Genetic Algorithms - Homework 3

Stefan Tomsa, Daniel Rusu

December 2019

Abstract

1 Introduction

1.1 Motivation

The goal of this paper is to observe how evoultive algorithms, genetic algorithms in particular handle combinatoric optimization problems and how genetic algorithms can yeald close results to NP-Complete problems in very little time. Troughout the paper we also observe the weight of all the evolutive operators for this problem on 3 types of instances: small(10-100), medium(100-300) and big(300-1200).

1.2 Traveling salesman problem

The travelling salesman problem (also called the travelling salesperson problem or TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

2 Method

2.1 Encoding

No encoding is used on this experiment because of the use of cyclic crossover that requires access to the decoded permutation.

2.2 Instances

The instances chosen for this experment can be grouped into 3 categories:

Small instances

- berlin52

Medium instances

- ch130
- ch150
- d198
- a280

Big instances

- d493
- d657
- d1291
- vm1084

2.3 Optimisation tehniques

Cyclic Crossover

Cycle crossover is used for chromosomes with permutation encoding. Cycle crossover performs recombination under the constraint that each gene comes from the parent or the other [Oliver et al. 1987]

Hyperparameter tuning

The parameters of the genetic algorithm were obtained using random search. The parameters were randomly generated then the evolutive algorithm was ran on ch130 for 500 generations. The number of generations can be lower here because most of the optimization is done in the early stage of the evolution.

The found parameters are:

- popSize': 90
- 'eliteSize': 12
- 'mutationRate': 0.004107141255506286,

2.4 Rank based selection paired with elitism

Rank Selection sorts the population first according to fitness value and ranks them. Then every chromosome is allocated selection probability with respect to its rank. Rank selection is used due to it being a more explorative tehnique. To counter balance the explorative nature of rank based selection the elitism is used.

3 Results

Instance	Min	Max	Stdev	Min / Optimum
a280	11541.963281496814	14114.035718612706	662.4211278470108	4.1833864739
berlin52	9108.31810460427	11561.510361347806	601.1099454779345	1.20767940926
ch130	12029.88615288532	14673.480881173615	726.664120499055	2.37657338516
ch150Best	14520.86338332076	17532.27371024473	711.7462395900847	2.22439696436
d198Best	48237.70414967518	59297.68831311495	2995.7032406940125	3.05688872938
d493Best	213460.27611452245	225937.60959290288	3476.8357241358026	6.09851654518
d657Best	479895.2073092939	505542.4972833134	6303.005824792107	9.81140021486
d1291Best	1202783.3917260212	1239318.4488844858	9308.80600380999	23.6763723495
vm1084Best	5416684.48968811	5673685.61075489	57788.89187585628	22.6358228047

Instance	Result
a280	30095.4
d198	60897.2
berlin52	22769.9
d493	275482
ch150	45760
d1291	1.15254e+006
d657	472038
ch130	39724.5
vm1084	8.06018e+006

