Assignment 3

# **Question 1:**

**Base Case:**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | 20 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | None |

**Graphical user interface

Description automatically generated with low confidence**

Min cost = 2594

**Case a: Changing the initial starting point (initial solution) 10 times**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | 20 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | None |
| \*\*Changing the initial starting point (initial solution) 10 times | |

**Text

Description automatically generated**

**A picture containing calendar

Description automatically generated**

**Analysis:**

By running multiple solutions, we have achieved the most optimal solution at *starting point 4 where cost is 2570.*

**Case bi: Changing the tabu list size smaller than the original (original length = 20, new length = 5)**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | 5 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | None |
| \*\*Changing the tabu list size smaller than the original (original length = 20, new length = 5) | |

**A screenshot of a computer

Description automatically generated with medium confidence**

Min cost = 2636

**Analysis:**

By making the tabu list size smaller, we have a more sub-optimal solution compared to the base case.

**Case bii: Changing the tabu list size larger than the original (original length: 20, new length = 50)**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | 50 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | None |
| \*\*Changing the tabu list size smaller than the original (original length = 20, new length = 50) | |

**A screenshot of a computer

Description automatically generated with medium confidence**

Min cost = 2650

**Analysis:**

By running with a larger tabu list size, we have achieved a less optimal solution than having a smaller tabu list size compared to the previous. This means that maybe we get stuck in a sub-optimal solution (local minima) when we try to find the optimal cost with a larger tabu list size.

**Case c: Changing the tabu list size dynamically**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | Dynamic (between 1-20) |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | None |
| \*\*Changing the tabu list size by producing a random number between 1 and 20 every 50 iterations | |

A picture containing text

Description automatically generated

Min cost = 2570

**Analysis:**

Changing the tabu list size diversifies the balance between exploration and exploitation. This actually resulted in us reaching the optimal solution.

**Case d: Aspiration Criteria – *best solution so far***

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | 20 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | Best solution so far |
| \*\*Aspiration Criteria – *best solution so far* | |

Text

Description automatically generated

Best solution so far = 2574

**Analysis:**

The aspiration criteria was equal to the minimum cost found overall, because the same aspiration criteria was used.

**Case e: Aspiration Criteria – *best solution in the neighbourhood***

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking the whole neighbourhood |
| Tabu list type | Recency-based |
| Tabu list size | 20 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | Best solution in the neighbourhood |
| \*\*Aspiration Criteria – *best solution in the neighbourhood* | |

Min cost = 2698

**Analysis:**

The aspiration criteria didn’t really help me to achieve an optimal solution.

Graphical user interface, text

Description automatically generated with medium confidence

A picture containing background pattern

Description automatically generated

**Background pattern

Description automatically generated**

**A screenshot of a computer

Description automatically generated with low confidence**

**Table

Description automatically generated with low confidence**

**A picture containing table

Description automatically generated**

**Text

Description automatically generated with medium confidence**

**Case f: Use less than the whole neighbourhood to select the next solution**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking less than the whole neighbourhood – don’t swap 10 neighbouring departments for each department |
| Tabu list type | Recency-based |
| Tabu list size | 20 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | Use less than the whole neighbourhood to select the next solution |
| \*\*Aspiration Criteria – *Use less than the whole neighbourhood to select the next solution* | |

Text

Description automatically generated

Min cost = 2644

**Analysis:**

Using less than the neighbourhood function allowed me to get a worse solution because sometimes the solutions that we weren’t considering could have been the optimal solution.

**Case g: Add a frequency based tabu list to encourage the search to diversify (count = 5)**

|  |  |
| --- | --- |
| Feature | Definition |
| Neighborhood and move operator | Single swaps from current location |
| Neighbourhood robustness | Checking less than the whole neighbourhood – don’t swap 10 neighbouring departments for each department |
| Tabu list type | Frequency based tabu list to encourage the search to diversify |
| Tabu list size | 20 |
| Stopping Criterion | 250 iterations |
| Aspiration Criteria | None |
| \*\*Frequency based tabu list to encourage the search to diversify | |

*A screenshot of a computer

Description automatically generated with medium confidence*

Min cost = 2636

**Analysis:**

Using a frequency-based list to diversify actually gave me suboptimal solutions compared to the base.

