FACULDADE DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE DE COIMBRA
2017/2018

INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

Mad AI – Fury Evolution

Trabalho prático nº3 – Meta 2

Nuno Filipe Cardia Ferreira

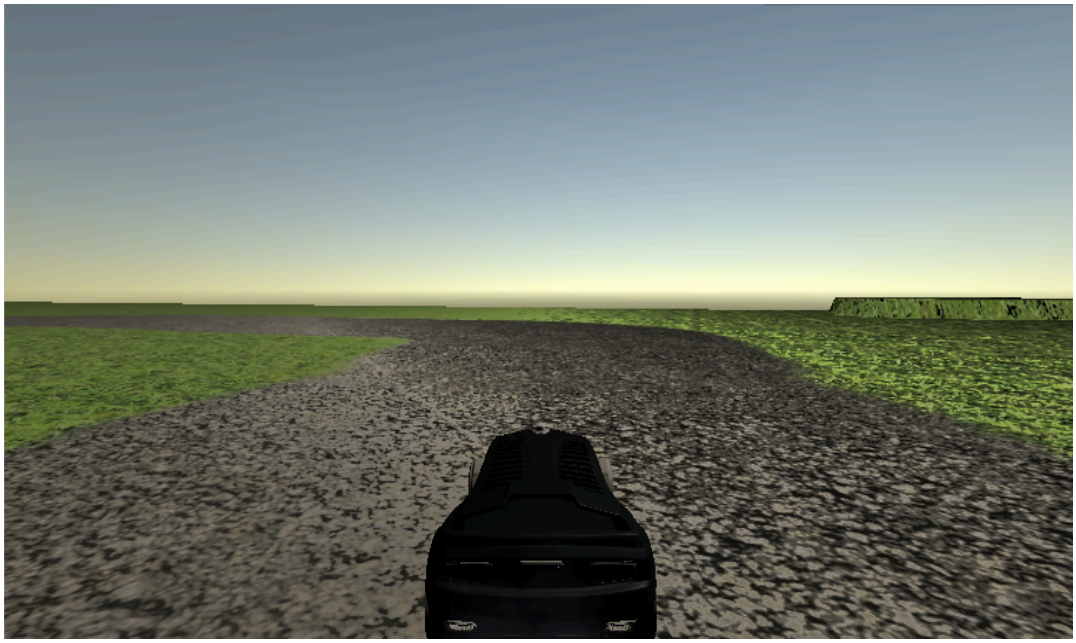
2013153319

nfcf@student.dei.uc.pt

PL2

1 – Algoritmos genéticos

Os algoritmos genéticos são capazes de resolver problemas complexos recorrendo a conceitos baseados na teoria da seleção natural e biologia evolucionária. Para este trabalho, usando uma abordagem evolucionária, pretende-se que o veículo complete a pista sem bater nas paredes.



1.1 – População e Indivíduos

A população de um algoritmo genético é um conjunto de indivíduos que são encontrados como uma solução e que são usados para criar uma nova geração. Desta forma, o tamanho da população pode afetar o desempenho e a eficiência dos algoritmos genéticos. Assim sendo, uma população muito pequena tende a perder a diversidade necessária para atingir uma boa solução, fornecendo uma pequena representação do espaço de procura do problema. No entanto, uma população com muitos indivíduos pode levar a uma perda de eficiência, sendo necessário maiores recursos computacionais.

1.2 – Representação

Para o desenvolvimento de um algoritmo genético é necessário a utilização de uma representação adequada para que seja possível chegar a resultados conclusivos. Para este projeto foi criada uma representação disponível em *GeneticAlgorithm.cs*.

1.3 – Aptidão

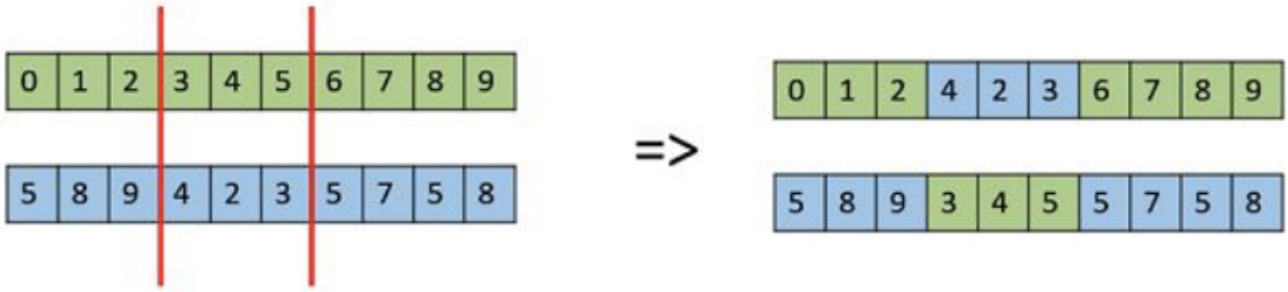
Para cada individuo da população é calculado um valor de aptidão, designado de fitness. O fitness é considerado dos elementos mais importantes de um algoritmo genético, uma vez que é através dele que se determina o quão próximo um individuo se encontra da solução pretendida ou quão boa é essa solução. Para tal, é fundamental que esta função seja representativa e diferencie as más soluções das boas. No caso apresentado a função de fitness é dada pela seguinte fórmula $(driveTime * numberOfLaps) + distanceFromStartingPoint + numberOfCheckpoints$. Esta função resolve vários problemas complexos como o caso em que o carro não avança na pista e fica às voltas no mesmo sitio mas não resolve outros problemas complexos como o caso em que o carro avança, dá umas voltas e volta para trás. Por isso, **a função de fitness encontrada não é ótima**.

