AIAB CW1 Report

By 246743

Abstract

Genetic algorithms (GA) solve many of the problems of today's society, but the exact results that are obtained when implementing them, depending on the variation of the parameters, need to be investigated on their own. The proposed approach is to obtain different results in the Knapsack problem [1] by continuously changing the hyper-parameters of the GA at the time of the study. A series of hyper-parameters are studied [3], mutation rate, crossover rate, number of tournaments and number of neighbour. All the results are then plotted using the matplotlib toolkit, making it easy for the reader to visualize what is happening. A spread value, or standard error of mean, is also calculated to indicate the accuracy of the data. Here, the results show that the results get progressively better as some of the rates of change are reduced. The change in mutation rate is most clearly reflected in the variance. This illustrates the random nature of variation and crossover, which is more likely to lead to the final result staying at a local maximum if the probability is adjusted too much. This paper will also compare and analyse the experimental results with the data in [4] (for small population sizes such as 30, crossover rate=0.9, mutation rate=0.01)

Introduction

1. Basic Principles of GA

Darwin's doctrine of natural selection is a widely accepted doctrine of natural evolution. This doctrine holds that for organisms to survive, they must engage in a struggle for survival. In the struggle for survival, individuals with favourable variations will survive, while those with unfavourable variations will be easily eliminated.

Genetic Algorithm is a computational model that mimic Darwin's genetic selection and natural elimination. It is a search method that randomises survival and detection iterations. In this, mutation, crossover becomes the genetic operation in the genetic algorithm. The design of fitness functions, and the design of genetic operations are the core elements of genetic algorithms.

2. GA Operations Description:

Individuals of length L (also known as genotypes) form an initial population, each of which contains several genes. There are three types of operations performed on this population.

2.1 Selection:

A pair of genotypes is randomly selected from the population. They need to compare their fitness values with each other, and the one with the higher score will be the winner.

2.2 Crossover

After the winner has been decided, the losers will cross over. The crossover will be with the winner, i.e. the loser's genes will be covered by those of winner in a certain probability. It is an important way to optimize the population, but negative change is also possible.

2.3 Mutation

After the crossover, we will allow the loser to continue mutating, this is essentially a gene changed randomly in a certain probability.

3. Problem Model

In this study, the object of application is the treatment of the Knapsack Problem, which is a problem of resource allocation. Imagine that you want to put some objects into a finite space A. Each object has a different Volume V, and Benefit B. V represents the amount of space the object will occupy, while B represents the value of the object. When calculating the fitness function, we add up the Benefit of the selected objects. If the final Volume of all objects does not exceed the total space capacity A, then the fitness score is the sum of all the Benefit; conversely, if the objects exceed the total space capacity, then the individual will not receive a fitness score, which is 0.

The data of Benefits and Volumes are from Google. [2]

Methods

1. **Fitness Function**

According to the Knapsack Problem:

- Each item is either taken or not taken, so the gene is either 1 or 0.
- The total volume of all items must not overflow the total volume of the backpack, otherwise it defeats the purpose of the question and is meaningless[5].

The pseudo-code of fitness function is shown below

```
An individual have 50 genes,each gene is either 1 or 0
Max_volumes is given, it is the maximum limit

Initialise total_benefits to 0
Initialise total_volumes to 0

For each gene in this individual:
    Total_benefits add up (gene * corresponding benefits)
    Total_volumes add up (gene * corresponding volumes)

If total_volumes > max_volume Then:
    Fitness value is 0
Otherwise:
    Fitness value is total_benefits
```

2. Crossover Rate

The first step in making improvements to the loser is to cross the loser according to the winner. The basic concept is that given a crossover rate of, say, 0.4, then for each gene in the loser genotype there is a 40% chance that the gene in that position will be converted to the gene in the corresponding position in the winner.

In this experiment, the crossover rate will be experimented with from 0 to 1 at intervals of 0.1. The default value is set to

0.9.

