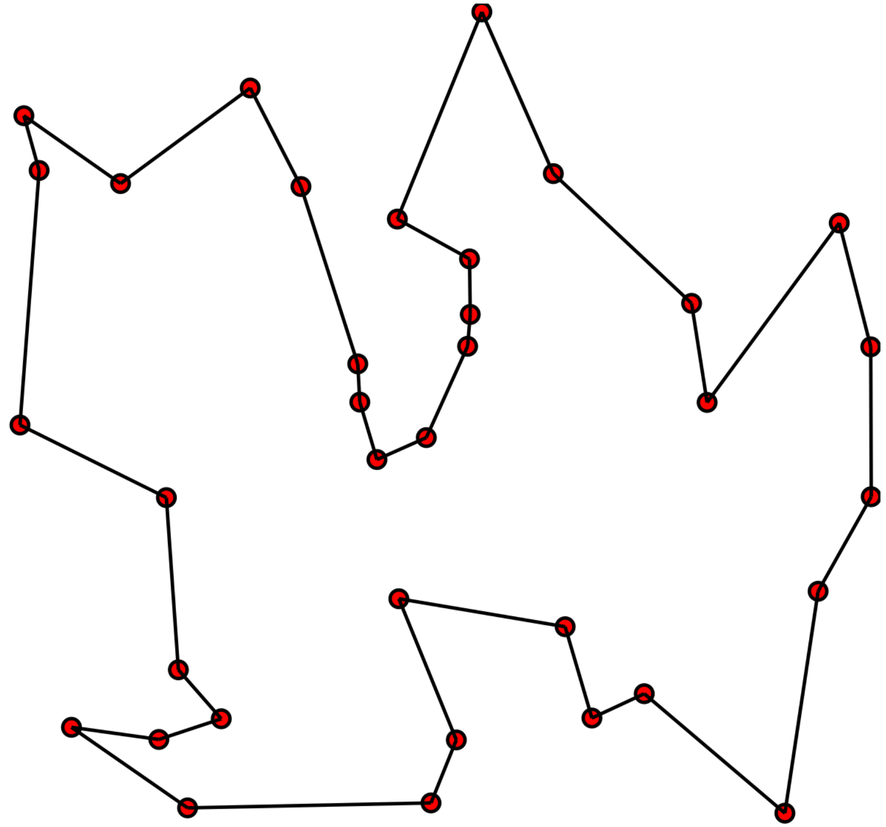
Solving Traveling Salesman Problem using Genetic Algorithms

525.770: Intelligent Algorithms

Azita Dadresan

Traveling Salesman Problem

- Find the shortest path such that all nodes are visited at least once and come back



Genetic Algorithm For TSP

- Solving TSP is NP-Hard! No exact solutions in polynomial time
- Genetic Algorithm is a meta-heuristic algorithm
- Gives approximate solutions in reasonable amount of time
- Many different variants are possible (different cross-over, mutate strategies)
- This work compares some of these methods!

Representation of Chromosome

- A binary chromosome representation is not useful for TSP
- We represent a chromosome as a simple path of nodes visited in order

1 2 3 5 6 4 7 9 8

Mutation

- Mutation means small perturbations made to the chromosome
- Make mutation with a small probability (pmutate)
- Two types considered:

1. Swap mutation
2. Insertion mutation

Swap Mutation

- Two adjacent alleles are swapped.
- The position is randomly sampled

1 2 3 5 6 4 7 9 8

1 2 3 6 5 4 7 9 8

Insertion Mutation

- alleles are swapped adjacent or not
- The positions are randomly sampled

1 2 3 5 6 4 7 9 8

1 2 3 9 6 4 7 5 8

Cross Over Operations

- Single point cross over fails in TSP
- Ref: J.-Y. Potvin, The traveling salesman problem

tour	(12564387)	:	**1**	**2**	| **5**	**6**	**4**	**3**	**8**	**7**
tour	(14236578)	:	1	4	| 2	3	6	5	7	8
offspring 1		:	**1**	**2**	2	3	6	5	7	8
offspring 2		:	1	4	**5**	**6**	**4**	**3**	**8**	**7**

Figure 5. Application of the one-point crossover on two parent tours.

- Note that the offsprings are not valid paths!

Partially Mapped Cross-Over (PMX)

- This is a path preserving cross-over (maintains absolute positions of allele)

parent 1	:	**1**	**2**	**5**	**6**	**4**	**3**	**8**	**7**
parent 2	:	1	4	2	3	6	5	7	8
offspring									
(step 1)	:	1	4	**5**	**6**	**4**	5	7	8
(step 2)	:	1	3	**5**	**6**	**4**	2	7	8

Figure 8. The partially-mapped crossover.

Order Cross-Over (OX)

- This is an order preserving cross-over (maintains the relative order of allele)

parent 1	:	**1**	**2**	**|**	**5**	**6**	**4**	**|**	**3**	**8**	**7**
parent 2	:	1	4	|	2	3	6	|	5	7	8
offspring											
(step 1)	:	–	–		**5**	**6**	**4**		–	–	–
(step 2)	:	2	3		**5**	**6**	**4**		7	8	1

