|  |
| --- |
| Group 8 |
| Vehicle Routing Problem with Time and Capacity Constraints |
| Delivery from online shopping is gaining huge popularity in the covid-19 era, and to cope with the growing demand, our team will solve 2 main problems address the creation of driver schedules and routes. |

|  |
| --- |
| TAY Paul Hong, Xu Junhui, Wen Jingzhou  8-9-2020 |

Contents

[Motivation 2](#_Toc48045245)

[Problem Objectives and Scope 2](#_Toc48045246)

[Problems Overview 2](#_Toc48045247)

[Problem Definition 3](#_Toc48045248)

[Assumptions 3](#_Toc48045249)

[Proposed Model 3](#_Toc48045250)

[Algorithm 4](#_Toc48045251)

[Problem 2 5](#_Toc48045252)

[Problem Definition 5](#_Toc48045253)

[Assumptions 6](#_Toc48045254)

[Assumptions are the same as Problem 1 with several other assumptions including: 1) All vans have no capacity constraints and all customers are purchasing the same items. 6](#_Toc48045255)

[Proposed Model 6](#_Toc48045256)

[Algorithm 6](#_Toc48045257)

[Experimental results demonstrating efficiency and effectiveness 7](#_Toc48045258)

[Problem 1 7](#_Toc48045259)

[Small Data Set – Base Model 7](#_Toc48045260)

[Large Data Set – Base Model 7](#_Toc48045261)

[Small Data Set – Bootstrapping Model 7](#_Toc48045262)

[Large Data Set – Bootstrapping Model 8](#_Toc48045263)

[Problem 2 9](#_Toc48045264)

[Model’s convergence 9](#_Toc48045265)

[Model’s performance against test datasets (largeDB) 9](#_Toc48045266)

[Insights and Lessons 10](#_Toc48045267)

[Why having a first solution strategy is very important 10](#_Toc48045268)

[Global optimal VS Local optimal 10](#_Toc48045269)

[Randomness may have a huge impact on the solution 10](#_Toc48045270)

[Why some models perform better than others 10](#_Toc48045271)

[Using Genetic Algorithm for MVRP 11](#_Toc48045272)

[Limitations 11](#_Toc48045273)

[Appendix 11](#_Toc48045274)

# Motivation

Nowadays, in our fast-paced daily lives, many of us want instant gratifications. Many of us wishes to have our delivery reach our doorsteps as soon as we check out our online shopping carts! However, the reality is that timetabling, and routing planning process is often painful and difficult, and its intractable complexity got a name “NP-hard” by our brilliant computer scientists!

As “NP-hard” as they are, our team believes there are things we can do to make the problem manageable. In our journey of solving this timetabling and routing planning problem, we have encountered many decision points where we need to make trade-offs between optimality and runtime, between model robustness and best individual outcomes and we have also ventured into evolutionary algorithms to get a flavor of its novelty in problem solving.

# Problem Objectives and Scope

Typically for delivery service there are often two types of orders: normal (or planned) orders and emergency (or spontaneous) orders. The objective for normal orders is to maximize estimated profit, by considering the following factors that have an impact to the profit:

1. Customer order, goods type, quantity and timeslot
2. Item price, weight
3. Customer location, warehouse location
4. Van type, speed, max load
5. Delivery cost, fixed van rental, petrol cost, failed delivery

Normal orders are received prior to the delivery date thus we have relatively larger amount of time for it to run through a more comprehensive search algorithm. However, there can still be many instances of failed deliveries due to customers’ last-minute cancelation or absentee from home etc. Therefore, the model needs to enhance its robustness to account for such possibilities.

The objective for emergency order is to minimize the maximum delivery time of all vans.

As the emergency orders are generated spontaneously, it is imperative to make sure the algorithm generates a good enough result within a short window of time, Therefore, we may need to select a model with high versatility and fast running speed at the expense of true optimality.

# Problems Overview

There are 2 main problems, the Problem 1 addresses “Normal Orders” and the Problem 2 addresses “Emergency Orders”. As the company has assigned 5 vans for “Normal Orders” and 5 vans for “Emergency Orders”, vans not needed for “Normal Orders” can potentially be used to support the “Emergency Orders” team for that day. The details are mentioned below.