1.4 – Operadores Genéticos

Os operadores genéticos transformam a população nas sucessivas gerações, estendendo a busca até encontrar um resultado satisfatório. Os operadores são necessários para que a população se diversifique, mas para que também se mantenham características de adaptação que as gerações anteriores adquiriram. Os mecanismos de recombinação e de mutação são fundamentais em algoritmos genéticos.

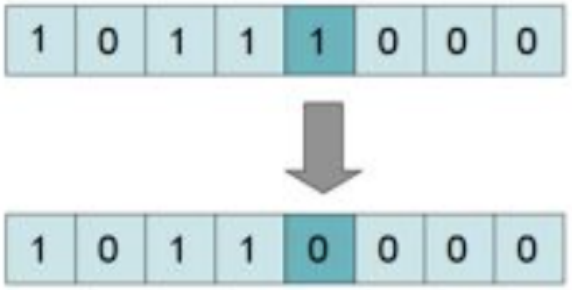
1.4.1 – Recombinação (Crossover)

Através da recombinação é possível criar novos indivíduos misturando características de dois indivíduos pais. Desta forma, a recombinação é considerada o operador genético mais predominante uma vez que emita, ainda que em abstração, a reprodução de genes entre células. O que acontece é que partes das características de um individuo são trocadas por outras de igual tamanho, de outro individuo. No trabalho são apresentados dois tipos de recombinações. O primeiro, designado de recombinação com 1 ponto de corte, troca a primeira metade das características de dois indivíduos. O segundo, designado de recombinação em N pontos, troca N-1 porções aleatórias de características entre dois indivíduos.



1.4.2 – Mutação

As mutações permitem modificar aleatoriamente alguma característica do individuo permitindo inserir diversidade genética. Estas modificações são importantes, pois acabam por criar novas características ou aumentar outras que existiam em pouca quantidade. Normalmente a probabilidade de mutação é pequena. As mutações são feitas de forma aleatória consoante a probabilidade escolhida pelo utilizador.



1.5 – Geração

A cada step um novo conjunto de indivíduos é gerado a partir da população anterior. Só com um grande número de gerações é que é possível obter bons resultados nos algoritmos genéticos.

1.6 – Seleção

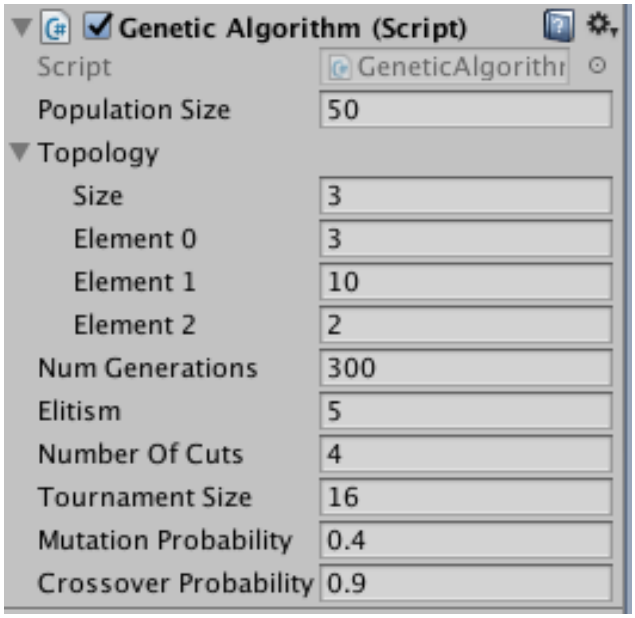
Quando se implementa um algoritmo genético é necessário definir a forma como será feita a seleção dos indivíduos que vão produzir a nova geração. Inicialmente, a seleção dos indivíduos era feita de forma aleatória o que leva a que o processo de evolução possa ser lento. Para combater essa situação, foi implementado outro mecanismo de seleção que quando combinado com os mecanismos de mutação e recombinação falados anteriormente permitem melhores resultados. É de ter em atenção que o mecanismo de seleção não deve ser demasiado exigente podendo levar à estagnação num máximo local. O método escolhido foi a seleção por torneio. Este mecanismo escolhe dois indivíduos de forma aleatória. Deste par de indivíduos, o individuo selecionado para fazer parte da nova geração dependerá de uma probabilidade que é indicada pelo utilizador. Este algoritmo consiste na execução de vários "torneios" entre um conjunto de indivíduos escolhidos ao acaso. O vencedor de cada "torneio", aquele com melhor *fitness*, é selecionado para *crossover*. Um factor importante neste algoritmo é o *tournament size*. Se esta variável for grande indivíduos mais fracos têm uma possibilidade mais pequena de serem selecionados.