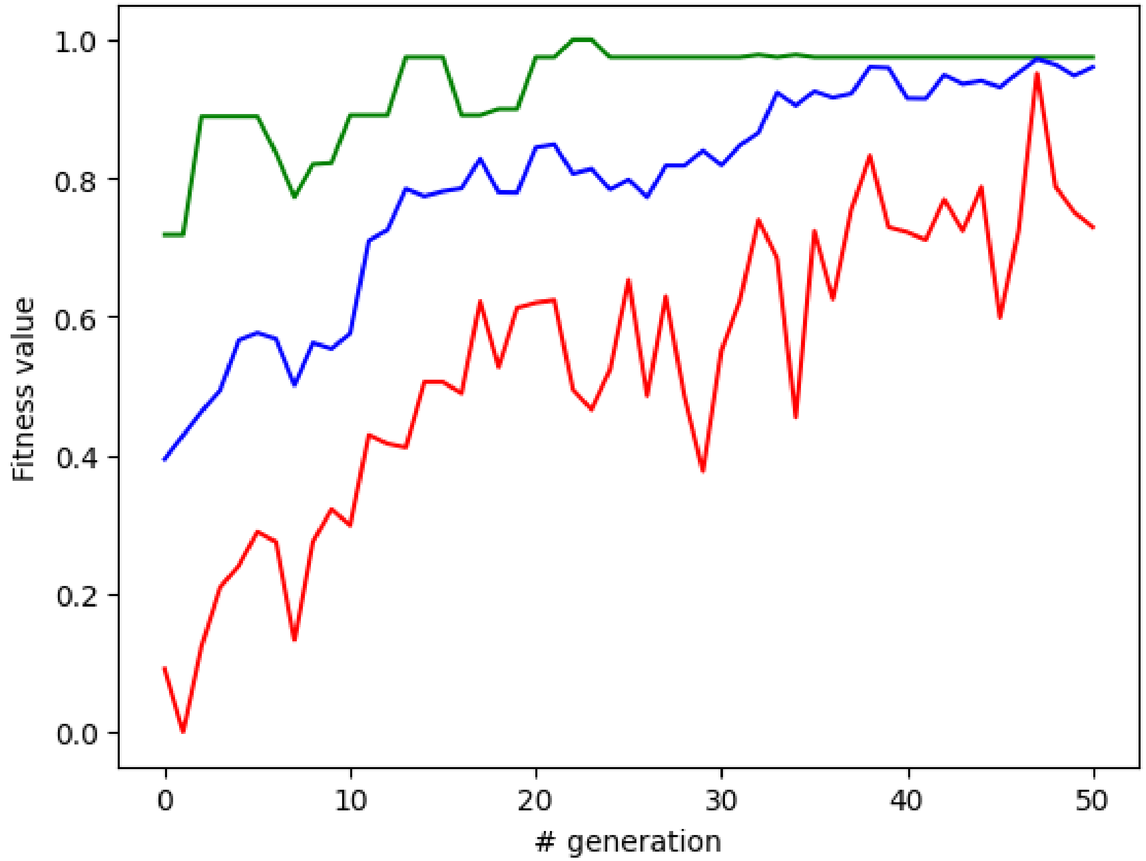
Figure 11. The order crossover.

Dataset

- The datasets are sampled Graphs
- Graph sizes: 10, 50, 100
- The edge weights are randomly sampled from 1 to 99

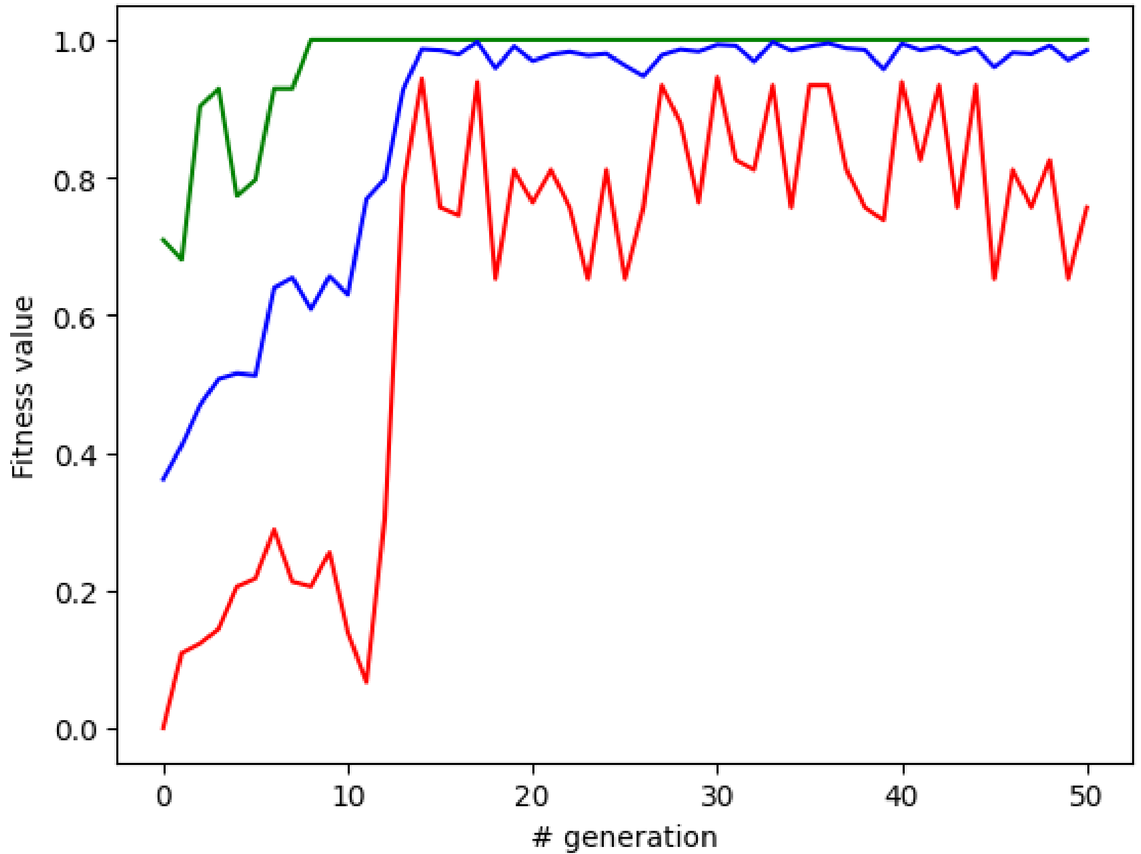
Experiments

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 50
- Mutation: swap, cross-over: Ox



