Multiple Constrained Knapsack Problem.

Student: Cathaoir Agnew *ID: 16171659*
Student: David Cherry *ID: 13056212*

Introduction

This paper details an investigation into the Multiple Constraints Knapsack Problem, also known as the bounded knapsack problem, using Genetic Algorithms (G.A.) implemented through Distributed Evolutionary Algorithms in Python (DEAP). The knapsack problem is a typical problem in combinatorial optimization where, when given a set of objects each with a weight and value, it is determined what the optimal combination of objects to pick are in order to maximize value whilst not exceeding a set weight limit. This problem is synonymous with resource allocation issues that can arise in many facets of life. Furthermore, in this project, the availability of each object is doubled, meaning that each object can be taken twice, once or not at all. This is the Multiple Constraints issue that is tackled in the investigation. Throughout this investigation two types of selection methods were chosen to examine, Roulette and Tournament Selection. Additionally, the crossover and mutation probabilities are examined and altered for both selection types in order to observe the diversity of the population and ultimately maximise the value output.

Details of the Problem

The Knapsack problem, as previously mentioned, is a generalised version of issues present in many facets of life. For example, if a hiker is going to go on a long hike, they will need to be able to carry everything that they bring with them. This sets a limit on the total weight of the items this hiker can take. The value of the items in this scenario could be regarded as the usefulness or necessity of the items to bring, such as food and water, shelter or various tools. These items would have a much higher value for a hiker to bring than items such as a television, an item of relatively heavy weight and little usefulness for a hike. With this in mind the hiker needs to pack their bag balancing the total value of each item against the limited weight capacity in order to optimise and prioritise which items to pack.

For this assignment, the standard 0-1 knapsack problem was taken as the beginning point, utilizing binary strings to represent the potential solutions, where a 1 represents an included item in the knapsack and a 0 represented an item not to be included. Each position on this string is initialized with a randomly assigned value between 1 and 10, as well as a randomly assigned weight between 1 and 100. The length of the binary string, denoted as 'n', is the amount of items available for selection. To accommodate the multiple constraints portion of this assignment, each item needed to be available for selection twice. To represent this, the string initialization was altered to be able to choose 0,1 and 2. The representation of the “0” and “1” remain unchanged, an item included or omitted respectively. The “2” represents an item selected twice.

Other important details considered were the variables used during the testing phase of this investigation, the fitness function utilized, and the number of evaluations computed for each run. The Crossover probability (Cx) and Mutation probability (Mx) where considered for 3 separate runs of each selection type, a low value (Cx=0.1, Mx=0.001), a mid value (Cx=0.7, Mx=0.01) and a high value (Cx=0.9, Mx=0.3) in order to observe various results. The fitness function utilized can be seen below. Finally, the number of evaluations which are represented by the multiple runs (30), generations (75) and population sizes (500), resulting in 1.125 million evaluations per test, were kept constant across all tests to ensure consistency for each test, of which there are six total.

Fitness Function:
```
def evalKnapsack2(individual):
        weight = 0.0
        value = 0.0
        for i in range(NBR_ITEMS):
                value += items[i][0]*individual[i]
                weight += items[i][1]*individual[i]

        if  weight > MAX_WEIGHT:
                return (value-(weight-MAX_WEIGHT)*2),weight
        else:
                return (value), weight
```

Experiment 1: Roulette Wheel Selection & Diversity.

Using the Roulette Wheel selection method and varying the probability and crossover parameters using the following values displayed in the table below, three separate results were achieved, one for each run. As previously mentioned, each run had a population size of five hundred and was allowed to run for seventy-five generations. In addition to this, each run had multirun capabilities enabled and was run thirty times, in order to ensure a statistically significant result. With these limits set, each run had to perform 1.125 million evaluations. Convergence was not achieved in all runs with these limits set.