# **Question 2:**



Kp = (2, 18)

[0, 16] – range

[(16 – 0) \* 100] + 1 = 1601

**Representation in bits = 11 bits**

TI = (1.05, 9.42)

[0, 8.37] – range

[(8.37 – 0) \* 100] + 1 = 838

**Representation in bits = 10 bits**

TD = (0.26, 2.37)

[0, 2.11] – range

[(2.11 – 0) \* 100] + 1 = 212

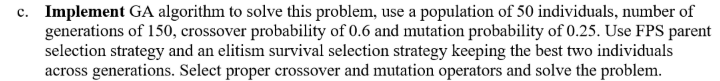
**Representation in bits = 8 bits**



For minimization of all the values, we want a fitness function that maximizes the reward to smaller values. We also want a function that considers all of the parameters.

Fitness function for GA that does maximization = 0.7(1/ISE) + 1/t\_r + 1/t\_s + 1/M\_p.

\*\*Key: Placed more weightage to the parameter ISE because it has more weight in this formula





Graphical user interface

Description automatically generated

Graphical user interface, text, application

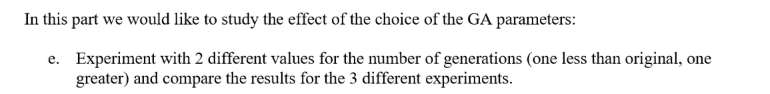
Description automatically generated

**K\_P =** 4.804

**T\_i =** 6.0424

**T\_d =** 2.36

**Fval** = 90.605



**Generations = 20**

**Chart, scatter chart

Description automatically generated**

A screenshot of a computer

Description automatically generated

**K\_P =** 4.8501

**T\_i =** 6.1401

**T\_d =** 2.3221

**Fval** = 90.838

**Generations = 300**

**Chart

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**K\_P =** 4.804

**T\_i =** 6.0424

**T\_d =** 2.36

**Fval** = 90.605

**Comparing Results:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Experiment 1 – Gen 150** | **Experiment 2** | **Experiment 3** |
| **K\_P** | 4.804 | 4.8501 | 4.804 |
| **T\_i** | 6.0424 | 6.1401 | 6.0424 |
| **T\_d** | 2.36 | 2.3221 | 2.36 |
| **Fval** | 90.6054 | 90.838 | 90.6054 |

Graphical user interface

Description automatically generated**Chart

Description automatically generatedChart, scatter chart

Description automatically generated**

Comparing the values and looking at the behaviour of the graphs , we see that the values have not converged yet at experiment 2. But the values have converged at generation before the max specified in experiment 1 and 3. This is why both the results from the experiments are the same.

****

**Population size = 200**

**Graphical user interface, chart

Description automatically generated**

**Chart

Description automatically generated**

**K\_P =** 4.8031

**T\_i =** 6.056

**T\_d =** 2.36

**Fval =** 90.605

**Population size = 10**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**K\_P =** 4.8173

**T\_i =** 7.5168

**T\_d =** 2.36

**Fval =** 90.937

**Comparing Results:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Experiment 1 – Pop = 50** | **Experiment 2 – Pop = 200** | **Experiment 3 – Pop = 10** |
| **K\_P** | 4.804 | 4.8031 | 4.8173 |
| **T\_i** | 6.0424 | 6.056 | 7.5168 |
| **T\_d** | 2.36 | 2.36 | 2.36 |
| **Fval** | 90.654 | 90.6054 | 90.9371 |

Graphical user interface

Description automatically generated**Graphical user interface

Description automatically generatedGraphical user interface, chart

Description automatically generated**

As I was running the simulation, experiment 3 ran the fastest because it had the least population size but experiment 2 took the longest. As well, we can see that we achieved optimal values faster with the greater population size, as we see convergence was reached much more early in experiment 2 than 1. This means that in experiment 1, we were still exploring lots of other population spaces in each generation and sometimes getting sub-optimal solution.

The conclusion is that, if there is time and enough computational capacity, it is better to use a greater population size to achieve convergence faster.