A screenshot of a cell phone

Description automatically generatedProblem 1

### Problem Definition

The first problem can be defined as choosing routes for a given number of vans to serve groups of customers nodes and each having different time windows. Each customer orders different items, with each item having different prices and weight per item. There is different limited weight capacity for each van and all vans must start and end at the depot node. Deliveries made must be within the customer’s stated time window, if the delivery is unable to happen, there will be a penalty (Failed Delivery Cost). Since distance between nodes varies, the travel time between nodes varies as well. There are also a petrol cost incurred while travelling and a van rental cost incurred if a van is needed for the day. The goal is to generate a route per van in order to maximize the total profits made, i.e. Revenue coming from delivering the items ordered by the customer and the costs coming from the petrol cost, van rental cost and failed delivery cost. Each customer must only be visited once and by one van.

There also exist 2 stochastic considerations that affects the creation of routes.

1. There can be cancellations of orders after the van leaves the depot
2. There can be changes in time window of some customer nodes once after the van leaves the depot

However, since the vans have left the depot, there is no way the van can change the route anymore.

Hence, this problem is stochastic and static.

### Assumptions

There is only 1 depot node, and all items ordered are available in unlimited quantity in that depot. The distance and time for between any two nodes are calculated using the Google Maps API and are deemed as accurate. All vans travel at the same speed. A complete delivery includes delivering all the items selected by the customer. There is no service time needed for all customer node, i.e. Once the van reached a node, the time taken to deliver the items and leave the node is assumed to be 0 sec.

### Proposed Model

The CP model was run, and 3 different First Solution strategies and 4 different Local Search options were tried. The details of strategies and options are explained as below:

**First Solution Strategy**

1. Global Cheapest Arc - Iteratively connect two nodes which produce the cheapest overall route
2. Path Cheapest Arc - Starting from a route "start" node, connect it to the node which produces the cheapest route segment, then extend the route by iterating on the last node added to the route

**Local Search Option**

1. Tabu Search – A procedure to iteratively move from one potential solution to an improved solution in the neighbourhood, until some stopping criterion has been satisfied (generally, an attempt limit or a score threshold)
2. Simulated Annealing – A procedure that considers some neighbourhood state based on a current state and probabilistically decides between moving to a new state or staying in the current state
3. Guided Local Search – A procedure that builds up penalties during a search and uses that to help local search algorithms to escape local minimal by modifying its objective function
4. Greedy Descent – A procedure to reach a local optimal fast by accepting improving local search neighbours greedily

### Algorithm

Variables

|  |  |
| --- | --- |
| N – number of customer and depot nodes | tij – time taken in minutes to travel from node i to j |
| K – number of vans in the depot node | qk – Weight capacity of van k |
| M – number of items available | STi – Start time of customer i |
| qmi-quantity of item m ordered by customer i | ETi – End time of customer i |
| pm – price of item m | v – speed of all vans |
| wm – weight of item m | cp – cost of petrol |
| gi – Total item weight of customer i. | cv – cost of van rental |
| ri – Total Revenue of customer i. | cf – cost of failed delivery |
| dij – Distance travelled in km from node i to j |  |

Decision Variable

Objective Function

where

1. Time take to travel from node to
2. Weights of items ordered by customer
3. Revenue from items ordered by customer

Constraints

1. Number of vans which start from depot node and go back to depot node is K
2. Each customer node can only be served by one van
3. All vans must start and end at depot node
4. Weight of items carried by van k cannot exceed its weight capacity
5. Vans can only deliver to customer node i within the time window

Post Processing

1. Total Revenue
2. Total petrol cost
3. Total Vehicle cost
4. Failed Delivery Cost
5. Total Profit

To add uncertainty to the model, we perform 10 iterations, with varying number of customer nodes and time windows. Once done, we have 10 xijk matrix. We then find the mode of all 10 xijk, and that will be the final result.

Problem 2

### Problem Definition

Problem 2 is similar Problem 1, with the main difference being that the objective is to minimize the maximum distance covered by a van, thus transforming the problem into a MVRP (multiple vehicle routing problem).

