Relatório Trabalho Prático 2

Rolling in the Hill

Evolutionary Edition

Fundamentos da Inteligência Artificial / Introdução à Inteligência Artificial

2022/2023

João Miguel Fernandes Moura || 2020235800 || joaomoura@student.dei.uc.pt || PL4

Filipe David Amado Mendes || 2020218797 || filipemendes@student.dei.uc.pt || PL1

Miguel Pedro de Noronha Pião || 2018280861 || piao@student.dei.uc.pt || PL3

Índice

Introdução

Hoje em dia a Inteligência Artificial está presente em diversos contextos do quotidiano, mesmo que nem sempre se interaja diretamente com as mesmas. Uma das matérias dessa área são os agentes evolucionários e é nisso que este trabalho prático assenta.

A partir de um algoritmo desenvolvido para o efeito, a nossa missão passou por fazer evoluir veículos para chegar ao final de três cenários diferentes: uma estrada com buracos progressivamente mais largos, uma composta por colinas e outra com obstáculos progressivamente maiores. Neste relatório analisamos os resultados das diversas experiências e documentamos o processo de desenvolvimento dos algoritmos capazes de gerar veículos que completem os cenários. Havendo sempre muitas opções e decisões a tomar relativamente à construção das funções de *fitness*, este relatório clarifica a nossa abordagem para criar a função.

Modelação e desenvolvimento do Algoritmo Genético

Para começar, optamos por implementar o pseudo-código fornecido nos locais devidos. Seguidamente, começamos a modelar as primeiras funções para calcular o *fitness*.

Para explicar o processo seletivo implementado. Após o início da execução, em cada geração, o algoritmo irá selecionar os pais no script Meta1/Roulette.cs, de forma estocástica. Aqui, são analisadas porções aleatórias de população já testada, e, para cada porção, escolhe-se o indivíduo mais apto(*fitness)* de acordo com o valor proveniente da função desenvolvida. Deste modo, obtemos uma lista dos melhores indivíduos em cada porção, que irá determinar a aptidão da geração seguinte. Tendo os pais selecionados, necessitamos de recombinar os seus genótipos de forma a formar os filhos. Isto acontece no script Meta1/UniformCrossover.cs. Com os filhos criados, e para estes não ficarem exatamente como sendo uma combinação dos seus pais, o algoritmo altera aleatoriamente alguns valores do genótipo utilizando um operador de mutação *Gaussiano* implementado em Meta1/GaussianMutation.cs. A geração que se sucede não será apenas constituída por filhos da geração anterior, os melhores exemplos da geração anterior sobrevivem e são testados novamente, isto é implementado através de um mecanismo de Elitismo presente em Meta1/Elitism.cs. Tal como referido acima, para calcularmos o valor de *fitness* de um veículo, necessitamos de uma função para o efeito. Isso é definido em CarFitness.cs.

Parâmetros não-treináveis

Para realizar as experiências foi nos fornecida uma tabela que contém parâmetros não--treináveis que influenciam a *performance* do algoritmo que desenvolvemos, ajudando a identificar um padrão removendo possíveis *outliers*. Em todas elas a probabilidade de recombinação foi de 0.9 e o número de gerações 30. O parâmetro mutação determina a probabilidade de mutação e o elitismo a quantidade de indivíduos que sobrevivem.

Após testar diferentes algoritmos de aptidão para todas as experiências presentes na Fig.1, três vezes para conseguir estabelecer um padrão, dado estas experiências serem de natureza estocástica, conclui-se que os resultados mais estáveis, isto é, com menos outliers, e satisfatórios se encontram na experiência 3 da tabela. O *fitness* tem um crescimento mais constante em todos os cenários. Esta experiência ter os melhores resultados não é surpreendente dado ter elitismo, que permite a sobrevivência de indivíduos mais aptos, e uma probabilidade de mutação de 0.05, visto que mais elevado poderia provocar uma maior perda de genes com qualidade. As experiências 1 e 2 dado não terem elitismo apresentam os piores resultados em todos os cenários, pois impede a sobrevivência de indivíduos aptos.

	Mutação	Elitismo	Crossover	Número Gerações
Experiência 1	0.05	0	0.9	30
Experiência 2	0.2			
Experiência 3	0.05	2		
Experiência 4	0.2			
Experiência 5	0.05			

Fig.1-Tabela de parâmetros não-treináveis

As Experiências

Foram desenvolvidos diferentes algoritmos para cada uma das pistas, tendo sido todos testados com os diferentes parâmetros, para a gap road foram feitos 3 algoritmos, para a hill road desenvolvemos 2, estes foram corridos 3 vezes cada com os 4 diferentes parâmetros com corridas de 30 gerações cada. Em contraste, para o obstacle road apenas foi feito um algoritmo que foi testado com os mesmos diferentes parâmetros mas com 500 gerações.

Para análise dos resultados serão apresentados gráficos que descrevem a qualidade do melhor indivíduo e a qualidade média da população ao longo das gerações da melhor experiência sendo os restantes deixados em anexo. Cada gráfico apresenta três linhas relativas a cada uma das tentativas da experiência.

No total temos 6 algoritmos diferentes de *fitness* para os vários cenários.

Cenários

Gap Road

Para o cenário Gap Road, optámos por utilizar um algoritmo simples como base para os outros algoritmos de cálculo do fitness, o algoritmo 1. A partir deste, desenvolvemos outros dois, ajustando a expressão de acordo com os parâmetros que achámos mais relevantes.