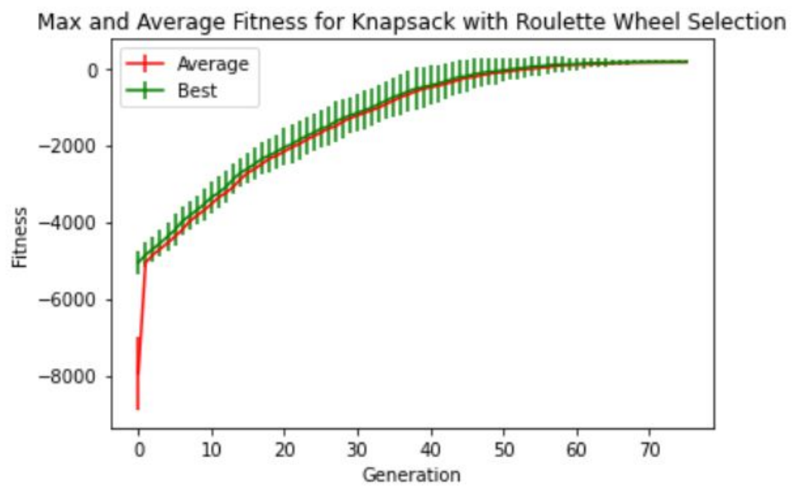
Roulette Wheel Selection Variables

Run Number	1	2	3
Population Size	500	500	500
Crossover Probability	0.7	0.1	0.9
Mutation Probability	0.01	0.001	0.3
Number of Generation	75	75	75
Number of Runs	30	30	30

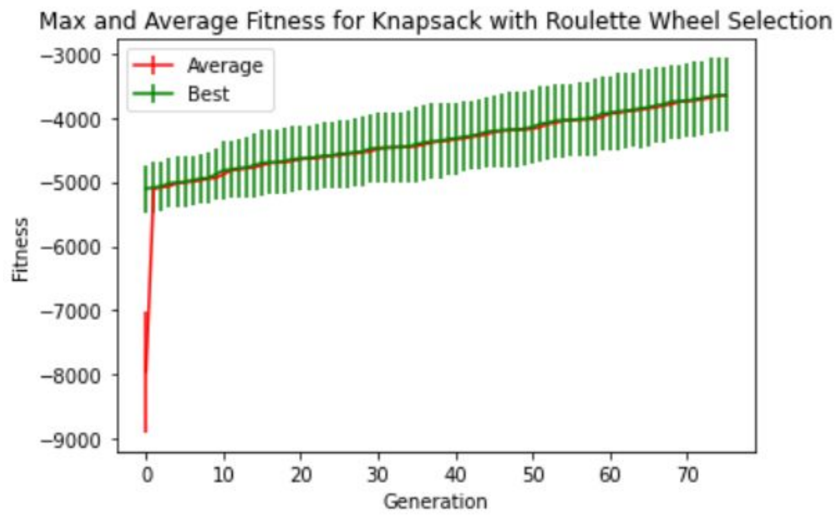
Best Individuals From Each Run

Value	226	-2086	288
Weight	987	2202	1000

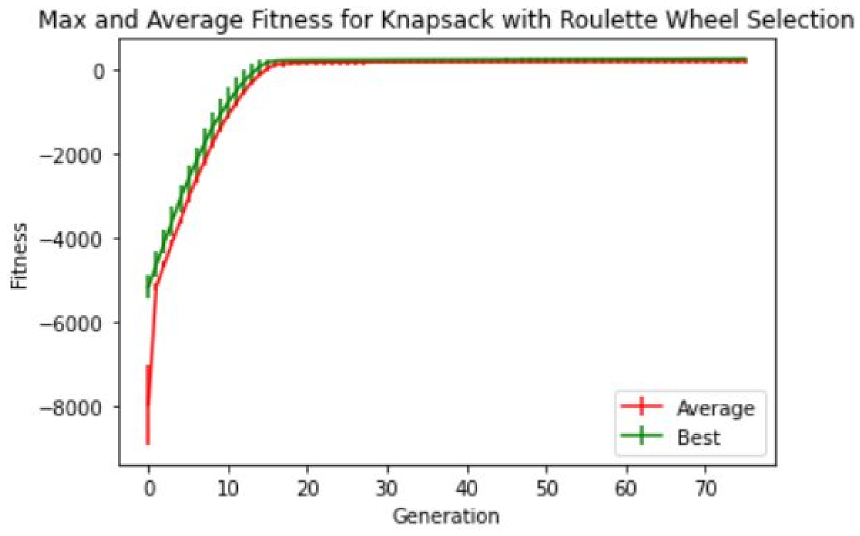
Run 1 Graph



Run 2 Graph



Run 3 Graph



Experiment 2: Tournament Selection & Diversity.

