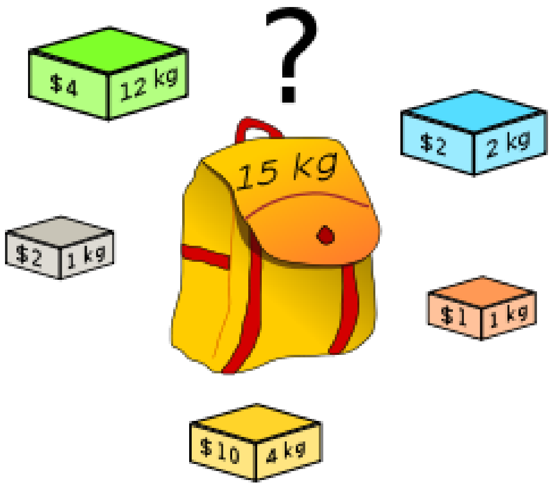

Computer Science w/ Artificial Intelligence - Acquired Intelligence Adaptive Behaviour

Effects of parameters on Genetic Algorithm Success

November 21, 2022

Report

234591

Contents

1 Introduction **2**

2 Methods **3**

2.1 Knapsack Problem . . . 3

2.2 Genetic Algorithms . . . 3

2.3 Genetic Operators . . . 4

2.3.1 Mutation . . . 4

2.3.2 Crossover . . . 4

3 Results **4**

3.1 Population Size and Fitness . . . 5

3.2 Mutation Rate Size and Fitness . . . 5

3.3 Crossover Probability Size and Fitness . . . 5

3.4 Neighbourhood Size and Fitness . . . 6

3.5 Tournament runs and Fitness . . . 6

4 Discussion **7**

4.1 Predictions . . . 7

4.2 Tournament runs . . . 7

4.3 Population size . . . 7

4.4 Mutation Rate . . . 8

4.5 Crossover probability . . . 8

4.6 Neighbourhood Size . . . 9

5 Conclusions **9**

6 References **9**

Abstract

Since Holland's Adaptation in Natural and Artificial Systems, Genetic algorithms have become a household name for dealing with all types of heuristic search problems. A genetic algorithm is part of a class of algorithms that search a solution space for the optimal solution to a problem. The search method used is modelled after the laws of 'survival of the fittest' and evolution, using genetic operators like mutations and chromosomal crossovers. We have found success using genetic algorithms to find an optimal solution for the knapsack problem and used this optimal solution to find out the best values for the parameters we used. This included population size, crossover probability, mutation rate, number of tournaments ran and neighbourhood size. We chose to use tournament selection because it is computationally more efficient and more amenable to parallel implementations if we were to expand this. We found that a delicate balance between mutation rate and crossover probability is necessary when implementing a successful genetic algorithm.

1 Introduction

In the theory of evolution the ability to survive and reproduce is defined as success. This drive to survive has compelled the human species to create a range of technologies that make our survival easier However, humans are at a precipice in the development of our species. We created automation to replace physical labour but now, through the use of methods like Genetic Algorithms, we are now creating AI to reduce mental labour as well. A genetic algorithm is an adaptive search heuristic, inspired by the theory of evolution by Charles Darwin where inspiration comes in the form of methods that imitate mutation, chromosomal crossover and selection. Since Holland's 'Adaptation in Natural and Artificial Systems',Genetic Algorithms have been applied to a wide range of problems including a subset of problems called the knapsack problem.In this report, I intend to discuss the Genetic Algorithm that we created to solve the knapsack problem and use this to investigate how the different parameters alter the success the Genetic Algorithms.

With this background, I aim to investigate the following question:

What effect does a change in the parameters of a Genetic Algorithm have on the success of the knapsack problem

---

2 Methods

2.1 Knapsack Problem

1. let V be the Volume
2. let N be the number of items
3. let B be the Benefit
4. let i be the index

The goal is to maximise benefit:

$$\sum_i^N B_i$$

Subject to the constraint that:

$$(\sum_i^N V_i) \leq V$$

Given a set of items, each weighted and valued, determine a set a items in the collection so that the total weight is equal or less than the limit and that value is the largest out of all the combinations of items. This problem is subset of the combinatorial optimization problems.