3. Mutation Rate

After the above crossover operation comes the mutation operation, which is again different from the crossover. For mutations, the loser will take a random sample of a gene to be genetically altered. In practice, this means reversing the gene, going from 0 to 1, or from 1 to 0.

In this experiment, the mutation rate will be chosen to be studied from 0 to 1 being chosen at intervals of 0.1. The default value is set to 0.1.

4. Number of Tournaments and Trails

They can be affected the result as the more tournaments/trails experienced, the more likely it will have a answer that reflect the facts.

In this study, different numbers of tournaments and runs are brought in to aid comparison.

5. Spread (Standard Error of Mean)

Also known as Standard Error of Mean, is the degree of dispersion between sample means in multiple sampling and reflects the representation of the sample mean to the overall mean and is used in inferential statistics.[6]

In this experiment, the results of the calculations will be discussed as the mean of the results after multiple runs, so it is appropriate to use spread as a consideration for the reliability of the results.

6. Number of Neighbour k

When the selection operation is performed, instead of picking two individuals out of all of them completely at random, a second individual will be randomly selected from its neighbours after one has been randomly selected. The number of neighbours is denoted by k. The specific formula is as follows.

```
gene_1 = random_int(0, num_individuals)
gene_2 = random_int(g_1+1, g_1+k) % num_individuals
```

In this experiment, Numpy.random.randint() will be used to compute random numbers. The cases k=2 to k=10 will be considered to explore whether the spacing of variant individuals directly affects the final result. k is set to the default value of 3.

Comparisons

1. Comparison of different tournaments number

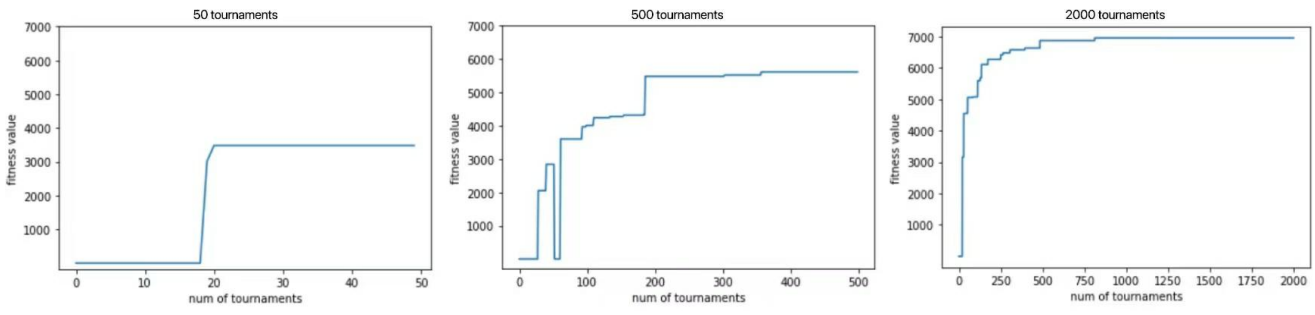

Fig 1

The best individuals are more likely to experience more updates when the parameters of the tournament are increased. As

shown in *Fig 1*, when there are only 50 tournaments, only one update occurs and it stays at the local minima (less than 4000). For the 500 and 2000 tournaments, the final fitness value also increased with the number of tournaments. This indicates that as an individual experiences more tournaments, it has the opportunity to progress more and thus end up with a higher score.