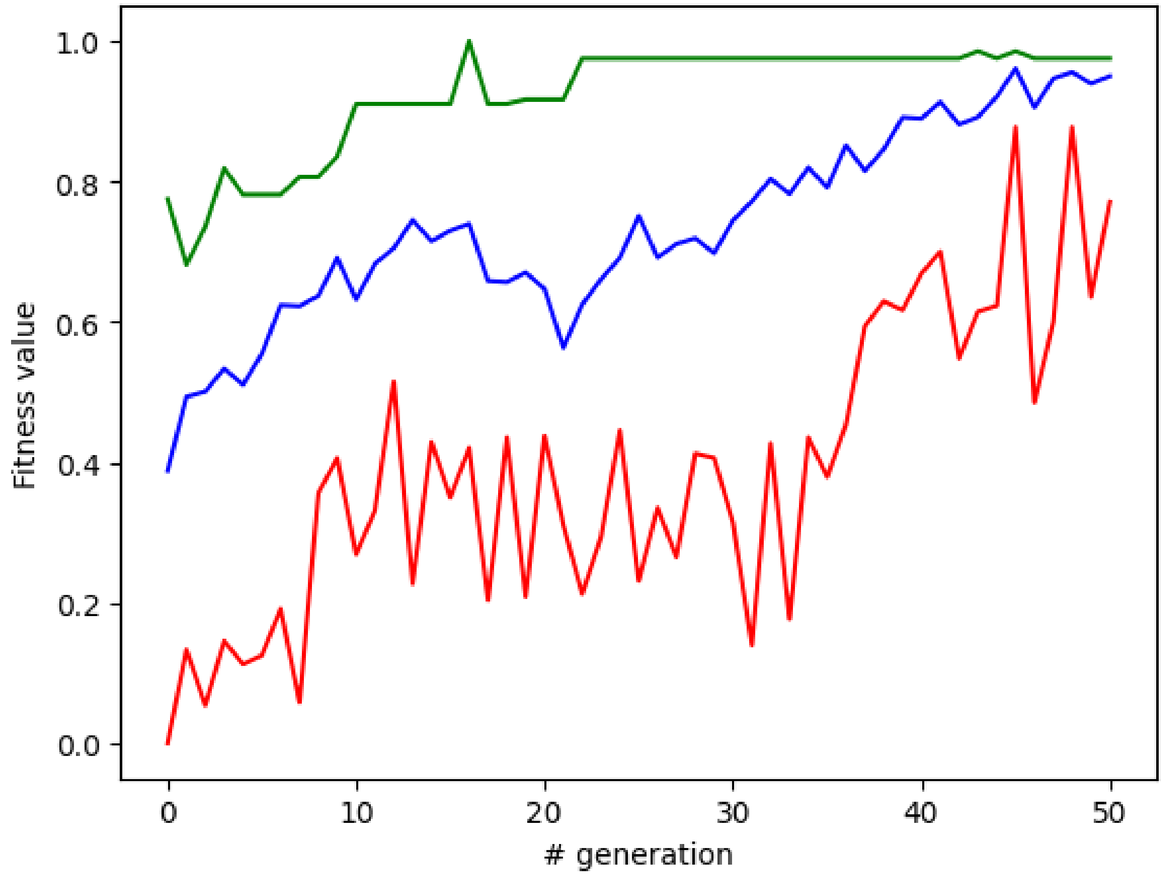
Experiments

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 50
- Mutation: swap, cross-over: pmx



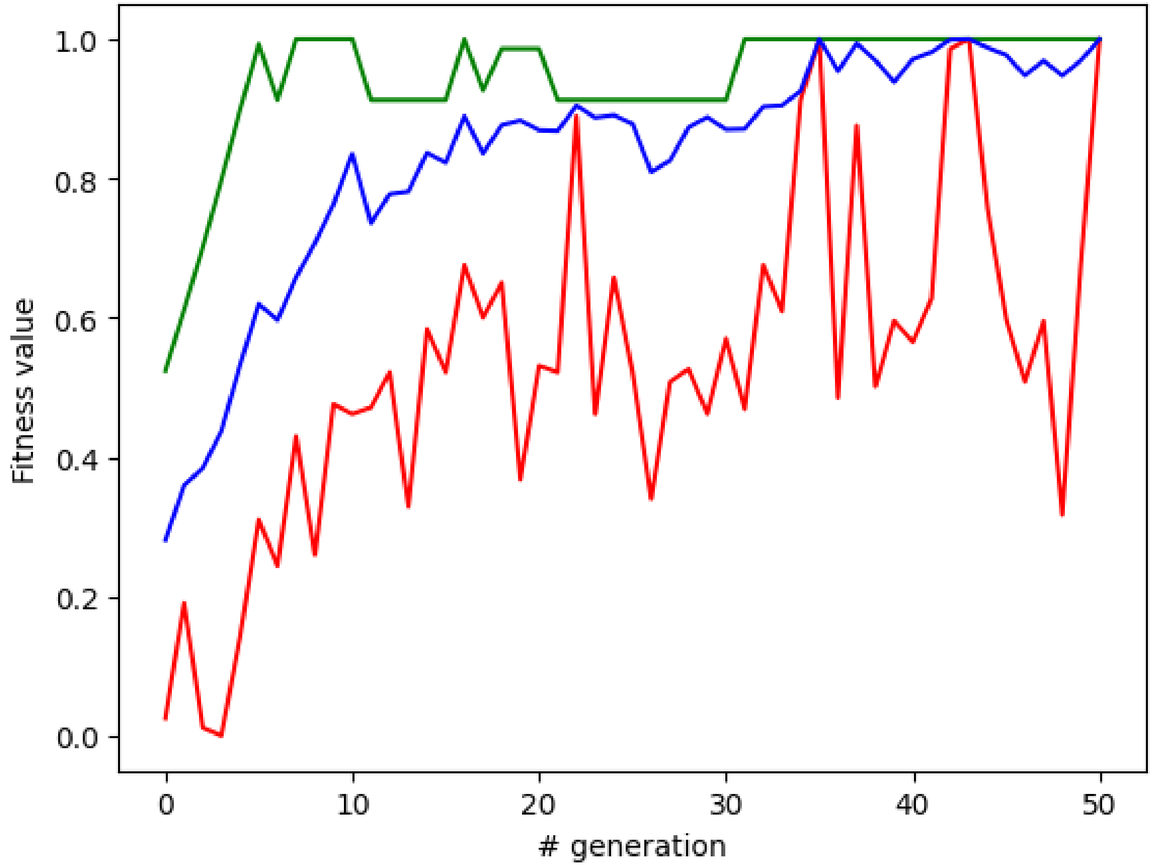
Experiments

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 50
- Mutation: insertion, cross-over: Ox