****

**Crossover Probability = 0.1**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

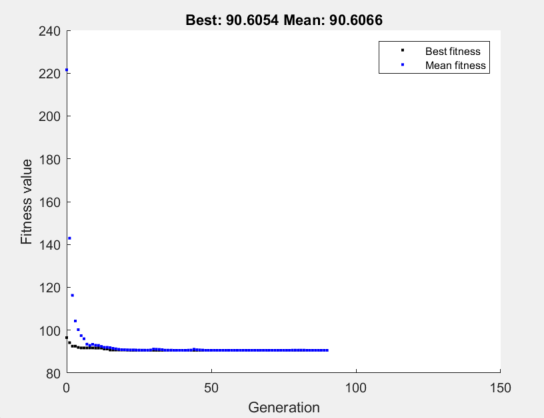
**K\_P =** 3.5619

**T\_i =** 9.3332

**T\_d =** 2.1071

**Fval =** 78.506

**Crossover Probability = 0.9**

****

Graphical user interface, text, application

Description automatically generated

**K\_P =** 4.8034

**T\_i =** 6.0509

**T\_d =** 2.36

**Fval =** 90.605

**Comparing Results:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Experiment 1 – crossover P = 0.6** | **Experiment 2 – crossover P = 0.1** | **Experiment 3 – crossover P = 0.9** |
| **K\_P** | 4.804 | 3.5619 | 4.8034 |
| **T\_i** | 6.0424 | 9.3332 | 6.0509 |
| **T\_d** | 2.36 | 2.1071 | 2.36 |
| **Fval** | 90.654 | 78.506 | 90.605 |

Graphical user interface

Description automatically generated

**Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated**

According to this, we can see that decreasing the cross over probability invites more randomness to the f value because we are accepting past solutions, and iteratively not picking the combination of the most optimal solutions. However, as we see in the last graph, for experiment 3, increasing the probability, ensures that we converge quicker because we increase the probability of combining the best solutions.



**Mutation Probability = 0.1**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**K\_P =** 4.804

**T\_i =** 6.0424

**T\_d =** 2.36

**Fval =** 90.605

**Mutation Probability = 0.9**

**Graphical user interface

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**K\_P =** 4.804

**T\_i =** 6.0424

**T\_d =** 2.36

**Fval =** 90.605

**Comparing Results:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Experiment 1 – mutation P = 0.25** | **Experiment 2 – mutation P = 0.1** | **Experiment 3 – mutation P = 0.9** |
| **K\_P** | 4.804 | 4.804 | 4.804 |
| **T\_i** | 6.0424 | 6.0424 | 6.0424 |
| **T\_d** | 2.36 | 2.36 | 2.36 |
| **Fval** | 90.654 | 90.605 | 90.605 |

**Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated**Graphical user interface

Description automatically generated

Comparing the results of this graph, we see that changing the mutation probability didn’t affect the f values or the convergence.

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Impact** | |
| **No** | **Change** | **Food foraging (exploration of search)** | **Transporting behaviour (finish time – when each of the food piles are complete )** |
| 1 | Effect of Increasing Population | As the population increases, the effect that was notice that food foraging also increases | As the population increases, the finish time was much faster for each of the food sources |
| 2 | Effect of Increasing Diffusion Rate | Food foraging/exploration was minimal here simply because most of the ants tended to cluster into areas that had more pheromone | As the diffusion increases, the finish time was much faster for each of the food sources (almost by double) |
| 3 | Effect of Increasing Evaporation Rate | Lower evaporation also reduces the rate of exploration, so most ants are clustered around one area | Increasing the evaporation reduced the time to consume all of the food sources |
| 4 | Effect of changing the location of the food sources | The closer the sources, the fewer the exploration of the ants searching for food | The closer the food sources were to the source of ants, the faster each of the food piles were depleted |

**No 1 Screenshots:**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

**Population = 30, 50, 90**

**No2 Screenshots:**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

**Diffusion Rate: 40, 80**

**No3 Screenshots:**

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

**Evaporation Rate: 10, 20**

**No 4 Screenshots:**

**Original Food Source Locations:**

Text

Description automatically generated

Icon

Description automatically generated

Chart, line chart

Description automatically generated

**New Food Source Locations:**

**Text

Description automatically generated**

**A picture containing text, light, traffic, vector graphics

Description automatically generated**

Chart, line chart

Description automatically generated