Analisando os resultados devolvidos por cada algoritmo, concluímos que a experiência 3 mostra uma consistência maior em relação às outras. Além de ser a experiência em que, em média, mais carros finalizam a corrida, observamos também que a evolução dos carros ao longo das gerações é mais linear.

De seguida, são apresentados todos os gráficos das experiências realizadas para cada um dos algoritmos. Em anexo segue também um ficheiro com as tabelas de resultados e os respetivos gráficos, caso seja necessária uma análise mais profunda das experiências realizadas.

Algoritmo 1

No algoritmo 1 optamos por usar uma condição, criando uma diferença significativa entre o fitness dos carros que cumprem os critérios desejados e outros.

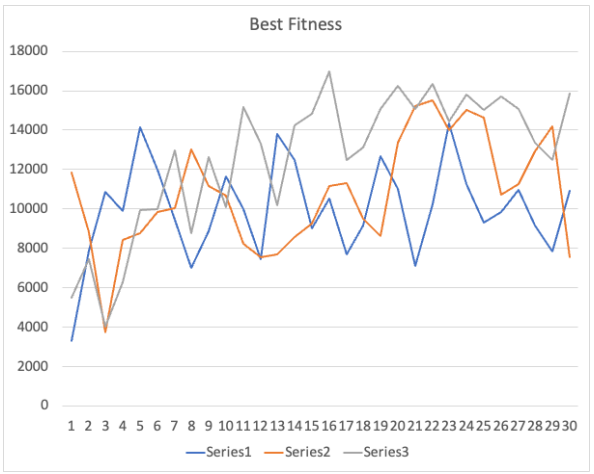
Depois de correrrmos algumas experiências sem utilizar nenhum algoritmo para calcular o fitness, percebemos que os carros entre uma massa de 250 e 450 tinham maior probabilidade de chegar ao final. Além disso, para excluir os carros que já começam a corrida parados, verificamos também na condição se a velocidade máxima atingida pelo carro é maior que zero.

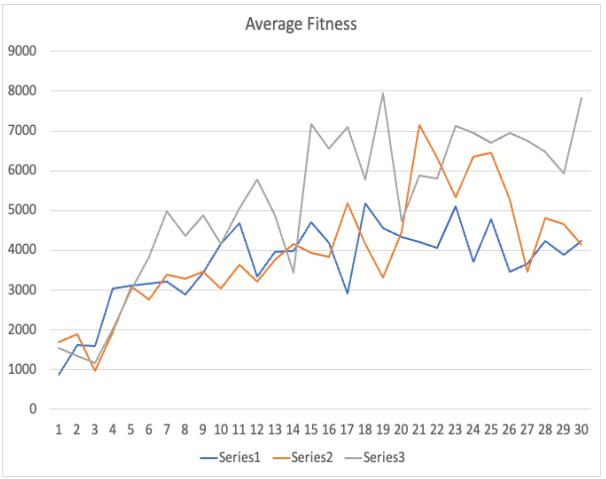
```
if ((CarMass >= 250 || CarMass <= 450) && MaxVelocity > 0){

    fitness = CarMass + (MaxDistance * MaxVelocity) + (((4 * MaxDistance) / MaxVelocity + 1) * IsRoadComplete) ;

}else{

    fitness =  (MaxDistance * MaxVelocity) + (((2 * MaxDistance) / MaxVelocity + 1) * IsRoadComplete);

}
```

Sendo este o primeiro algoritmo desenvolvido, considerámos importante dar bastante valor à distância e à velocidade máxima dos carros, isto porque carros que chegam ao final mais rapidamente, cumpriram de forma mais eficaz o seu objetivo. Além disso, para distinguir dos carros que não cumprem a condição inicial, é também somada a massa do carro, dando assim algum valor a esse atributo.

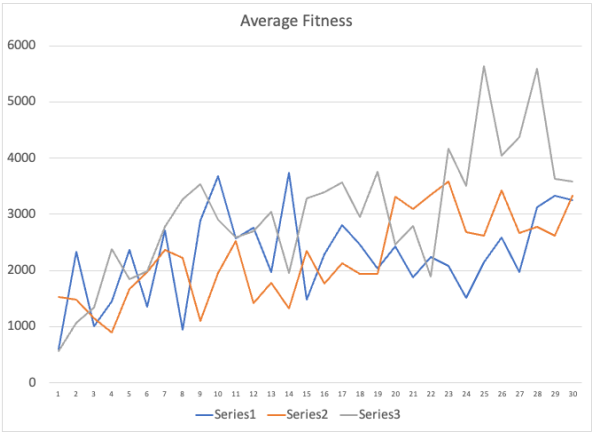
Considerámos importante também, fazer a distinção entre os carros que chegam à linha da meta e os que não conseguem terminar o percurso. Para tal, fizemos uso da variável IsRoadComplete numa parcela da expressão. Para realçar estes carros, foram também utilizados quantificadores nesta parcela. Os valores utilizados foram escolhidos depois de alguns testes, pois mostram uma diferença significativa nos diferentes carros.