### Assumptions

## Assumptions are the same as Problem 1 with several other assumptions including: 1) All vans have no capacity constraints and all customers are purchasing the same items.

### Proposed Model

Genetic algorithms are used to solve the MVRP with a fixed number of generations to control the model run-time.

### Algorithm

Variables used are the same as problem 1

Parameters

|  |  |
| --- | --- |
| P – population size | Pm – probability of mutation |
| G – number of generations | E – size of elite population |
| Pc – probability of crossover |  |

Decision Variable

Objective Function

Where

1. Solution space is represented by y, such that

Selection methodology

Tournament of two – at each selection, two solutions will be randomly selected from the population and the one with higher fitness value will be selected. This is to ensure population variety is preserved to some extent.

Ordered crossover – solution for a VRP is an ordered list and therefore it is important to preserve the ordering of the list to a certain degree. This is achieved by a two-point cut over at random locations and then filling in the gaps with the same ordering.

Example:

Parent solution 1 – 143265, Parent solution 2 - 624135

**Two-point cut over: at locations 2 and 5**

Offspring 1 – \_2413\_, left with 6 and 5, Offspring 2 – \_4523\_, left with 6 and 1

**Fill in gaps with parents’ sequence of order**

Offspring 1 – 624135, Offspring 2 – 645231

**Shuffled mutation** – we favored shuffled mutation because it can introduce more random solutions and always keep the model on explorative mode.

**Swap mutation** – swap mutation can also be considered if the model struggles to stabilize on a good solution.

**Elitism** – It is a good idea to always save a small pool of best solutions so as to not be lost to randomness introduced by mutations and crossovers.

**Termination criteria** – number of generations, since model run-time is the primary concern here, so limiting the number of generations is the best choice for termination criteria.

# Experimental results demonstrating efficiency and effectiveness

## Problem 1

### Small Data Set – Base Model

For the base model on the small dataset, Tabu Search with Path Cheapest Arc outperformed other first solution strategy and search method combination. Even though Greedy Descent method did not give the best profit, its running speed of 1 second is fastest among all, with each of the rest taking about 10 minutes.

For the best model, this is the route for each van, presented in this for format – {vanId: [nodeId, …], …}

{1: [0, 6, 7, 5, 4, 11],3: [0, 9, 10, 8, 14, 12, 15],4: [0, 13, 2, 1, 3]}

### Large Data Set – Base Model

For the base model on the large dataset, Tabu Search with Path Cheapest Arc outperformed other first solution strategy and search method combinations. For all Local Search Options with Global Cheapest Arc, no results could be found given a 1 min max time limit.

For the best model, the results for each van are:

{3: [0, 44, 48, 50, 94, 96, 98, 100, 102, 26, 6], 5: [0, 92, 99, 52, 54, 55, 57, 58, 59, 61, 3], 8: [0, 70, 75, 77, 68, 62, 64, 25, 27, 17, 4], 9: [0, 63, 65, 83, 86, 87, 67, 72, 66, 10], 13: [0, 49, 45, 47, 51, 53, 56, 60, 20, 21], 15: [0, 69, 71, 74, 76, 78, 80, 82, 84, 85, 88, 90, 9], 16: [0, 33, 15, 16, 13, 19, 22, 5], 17: [0, 41, 43, 46, 2], 19: [0, 34, 35, 37, 38, 39, 42, 28, 23, 11], 25: [0, 1, 12, 14, 18, 103, 36, 40], 28: [0, 30, 29, 32, 31, 24, 7], 29: [0, 73, 79, 81, 89, 91, 93, 95, 97, 101, 8]}

### Small Data Set – Bootstrapping Model

For the bootstrapping model on the small dataset, the profits provided from variant dataset does not differ much. The profits obtained from different iterations did not differ much for each method and strategy combination. Tabu Search and Guided Local Search gives the best average profit.

