# 1) Genetic Algorithm

## Introduction:

The Genetic Algorithm (GA) is a powerful optimization technique inspired by the principles of natural selection and genetics. It iteratively evolves a population of candidate solutions toward an optimal solution by mimicking the process of natural selection, crossover, and mutation.

## Problem Definition:

The Knapsack Problem is a classic combinatorial optimization problem that asks the following question: "Given a set of items, each with a weight and a value, determine the maximum value that can be obtained by selecting a subset of the items such that the total weight does not exceed a given limit."

## Hardware Setup:

Hardware: AMD Athlon Gold 3150U CPU, 8GB RAM.

Software: Java.

Environment: Window 11.

## Algorithm Configuration:

Initial Population Generation: Randomly select items to place in the array from the list of available items. The array represents the knapsack. Always ensure that the total weight of the items don't exceed the give knapsack limit. Hence after inserting an item ensure to check if the weight is not exceeded before considering adding another item.

The Knapsack limits are given. The population depends on the number of items to choose from. If there less than 6 items, then population size is 10, if there between 6-9(inclusive) items then population size is 13, if the population is between 11-20(inclusive) the population is 25, else the population size is 35.

The fitness of each is solution is calculated based total value of items in the knapsack. Of course, the solution must be within the given knapsack limit. The high the value, the fitter the solution. The parents that will participate in reproduction will be chosen via tournament selection. The tournament size changes according to the number of items to choose from. If there less than 6 items, then tournament size is 2, if is between 6-15 (inclusive) the tournament size if 3, if it between 16-30(inclusive) then tournament size is 4 else the size 20% of the number of items.

A multipoint cross over is used. Where Cross over is at a gene level. Meaning genes are swapped between parents. The swapping of genes depends on the two parents, the swap depends on different genes between the parents. This is to ensure to prevent duplication of genes in the offspring. Mutation randomly swaps out an item from the knapsack with another item ensuring that the knapsack limit is not violated.

Steady Generational Replacement is used. Where the parents are swapped out with the offspring.

Stopping Criterion: Maximum number of generations which was set 200.

### Experimental Procedure:

```
1: Create initial population
2: Calculate fitness of all individuals
3: while termination condition not met do
4:      Select fitter individuals for reproduction
5:      Recombine to produce offspring
6:      Mutate the offspring
7:      Generate a new population
8: end while
9: return best individual
```

## 2) Genetic Algorithm + Local Search(Memetic Algorithm)

### Introduction:

The Genetic Algorithm (GA) with Local Search is a powerful optimization technique inspired by the principles of natural selection and genetics. It iteratively evolves a population of candidate solutions toward an optimal solution by mimicking the process of natural selection, crossover, and mutation. Additionally uses a local search ensure the best solution is found. The local search be applied any  after crossover, mutation or selection.

### Problem Definition:

The Knapsack Problem is a classic combinatorial optimization problem that asks the following question: "Given a set of items, each with a weight and a value, determine the maximum value that can be obtained by selecting a subset of the items such that the total weight does not exceed a given limit."

### Hardware Setup:

Hardware: AMD Athlon Gold 3150U CPU, 8GB RAM.

Software: Java.

Environment: Window 11.

### Algorithm Configuration:

Initial Population Generation: Randomly select items to place in the array from the list of available items. The array represents the knapsack. Always ensure that the total weight of the items don’t exceed the give knapsack limit. Hence after inserting an item ensure to check if the weight is not exceeded before considering adding another item.

The Knapsack limits are given.  The population depends on the number of items to choose from. If there less than 6 items, then population size is 10, if there between 6-9(inclusive) items then population size is 13, if the population is between 11-20(inclusive) the population is 25, else the population size is 35.

The fitness of each is solution is calculated based total value of items in the knapsack. Of course, the solution must be within the given knapsack limit. The high the value, the fitter the solution. The parents that will participate in reproduction will choose via tournament selection. The tournament size changes according to the number of items to choose from. If there less than 6 items, then tournament size is 2, if is between 6-15 (inclusive) the tournament size if 3, if it between 16-30(inclusive) then tournament size is 4 else the size 20% of the number of items.

A multipoint cross over is used. Where Cross over is at a gene level. Meaning genes are swapped between parents. The swapping of genes depends on the two parents, the swap depends on different genes between the parents. This is to ensure to prevent duplication of genes in the offspring.  Mutation randomly swaps out an item from the knapsack with another item ensuring that the knapsack limit is not violated.

A Tabu Search then applied after the mutations step to possible obtain better offspring. The reason for choosing Tabu Search ensure that we escape local optima in the processing of finding the better offspring.

Steady Generational Replacement is used. Where the parents are swapped out with the offspring.

Stopping Criterion: Maximum number of generations which was set 200.