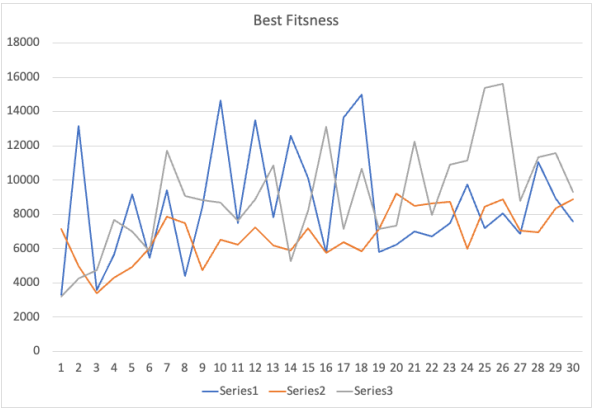
Experiência 1



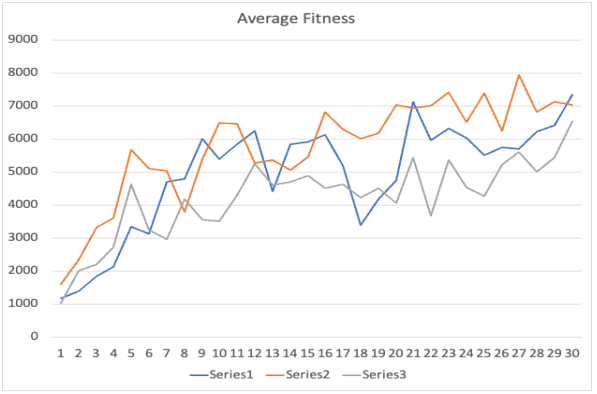


Experiência 2

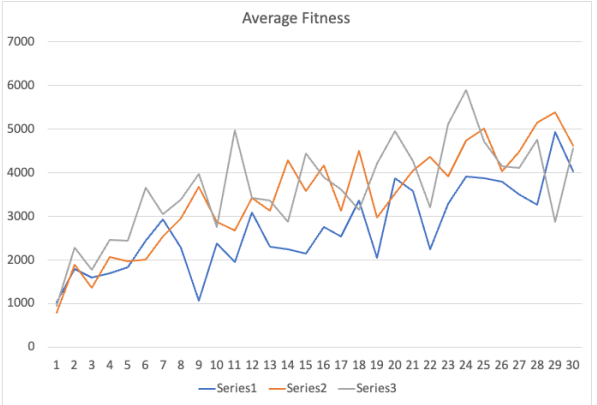




Experiência 3



Experiência 4



Tal como referido anteriormente, a experiência 3 foi a que mostrou resultados mais consistentes entre as várias execuções. Apesar de os resultados obtidos não terem sido ideais, conseguimos observar uma evolução contínua. Observando também o gráfico do

Average Fitness da experiência 3, verificamos que a média do fitness de cada geração também sobe, mostrando assim que a geração toda de carros tende a evoluir, e não apenas alguns.

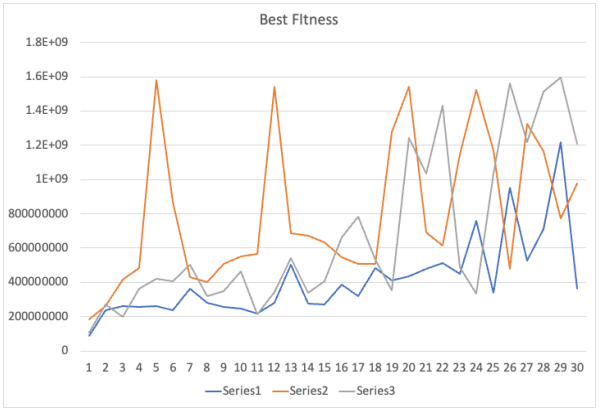
Algoritmo 2

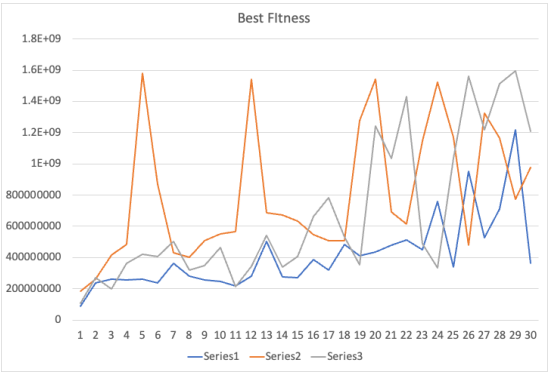
Seguindo a lógica do primeiro algoritmo, utilizámos a mesma condição inicial para separar os carros. Porém, neste algoritmo optámos por ter também em conta o tempo que os carros levam a chegar ao final, garantindo assim que carros mais rápidos têm mais peso na roleta que carros mais lentos.

```
if ((CarMass >= 250 || CarMass <= 450) && MaxVelocity > 0){

    fitness = (float) Math.Pow(CarMass + ((MaxDistance * MaxVelocity)/(MaxDistanceTime + 1))+ (((4 * MaxDistance) / (MaxVelocity+1)) * IsRoadComplete), 3);

}else{

    fitness = ((MaxDistance * MaxVelocity) / (MaxDistanceTime + 1)) + (((2 * MaxDistance) / (MaxVelocity+1)) * IsRoadComplete);

}
```

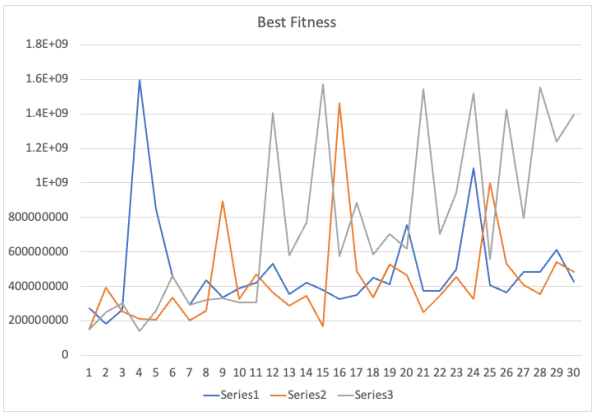
Além disso, recorremos ainda à exponencialização da expressão de cálculo para os carros que cumprem os requisitos iniciais. Este uso de expoentes, ajuda-nos a garantir que os carros com as características "ideais" têm uma probabilidade maior de passar os seus genes às gerações seguintes.