### Large Data Set – Bootstrapping Model

For the bootstrapping model on the large dataset, each combination of search method and first solution strategy ran for 10 iterations with varying dataset. There were 10 variant datasets for each of the 10 iterations. For each iteration, each combination is tested using the same dataset to compare with the other combinations. For some combinations, there were no feasible solution. The combinations that were not able to provide a feasible solution were less robust than those that could provide one. The process of comparing robustness is very lengthy as each model (other than Greedy Descent) took 10 mins for one iteration.

The best routes from iterations are: {3: [0, 1, 12, 16, 19, 21, 9, 11],4: [0, 92, 29, 31, 33, 35, 37, 39, 34, 4],5: [0, 42, 44, 46, 48, 49, 53, 54, 56, 7],9: [0, 55, 83, 85, 84, 86, 87, 89, 91, 30, 3],15: [0, 17, 18, 20, 24, 25, 27, 28, 5],18: [0, 59, 80, 75, 77, 79, 81, 82, 88, 90, 14],19: [0, 41, 43, 45, 47, 50, 52, 57, 58, 60, 23, 6],25: [0, 2, 10, 32, 36, 38, 40, 26, 15],28: [0, 61, 63, 64, 67, 69, 71, 73, 62, 51, 8],29: [0, 65, 66, 68, 70, 72, 74, 76, 78, 22, 13]}

The first solution strategy of Path Cheapest Arc was more robust to generate the solutions given different datasets and the profit generated by Path Cheapest Arc was higher than Global Cheapest Arc. The two best profits came from Tabu Search and Guided Local Search. Greedy Descent is worst performing among all the search options.

The highest profit among all iterations come from Guided Local Search with Path Cheapest Arc as the first solution strategy. The highest profit solution selected 10 vans out of 31 and designed the routes for each selected van. To illustrate our solution, 5 routes out of 10 are plotted below.

A picture containing text, map

Description automatically generated

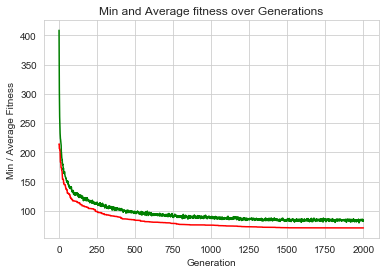
In a word, among all the local search methods, Guided Local Search and Tabu Search outperformed the rest methods. Path Cheapest Arc or Global Cheapest Arc can be the most suitable first solution strategy depending on the nature of the dataset. For a larger dataset, it is likely that Global Cheapest Arc may not be able to generate feasible solutions due to the complexity of the algorithms and the dataset constraints.

After different strategies and options having been tested with variant small dataset and large dataset, it can be concluded that Guided Local Search and Tabu Search with Path Cheapest Arc are the best options and strategy to generate good solutions. They are robust to provide quality result with efficiency and effectiveness.

## Problem 2

### Model’s convergence

It is generally difficult to know the convergence point for GA models due to its inherited randomness, and based on trial-and-error, we observed the GA model typically converge after 2000 generations.



### Model’s performance against test datasets (largeDB)

While the model runtime is kept well within 5mins, its performance is not satisfactory against the global optimal solutions. As the number of vans decreases, the model’s performance deteriorates progressively, it is apparent that the GA model, while useful in certain scenarios, is sub-par in its performance against other state-of-the-art models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of vans | Model run-time (sec) | Optimal solution found by GA | Global Optimal | Solution Quality |
| 31 | 77 | 59.28 | 57 | 96% |
| 25 | 74 | 63.24 | 57 | 90% |
| 20 | 71 | 67.18 | 57 | 85% |
| 15 | 68 | 70.76 | 57 | 81% |
| 10 | 59 | 130.42 | 57 | 44% |
| 5 | 59 | 173.45 | 61 | 35% |
| 3 | 59 | 175.62 | 74 | 42% |
| 1 | 59 | 482.43 | 157 | 33% |

Model’s performance against randomized datasets (30 nodes)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of vehicles | Model run-time (sec) | Optimal solution found by GA | Global Optimal | Solution Quality |
| 5 | 31 | 629.35 | 628 | 100% |
| 2 | 27 | 1065.42 | 849 | 80% |
| 1 | 27 | 1499 | 1300 | 87% |

Model’s performance against randomized datasets (500 nodes)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of vehicles | Model run-time (sec) | Optimal solution found by GA | Global Optimal | Solution Quality |
| 20 | 304 | 2735.09 | 1005 (65min) | 37% |

While the GA’s solution is inferior to the global optimal solution, but it takes 65mins for the system to generate the global optimal solution.