2. Comparison of different mutation rate

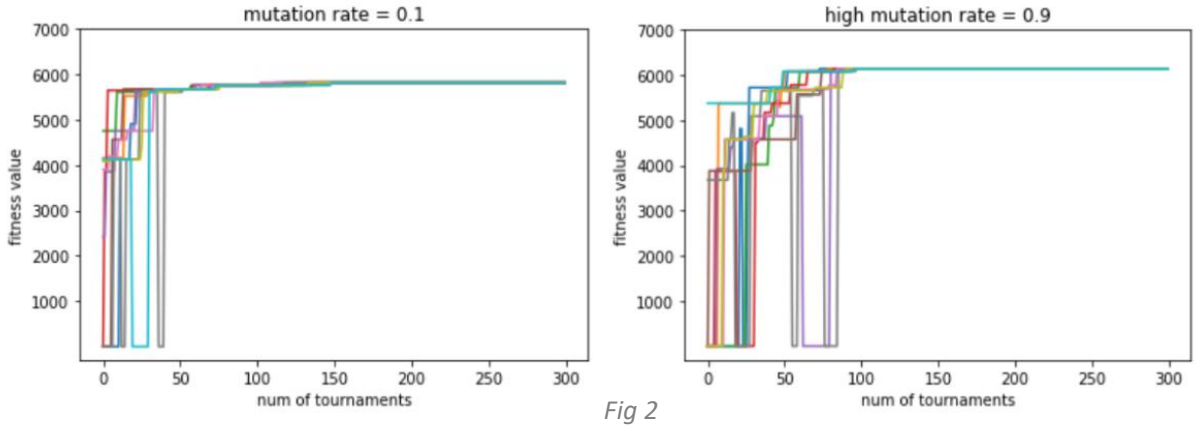

Fig 2

Mutation does not play a decisive role in the size of the final fitness value (many experiments have been run, with mixed results, all of which are similar in aggregate). However, there is no doubt that a higher mutation rate will give the data a greater chance of being updated. (*Fig 2*)

3. Comparison of different crossover rate

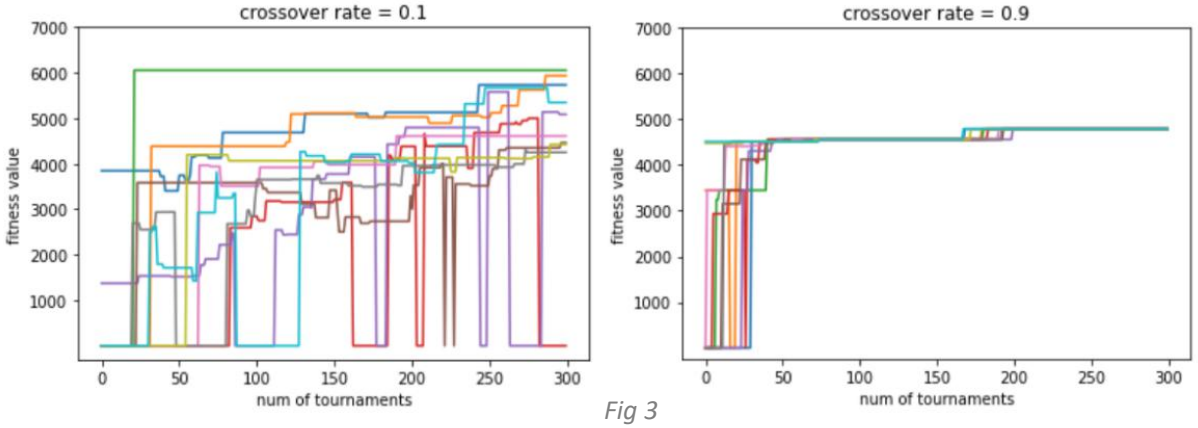

Fig 3

Similar to the mutation rate, crossover needs to be used in conjunction with other parameters in order to control the final result. This number should be large, as it is clear from *Fig 3* that when crossover equals a small number, the genotypes are very unstable and not robust throughout the process.