Experiments

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 50
- Mutation: insertion, cross-over: pmx



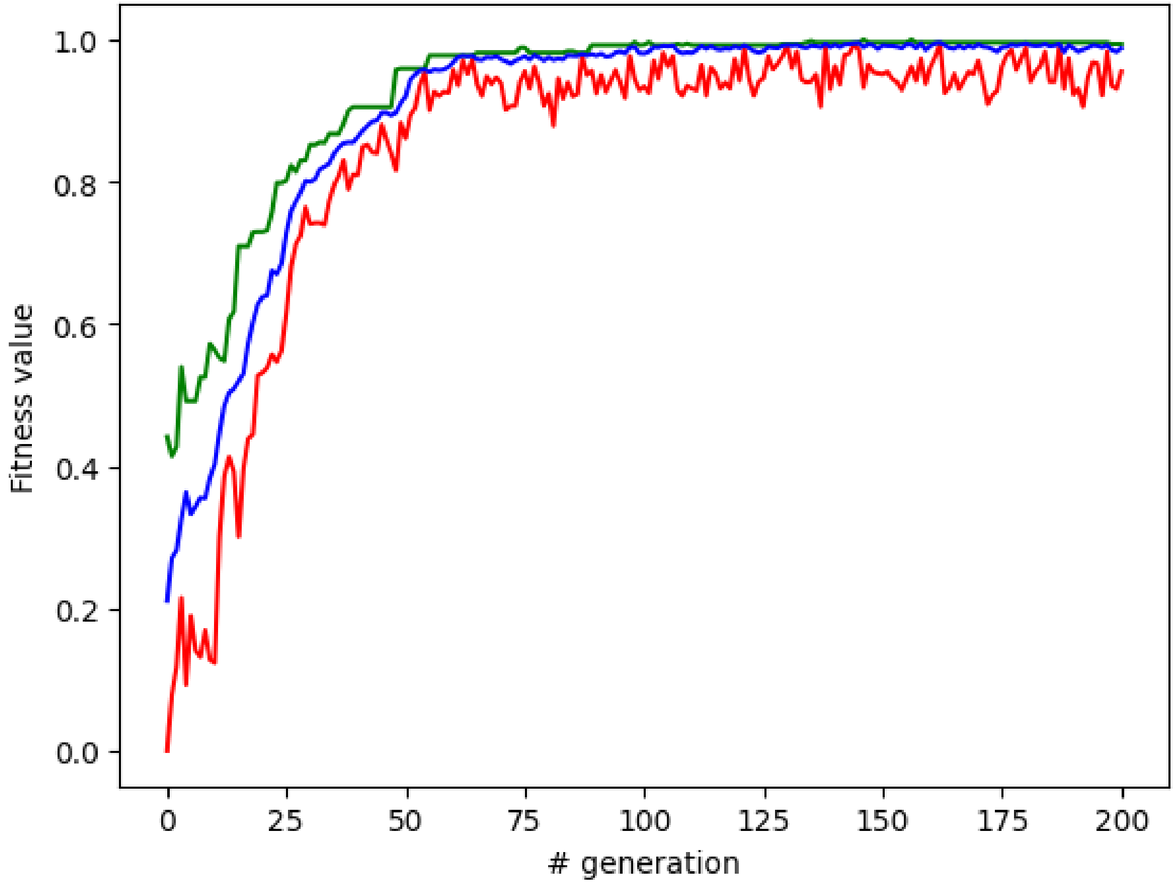
Comparison

- The minimum path length found by the different combinations of mutation & cross-over functions are

	Swap	Insertion
OX	259	261
PMX	291	261

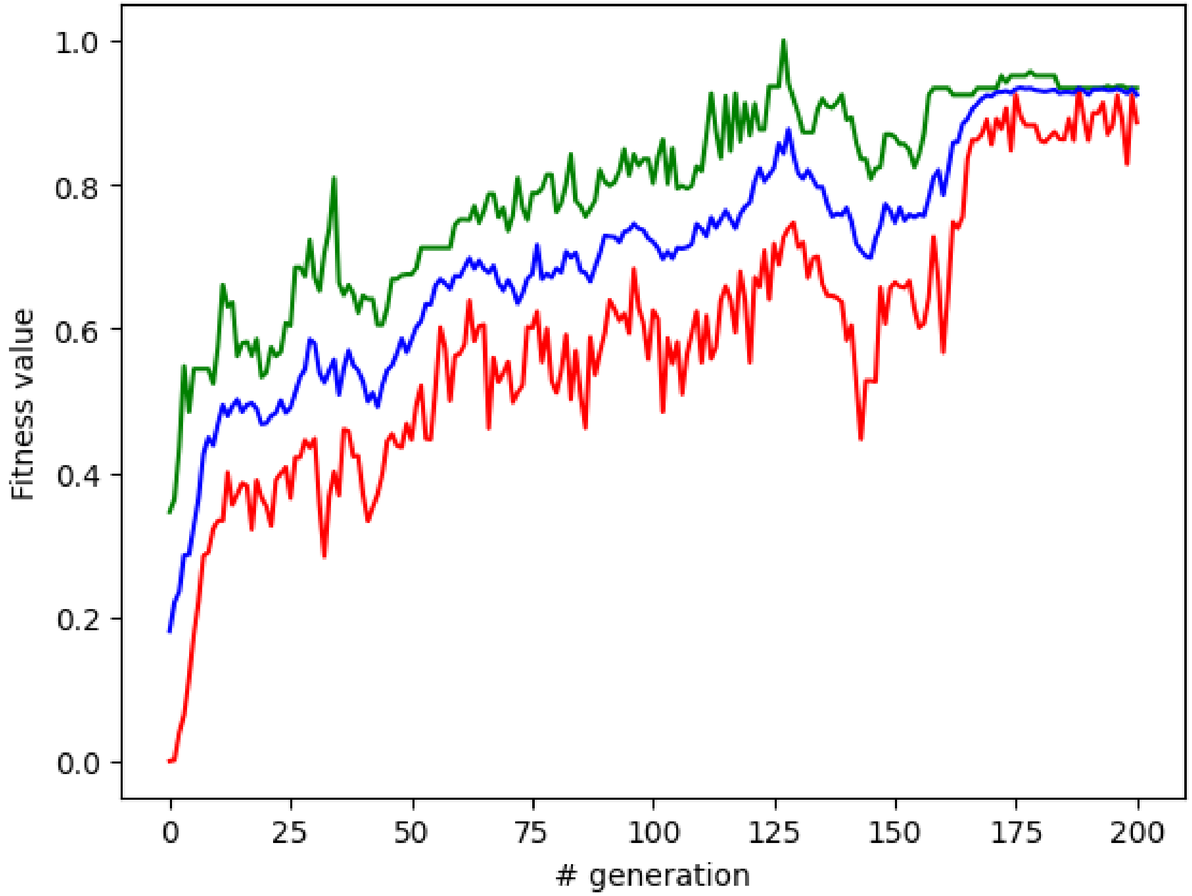
Larger Graph (N = 50)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 200
- Mutation: swap, cross-over: pmx



