COMP9058: Metaheuristic Optimization

Assignment 1

Student: William Barrett

Student ID: R00029480

Lecturer: Dr Diarmuid Grimes

Due Date: October 12th, 2025

1.1 Genetic Algorithms

The Following document represents and investigation into using Genetic algorithms to produce meaningful results in the travelling salesman problem.

Run Configurations:

python Barrett_William_R00029480_MH1.py data/inst-a.tsp 10 1000 100 0 0.9 0 0.2 0.1 0.5		
Parameter	Value	Description
Inst_file	data/inst-a.tsp	The file containing dictionary of coords to the TSP problem. Instance A and B apply to my student number as the last digit is < 3
nRuns	10	How many times to Run GA.
nIters	1000	Number of generations in a run of the GA
popSize	100	Number of chromosomes/individuals
xoverH	0	Crossover operator
Pc	0.9	Crossover probability
mutH	0	Mutation Operator
Pm	0.2	Mutation probability
elites	0.1	Proportion of the population that are elite
trunk	0.5	Proportion used for truncation selection

Table A: program run arguments.

The seed was update to 29480 as per R00029480. Throughout the investigation what I was outputting evolved into the following:

```
Runs:
        run: 3, fitness: 3632457, Best Fitness: 3699019
        run: 4, fitness: 3601892, Best Fitness: 3632457
        run: 7, fitness: 3564541, Best Fitness: 3601892
        run: 9, fitness: 3384050, Best Fitness: 3564541
Best Solution: [32, 96, 21, 48, 10, 4, 5, 60, 55, 100, 3
, 41, 87, 11, 85, 99, 33, 8, 57, 22, 14, 28, 89, 40, 66,
Best Run Time: 0m 3s 268ms
Average Run Time: 0m 3s 295ms
Best Distance: 3384050
Average Distance: 3673536.6
Average Initial Distance: 11445272.6
Overall Run Time: 0m 32s 957ms
```

Figure 1: Final Output

The image **[Figure 1]** shows the final output using the default parameters.

1.1.1 Population initialisation

The default initialization heuristic in the sample code is “random tours”. The Provided TSP_Individual.py also provides a configuration for using insertion heuristic. While this is not configurable at the program level we can update BasicTSP initializer **[Figure 2]**.

```
self.initH          = 1
```

Figure 2: configure insertion heuristic

Experiment 1 – Random vs Insertion Initialization

Justification:

Random has very low possibility of generating a good solution. As Insertion heuristic takes a methodical approach to producing meaningful results, I expected Insertion heuristic to produce much better results. Each chromosome or individual starts at a random location and uses nearest neighbour. This leaves the population with lots of small, good sections within them around the random index chosen. So, it should perform much better random initialization.

Results:

	Initialization	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**Random**	3384050	3673536.6	11445272.6	0m 37s 466ms
	Insertion	2368338	2432228.1	2530346.5	0m 34s 685ms
	Improvement	30.02%	33.78%	77.90%	7.43%
Inst-b	**Random**	27175913	30025708.6	11445272.6	3m 11s 789ms
	Insertion	6702819	6849429.4	7333745.4	3m 14s 678ms
	Improvement	75.34%	33.78%	35.92%	-1.51%

Table 1: Comparison of random vs insertion initialization

Summary:

We can see a huge improvement in best distance at 30.02% for inst-a and 75.34% for inst-b. With the larger inst-b dataset we see that the larger dataset showed a significant improvement in solution quality.

Experiment 2 – Order crossover vs uniform crossover

Justification:

Crossover is the method in which two parents create a child. While order crossover keeps a lot of the characteristics of each parent. Uniform crossover will use a binary mask that can introduce a lot of diversity. We use random insertion, so the diversity is already there in individuals. I think overall uniform crossover should perform worse than order crossover due to fact it will likely break up good sections of solution.

Results:

	Crossover	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**Order**	3384050	3673536.6	11445272.6	0m 34s 255ms
	Uniform	3419571	3970840.0	11445272.6	0m 23s 680ms
	Improvement	-1.05%	-8.09%	0%	30.86%
Inst-b	**Order**	27175913	30025708.6	97302786.9	3m 10s 542ms
	Uniform	34389088	37543730.8	97302786.9	1m 9s 133ms
	Improvement	-26.55%	-25.04%	0%	63.72%

Summary:

Indeed, uniform crossover performs worse. On lager data sets it performed 26.55% worse on best distance. However, we see a significant improvement in the time to

process. I think the time improvement is due to the implementation. The mask/replace strategy is less intensive than the copy and iterate in order crossover.