1.7 – Elitismo

O Elitismo garante que os *n* melhores indivíduos de cada geração são levados para a geração seguinte. Assim só serão necessários criar os restantes elementos e submetê-los aos operadores genéticos. Este valor é indicado pelo utilizador no início da simulação.

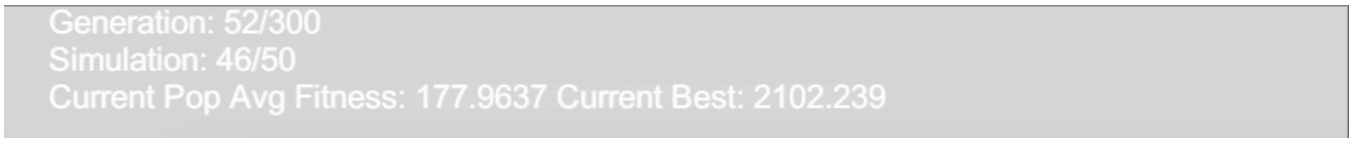
1.8 – Parametrização

Em cada simulação os valores e os mecanismos podem ser alterados no *Unity* sem que seja necessário recorrer a alterações no código. Aqui podem ser alterados o número da população, a topologia da rede neuronal, o número de gerações, o número de indivíduos a preservar para a próxima geração, número de cortes, tamanho do torneio, probabilidade de mutação e probabilidade de crossover.



1.9 – Recolha de dados

Na consola do *Unity* é possível ver para cada geração o valor do *fitness*. No ecrã da simulação pode ser visto o número da geração, o número de simulações e o melhor resultado alcançado na simulação.



1.10 – Casos de teste

Os casos de teste criados **visam estudar a influência** dos seguintes parâmetros: **elitismo, probabilidade de crossover, número de cortes e tamanho de torneio**. As variáveis que se mantêm são o número de gerações, tamanho da população e mutação. O número de gerações fixou-se em 300, o tamanho da população em 50 e o a probabilidade de mutação em 0,4. Os seguintes casos de teste têm todos um tournament size e probabilidade de crossover altos para se obterem bons resultados e o elitismo baixo para se obter mais diversidade na população.

Caso 1	
Elitismo	2
Crossover(%)	80
Nº de cortes	2
Tournament Size	16

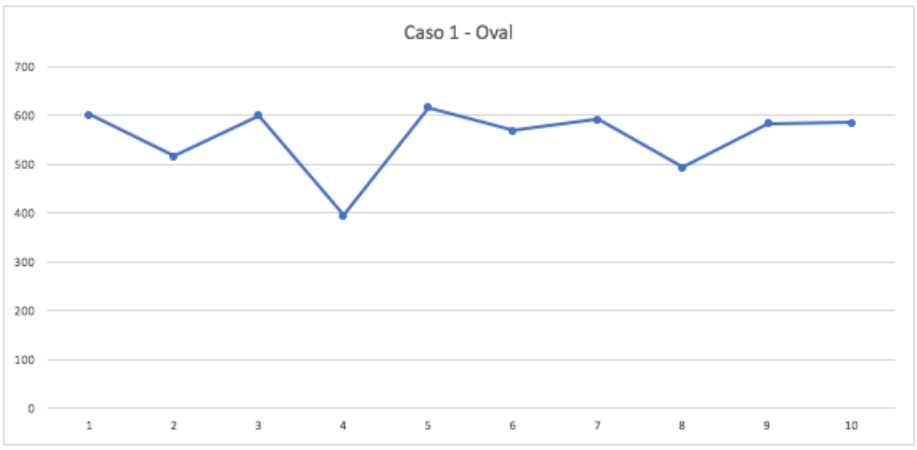
Caso 2	
Elitismo	5
Crossover(%)	90
Nº de cortes	4
Tournament Size	16