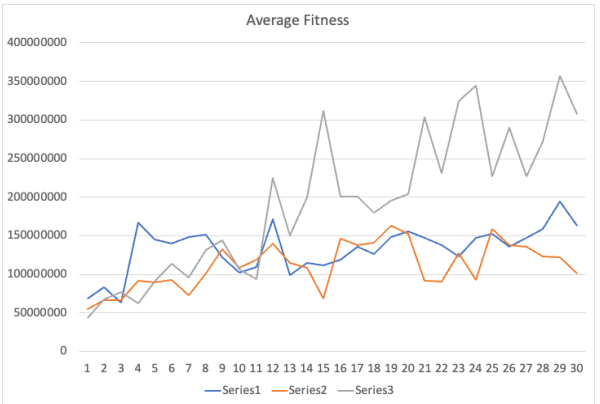
Experiência 1



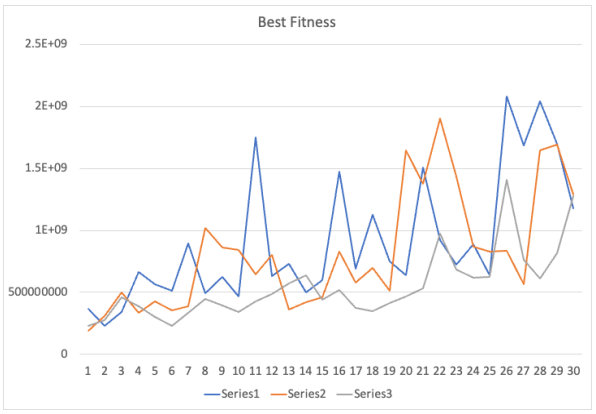


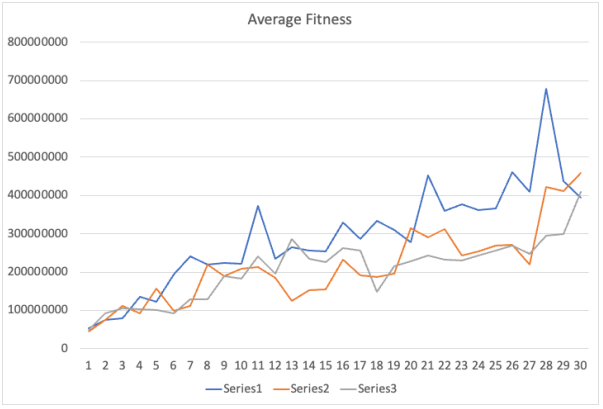
Experiência 2



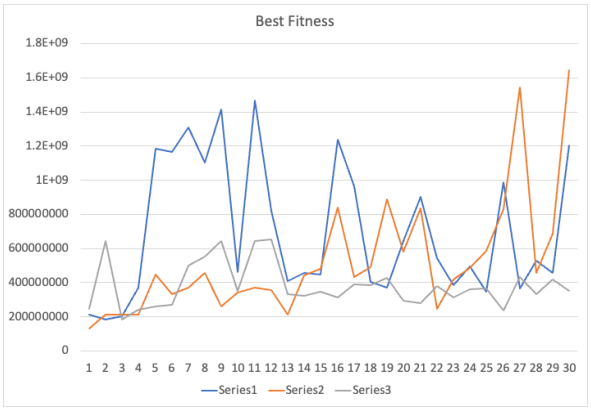


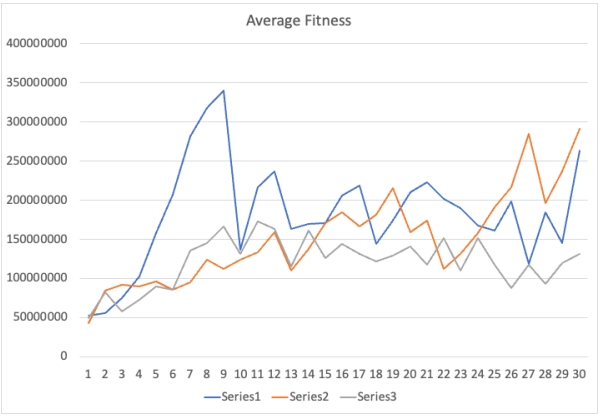
Experiência 3





Experiência 4





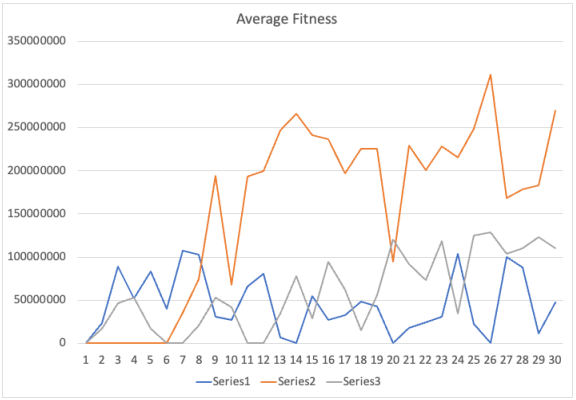
Depois da testagem do segundo algoritmo, podemos analisar os resultados e compará-los com os do algoritmo anterior. As experiências 1, 2 e 4, mostram resultados pouco promissores devido à sua inconsistência. Por outro lado, na terceira experiência podemos observar uma clara melhoria das gerações. Apesar de os resultados no gráfico Best Fitness mostrarem alguns carros com o fitness alto e outros com um fitness bastante mais baixo, analisando o gráfico Average Fitness vemos que a evolução das gerações como um todo é constante e destaca-se relativamente às outras experiências.