For the second set of experimental runs, tournament selection was used. The values previously used in the roulette wheel selection for crossover and mutation probabilities for each respective run were also maintained, along with the total evaluations number derived from population, generations and multi-runs. This was done in order for a reasonable comparison of the results to be drawn with respect to tournament and roulette wheel selection for this multi-constraints knapsack problem.

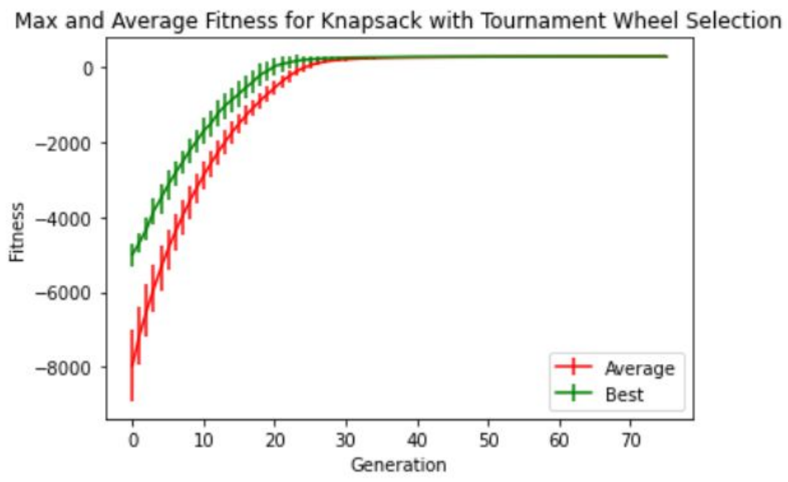
Tournament Selection Variables

Run Number	**1**	**2**	**3**
Population Size	**500**	**500**	**500**
Crossover Probability	**0.7**	**0.1**	**0.9**
Mutation Probability	**0.01**	**0.001**	**0.3**
Number of Generation	**75**	**75**	**75**
Number of Runs	**30**	**30**	**30**