# Insights and Lessons

## Why having a first solution strategy is very important

The first solution strategy is the method the solver uses to find an initial solution. It is a greedy approach that follows the problem-solving heuristic of making the locally optimal choice at each stage. Although it does not guarantee the global optimal solution, it is helpful to decide a starting point, i.e. a combination that satisfies all constraints. Without it, the model might have to run many different iterations before finding a valid combination. This might take a long time, and it does not guarantee a better starting point anyway.

That said, implementing a first solution strategy without any local search methods like Tabu, Simulated Annealing, etc., is not good enough as it only produces a combination that satisfy all constraints, but does not produce a local optimal that is close to the global optimal. Hence, complimenting this with a local search method that helps us to consider nearby local optimal might give us a better solution without compromising so much in terms of search time. When we started this project, we used a first solution strategy without a Local Search Option, and that caused us to get solutions that can satisfy all constraints but by just eyeballing the solution, we can tell that they are not optimal. For example, every van is travelling all around Singapore instead of focusing on one area. In our model, we explored 2 different first solution strategy – 1) Path Cheapest Arc, and 2) Global Cheapest Arc. Although both methods caused the model to land on a different starting point, the time taken to reach that starting point is much faster than if we randomly generate different combinations until we hit a valid combination.

## Global optimal VS Local optimal

Sometimes, the “good enough” solution might be better than the “best” solution for a given use case due to practical limitations, such as time taken for the model to run. For our use case, we don’t have all the time in the world to generate the “best” solution due to time pressure from the business to generate a solution in a relatively short amount of time so that the implementation of the solution can be executed. In fact, creating a solution in a short amount of time that meets a business deadline which might not give the highest profit is still better than creating a solution that takes too long to run and in the end, not being able to execute the solution as the day is over.

Furthermore, for an e-commerce firm, the opportunity cost of not being able to create the global optimal route is loss of some profits which is not as devastating as compared to the healthcare industry where it could lead to lives loss.

## Randomness may have a huge impact on the solution

The first randomness in our project is the variant number of customers to delivered due to unseen events due to traffic or weather or customers not at home. The second randomness is the variant time window of customers. Due to the randomness of the data, the most suitable search method and first solution strategy and routes for each van varied compared to the base model which is static. For example, Tabu Search with Path Cheapest Arc gave the best profit in the base model on the small dataset, while Guided Local Search with Path Cheapest Arc gave the best profit in the bootstrapping model when the randomness was introduced. The randomness will cause the dataset to change, and then the solution will change, with the method to generate the solution changing too.

## Why some models perform better than others

For most experiments (more than 90%) in our project, Tabu Search works better than Greedy Descent local search method. The reason is that Tabu search method can get out of the local minimum due to its metaheuristic nature, and it has a higher chance to reach the global minimum. Simulated Annealing applies a probabilistic heuristic technique that gives the algorithm a percentage to get out of local minimum like the process of heating cooling annealing in metallurgy. Guided local search is a metaheuristic search method that builds up penalties during a search. It uses penalties to help local search algorithms escape from local minimal and plateaus. In a word, Tabu Search, Simulated Annealing and Guided Local Search are supposed to provide better solutions than Greedy Search method but will take much longer time. Guided Local Search is said to be the most efficient metaheuristic for vehicle routing. However, in our experiments, we find Tabu search also gives better solutions overall. Both Guided Local Search and Tabu Search provide very good results in our experiments. The possible reasons could be due to the nature of the data.

## Using Genetic Algorithm for MVRP

GA model has a sub-par performance against other state-of-the-art models, and it is generally less preferred for MVRP problems. However, GA model can potentially add value when the dataset is very huge and there is a tight model runtime constraint.