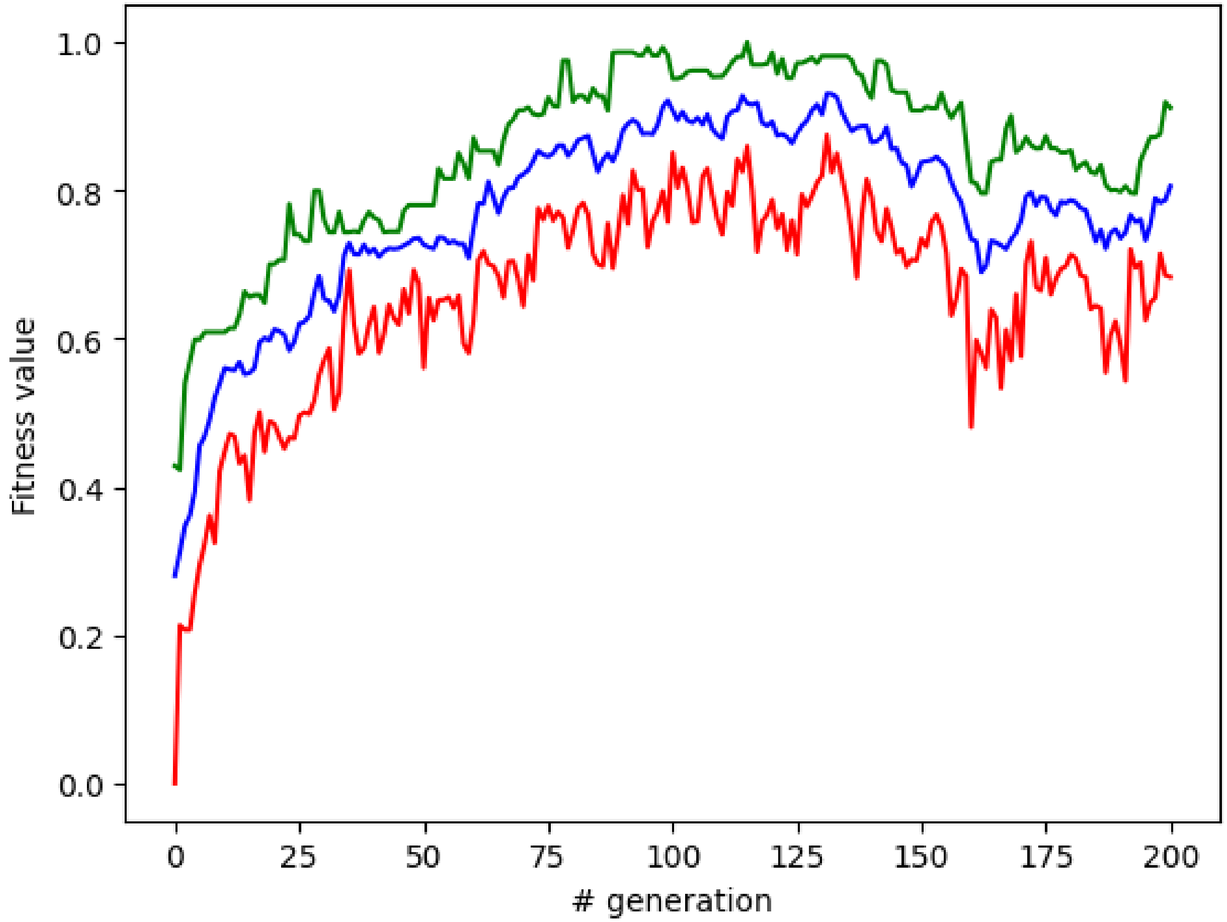
Larger Graph (N = 50)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 200
- Mutation: swap, cross-over: Ox