4. Comparison of number of neighbour

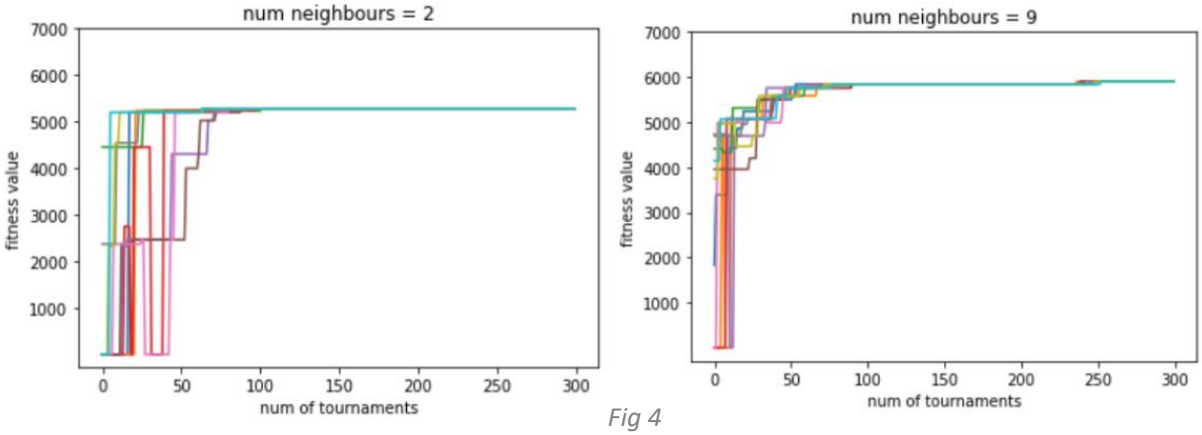

Fig 4

As shown in *Fig4*, when the number of neighbours is small, the genotypes in the early stages are dispersed, i.e. the scores are high and low. In contrast, when there are more neighbours, the genotypes are well integrated. This is because a larger number of neighbours brings more possibilities for optimising good genotypes.

Results

1. Best Genotype fitness vs tournaments

In the current model of the problem, the definition of "best" is to have as many benefits as possible. This means that the individual with the highest fitness score after the final tournament round will be considered the best genotype.

From the only run of Full_Microbial_GA, with the parameters of
individuals = 20, tournaments = 200, prob_mutate = 0.1, porb_cross = 0.9
The result is shown below

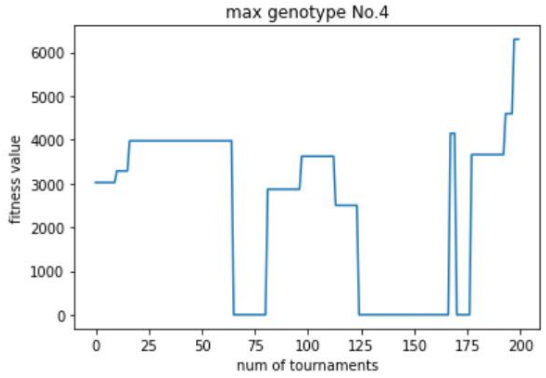

Fig 5

In *Fig 5*, it shows the changes during 200 tournaments of fitness of best genotype among these 20 ones. It is easy to see that this No.4 genotype has gone through several reversions to zero, which means that it keeps running into stronger and then improving, and although for a period of time (tournament 125 - tournament 175) this individual was in the trough zone for a long time, fortunately, after that it was successfully genetically modified to return to the peak and finally became the highest-fitness individual.

2. Change Mutation Rate

The mutation rate is calculated by varying the mutation rate from 0 to 1, with an interval of 0.1. During this period, the number of tournaments, the number of neighbour, and the crossover rate are left at their default values of 500, 3, and 0.5, respectively.