Experiment 3 – Reciprocal vs inversion mutation

Justification:

Mutation introduces randomness to children. Our current implementation uses reciprocal mutation which swaps two random genes. I have implemented inversion-based mutation which reverse the order between those two random genes. I think inversion should perform better in this scenario. When we move two genes in reciprocal it can create 4 long paths. When we move two genes in inversion it will make 2 longs paths as what's in between is inverted and maintained.

Results:

	Mutation	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**Reciprocal**	3384050	3673536.6	11445272.6	0m 33s 444ms
	Inversion	2390514	2501026.9	11445272.6	0m 33s 354ms
	Improvement	29.36%	31.91%	0%	0.27%
Inst-b	**Reciprocal**	27175913	30025708.6	97302786.9	3m 17s 495ms
	Inversion	12560074	13199131.3	97302786.9	3m 17s 294ms
	Improvement	53.78%	56.04%	0%	0.10%

Summary:

Inversion does perform significantly better that reciprocal. As mentioned previously this is likely due to that fact inversion inverts the contents of what's between the two selected genes. This means it maintains more of the "good solution". Reciprocal then just swaps two genes so it's more destructive. The time impact for such and improvement is also shown to be negligible.

Experiment 4 – Crossover Probability.

Justification

Crossover probability is the probability a crossover of parents is going to produce a child. Should crossover not happen a random chosen parent's genes are returned which then proceed to the mutation step as normal. This experiment tests the impact of crossover probability. The default crossover probability is 0.9.

Results:

	Crossover Probability	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**0.9**	3384050	3673536.6	11445272.6	0m 33s 396ms
	0.5	3509394	3861898.4	11445272.6	0m 24s 307ms
	0.2	3775758	4139035.7	11445272.6	0m 18s 439ms
Inst-b	**0.9**	27175913	30025708.6	97302786.9	3m 15s 925ms
	0.5	27571379	30106857.4	97302786.9	2m 0s 408ms
	0.2	30119657	33064076.2	97302786.9	1m 13s 889ms

Summary:

The overall time to process each generation reduces as we reduce the crossover probability. This makes perfect sense there's less work to do if there's less crossover. Preformance on the best and average distances also reduce significantly. I believe that the increased randomness of higher crossover probability helps to stop uniform crossover getting stuck in a local optimum thus making it perform better with high crossover probability. Each crossover creates more diversity in the population.

Experiment 5 – Mutation Probability.

Justification

This experiment is to test the impact of mutation probability in both reciprocal and inversion mutations. We mentioned previously that inversion performed better than mutation probability due to maintaining the order between swapped genes. I think we will see something similar here, Increasing the mutation probability should exhibit a negative effect on the solution.

Results:

	Mutation Probability	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**0.2**	3384050	3673536.6	11445272.6	0m 33s 844ms
	0.5	3019650	3400953.1	11445272.6	0m 34s 490ms
	0.9	3690576	4219941.2	11445272.6	0m 33s 505ms
Inst-b	**0.2**	27175913	30025708.6	97302786.9	3m 20s 260ms
	0.5	26316499	28812669.1	97302786.9	3m 18s 798ms
	0.9	29040657	32273230.1	97302786.9	3m 18s 105ms

Summary:

My initial thoughts were inaccurate. From the results a mutation probability of 0.5 performed better than 0.2. Initially I thought that more mutation means were breaking apart good solutions. However, it seems that the diversity it introduces to the population over generations improves the solutions. This only works to a point though. It seems that 0.9 performed worse in both instances. It seems that 0.5 is the better spot which explores enough to get more meaningful results and escape local optima. Mutation probability appears to have little to no meaningful impact on processing time.

Experiment 6 – Elitism

Justification

In this experiment I'd like to compare the impact of elitism. I believe that a low level of elitism helps to keep the population strong and advance. Removing elites allows the solution to explore more of the search space with the cost of forgetting previous best fitness. However, I expect that removing elitism will produce worse results.

Results

	elitism	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**0.1**	3384050	3673536.6	11445272.6	0m 33s 2ms
	0	3827324	4136142.1	11445272.6	0m 37s 13ms
	Improvement	-11.59%	-11.19%	0%	-10.84%
Inst-b	**0.1**	27175913	30025708.6	97302786.9	3m 15s 684ms
	0	28329441	31118198.3	97302786.9	3m 36s 698ms
	Improvement	-4.07%	-3.51%	0%	-9.70%

Summary

These results show removing elitism produced worse results. Keeping an elitism of 0.1 produces better best and average distances at a faster speed. While removing elitism allows more exploration, it is less likely to produce better results. Keeping elites over generations allows the coming generations to improve upon existing solutions while still exploring new space.