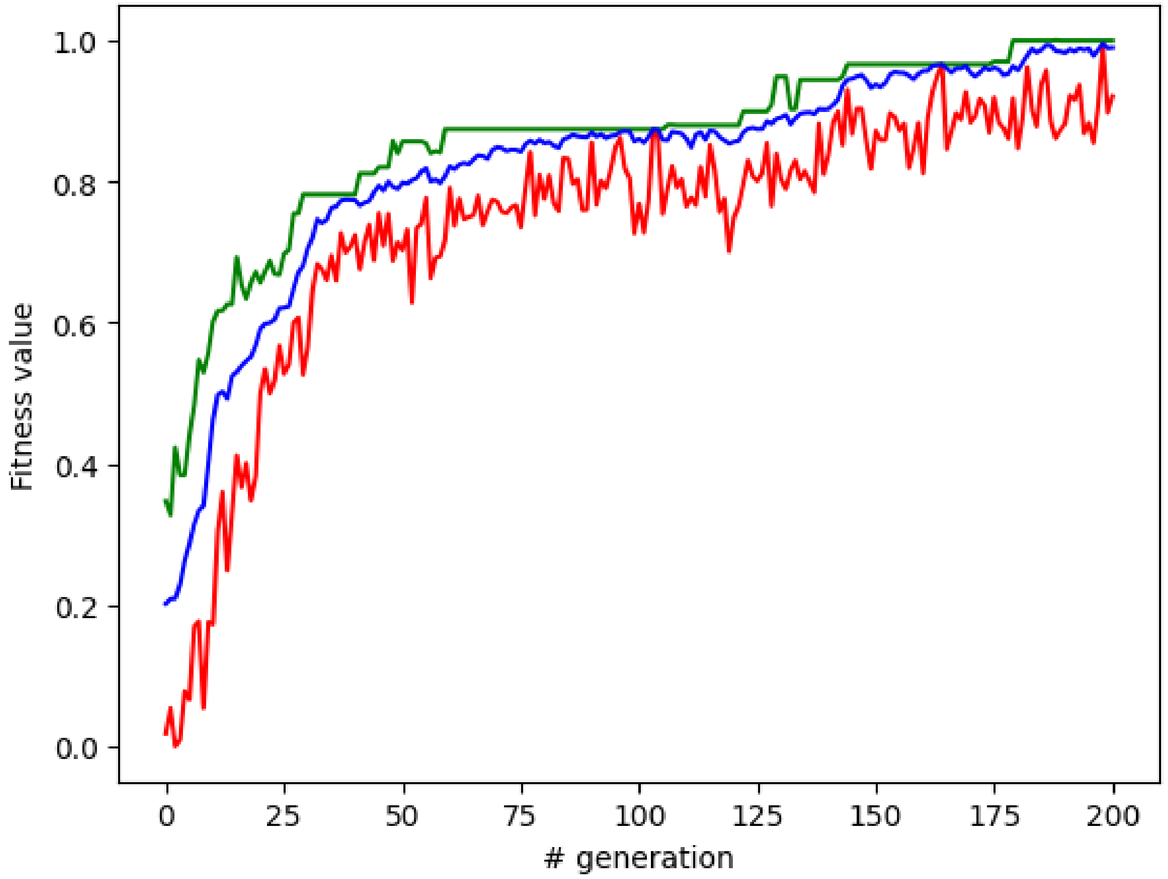
Larger Graph (N = 50)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 200
- Mutation: insertion, cross-over: Ox



Larger Graph (N = 50)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 200
- Mutation: insertion, cross-over: pmx



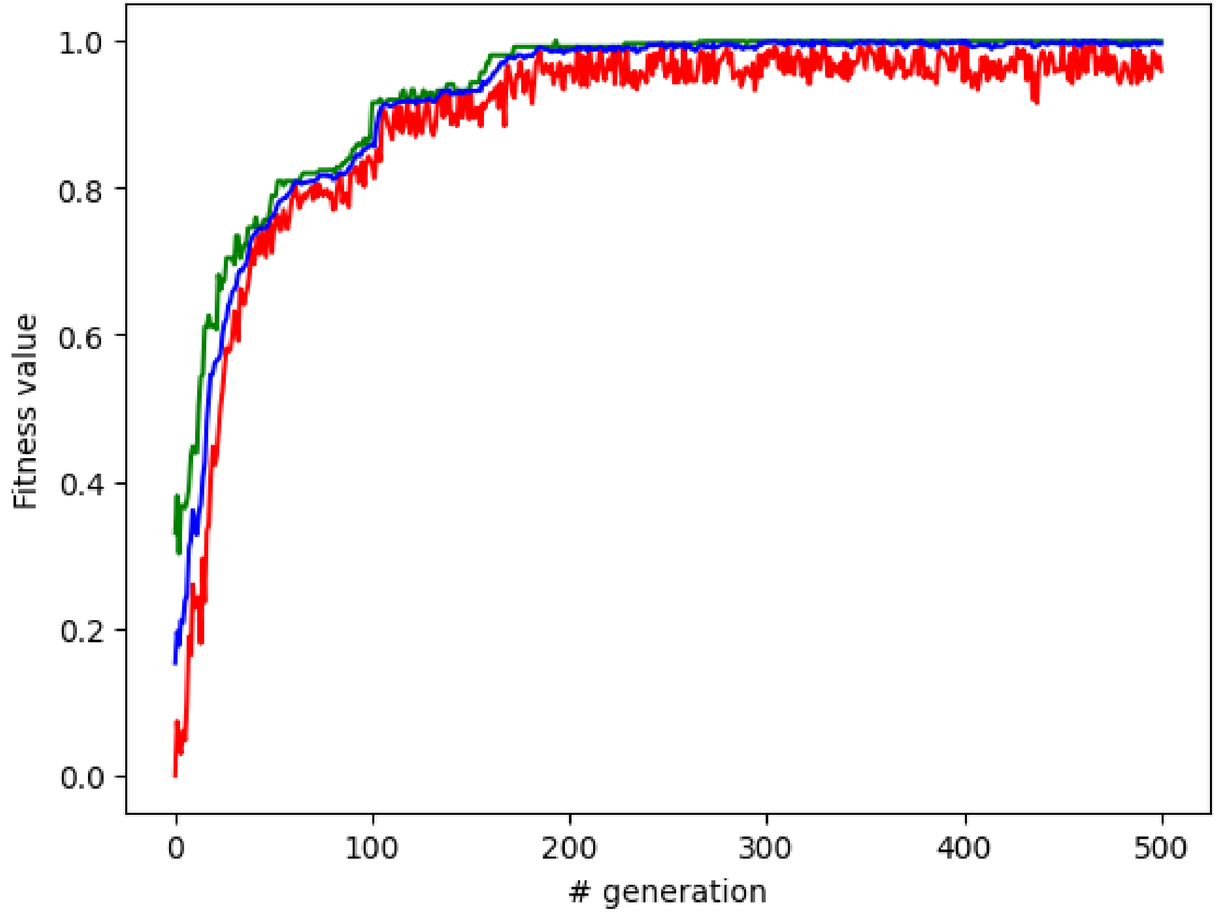
Comparison

- The minimum path length found by the different combinations of mutation & cross-over functions are for N = 50