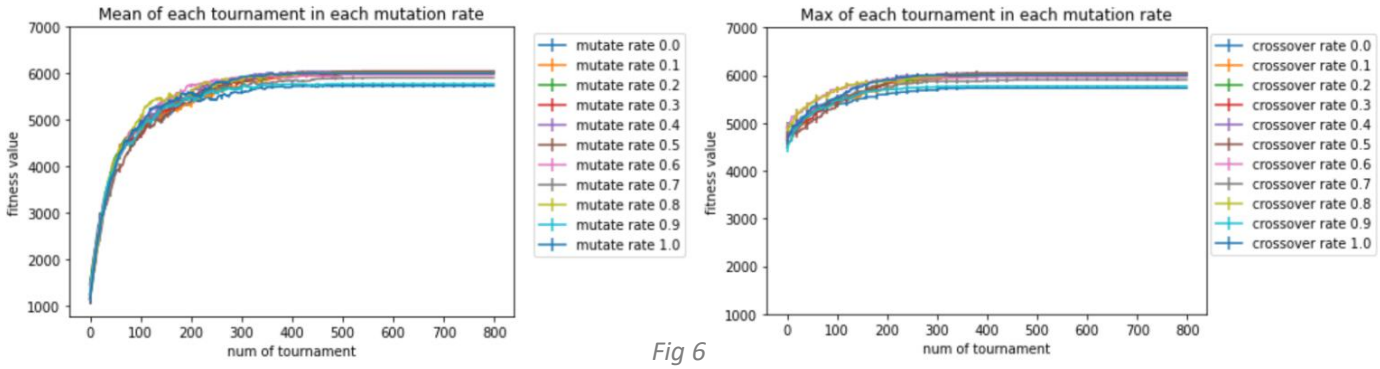

Fig 6

The graph on the left shows the trend of the mean values at different mutation rates. All the means start at a low modest

level (around 1000) and gradually climb to near the maximum (around 6000), with a large overall span. Error bars are used to indicate the spread.

The right graph is similar with the left, except all values start at a high fitness (around 5000).

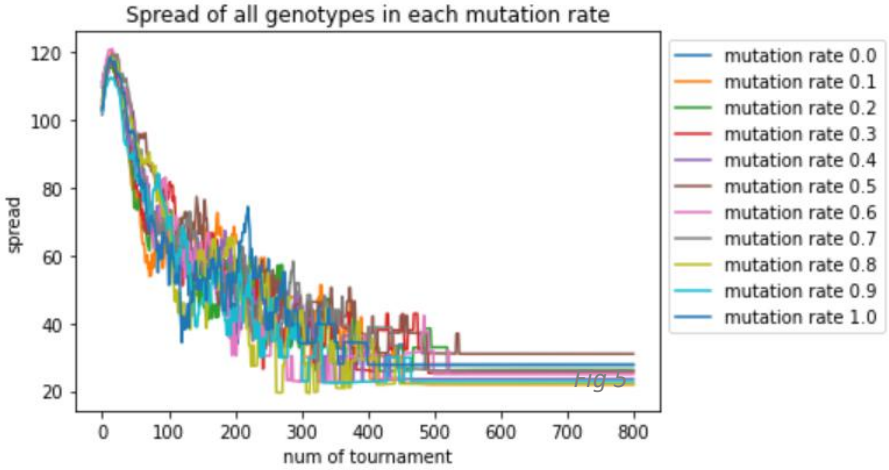

Fig 7

In *Fig 7* , After a short rise, the spread values become smaller as the number of tournaments increases. Although the lines start to fluctuate sharply after about the 100th tournaments, overall the spread drops by between 80 and 100 points.

3. Change Crossover Rate

The crossover rate is calculated by varying the crossover rate from 0 to 1 at intervals of 0.1. During this period, the number of tournaments, the number of neighbour and the mutation rate are kept at their default values of 500, 3 and 0.1 respectively.