## Experimental Procedure:

```
1: Create initial population
2: Calculate fitness of all individuals
3: while termination condition not met do
4:        Select fitter individuals for reproduction
5:        Recombine to produce offspring
6:        Mutate the offspring
7:        Apply Tabu on the offspring’s
8:        Generate a new population
9: end while
10: return best individual
```

## 3) Reason Why Tabu Search is used

Tabu is very explorative as it randomly chooses a neighboring solution.

It keeps a list of previous explored solution and as result will no re-explored visited solution.

In the case of knapsack problem to search neighboring solution, we substitute items in the knapsack with item out the knapsack. This substitution process is random so it's possible to swap back to a previous solution (already visited). So, the Tabu Search remembers this act according.

The list in the Tabu Search hold the best-found solution, which can be found by iterating through the list.

## 4) Parameter Set Up

### GA:

Number Of Generation is 200: A couple of generations were tried, e.g., 20, 50, 100 and 120, 170, 200 generations. Even though it possible to get best possible solution (in the entire search space) using 20 and 50 but on number of simulations, GA did not return best possible solution (not close to optima). Hence the number of generations was increase to 200. At this point there is a high chance it will return the best possible solution even for larger files.

The length is chromosome is set number of items available. However, the not all genes are used. A gene is only active if adding an item to gene will not violate the knapsack limit. Hence solutions look like they have different length because they fit different number items.

Population Size vary according to the number items available. If there less than 6, then population size is 10, if there between 6-9(inclusive) items then population size is 13, if the population is between 11-20(inclusive) the population is 25, else the population size is 35. The reason is that if the number of items available are small and the population is large then the population will contain duplicates. Again, if the number of items is many and population is small then population generate will evenly spread across the search space. Hence there is need to be balance.

Tournament size also varies according to the number of item available. According to paragraph above when there is large number of available items then population size should be large and vice versa. Hence Tournament selection try reducing convergence by randomly choose evenly across the population.

### GA Local Search:

Number Of Generation is 200: A couple of generations were tried, e.g., 20, 50, 100 and 120, 150, 180, 200 generations. Even though it possible to get best possible solution (in the entire search space) using 20 and 50 but on number of simulations, GA Local Search did not return best possible solution (not close to optima). Hence the number of generations was increase to 200. At this point there is a high chance it will return the best possible solution even for larger files.

The length is chromosome is set number of items available. However, the not all genes are used. A gene is only active if adding an item to gene will not violate the knapsack limit. Hence solutions look like they have different length because they fit different number items.

Population Size vary according to the number items available. If there less than 6, then population size is 10, if there between 6-9(inclusive) items then population size is 13, if the population is between 11-20(inclusive) the population is 25, else the population size is 35. The reason is that if the number of items available are small and the population is large then the population will contain duplicates.  Again, if the number of items is many and population is small then population generate will evenly spread across the search space. Hence there is need to be balance.

Tournament size also varies according to the number of item available. According to the paragraph above when there is large number of available items then population size should be large and vice versa. Hence Tournament selection try reducing convergence by randomly choose evenly across the population.