Experiment 7 – Population Size and generations

Justification

With an increased population comes increased diversity and the higher chance to find a good solution. This comes at a cost to time and resources. Increasing the population should naturally produce better results. More generations should have a direct impact on the exploration of the current data set and again improve results.

Results

	Population: Generations	Best Distance	Average Distance	Average Initial Distance	Overall Run Time
Inst-a	**1000:100**	3384050	3673536.6	11445272.6	0m 33s 298ms
	500:250	3437752	3761589.1	11366489.1	0m 42s 130ms
	2000:50	3368913	3702507.8	0m 33s 541ms	0m 33s 541ms
Inst-b	**1000:100**	27175913	30025708.6	97302786.9	3m 4s 114ms
	500:250	29408947	32698922.3	95576823.5	4m 6s 699ms
	2000:50	24222392	26516837.4	97827567.2	3m 13s 533ms

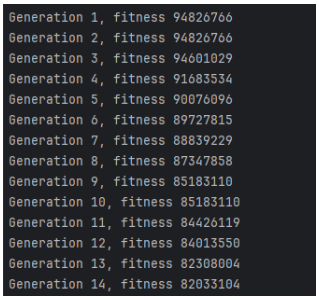
Summary

In the results we see the output for a small population (500) and large generations (250). While the overall best distance and average distance better in the larger dataset. More

generations can allow to explore more of the solution, but a larger population is needed to add diversity.

For the results of the large population (2000) and small generation (50) we see an improvement in quality of the best fitness, but the average distance is worst.

In the following image we show a plot the evolution of the fitness over runs and generations.



Essentially if we add more population and more generations it will find better solutions but at a cost to time and resources.

1.2 NP Completeness

NP-Complete problems are problems that can be reduced and verified.
For this exercise the following formula is used based on R00029480

F = (q1V ¬q2 V ¬q3 V q4 V ¬q5) Λ (q1 V ¬q3) Λ (q5)

This problem concerns the proof of the NP-completeness of 3-SAT

a) First, we convert the formula
F = (q1V ¬q2 V ¬q3 V q4 V ¬q5) Λ (q1 V ¬q3) Λ (q5)
into a 3SAT formula, using the construction/reduction
(q1V ¬q2 V ¬q3 V q4 V ¬q5): k=5 literals so -> $(q1 \vee \neg q2 \vee x^1) \wedge (\neg x^1 \vee \neg q3 \vee x^2) \wedge (\neg x^2 \vee q4 \vee \neg q5)$
(q1 V ¬q3): k=2 literals so -> $(q1 \vee \neg q3 \vee y^1) \wedge (q1 \vee \neg q3 \vee \neg y^1)$
(q5): k=1 literal so -> $(q5 \vee z^1 \vee z^2) \wedge (q5 \vee \neg z^1 \vee z^2) \wedge (q5 \vee z^1 \vee \neg z^2) \wedge (q5 \vee \neg z^1 \vee \neg z^2)$
Therefore, 3SAT formula is

$F' = (q1 \vee \neg q2 \vee x^1) \wedge (\neg x^1 \vee \neg q3 \vee x^2) \wedge (\neg x^2 \vee q4 \vee \neg q5) \wedge (q1 \vee \neg q3 \vee y^1) \wedge (q1 \vee \neg q3 \vee \neg y^1)$
$\wedge (q5 \vee z^1 \vee z^2) \wedge (q5 \vee z^1 \vee z^2) \wedge (q5 \vee z^1 \vee \neg z^2) \wedge (q5 \vee \neg z^1 \vee \neg z^2)$

b) Next, we find a solution for the 3SAT instance of F and verify that it is a solution to the original problem.
Solution:

$q1=T, q2=F, q3=F, q4=F, q5=T, x^1= F, x^2= F, y^1= F, y^2= F, z^1= F, z^2= F$

Verification:

$(q1 \vee \neg q2 \vee x^1) \wedge (\neg x^1 \vee \neg q3 \vee x^2) \wedge (\neg x^2 \vee q4 \vee \neg q5) \wedge (q1 \vee \neg q3 \vee y^1) \wedge$
$(q1 \vee \neg q3 \vee \neg y^1) \wedge (q5 \vee z^1 \vee z^2) \wedge (q5 \vee z^1 \vee z^2) \wedge (q5 \vee z^1 \vee \neg z^2) \wedge (q5 \vee \neg z^1 \vee \neg z^2)$
=
$(T \vee T \vee F) \wedge (T \vee T \vee F) \wedge (T \vee F \vee F) \wedge (T \vee T \vee F) \wedge (T \vee T \vee T) \wedge (T \vee F \vee F) \wedge$
$(T \vee T \vee F) \wedge (T \vee F \vee T) \wedge (T \vee T \vee T)$