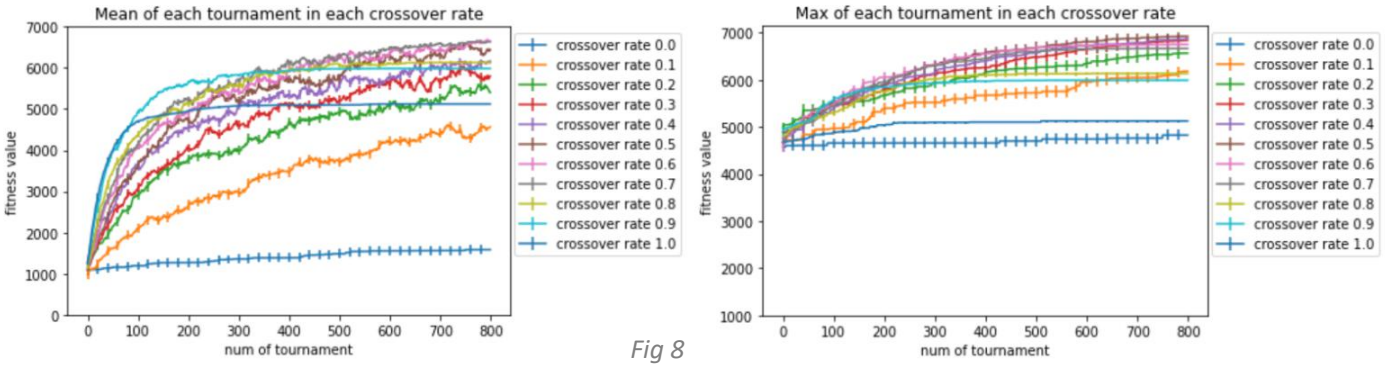

Fig 8

Left graph and right graph shows both have a similar trend. They all have diverge as they climb. Some have a larger gradient, which also means a higher final score, while others have a slower gradient, resulting in a very low score. The results peaked at rate = 0.7 and then fell again. Error bar uses for spread.

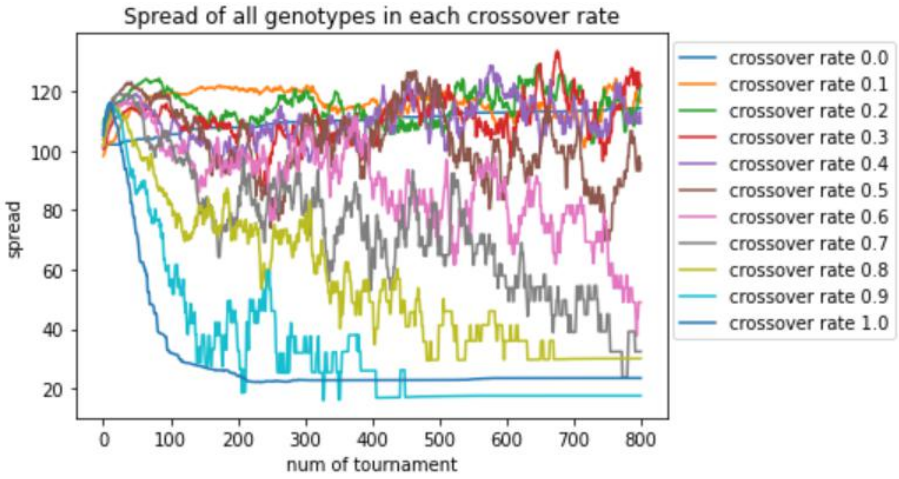

Fig 9

Fig 9 shows the trend in spread values for the above results. All results show a decreasing trend, but for small rates (e.g. rate = 0.3) the decrease in spread is negligible, while for larger rates (e.g. rate = 0.9) the decrease in spread is significant. In general, the value of spread decreases as the crossover rate increases.

4. Change Number of Neighbour

The neighbour number changes from 2 to 10 (max), not starts from 1 because one cannot have a neighbour for itself.