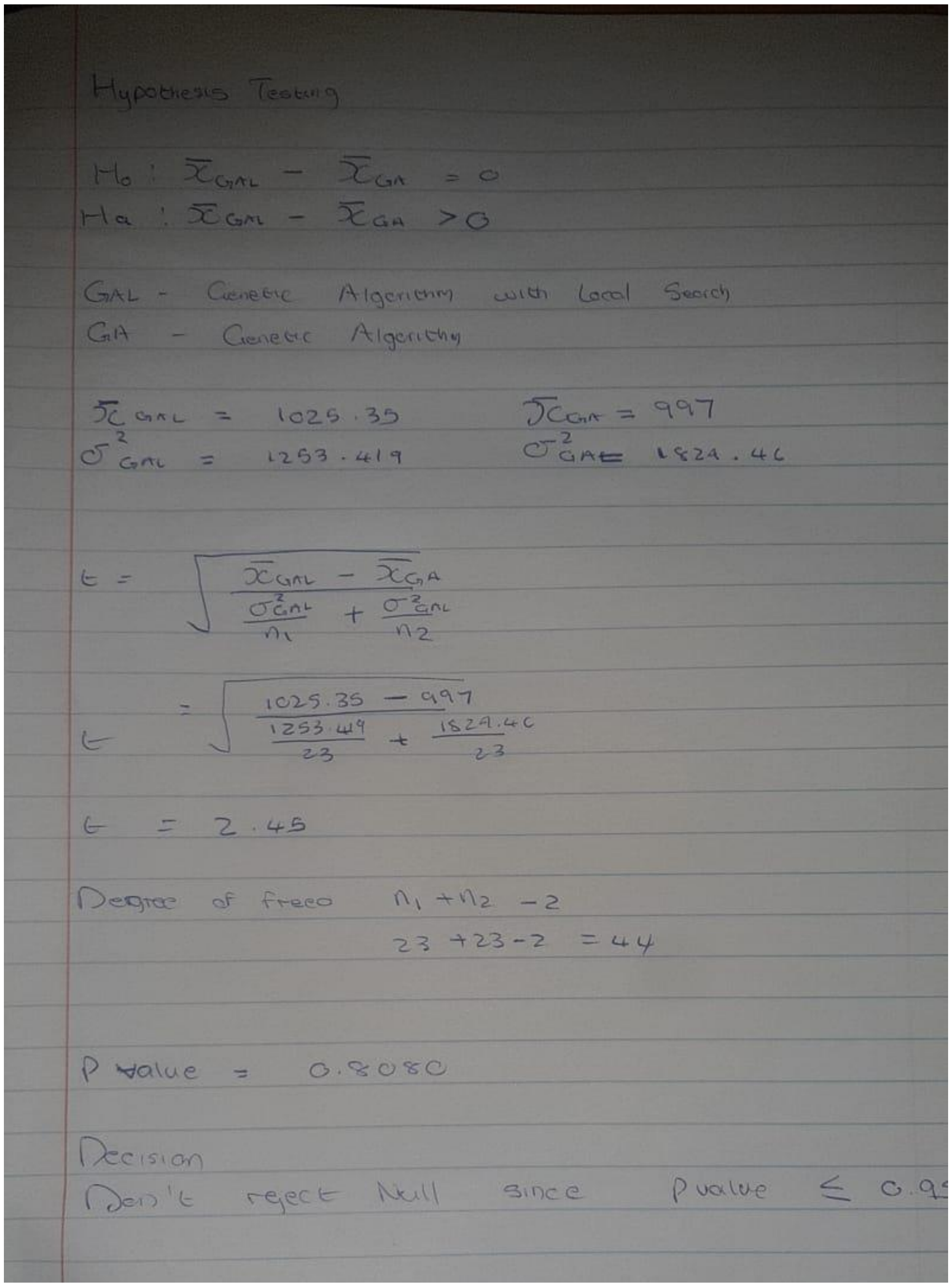
## 5) Table and Analysis

| Problem Instance | Algorithm | Seed Value | Best Solution | Known Optimum | Runtime(second) |
|---|---|---|---|---|---|
| f1−ld−kp−10−269 | GA-LS | 1714226074350 | 61.0 - 62.0<br>8.0 - 80.0<br>5.0 - 32.0<br>10.0 - 4.0<br>55.0 - 95.0<br>50.0 - 72.0<br>47.0 - 60.0<br>8.0 - 80.0<br>4.0 - 23.0<br>**Total Value 508.0**<br>**Total Weight 248.0**<br>(Weight-Value) | 295 | 0.3095915 |
|  | GA | 1714226074350 | 8.0 - 80.0<br>10.0 - 4.0<br>55.0 - 95.0<br>61.0 - 62.0<br>47.0 - 60.0<br>50.0 - 72.0<br>5.0 - 32.0<br>4.0 - 23.0<br>4.0 - 23.0<br>**Total Value 451.0**<br>**Total Weight 244.0** | 295 | 0.23154579 |
| f2−l−d−kp−20−878 | GA-LS | 1714226546977 | 77.0 - 56.0<br>63.0 - 58.0<br>17.0 - 96.0<br>75.0 - 92.0<br>40.0 - 18.0<br>8.0 - 6.0<br>35.0 - 82.0<br>91.0 - 84.0<br>29.0 - 48.0<br>44.0 - 92.0<br>75.0 - 70.0<br>72.0 - 83.0<br>61.0 - 25.0<br>15.0 - 83.0<br>78.0 - 32.0<br>46.0 - 4.0<br>40.0 - 68.0<br>**Total Value 997.0**<br>**Total Weight 866.0** | 1024 | 0.2976005 |
|  | GA | 1714226546977 | 75.0 - 92.0<br>77.0 - 56.0<br>54.0 - 44.0<br>35.0 - 82.0<br>8.0 - 6.0<br>91.0 - 84.0<br>15.0 - 83.0<br>61.0 - 25.0<br>17.0 - 96.0<br>63.0 - 58.0<br>46.0 - 4.0<br>44.0 - 92.0<br>40.0 - 18.0<br>40.0 - 68.0<br>29.0 - 48.0<br>72.0 - 83.0<br>78.0 - 32.0<br>**Total Value 971.0**<br>**Total Weight 845.0** | 1024 | 0.2240745 |
| Instance X<br>f8_l- | GA-LS | 1714226811531 | 977.0 - 979.0<br>970.0 - 972.0<br>980.0 - 982.0 | 9767 | 0.3486164 |

