

Muitas outras experiências foram realizadas, algumas que não tiveram bons resultados, outras que nos conduziram às presentes neste relatório. Elas encontram-se neste link <https://we.tl/t-tuw48ww33Z>, disponível no máximo sete dias após a entrega deste relatório.

Referências

- [1] Geek for Geeks Website. *Mutation Algorithms for Real-Valued Parameters (GA)*. <https://www.geeksforgeeks.org/mutation-algorithms-for-real-valued-parameters-ga/>. 2023.
- [2] Natsuki Higashi e Hitoshi Iba. «Particle swarm optimization with Gaussian mutation». Em: *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*. IEEE. 2003, pp. 72–79.
- [3] Padmavathi Kora e Priyanka Yadlapalli. «Crossover operators in genetic algorithms: A review». Em: *International Journal of Computer Applications* 162.10 (2017).
- [4] tutoriais.edu. *Algoritmos Genéticos - Seleção de País*. <https://tutoriais.edu.lat/pub/genetic-algorithms/genetic-algorithms-parent-selection/algoritmos-geneticos-selecao-de-pais>. 2023.