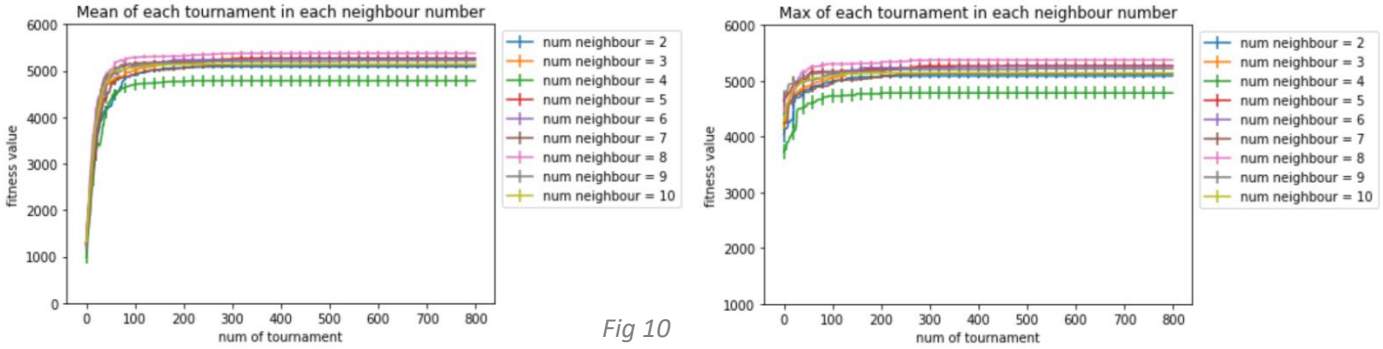

Fig 10

The results of *Fig 10* are very similar to *Fig 6*, except that *Fig 10* ends up with a smaller fitness value, and both graphs rise very smoothly and stop at a certain value (possibly the local minima).

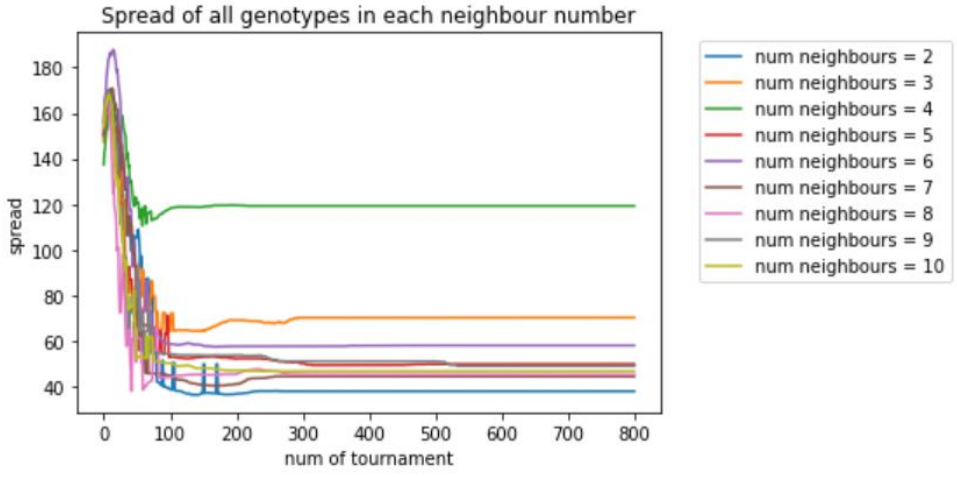

Fig 11

Fig 11 shows that the spread reaches a minimum at neighbour number equal to 2, but rises abruptly at 3 and 4, returns to a low at 5 and keeps small from that.

Discussion

In *Fig 1* and *Fig 5*, we can see more tournaments lead to more chances of variation. When an individual is at a low fitness value, it will have a high chance of falling and then improving. As it improves more often, the opponents who beat it are those with high fitness values, which makes each improvement very efficient. Thus, by doing this in huge numbers of tournament, it is easier to create a genotype with a very high fitness in the end.

In the *Fig 6* it can be found that for each max value in each tournament that was extracted in Fig4, they all start at a higher value than the mean in Fig3. This illustrates the random nature of the data at the beginning. In addition, what both graphs have in common is that the rising curves are relatively smooth and do not have many ripples.

The spread in errorbar represents the accuracy of the data to some extent. A more intuitive way to see this is to look at *Fig 7*, where spread is listed separately, and it is clear that the data starts to become more accurate after about 20 tournaments. The final spread values are basically in the range of 20 to 40, which gives the data a degree of reliability and accuracy.