Algoritmo 3

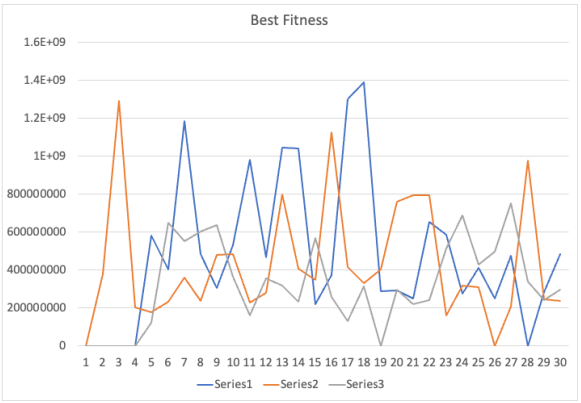
No desenvolvimento do algoritmo 3 considerámos que poderíamos melhorar a condição inicial, fazendo assim uma triagem mais eficiente dos carros. Para tal, decidimos valorizar também o atributo da distância nessa verificação. Assim, partindo do segundo algoritmo, adicionámos esta nova condição.

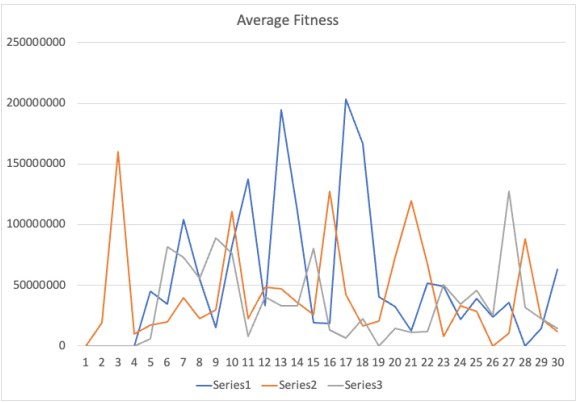
```
if ((CarMass >= 250 || CarMass <= 450) && MaxVelocity > 0 && MaxDistance > 400){

    fitness = (float) Math.Pow(CarMass + ((MaxDistance * MaxVelocity)/(MaxDistanceTime + 1))+ (((4 * MaxDistance) / (MaxVelocity+1)) * IsRoadComplete), 3);

}else{

    fitness = ((MaxDistance * MaxVelocity) / (MaxDistanceTime + 1)) + (((2 * MaxDistance) / (MaxVelocity+1)) * IsRoadComplete);

}
```

Experiência 1

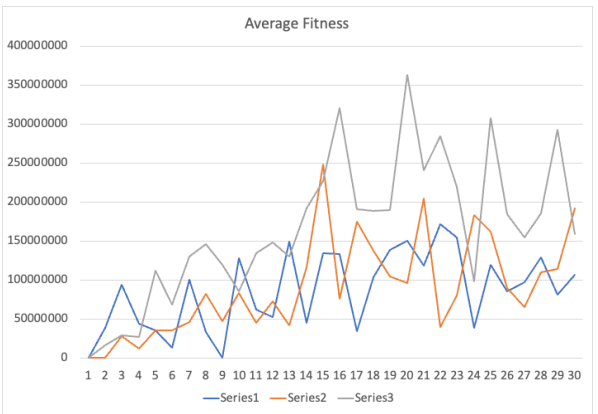


Experiência 2

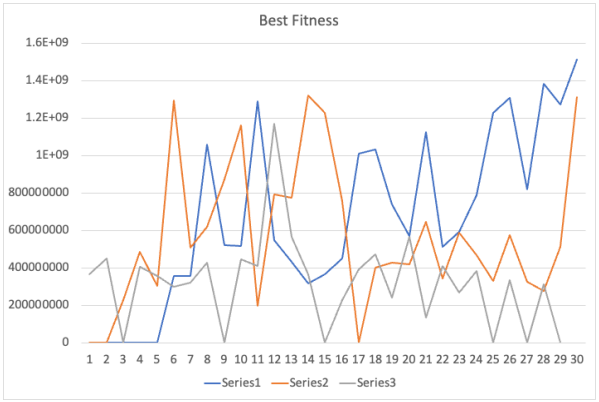


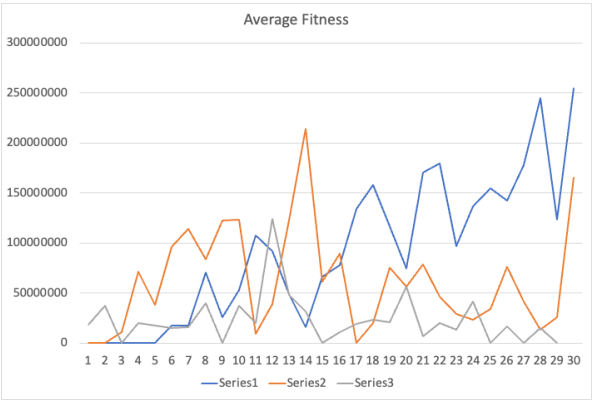


Experiência 3



Experiência 4





O terceiro e último algoritmo desenvolvido para este cenário mostrou resultados um pouco inesperados. Mais uma vez a experiência 3 destacou-se como sendo a mais adequada.