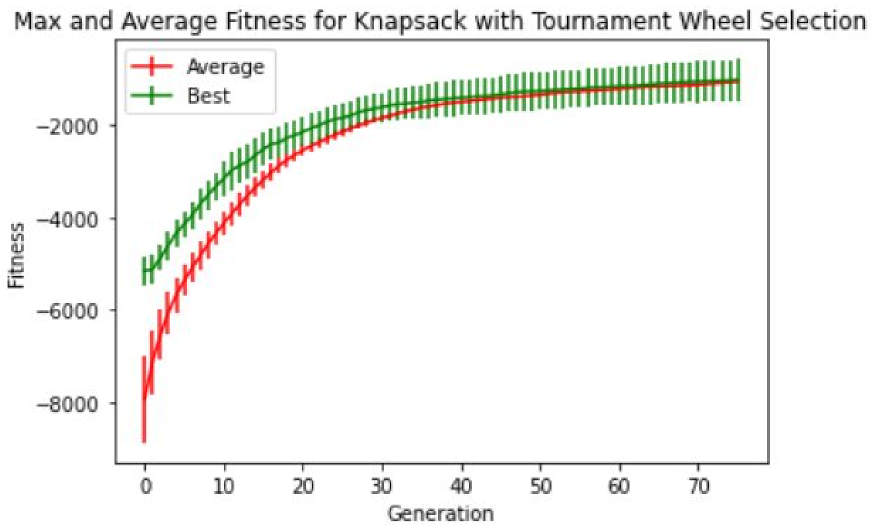
Best Individuals From Each Run

Value	**320**	**102**	**356**
Weight	**1000**	**1047**	**986**

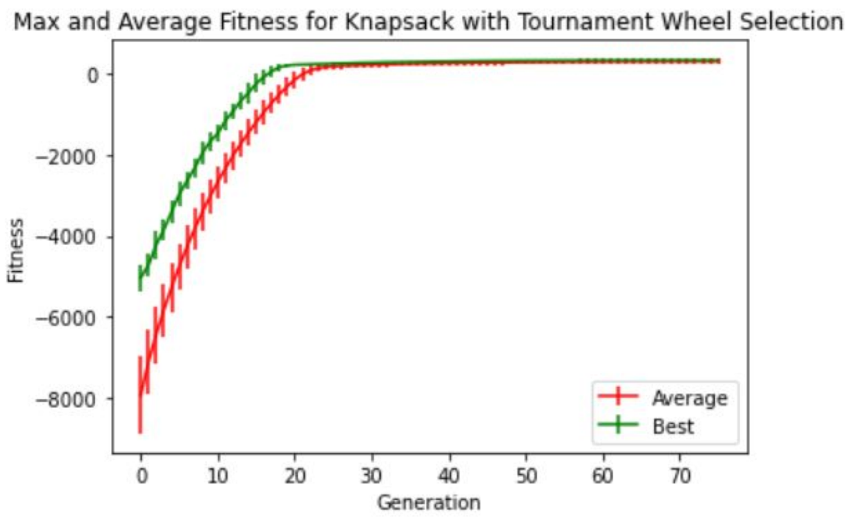
Run 1 Graph



Run 2 Graph



Run 3 Graph



Results

It can be observed from the experiments that the tournament selection process worked better than the roulette wheel selection for this application of the multiple constraints knapsack problem, also referred to as a bounded knapsack problem. This is most likely due to the fitness bias that tournament selection presents inherently. Tournament selection means that individuals are randomly selected for comparison, then the individual with the higher fitness level progresses to the next generation. As a result of this, the least fit individual in any given generation has no chance to progress as any other individual it is compared to will have a higher fitness. Roulette wheel selection on the other hand gives the least fit individual a chance, albeit a relatively small chance, to progress. For a relatively simple problem such as the knapsack problem, tournament selection excels in performance compared to roulette wheel.

Next looking at the probabilities for crossover and mutations, it can be observed from the results that the higher probabilities for both allowed for a better mix of diversity in the population in both roulette wheel and tournament selection. This in turn allowed for individuals both more and less fit to appear within the population, in other words a greater diversity. However, as each generation was evaluated, it can be observed from the graphs that the overall average fitness improved. This is due to the fitter individuals of a given generation being able to progress to the next generation and the weaker individuals "dying out".

It should be noted for run 2 in both tournament and roulette wheel selection, convergence was never reached within the limit of 75 generations. In fact, both selection methods returned a best individual with an overweight bag. This was directly due to the low crossover and mutation probabilities. The population diversity changed much more slowly compared to the other runs, meaning that improvements would appear less frequently. Had these runs been allowed to continue until convergence by increasing the generations limit, a comparably good result may have been achieved, but the computation time and evaluations required would have made this impractical to pursue.

Taking all the previously mentioned points into account, as well as the recorded results from the experiments, it can be observed that the best result, with a value of 356 and weight under the 1000 limit at 986 was accomplished. This was achieved by the tournament selection used in experiment 2, utilizing high mutation and crossover probabilities as demonstrated in run 3.

Conclusion

Tournament selection performed remarkably better due to its inherent fitness bias previously mentioned. Should the data being used in the population have had more variables than simply value, weight and quantity, this would have made the problem far more complex and may have led to the roulette wheel selection method performing better. The No Free Lunch Theorem applies here.

It can also be seen that this experiment benefited greatly from higher diversity within each generation, generated from high crossover and mutation probabilities, as demonstrated in run 3 for both selection methods. The opposite can also be demonstrated, that low diversity caused the experiment to improve much more slowly. It should be noted however that even with low diversity, the generation average fitness was still improving with each iteration, particularly the roulette wheel selection method.

In conclusion our experiments tell us that tournament selection with tournament size = 3 performs better than roulette wheel selection in the multiple constraint knapsack problem.