Caso 3	
Elitismo	5
Crossover(%)	80
Nº de cortes	4
Tournament Size	18

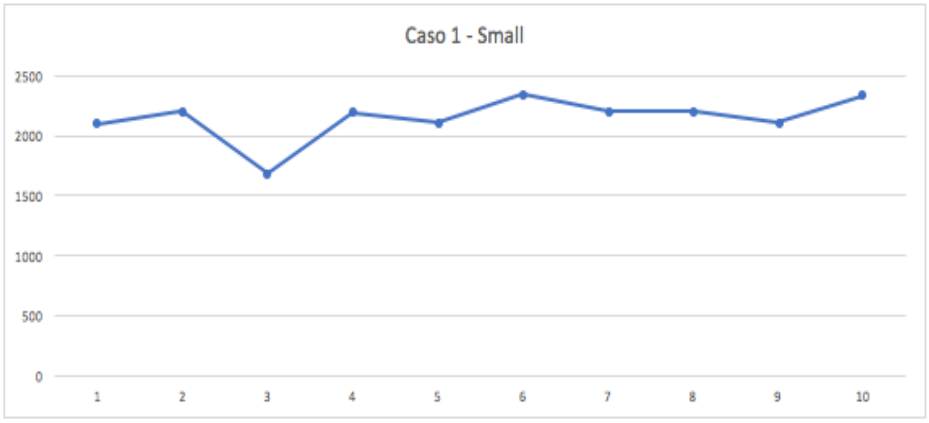
Os casos de teste têm o objetivo de provar os conceitos teóricos estudados nas aulas práticas. Espera-se que com um maior elitismo a diversidade de população diminua o que conduz a um pior resultado. Espera-se que uma maior probabilidade de crossover e um número menor de cortes conduza a uma maior diversidade o que leva a melhores resultados. Um maior valor de torneio conduz a melhores resultados mas pouco significativos.

1.11 – Resultados

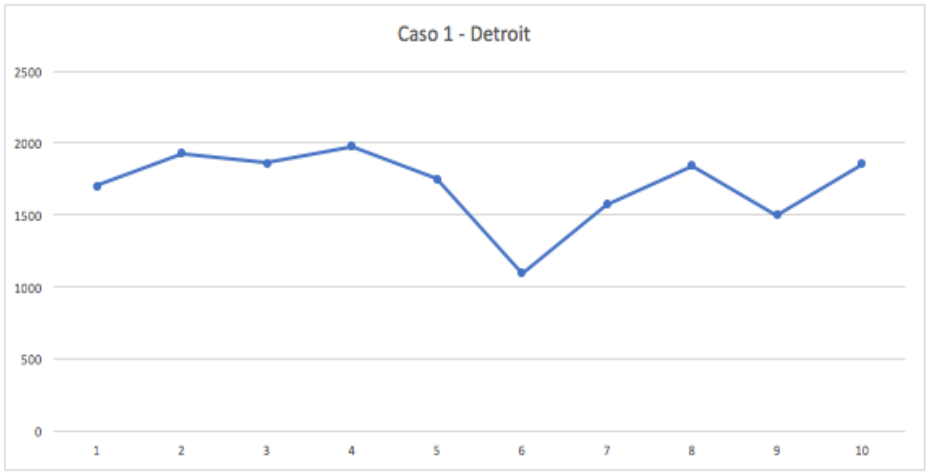
Testes	Caso 1 - Oval
1	601,044
2	515,2852
3	599,433
4	394,1745
5	616,2787
6	567,7629
7	592,0065
8	492,9373
9	583,7963
10	585,0456



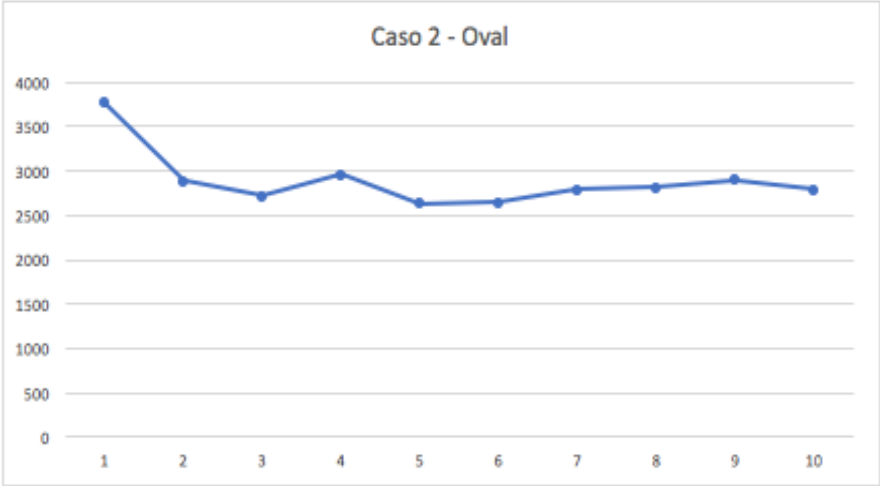
Testes	Caso 1 - Small
1	2100,6266
2	2202,8105
3	1681,231
4	2194,766
5	2105,762
6	2345,455
7	2202,8483
8	2203,3952
9	2109,1573
10	2334,792