	Swap	Insertion
OX	1218	1220
PMX	1028	1093

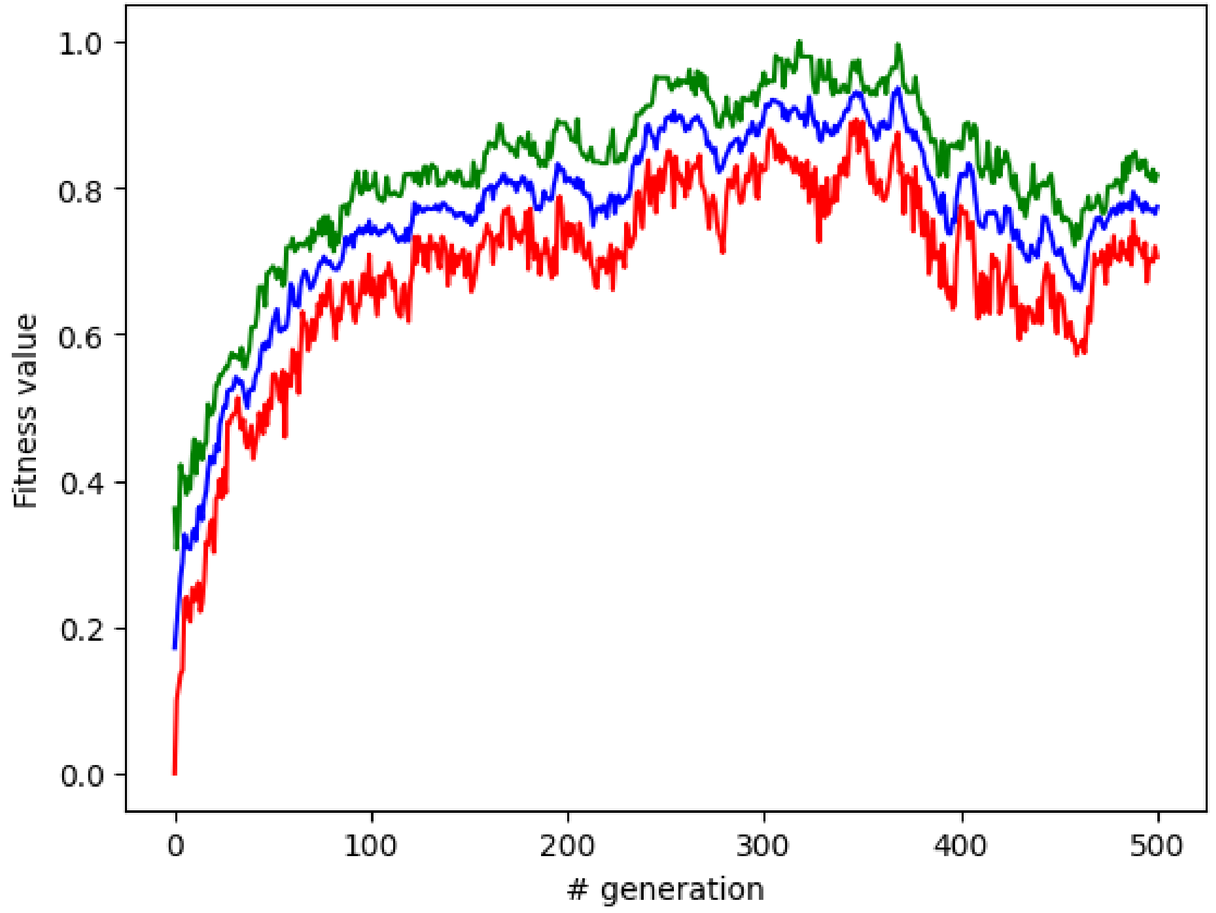
Larger Graph (N = 100)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 500
- Mutation: swap, cross-over: pmx



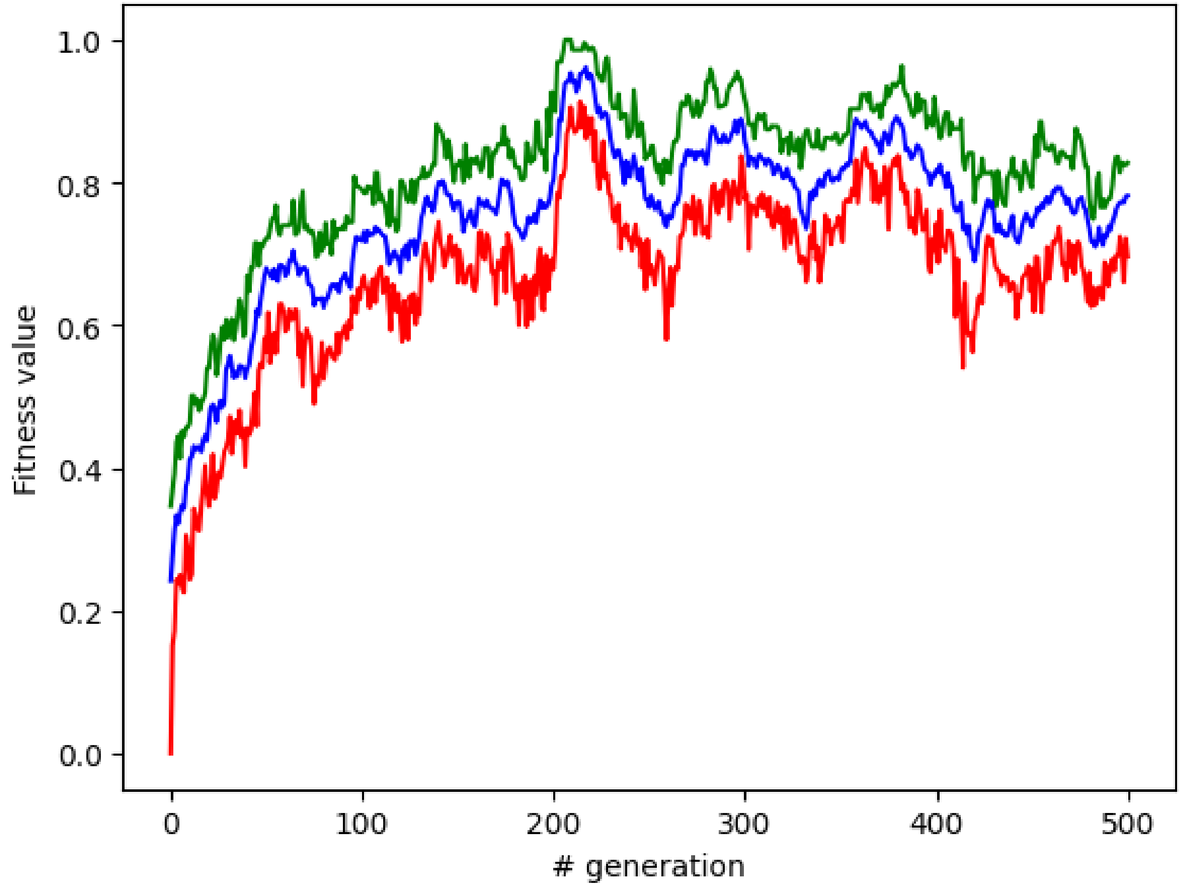
Larger Graph (N = 100)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 500
- Mutation: swap, cross-over: Ox