| | | | | | |
|---|---|---|---|---|---|
| d_kp_23_10000 | | | 981.0 - 983.0<br>484.0 - 485.0<br>487.0 - 488.0<br>978.0 - 980.0<br>979.0 - 981.0<br>482.0 - 483.0<br>976.0 - 978.0<br>485.0 - 486.0<br>980.0 - 982.0<br>**Total Value 9779.0**<br>**Total Weight 9759.0** | | |
| | GA | 1714226811531 | 962.0 - 964.0<br>976.0 - 969.0<br>970.0 - 972.0<br>976.0 - 978.0<br>484.0 - 485.0<br>980.0 - 982.0<br>974.0 - 966.0<br>485.0 - 486.0<br>979.0 - 981.0<br>857.0 - 959.0<br>978.0 - 980.0<br>**Total Value 9722.0**<br>**Total Weight 9621.0** | 9767 | 0.1527183 |
| f10−l−d−kp−20−879 | GA-LS | 1714227094394 | 46.0 - 4.0<br>44.0 - 92.0<br>55.0 - 44.0<br>17.0 - 96.0<br>35.0 - 82.0<br>63.0 - 58.0<br>75.0 - 92.0<br>40.0 - 68.0<br>75.0 - 70.0<br>8.0 - 6.0<br>29.0 - 48.0<br>40.0 - 18.0<br>91.0 - 84.0<br>15.0 - 83.0<br>77.0 - 56.0<br>72.0 - 83.0<br>63.0 - 58.0<br>**Total Value 1042.0**<br>**Total Weight 845.0** | 1025 | 0.26856452 |
| | GA | 1714227094394 | 29.0 - 48.0<br>55.0 - 44.0<br>72.0 - 83.0<br>75.0 - 70.0<br>75.0 - 92.0<br>63.0 - 58.0<br>35.0 - 82.0<br>15.0 - 83.0<br>75.0 - 14.0<br>17.0 - 96.0<br>77.0 - 56.0<br>40.0 - 68.0<br>61.0 - 25.0<br>46.0 - 4.0<br>44.0 - 92.0<br>8.0 - 6.0<br>15.0 - 83.0<br>77.0 - 56.0<br>**Total Value 1060.0**<br>**Total Weight 879.0** | 1025 | 0.16111101 |

6) Statistical analysis of differences in performance needs to be presented. Specifically, a one-tailed z-test (5% level) is used. The null hypothesis is that the means are equivalent.

Both algorithms' results were based on the "f8_l-d_kp_23_10000". Both algorithms were ran 23 times. This created a sample base to calculate the mean and variance need for the test. Sample values were stored in GA.txt and GALocal.txt The calculation was made in the Calculation.xlsx



Hypothesis Testing

$H_0 : \bar{x}_{GAL} - \bar{x}_{GA} = 0$

$H_a : \bar{x}_{GAL} - \bar{x}_{GA} > 0$

GAL - Genetic Algorithm with Local Search

GA - Genetic Algorithm

$\bar{x}_{GAL} = 1025.35$ $\qquad \bar{x}_{GA} = 997$

$\sigma^2_{GAL} = 1253.419$ $\qquad \sigma^2_{GA} = 1824.46$

$$t = \sqrt{\frac{\bar{x}_{GAL} - \bar{x}_{GA}}{\frac{\sigma^2_{GAL}}{n_1} + \frac{\sigma^2_{GAL}}{n_2}}}$$

$$t = \sqrt{\frac{1025.35 - 997}{\frac{1253.49}{23} + \frac{1824.46}{23}}}$$

$t = 2.45$

Degree of freeo $n_1 + n_2 - 2$

$23 + 23 - 2 = 44$

P value = 0.8080

Decision

Don't reject Null since P value $\leq$ 0.9

## 7) A critical analysis of the results

Considering both the result from the table and Hypothesis.

### Exploration (Escape local optima)

GA with Local Search is more explorative than GA. This mostly like due the Tabu Search applied the offspring. Tabu is explorative by nature as allows itself to choose neighboring solution which could worse or better. Addition both use crossover and mutation.

### Exploitation

Both algorithms using tournament selection, choose the parents for reproduction. Hence, they very equal in this area.

### Sensitivity to Parameters

Both algorithms use same set of rules to initialize the parameter. However, GA with Local Search is affected by chosen Local Search Algorithm and the frequency of the local Search. Possible if Local search are applied more time, the best solution can be found quicker. However, the additionally local search make run longer.

### Fitness of Solutions

Generally, GA with local search produces the best solution more times than the GA from table and always have a higher mean value from the Hypothesis Test.

### Hypothesis Test

The Decision is to not reject Null Hypothesis that Mean (GAL) is equal to Mean(GA). But it does not directly mean that GA average an GA Local Search average are equal. It just means there isn’t sufficient evidence to conclude that averages are not equal.

### Table Results

GA with Local Search produces the better result most of the time. The results  closer or better than optimum values.

**GA with local search outperforms GA.**