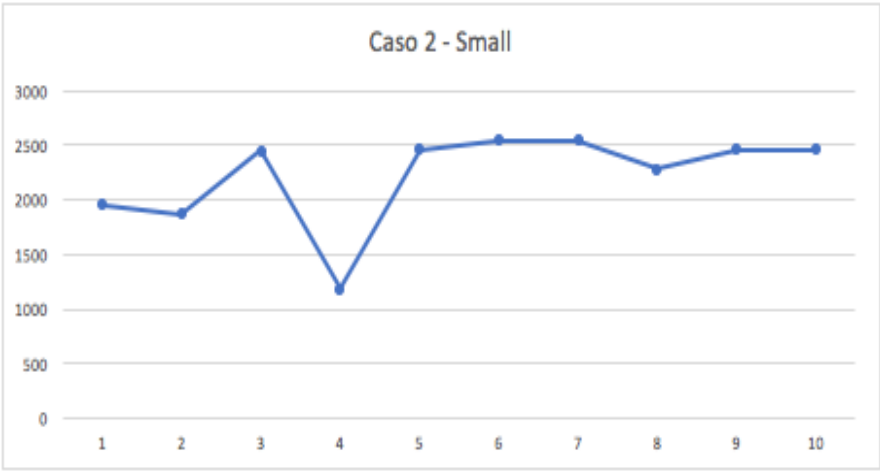
Testes	Caso 1 - Detroit
1	1704,019
2	1926,711
3	1864,238
4	1979,907
5	1749,197
6	1098,459
7	1572,874
8	1846,345
9	1498,121
10	1856,488



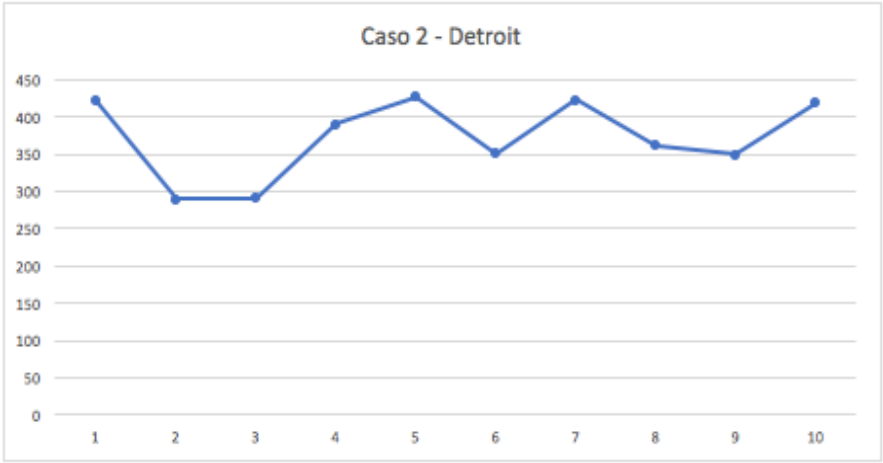
Testes	Caso 2 - Oval
1	3769,791
2	2893,177
3	2717,589
4	2966,002
5	2631,872
6	2651,525
7	2795,472
8	2819,452
9	2894,577
10	2792,836



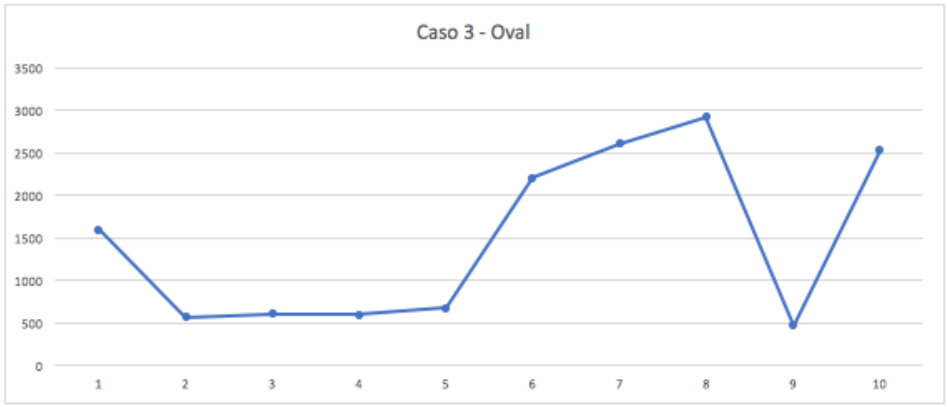
Testes	Caso 2 - Small
1	1957,415
2	1873,216
3	2452,636
4	1183,074
5	2458,901
6	2547,372
7	2548,925
8	2285,777
9	2461,847
10	2464,686