2.2 Genetic Algorithms

In a Genetic Algorithm, a population of genotypes, with encoded answers for a given problem, are pushed towards a high-quality solution. Each phenotype has its own attributes usually represented in form a set of randomly generated binary digits which can be altered by mutation and other methods.Our algorithm uses tournament selection

Each genotype/individual in the population had a one-dimensional array of binary digits where the length of the array was set by the number of genes variable. We then created a 2-dimensional array of the population of genotypes. The binary digits in the genotype correspond to whether that genotypes is active for that index of benefit and volume. As a result, our genetic algorithm finds the best combination of binary digits to maximise the benefits while not exceeding the maximum volume

The steps in our algorithm proceeded as follows:

1. Initialise random pop P
2. Associate each individual with a position x , i.e, let the position of the genotype in the population matrix indicate the position on a 1D grid.
3. Pick one individual at random, i.e. genotype $G1$ at position $x1$
4. Pick a second individual G2 in the local neighbourhood of the first, i.e., pick a competitor from the local neighbourhood in the range of $x1+1$ to $x1+k$
5. Compare $G1$ and $G2$ finding a winner (W) and loser (L)
6. Copy each gene of the winner W to the L with crossover probability (Pcrossover, say 0.5 to start)
7. Add a mutation to the L and insert it back to the population.
8. Until success or give up, goto 3

2.3 Genetic Operators

Genetic operators are functions used in GA's to steer an algorithm to the highest quality solution. There are a range of genetic operators that can be used but I will discuss the ones that were used in our GA.

2.3.1 Mutation

In our GA we used the bit mutation genetic operator which took in a mutation rate and a randomly generated number between 0 and 1 for each gene in the genotype. If this number was bigger than the mutation rate then that gene would be flipped (e.g. 1 -> 0). Our intention with mutation operator was to assist with genetic diversity and discourage the algorithm from converging at a local min

2.3.2 Crossover

Given two genotypes, a loser and a winner, and a crossover probability our crossover function went through each gene in the genotype and decided whether the winner gene should be copied into the loser gene. We did this by generating a random number between 0 and 1 - if this number was bigger than the mutation rate then the winners gene would be copied to the losers gene. We used the genetic crossover operator because it allows for beneficial gene modification due to the guarantee that the genotype of the winner has a higher fitness.

3 Results

Optimal Parameters:
Neighbourhood - 8 (on a population of 25)
Crossover Probability - 0.9
Tournament Runs - 1500
Mutation Rate - 0.1
Population Size - 20