Correndo o terceiro algoritmo com os valores da terceira experiência resultou no maior número de carros a concluírem a corrida de todas as experiências realizadas. Porém, os resultados obtidos não foram de encontro às nossas expectativas. Apesar do número alto de carros a chegar à meta, podemos ver uma grande inconsistência ao longo das diferentes gerações de carros.

Hill Road

Para o cenário Hill Road, optámos por utilizar um algoritmo usando a mesma lógica que nos anteriores com as alterações que, depois de testes, achámos que fariam mais sentido para o melhor desenvolvimento dos carros.
De seguida, são apresentados todos os gráficos das experiências realizadas para cada um dos algoritmos. Em anexo segue também um ficheiro com as tabelas de resultados e os respetivos gráficos, caso seja necessária uma análise mais profunda das experiências realizadas.

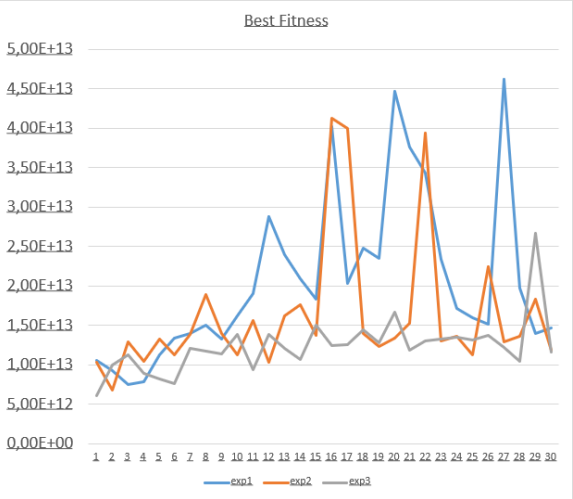
Algoritmo 1

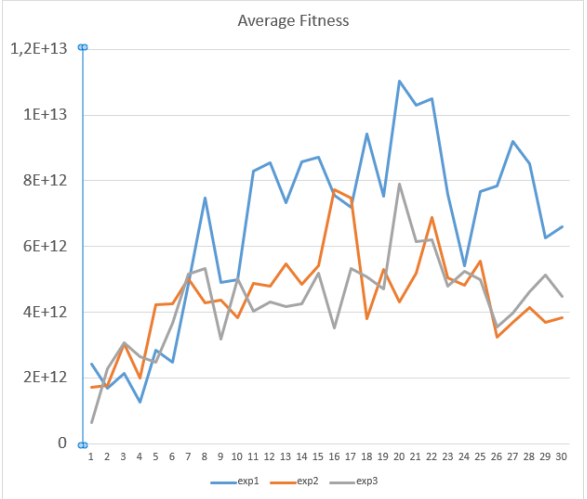
Seguindo a lógica dos algoritmos anteriores mantivemos a mesma importância no valor da distância e da velocidade máxima dos carros , utilizámos a mesma condição inicial e adicionàmos o número de rodas maior que 10 para separar os carros.

```
if ((mass >= 250 || mass <= 450) && MaxVelocity > 0 && wheels>10){

        fitness = (float) Math.Pow((mass + ((distance * MaxVelocity))-(5*min_time + 1))+ (((4 * distance) / (MaxVelocity+1)) * IsRoadComplete), 3);
    }

    else{

        fitness = (((distance * MaxVelocity))-(2*min_time + 1))+ (((2* distance) / (MaxVelocity+1)) * IsRoadComplete);
    }
```

Mantivemos também a exponencialização da expressão de cálculo para os carros que cumprem os requisitos iniciais.