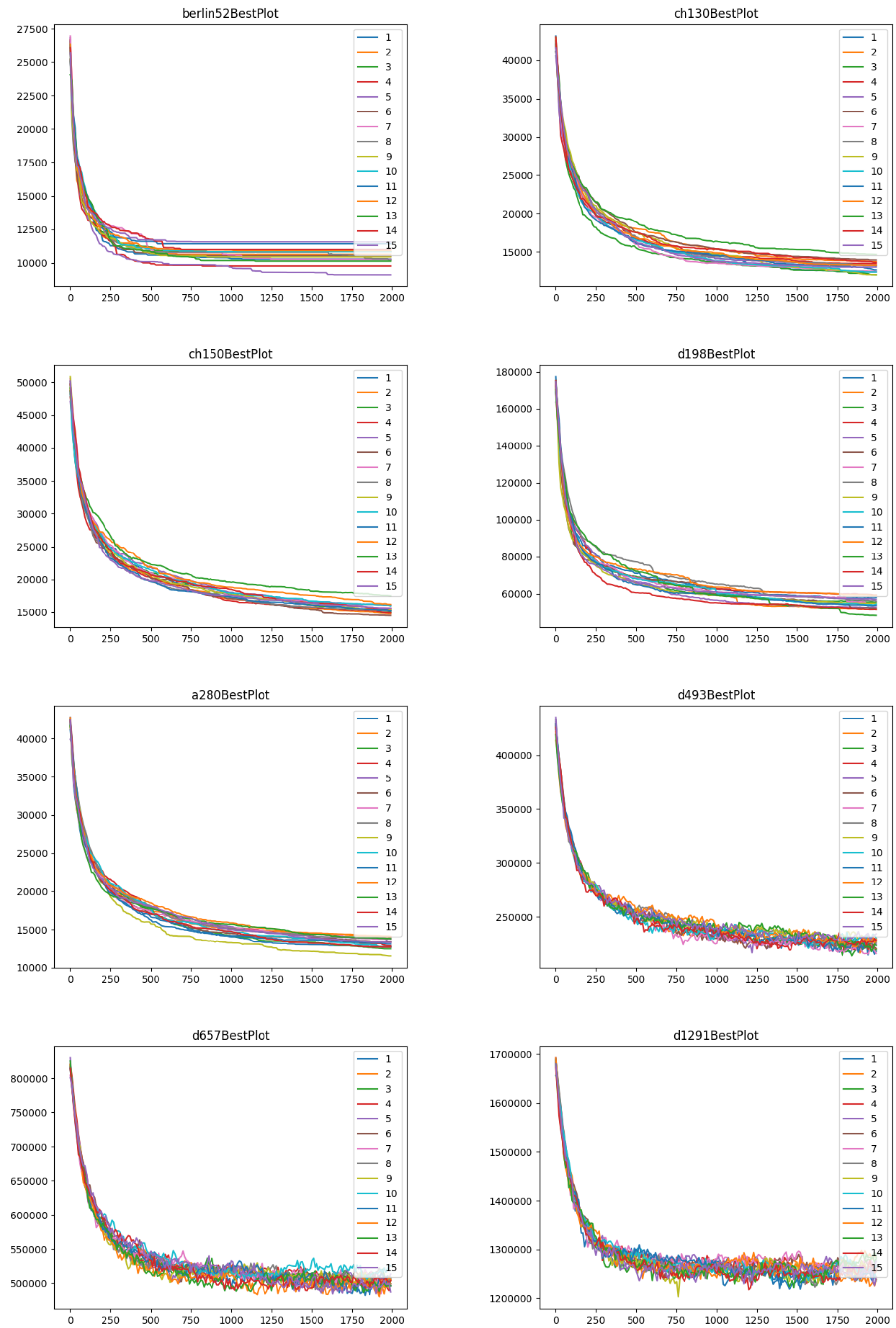

Figure 1: Convergence of the best member of the population during 15 runs

4 Interpretation

On the small instance the result is close to the global optimum and this should happen on smaller problems too.

On all runs we can observe on the plots that almost all the improvement was done during the first 500 generations making it seem like the 80/20 rule applies to combinatorical optimization.

We can observe from the graphs that the results tend to modify in a similar manner on all runs and on all 3 classes of instances.

The results could probably be further improved by using a nearest neighbour algorithm to generate the initial population.

The results could probably be further improved by using an algorithm based on nearest neighbour to reorder elements of aleles every once in a while.

The $Min/Optimum$ raport tends to increase linearly with the dimensionality.

The results are better than the ones given by the random search algorithm on all 3 classes of inputs but the difference tends do decrease as the dimensionality increseas this would suggest that on a very big instance random search might yeald better results without further optimisation on the genetic algorithm.

5 Conclusion

The mutation rate should be lower on Traveling Salesman Problem and probably on most combinatorical optimization problems compared to numerical optimization problems. This can be due to the discrete nature of combinatorical optimization the steps on the gradient can be way bigger than in combinatorical optimization.

The mutation rate being low the evolution of the solutions is highly dependent on the crossover and the selection.

Genetic Algorithms yeald better results than a random search by a great margin on instances where the number of nodes is lower than 1000

5.1 Open questions

How would the results be impacted if Hyperparameter tuning was ran on a bigger instance?

How would the results would be impacted if Bayesian search would be used for Hyperparamer tuning?

How do different crossover variation affect the results

References

[1] https://shodhganga.inflibnet.ac.in/bitstream/10603/32680/19/19_chapter%209.pdf

[2] https://www.researchgate.net/publication/226665831_Genetic_Algorithm_for_the_Travelling_Salesman _Problem_A_Review_of_Representations_and_Operators

[3] https://profs.info.uaic.ro/p̃mihaela/GA/laborator3.html

[4] Stefan Tomsa, Genetic Algorithms Homework 1

[5] Stefan Tomsa, Genetic Algorithms Homework 2

[6] Stefan Tomsa, Genetic Algorithms Homework 2’

[7] Daniel Rusu, Genetic Algorithms Homework 1

[8] Daniel Rusu, Genetic Algorithms Homework 2

[9] Daniel Rusu, Genetic Algorithms Homework 2’

[10] https://en.wikipedia.org/wiki/Travelling$_{s}alesman_{p}roblem$