3.1 Population Size and Fitness

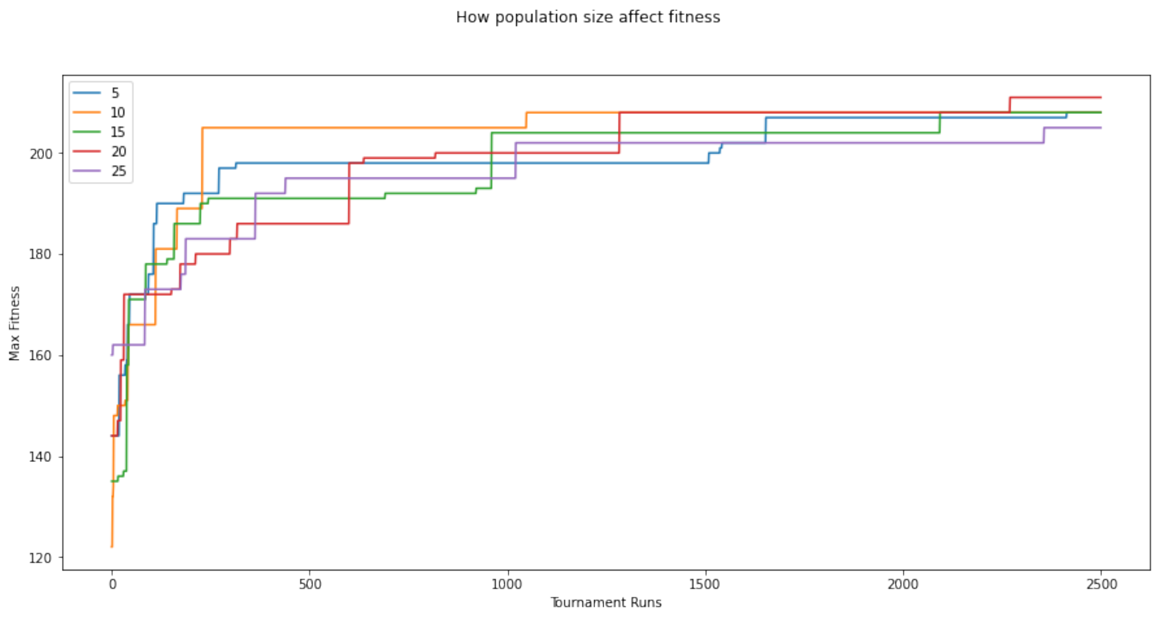

Figure 1: Performance comparison of population size from 5 to 25

Neighbourhood - Population / 2

Crossover Probability - 0.6

Tournament Runs - 2500

Mutation Rate - 0.1

3.2 Mutation Rate Size and Fitness

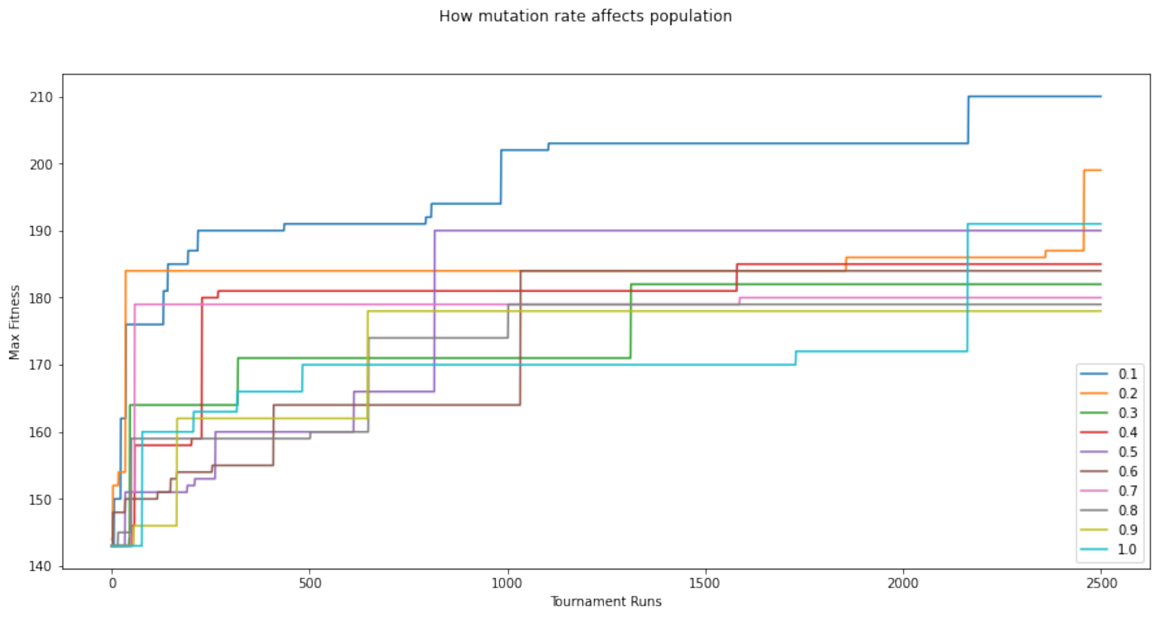

Figure 2: Performance comparison of mutation rate from 0.1 to 10

3.3 Crossover Probability Size and Fitness

Neighbourhood - 12

Tournament Runs - 2500

Mutation Rate - 0.1

Population size - 20

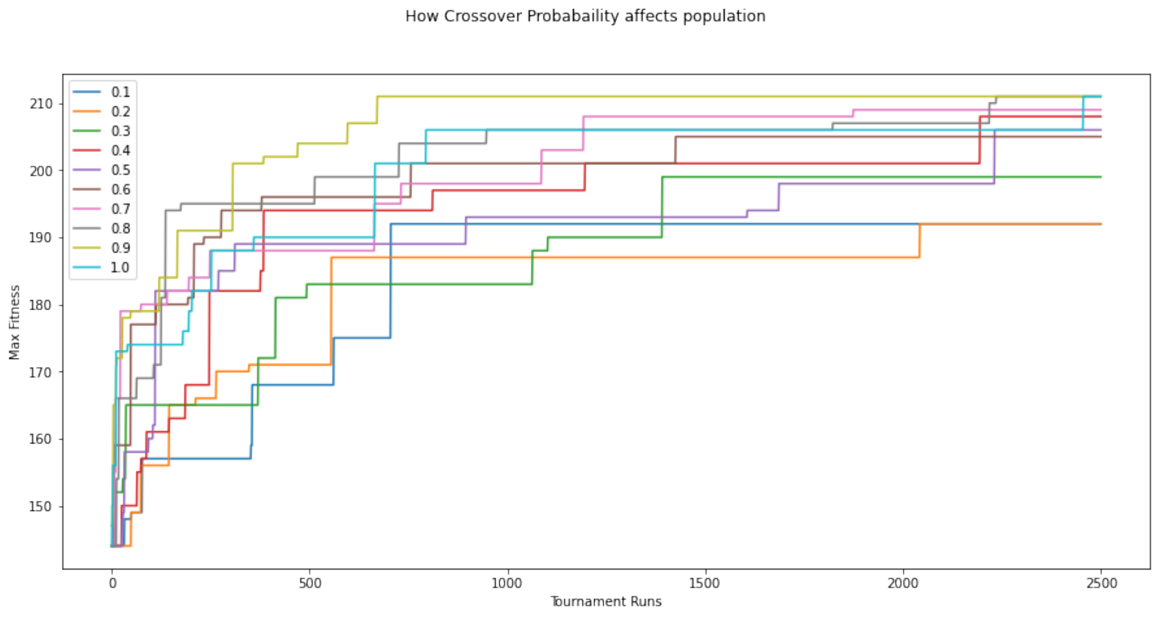

Figure 3: Performance comparison of Crossover Probability from 0.1 to 10

3.4 Neighbourhood Size and Fitness

Crossover Probability - 0.6

Tournament Runs - 2500

Mutation Rate - 0.1

Population size - 20

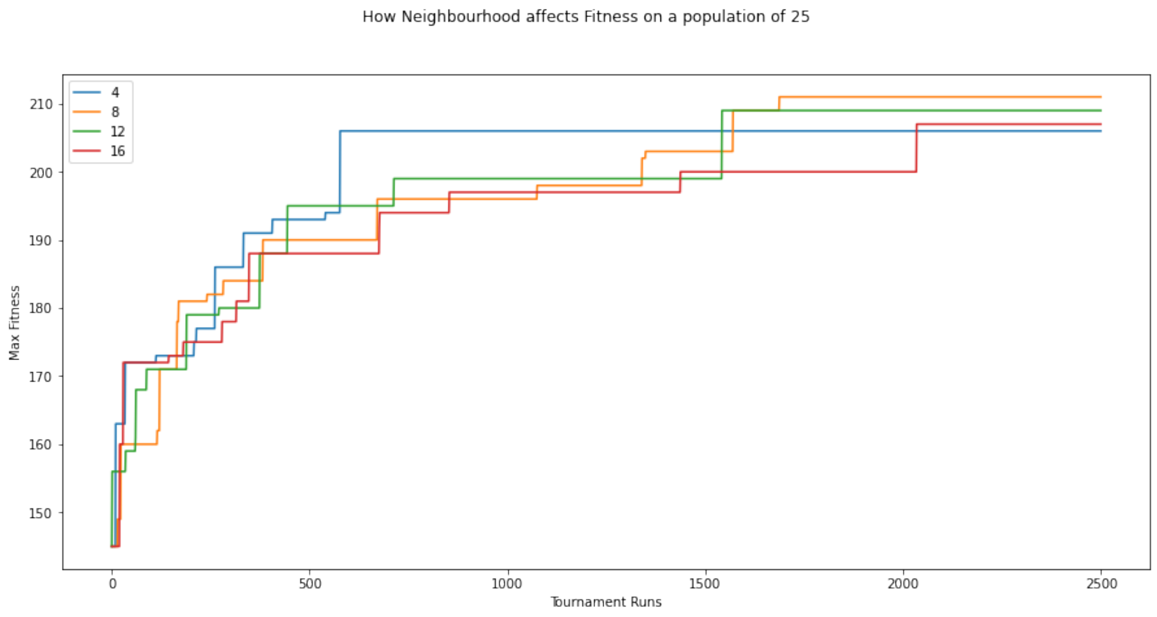

Figure 4: Performance comparison of Neighbourhood Size from 0.1 to 10

3.5 Tournament runs and Fitness

Neighbourhood - 4

Crossover Probability - 0.6

Population Size - 20
Mutation Rate - 0.1

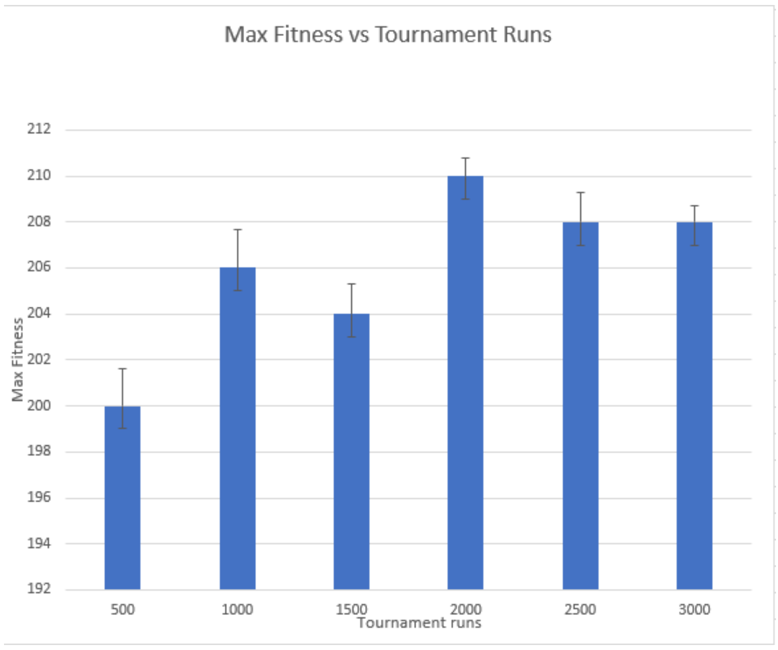

Figure 5: Performance comparison of Tournament runs against Max Fitness

4 Discussion

4.1 Predictions