Experiência 3



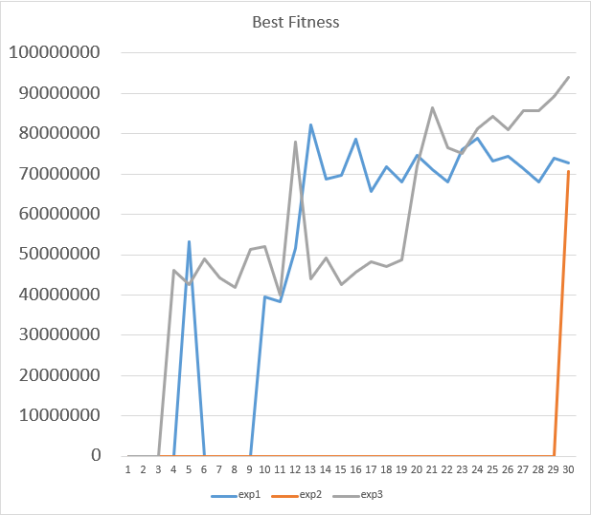


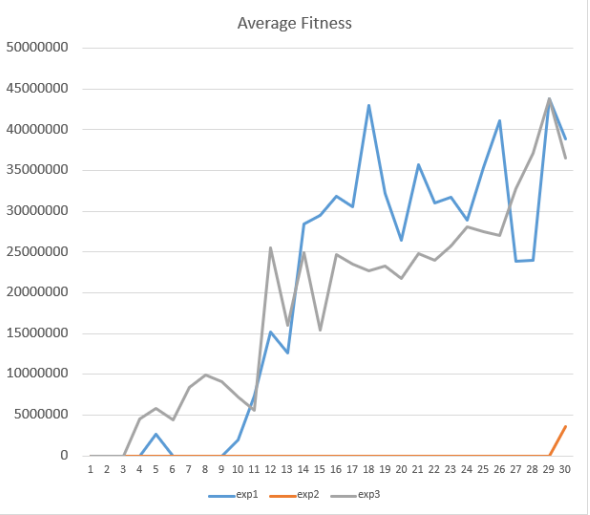
Algoritmo 2

As alterações deste algoritmo para o anterior são mínimas e revelam-se mais na expressão em si, a maior alteração mostra na reintegração de uma divisão aplicada para baixar os valores elevados de fitness que aconteceram no algoritmo anterior desta estrada.

```
if ((mass >= 250 || mass <= 450) && MaxVelocity > 0 && wheels>10)
        {
                fitness = (float) Math.Pow(((distance * MaxVelocity)/(mass * min_time + 1))+
(((4 * distance) / (MaxVelocity+1)) * IsRoadComplete), 3);

        }
        else
        {

                fitness = ((distance * MaxVelocity)/(mass * min_time + 1))+ (((2 * distance) /
(MaxVelocity+1)) * IsRoadComplete);

        }
```

Experiência 3





O segundo algoritmo desenvolvido para este cenário mostrou resultados um pouco inesperados. Mais uma vez a experiência 3 destacou-se como sendo a mais adequada.

Ambos os algoritmos revelaram que a terceira experiência resultou no maior número de carros a concluírem a corrida de todas as experiências realizadas. Porém, os resultados apresentados diferem dos que nós esperamos. Apesar do elevado número de carros a chegar à meta, a inconsistência mostrada nas gerações mostra uma possível ineficiência dos algoritmos.

Obstacle Road

Para construir o algoritmo de cálculo de *fitness* para este cenário utilizou-se uma estrutura lógica semelhante ao que já tinha sido utilizado nos restantes cenários, no entanto, foram tomadas algumas considerações para adaptar para o contexto deste cenário. Dado que os veículos necessitam de ser capazes de "trepar" os obstáculos, considerámos que os veículos deveriam ter uma grande quantidade de rodas para que, caso se virem, tenham capacidade de continuar a mover-se. Para além da distância, que é sempre relevante dado o objetivo principal dos indivíduos ser chegarem ao fim do cenário, a velocidade foi dos fatores mais considerados e com maior peso dado os veículos necessitarem de velocidade para conseguirem ultrapassar o maior número de obstáculos sem perder potência. Outros fatores foram considerados, mas com menos importância.

Uma mudança em relação à testagem dos algoritmos prende-se com a quantidade de gerações utilizadas. Após testagem prévia com as 30 gerações indicadas na tabela (Fig.1) verificou-se que, independentemente das alterações no algoritmo, não haveria tempo suficiente para os indivíduos evoluirem ao ponto de obter resultados satisfatórios. Optou-se por utilizar 500 gerações para fazer a testagem, tendo obtido resultados mais satisfatórios.

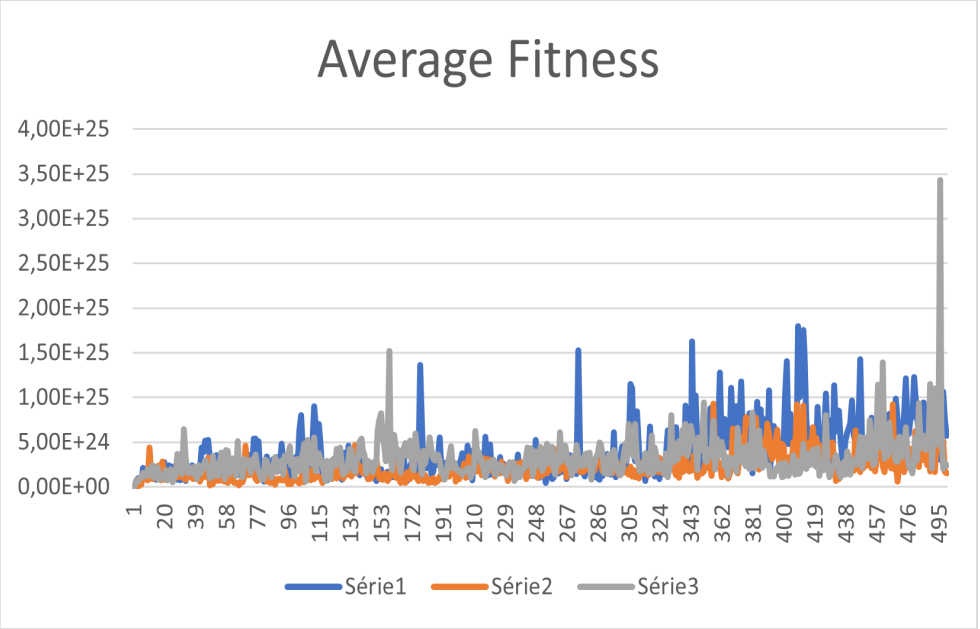
Apresento de seguida o algoritmo utilizado:

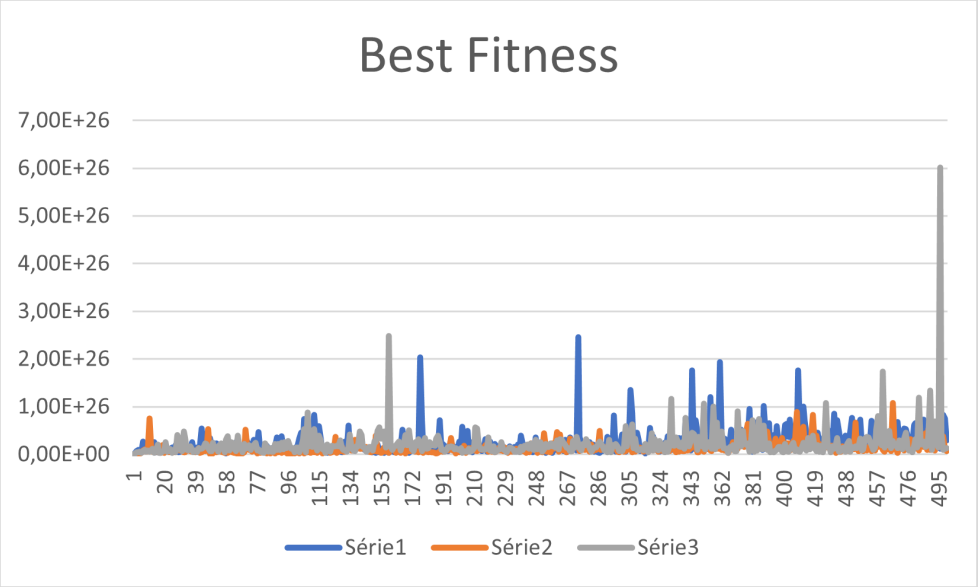
```
if ((CarMass>=200 || CarMass<=350) && MaxWheels>=12 && MaxVelocity > 0 ){
        fitness=(float)Math.Pow((((CarMass*MaxWheels*5)*((MaxVelocity+1)*MaxDistance * 12)) + ((10 * (MaxVelocity+1) * MaxDistance)) + (MaxDistance* 15 * IsRoadComplete)),3);
    }
else{
    fitness = (CarMass*MaxDistance*6) + (3*(MaxVelocity+1)) + (MaxDistance * 3 *IsRoadComplete);
    }
```

Explicando a expressão mais detalhadamente, podemos observar que, após alguma experimentação prévia com as primeiras versões do algoritmo, a massa deveria entre 200 e

350, ou seja, carros mais leves teriam melhor performance a subir os obstáculos; o número de rodas deverá ser superior a 12 pela razão apresentada anteriormente. A parcela que atribui maior valor à expressão é uma multiplicação entre a velocidade e a distância percorrida dado que estes são os que mais influenciam a aptidão de um indivíduo. Uma parcela adicional utiliza a variável *isRoadComplete*, para criar uma distinção mais significativa para todos os carros que chegam ao fim do percurso. A exponenciação utilizada na primeira condição cria uma pressão seletiva maior ao aumentar significativamente o valor do *fitness*, o que fará com que veículos que cumpram requisitos que consideramos "ideais" tenham uma probabilidade muito maior de transmitir genes para a geração seguinte. Todos os quantificadores utilizados foram selecionados com base em experimentação e testagem prévia com versões iniciais do algoritmo de aptidão.

De seguida apresentamos os gráficos relativos ao *BestFitness* e ao *AverageFitness* utilizando os parâmetros da experiência 3 da tabela apresentada (Fig.1). Esta experiência foi selecionada dado ter obtido os melhores resultados e mais consistentes neste cenário, os restantes resultados serão enviados em anexo para consulta posterior.





Cada "Série" corresponde a uma das tentativas da experiência realizada, tendo sido reunidas todas as tentativas num único gráfico para ser mais simples a análise e por ser irrelevante analisar isoladamente dada a natureza estocástica das experiências.

Embora os resultados não sejam ideais, é facilmente percetível que a média de *fitness* da população cresce de maneira consistente e constante ao longo das 500 gerações, o que demonstra que, como teoricamente previsto, o fundo genético da espécie melhora progressivamente a cada geração. Os resultados são satisfatórios neste aspeto. Relativamente ao melhor indivíduo, o crescimento não é tão acentuado nem tão notório como em relação à média de aptidão, no entanto, é possível verificar facilmente através dos máximos locais visíveis no gráfico que nas gerações mais próximas do fim da experiência são gerados vários veículos com fitness significativamente mais elevado que os restantes. Consideramos que os resultados são globalmente satisfatórios, tendo como recomendações para melhoria do algoritmo alguns pontos:

- Não valorizar em demasia a massa dos veículos: veículos mais pesados podem não ser necessariamente mais aptos para conseguir escalar os obstáculos
- Diminuir o desequilíbrio seletivo provocado pelo algoritmo: embora uma pressão seletiva mais desequilibrada possa fazer uma seleção mais precisa dos genes, também pode acontecer o oposto e haver perda de genes aptos

Conclusão

Este trabalho prático tinha como objetivo a evolução de veículos de modo a conseguir atingir o final dos cenários propostos. Começámos por descobrir quais os parâmetros ideais para testar as várias funções de fitness por nós desenvolvidas. Para tal, utilizamos o cenário Gap Road como sugerido, uma vez que era o mais fácil de completar logo, de tirar conclusões.

Tendo esses parâmetros, e tendo conseguido completar o cenário acima referido, passamos para o cenário Hill Road. A criação dos algoritmos para este cenário teve como base as funções desenvolvidas no cenário anterior mas tendo em consideração as especificidades do cenário. Depois da conclusão deste cenário foi desenvolvido o algoritmo para o Obstacle Road, uma tarefa mais árdua e difícil que para os cenários anteriores, embora se tenha obtido resultados com qualidade.

Com este trabalho, aprendemos o mecanismo de evoluir agentes através de algoritmos de aptidão e como os melhorar : beneficiar quem tem as características mais necessárias e prejudicar quem não as tem. Isto ajudou a perceber melhor a matéria lecionada nas aulas teóricas da cadeira e a compreender a importância atual de agentes evolucionários em Inteligência Artificial.

Notas:

- Será enviado em anexo uma pasta zip que contém resultados e gráficos relativos às experiências efetuadas