In the random 500 node test, there is a trade-off between model run-time and performance, GA model, with its light-weightiness, can provide a reasonable alternative solution in the event that the state-of-the-art model cannot generate a solution within the stipulated time.

In our opinion, our GA model didn’t capitalize on the unique structure of MVRP, and therefore many of the solution generated are too random to provide any performance enhancement. One area is to look deeper at the crossover operation and to ensure that on average the offspring from the crossover operation have a higher fitness score than that those of their parents.

## Limitations

There can be multiple warehouses in real scenario whereas there is just one warehouse as the depot in our vehicle routing base and bootstrapping models. All the items are assumed to be stored in that one warehouse and the warehouse is assumed to have unlimited items. Given more time, how to introduce more depots and how to apply the item-warehouse constraints can be further explored. The item-warehouse constraints include that items can be retrieved from multiple warehouses, and the items in customers’ orders cannot exceed the capacity of all warehouses.

# Appendix

Robustness Testing Results for Small Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | LocalSearchMetaheuristic | Profit | Distance | Vans |
| 0 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2967.402 | 210 | [1, 2, 3] |
| 1 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2972.008 | 198 | [1, 2, 3] |
| 2 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 3 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2968.17 | 208 | [1, 3, 4] |
| 4 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2963.18 | 221 | [1, 2, 3] |
| 5 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2972.008 | 198 | [0, 3, 4] |
| 6 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2965.483 | 215 | [1, 2, 4] |
| 7 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2970.472 | 202 | [0, 1, 4] |
| 8 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2858.575 | 233 | [0, 1, 2, 3] |
| 9 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 2971.624 | 199 | [0, 1, 3] |
| 0 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2967.402 | 210 | [1, 2, 3] |
| 1 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 2 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2954.353 | 244 | [1, 2, 4] |
| 3 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2968.17 | 208 | [0, 3, 4] |
| 4 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2962.029 | 224 | [1, 2, 3] |
| 5 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2970.472 | 202 | [0, 3, 4] |
| 6 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2968.17 | 208 | [1, 2, 4] |
| 7 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2970.856 | 201 | [0, 1, 4] |
| 8 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2858.575 | 233 | [0, 1, 2, 3] |
| 9 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 0 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2967.786 | 209 | [1, 2, 3] |
| 1 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 2 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 3 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2958.958 | 232 | [1, 3, 4] |
| 4 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2966.251 | 213 | [1, 2, 3] |
| 5 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2972.391 | 197 | [0, 3, 4] |
| 6 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2968.553 | 207 | [1, 2, 4] |
| 7 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2958.958 | 232 | [0, 2, 3] |
| 8 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2960.11 | 229 | [1, 2, 3] |
| 9 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 2973.159 | 195 | [0, 1, 3] |
| 0 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2970.089 | 203 | [0, 1, 2] |
| 1 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2971.24 | 200 | [2, 3, 4] |
| 2 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2967.786 | 209 | [2, 3, 4] |
| 3 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2968.553 | 207 | [0, 3, 4] |
| 4 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2966.251 | 213 | [0, 3, 4] |
| 5 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2967.402 | 210 | [2, 3, 4] |
| 6 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2967.402 | 210 | [1, 3, 4] |
| 7 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2968.553 | 207 | [1, 3, 4] |
| 8 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2963.948 | 219 | [0, 3, 4] |
| 9 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 2972.391 | 197 | [0, 1, 3] |
| 0 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2969.321 | 205 | [2, 3, 4] |
| 1 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2972.008 | 198 | [2, 3, 4] |
| 2 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2968.553 | 207 | [2, 3, 4] |
| 3 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2953.585 | 246 | [2, 3, 4] |
| 4 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2965.867 | 214 | [1, 3, 4] |
| 5 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2972.391 | 197 | [2, 3, 4] |
| 6 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2960.11 | 229 | [2, 3, 4] |
| 7 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2970.856 | 201 | [0, 1, 4] |
| 8 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2961.645 | 225 | [2, 3, 4] |
| 9 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 2971.24 | 200 | [2, 3, 4] |
| 0 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2967.402 | 210 | [1, 2, 3] |
| 1 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2972.008 | 198 | [1, 2, 3] |
| 2 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 3 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2968.553 | 207 | [0, 3, 4] |
| 4 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2965.483 | 215 | [1, 2, 3] |
| 5 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2972.391 | 197 | [0, 3, 4] |
| 6 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2967.786 | 209 | [1, 2, 4] |
| 7 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2970.856 | 201 | [0, 1, 4] |
| 8 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2965.099 | 216 | [0, 3, 4] |
| 9 | TABU\_SEARCH\_Global\_Cheapest\_Arc | 2971.24 | 200 | [0, 1, 3] |
| 0 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2970.089 | 203 | [2, 3, 4] |
| 1 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2971.24 | 200 | [2, 3, 4] |
| 2 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2965.867 | 214 | [2, 3, 4] |
| 3 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2968.553 | 207 | [2, 3, 4] |
| 4 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2962.029 | 224 | [0, 2, 4] |
| 5 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2972.391 | 197 | [0, 3, 4] |
| 6 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2967.018 | 211 | [2, 3, 4] |
| 7 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2970.856 | 201 | [2, 3, 4] |
| 8 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2966.251 | 213 | [1, 3, 4] |
| 9 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 2968.17 | 208 | [2, 3, 4] |
| 0 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2970.089 | 203 | [1, 2, 3] |
| 1 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2971.624 | 199 | [1, 2, 3] |
| 2 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2970.472 | 202 | [1, 2, 3] |
| 3 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2968.17 | 208 | [0, 3, 4] |
| 4 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2967.402 | 210 | [1, 2, 4] |
| 5 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2968.17 | 208 | [2, 3, 4] |
| 6 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2968.17 | 208 | [1, 2, 4] |
| 7 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2970.856 | 201 | [0, 1, 4] |
| 8 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2966.251 | 213 | [0, 3, 4] |
| 9 | GUIDED\_LOCAL\_SEARCH\_Global\_Cheapest\_Arc | 2973.159 | 195 | [0, 1, 3] |