In order to investigate how the different genetic operators affect fitness, we ran the tournament selection function several times, calculating the mean and the spread of each instance of a parameter. We initially predicted that population size,crossover probability and the number of tournament runs would be the most important factors when increasing fitness.An increase in population size will allow for more genes to be randomly generated giving a higher chance for genes with a higher base fitness to be encoded and subsequently passed on and evolved through the generations. A high crossover probability guarantees that the loser will increase in fitness as the winner is the transferring higher quality genes to the loser[**riazi_2019**]. Lastly, by increasing Tournament runs, the individuals will have more of a chance to interact and share genes which will consequently lead to a higher max fitness. On the other hand, we expected a high mutation rate to actually be detrimental to the genetic algorithm as a high mutation rate will reduce the number of useful genes that a loser will receive from crossing over with a winner.[**marek**]

4.2 Tournament runs

We expected the tournament runs to be directly proportional in it's increase because more tournament runs leads to more of a chance to share beneficial genes. However, we theorise that the data would show this trend had we measure total population fitness or average fitness. Another simple explanation could be due to the crossover probabilities and mutation rates not being as favourable for the higher tournament runs.

4.3 Population size

We estimated population size to be 25 (the highest population size) because we believed that a higher population size would bring more genetic diversity in the population which is beneficial to finding new local minimums. Interestingly, the population size did not seem to make that much of a difference with the max fitness and all the populations turned out to finish within in a range of 10.

4.4 Mutation Rate

Our predictions regrading the mutation rate was correct for the most part - a low mutation rate gave the beneficial genes from crossing over a chance to flourish and evolve into grander solutions. Despite this, the mutation rate is still highly important as mutations retain genetic diversity which reduces the chance for a population to be isolated in a local minmum.

4.5 Crossover probability

Crossover probability was the most straight-forward prediction as the benefit of crossing over is the gain of beneficial genes to the loser. Irrespective of this, it is important to not have too high of a crossover probability because genetic diversity can easily be lost if there is a straight asexual reproduction of genes and no mutations to provide genetic diversity.[**marek**]