A similar experiment was implemented in “Change Crossover Rate”. The only similarity to the “Change Mutation Rate” is that the max value in *Fig 8* has a relatively high starting point compared to the mean value in the same figure. It is also worth noting that the growth rates of the individuals in *Fig 8* are different. Their growth rates increase as the crossover rate increases, and then gradually decrease after about when the crossover rate = 0.7.

Similarly, the change in the errorbar shows that the final spread decreases as the crossover rate increases, which means that the data can reach a high level of precision at high values of the crossover rate. (*Fig 8* and *Fig 9*)

The combination of *Fig 10* and *Fig 11* shows that the precision of the data increases with the number of neighbours in the overall trend. However, in the front part of the increasing number, there is a possibility of causing a decrease in precision.

For the points in [4], mutation rates is considered reasonable as the best data of mutation rate = 0.2 is relatively close to 0.1 that was given, and the growth curve is very smooth. And for crossover rates, the best performance found within this study tended to be when the crossover rate = 0.7, has a bit more differences with that in [4], which crossover rate = 0.9. Although 0.9 did not score highly in the final tournament, it was excellent in terms of the accuracy of the data. This also explains the reason for the choice of 0.9 in [4] in another way - the degree of error or dispersion of the data. In addition, due to the limited nature of the experiment, it is not possible to exclude changes in the data that would be introduced after more tournaments were conducted

Conclusion

To conclude, this experiment is a study of the Full Microbial Genetic Algorithm based on the knapsack problem. The experiments were carried out to quantify the different hyper-parameters (number of tournaments, mutation rate, crossover rate) to see if they were as stated in [4].

Comparisons were made for different values between the parameters. The result of the study was that:

- Genotypes are given more chances to change with an increasing number of tournaments and are thus more likely to score high.
- A higher mutation rate leads to more opportunities for change, while a lower crossover rate makes genotypes less robust.

- The number of neighbours affects the consistent progressiveness of genotypes, and a larger number of neighbours leads to a greater chance of picking better genotypes.

The results obtained are that the number of tournaments will have a good effect when reaching 500. For the mutation rate the results are fine, as the best results are similar to the citation, being close to 0.1. While for the crossover rate, although there is no consensus on the best genotype at the moment, the high accuracy rate and the inevitable one-sidedness of the experiment make the answer of 0.9 acceptable.

Reference

1. Jooken, J., Leyman, P., & De Causmaecker, P. (2022). *A new class of hard problem instances for the 0–1 knapsack problem. European Journal of Operational Research*, 301(3), 841–. https://doi.org/10.1016/j.ejor.2021.12.009
2. Google Developer (2022) *The Knapsack Problem* https://developers.google.com/optimization/bin/knapsack
3. Lugo-González, E. (2012). *Performance of Simple Genetic Algorithm Inserting Forced Inheritance Mechanism and Parameters Relaxation*. sine loco: IntechOpen. https://torl.biblioboard.com/content/80232c50-9df6-4d7e-ba29-fda52925fef4
4. Man, K. F. (Kim F. ., Tang, K. S., & Kwong, S. (1999). *Genetic algorithms : concepts and designs*. London, [England: Springer. https://ebookcentral.proquest.com/lib/suss/detail.action?docID=3073392
5. Mahato, M., Gedam, S., Joglekar, J., & Buddhiraju, K. M. (2019). Dense Stereo Matching Based on Multiobjective Fitness Function-A Genetic Algorithm Optimization Approach for Stereo Correspondence. IEEE Transactions on Geoscience and Remote Sensing, 57(6), 3341–3353. https://doi.org/10.1109/TGRS.2018.2883483
6. Garg, P. K., & Mohanty, D. (2012). Mean (Standard Deviation) or Mean (Standard Error of Mean): Time to Ponder. World Journal of Surgery, 37(4), 932–932. https://doi.org/10.1007/s00268-012-1854-z