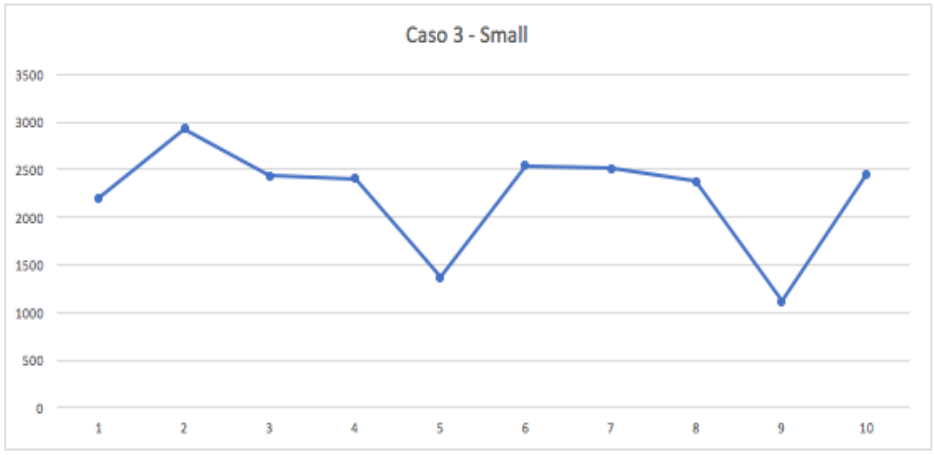
Testes	Caso 2 - Detroit
1	421,2116
2	289,4864
3	291,0179
4	390,0462
5	426,3089
6	350,3058
7	422,1268
8	362,2086
9	349,328
10	418,8194



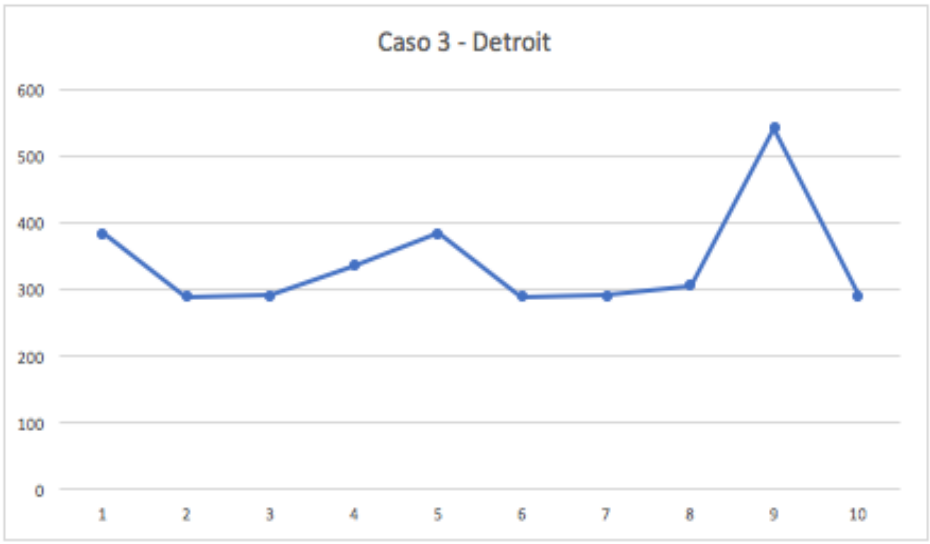
Testes	Caso 3 - Oval
1	1594,893
2	569,1783
3	610,8101
4	601,0137
5	676,8298
6	2211,256
7	2603,541
8	2924,882
9	477,1108
10	2530,301