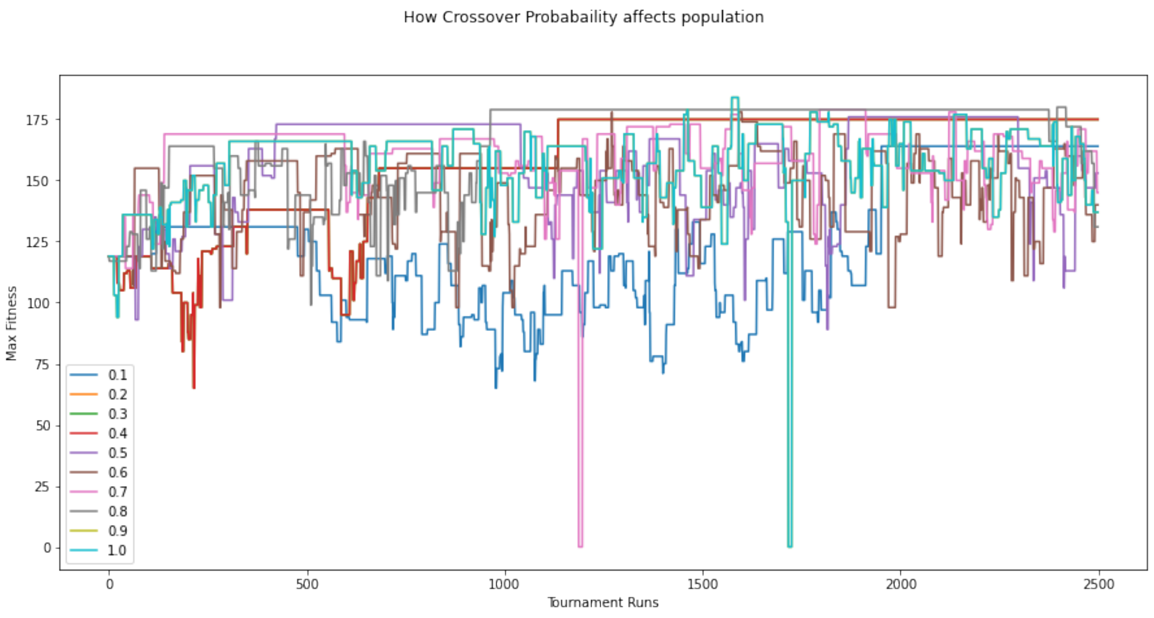

Figure 6: Performance of the best base individual with different Crossover Probabilities

Crossover Probability	Standard Deviation
0.1	1.24
0.2	1.17
0.3	1.09
0.4	1.02
0.5	0.97
0.6	0.94
0.7	0.91
0.8	0.89
0.9	0.87
1.0	0.85

Figure 7: Standard deviations and crossover probabilities for the best individual

4.6 Neighbourhood Size

Neighbourhood size is an interesting factor that we involved. Neighbourhoods do not alter a genes however, with the addition of neighbours, Hillis found that spatial neighbourhoods helped preserve diversity by maintaining spatially isolated species, with innovations largely occurring at the boundaries between species. As a result, Neighbourhoods do not impact the success of the GA when compared to the other parameters.

5 Conclusions

The Theory of evolution provides a solid foundation to solving problems that computer scientists do not know the answer to initially. We found that with a delicate balance between the mutation rate and the crossover probability provides the best chance of success. Having too high of a mutation rate restricts progression in a known beneficial direction but, a high crossover rate restricts genetic diversity in a population which causes species to be isolated in local minima [**riazi_2019**].We also discovered that Many advocates of genetic algorithms believe that in order to create intelligent AI we need highly adaptable and massively parallel systems that are able handle complexity. Genetic algorithms are a good first step in the evolution of computing systems based of biology.

6 References

Anon, 2020. Genetic algorithm: Application of genetic algorithm. [online] Analytics Vidhya. Available at: lt;https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/gt; Anon, n.d. Modelling and simulation of Integrated Systems in engineering. [online] ScienceDirect. Available at: lt;https://www.sciencedirect.com/book/9780857090782/modelling-and-simulation-of-integrated-systems-in-engineeringgt;

Mallawaarachchi, V., 2020. Introduction to genetic algorithms - including example code. [online] Medium. Available at: lt;https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3gt;

Marek Obitko, marek@obitko.com, n.d. Xi. crossover and mutation. [online] Crossover and mutation - Introduction to Genetic Algorithms - Tutorial with Interactive Java Applets. Available at: lt;https://www.obitko.com/tutorials/ algorithms/crossover-mutation.phpgt;

Riazi, A., 2019. Genetic algorithm and a double-chromosome implementation to the traveling salesman problem - SN Applied Sciences. [online] SpringerLink. Available at: lt;https://link.springer.com/article/10.1007/s42452-019-1469-1gt;