H3
Solving the Travelling Salesman Problem using a genetic algorithm underpinned by a heuristic method

Opris Vlad, 2A3

January 10, 2024

Abstract

This report studies the behaviour of two possible solutions for a NP-Hard problem, that being the Travelling Salesman Problem[2]. This paper presents two methods: a heuristic approach (Nearest Neighbor Heuristic) and an evolutionary one through using a genetic algorithm[2]. The report explains how the the algorithms works and show the results of tests performed. The experiments will be performed on ten instances of sample instances for the TSP[3].

1 Introduction

Genetic algorithms are a optimization technique which can be applied to various problems, including those that are NP-Hard. The technique does not ensure an optimal solution, however it usually gives good approximations in a reasonable amount of time. This, therefore, would be a good algorithm to try on the traveling salesman problem, one of the most famous NP-hard problems.
In the traveling salesman problem we wish to find a tour of all nodes in a weighted graph so that the total weight is minimized. The traveling salesman problem is NP-Hard but has many real world applications so a good solution would be useful.

2 Methodology

The nearest neighbour (NN)[2] algorithm (a greedy algorithm) lets the salesman choose the nearest unvisited city as his next move. This algorithm quickly yields an effectively short route. For N cities randomly distributed on a plane, the algorithm on average yields a path 25% longer than the shortest possible path. However, there exist many specially arranged city distributions which make the NN algorithm give the worst route and because often the last edge added to the tour can be quite large.

The next algorithm follows the structure of a standard genetic algorithm. Usually the first step in a genetic algorithm is to generate a random population of candidate solutions (in our case a cycle that visits all the cities exactly once), but instead, this time the algorithm starts from the tours given by the NN heuristic and evolve these solution over multiple generations. Roulette wheel method is used for the selection step. This method follows the idea that each individual gets a number of descendants proportional with their fitness. Elitism is also implemented along the selection to ensure that the best solutions will pass to the next generation. The role of a fitness function is to measure the quality of the chromosomes and make sure the best solutions get a high score, thus having a higher change of getting selected. The fitness function works inversely proportional in relation to the route length.

The next components of the algorithm are crossover and mutation. The crossover of the population works as follows: in every generation an individual gets a change to be selected for the crossover operation according to a crossover probability. For the actual crossover, "Order-Crossover"[4] is used. This results in two offspring which are added to the population. The selection process will subsequently take care to restore the size of the population to a base value. Regarding mutation ”Inverse Mutation” and ”Exchange Mutation”[4] are employed. Both are used at the same time and applied according to different mutation probabilities. The algorithm ends when it performs a predetermined number of generations.