Robustness Testing Results for Large Dataset

|  |  |  |  |
| --- | --- | --- | --- |
|  | LocalSearchMetaheuristic | Profit | Distance |
| 0 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 11735 | 645 |
| 0 | SIMULATED\_ANNEALING\_Global\_Cheapest\_Arc | 11733 | 652 |
| 10 | GREEDY\_DESCENT\_Global\_Cheapest\_Arc | 11704 | 727 |
| 2 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11986 | 514 |
| 9 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11985 | 517 |
| 1 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11984 | 519 |
| 14 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11982 | 525 |
| 4 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11979 | 531 |
| 16 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11979 | 532 |
| 0 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11977 | 537 |
| 5 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11976 | 539 |
| 15 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11972 | 550 |
| 19 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11968 | 560 |
| 8 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11965 | 569 |
| 6 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11889 | 506 |
| 12 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11884 | 518 |
| 3 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11874 | 545 |
| 13 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11872 | 551 |
| 11 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11871 | 552 |
| 7 | GUIDED\_LOCAL\_SEARCH\_Path\_Cheapest\_Arc | 11871 | 553 |
| 20 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11867 | 562 |
| 0 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11857 | 588 |
| 18 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11857 | 589 |
| 5 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11850 | 608 |
| 3 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11845 | 620 |
| 8 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11814 | 701 |
| 2 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11806 | 723 |
| 17 | TABU\_SEARCH\_Path\_Cheapest\_Arc | 11765 | 567 |
| 9 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11744 | 624 |
| 1 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11738 | 639 |
| 5 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11735 | 645 |
| 6 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11735 | 645 |
| 3 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11734 | 649 |
| 2 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11729 | 661 |
| 0 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11724 | 676 |
| 4 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11722 | 680 |
| 8 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11718 | 690 |
| 4 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11716 | 696 |
| 7 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11714 | 702 |
| 7 | SIMULATED\_ANNEALING\_Path\_Cheapest\_Arc | 11714 | 702 |
| 1 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11713 | 703 |
| 6 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11712 | 705 |
| 9 | GREEDY\_DESCENT\_Path\_Cheapest\_Arc | 11684 | 778 |