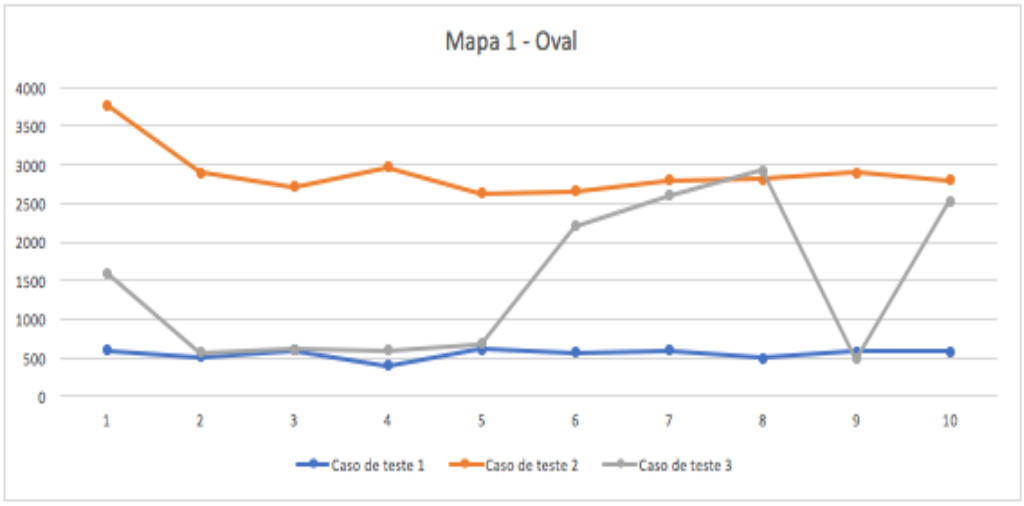


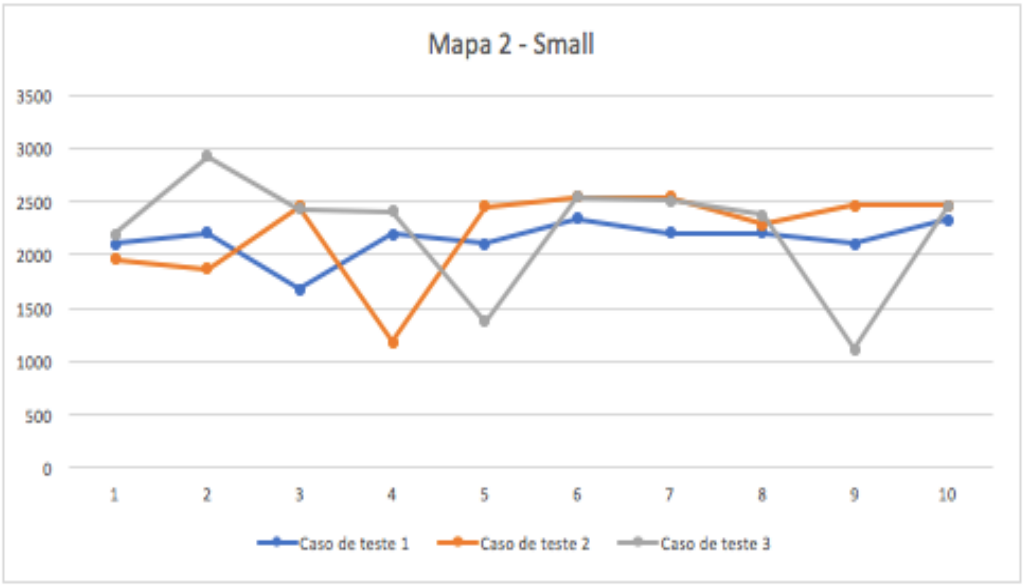
Testes	Caso 3 - Small
1	2197,579
2	2923,154
3	2432,175
4	2404,344
5	1372,335
6	2544,746
7	2513,648
8	2378,434
9	1121,03
10	2453,424