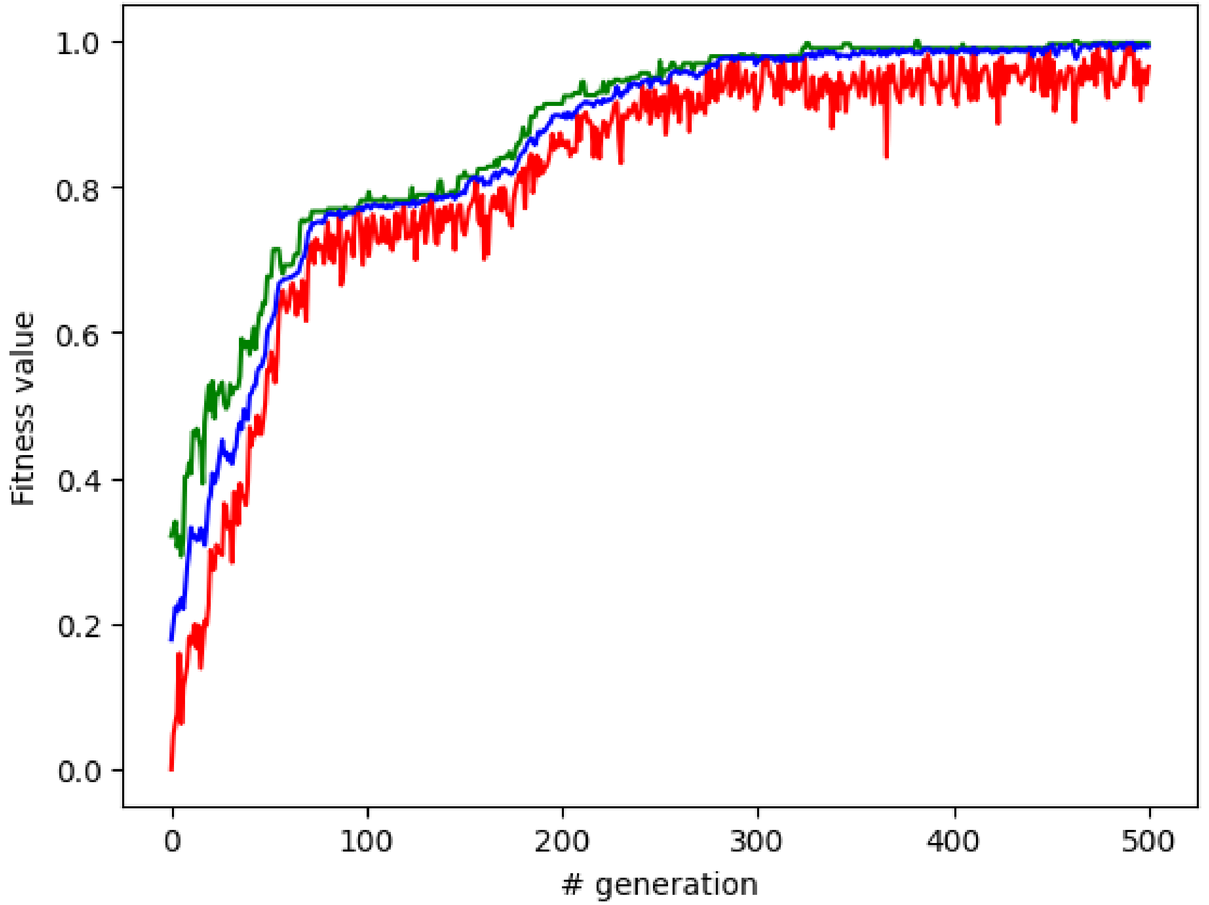
Larger Graph (N = 100)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 500
- Mutation: insertion, cross-over: Ox



Larger Graph (N = 100)

- # parameters: popsize = 20, pmutate = 0.1, pcross = 0.9, ngen = 500
- Mutation: insertion, cross-over: pmx



Comparison

- The minimum path length found by the different combinations of mutation & cross-over functions are for N = 100

	Swap	Insertion
OX	2652	2830
PMX	2733	2197

Conclusion & Future Work

- In general PMX cross over seems to be better than OX, especially for large graphs
- This is in contradiction to the claim in paper [2] which claims that OX is better in general.
- There is unlikely to be major difference between swap & insertion mutation
- Running the code takes less than a second N = 100. Could consider larger graphs.
- Could consider different parameter values for pmutate, pcross, population size etc.
- Could introduce elitism
- Overall Genetic algorithm converged & found good solutions to Travelling salesman problem for varying graph sizes.

References

[1] An Improved Genetic Algorithm for Solving the Traveling Salesman Problem by Pen Chang

[2] Genetic algorithms for the traveling salesman problem Jean-Yves Potvin