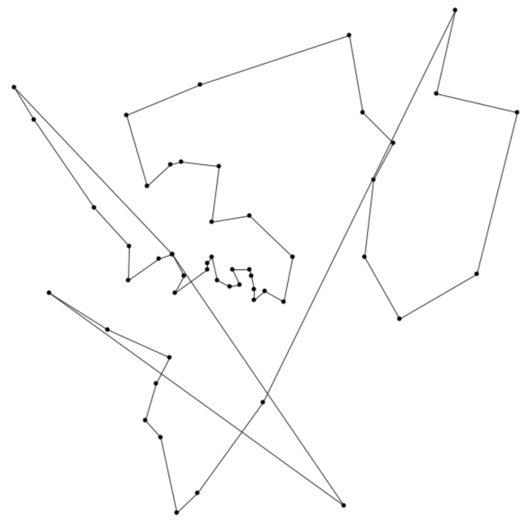
3 Experiment results

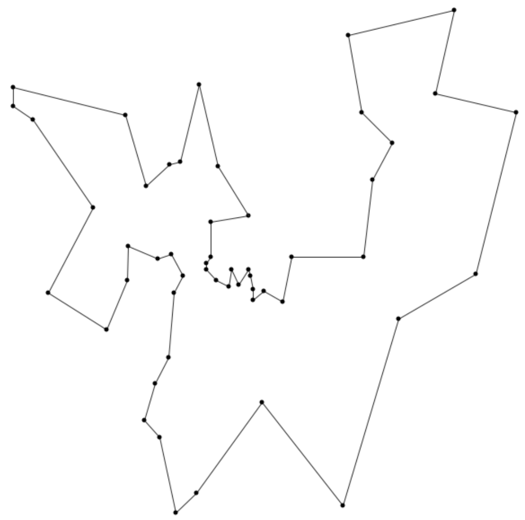
3.1 Parameters

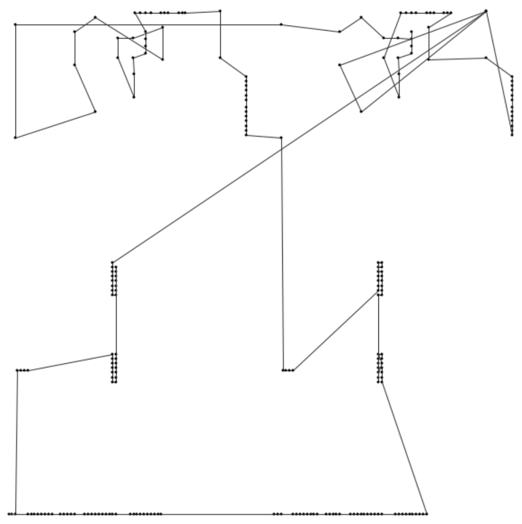
Tests	30
Population size	200
Generations	2000
Elitism	10%
Crossover prob.	0.9
Mutation prob.	1/N

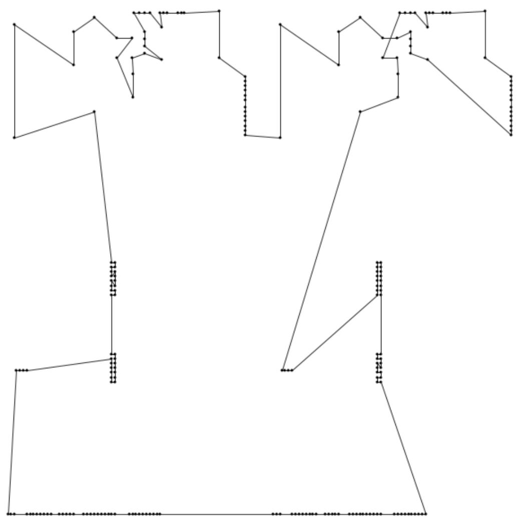
3.2 Results

Sample instances			Heuristic	Genetic Algorithm			
Name	#cities	Optimal Len.	NN(Avg)	Best Len.	Average Len.	Sd	Time(s)
berlin52	52	7542	9192	7544	7632	110	2
eil101	101	629	812	685	701	13	5
d198	198	15,780	18,772	16,814	17,053	231	9
pr226	226	80,369	99,398	83,364	84,934	954	10
a280	280	2579	3287	2843	2889	43	12
rat575	575	6773	8532	7712	7903	94	25
pr1002	1002	259,045	327,288	296,349	298,770	2665	66
nrw1379	1379	56,638	70,318	65,558	65,911	255	92
pr2392	2392	378,032	472,092	448,319	449,339	721	134
rl5915	5915	565,040	691,727	671,942	675,348	2409	148


(a) berlin52 - NN


(b) berlin52 - GA


(a) pr226 - NN


(b) pr226 - GA

4 Conclusion

There are many other ways to approach TSP, and genetic algorithms are just one of the many approaches one may take. It is also not the most effective way, as iterating over generations and generations can often take a lot of time.

References

[1] Course page
`https://profs.info.uaic.ro/~eugennc/teaching/ga/`

[2] Wikipedia
`https://en.wikipedia.org/wiki/Genetic_algorithm`
`https://en.wikipedia.org/wiki/Travelling_salesman_problem`
`https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm`

[3] TSPLIB
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf`

[4] Genetic operators
`https://mat.uab.cat/~alseda/MasterOpt/GeneticOperations.pdf`