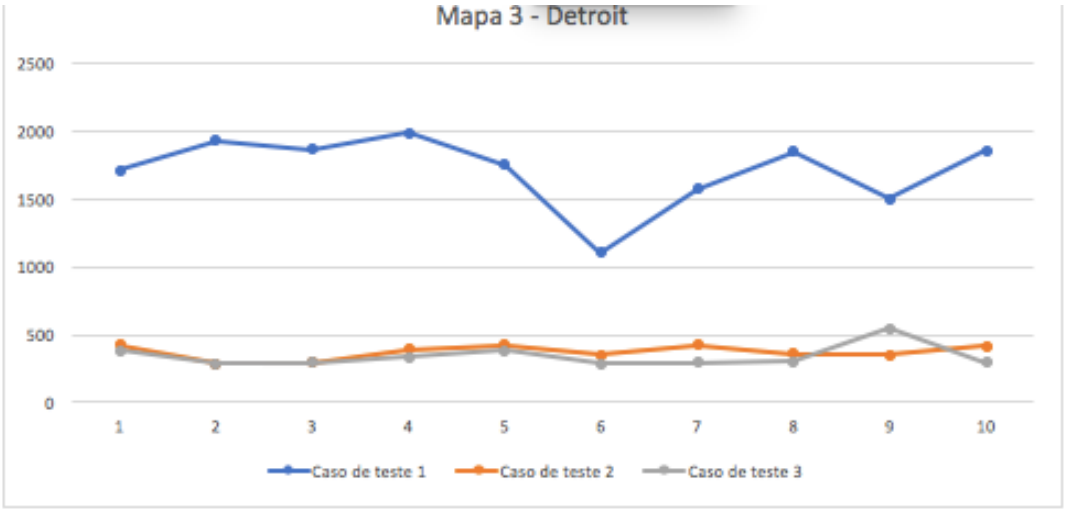


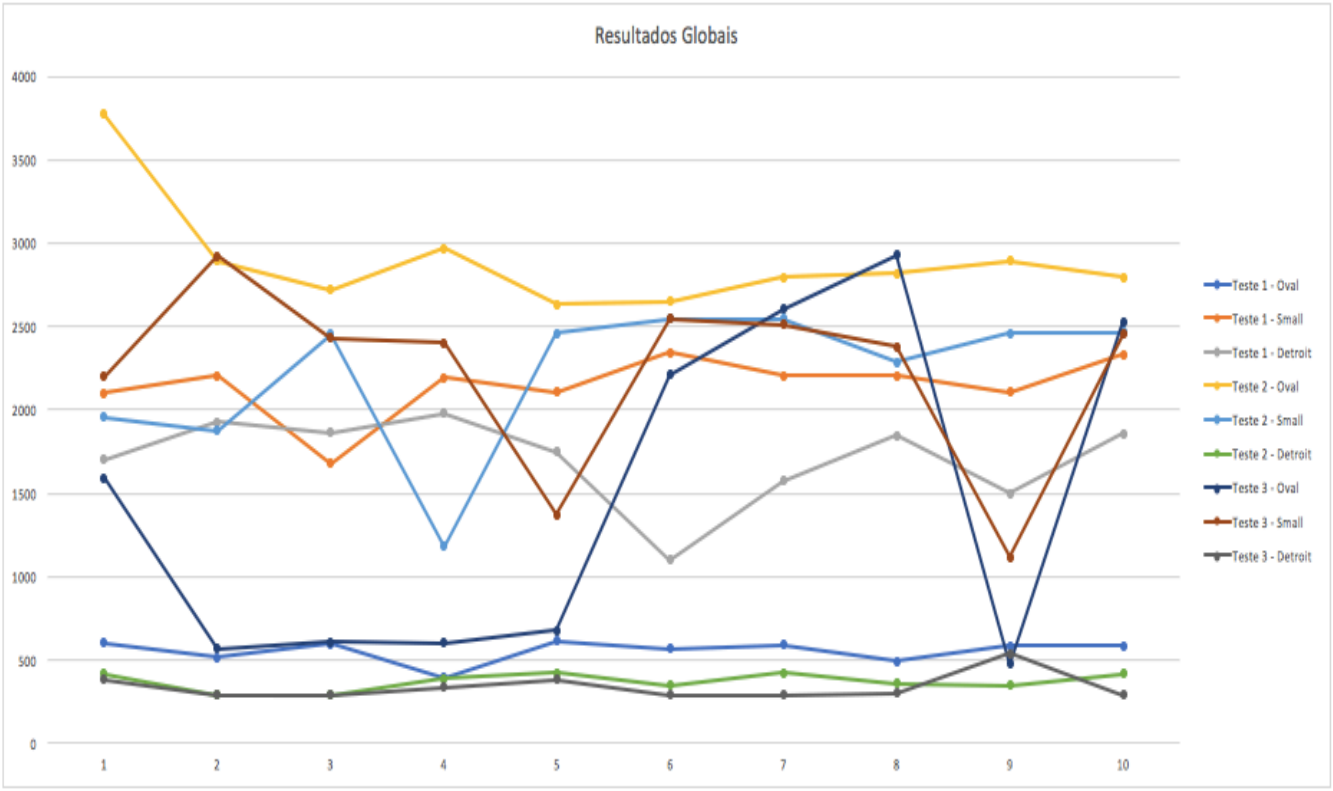
Testes	Caso 3 - Detroit
1	383,2214
2	289,601
3	291,3062
4	335,9079
5	384,1674
6	290,1594
7	291,6574
8	305,5882
9	541,3471
10	291,2991




Mapa 1 - Oval
4000
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
Caso de teste 1
Caso de teste 2
Caso de teste 3


Mapa 2 - Small
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
Caso de teste 1
Caso de teste 2
Caso de teste 3


Mapa 3 - Detroit
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8
9
10
Caso de teste 1
Caso de teste 2
Caso de teste 3



1.12 – Conclusões finais

Analisando os dados anteriores é de notar que quanto maior for o número de indivíduos a transitar para a geração seguinte pior será o resultado obtido. Esta situação seria de esperar dado que, quanto maior for esse valor menor será a variedade introduzida. No caso do *Crossover*, ainda que este apresente sempre valores elevados em todos os casos de teste, **quanto maior for a probabilidade de ocorrência e o número de cortes melhor serão os resultados obtidos**. Na seleção, usando o mecanismo do Torneio, desde que a probabilidade de ocorrência seja elevada os dados daí resultantes são bons. Pequenas variações na probabilidade do Torneio não são significativas. Pode-se concluir também que **um número maior de elitismo** conduz a uma **perda de diversidade** na população o que leva **à obtenção de piores resultados**.