# Executive summary Problem Defination

# Data and Pre-processing

Our dataset *vrptw\_team13\_100\_orders.csv* is attached in zip folder. The data file includes 6 key columns as below:

1. SG postal code
2. Latitude
3. Longtitude
4. **Load (demand)**
5. **Earliest**
6. **Latest**
7. start\_tw (hh:mm)
8. end\_tw (hh:mm)

The data was generated from VRPTW’s standard [data set Solomon R101](http://web.cba.neu.edu/~msolomon/r101.htm).

The X coordinate and Y coordinate columns are replaced with **Geocoding** data so that we can do rout visualization in later part of the project.

From the original earliest and latest time window data, we derive two time fieds to be used in heuristic algorithm (Routific) API implementation later.

# Model part 1 - Lingo

Objective is to minimize cost (distance). To minimize the number of vehicles, the formulation is to sum each demand divided by capacity.

# Distance Matrix

We applied [Google's Distance Matrix](https://developers.google.com/maps/documentation/javascript/distancematrix" \t "_blank) service computes travel distance between orders using a “DRIVING” mode of travel. The prameters like **avoidHighways** and **avoidTolls** are set to false to meet real-world situation.

The objective is to solve the optimization problem in lingo code, we will reference data matrix as external file after calculation. This will make the lingo algorithm more efficient and useable for larger problem such as more than 100 orders.

Distance calculation file - Team13 folder->googleDistanceMatrix.html

Lingo data matrix file - Team13 folder->5Order> lingo> vrptwdataGoogleMatrix.txt

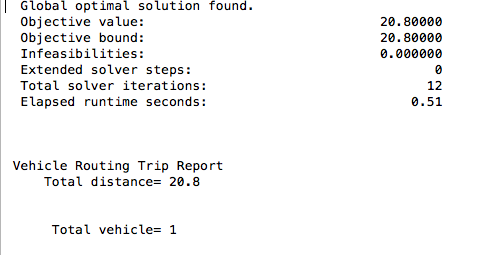
# 20 mins time window

# Lingo Result

Due to limitation of demo version of lingo, we can only solve max 5 orders for our problem. The result is shown below.

We need only 1 vehicle for 5 orders and total distance is 20.8km. It took only 0.3 seconds to solve the problem.

Since we can’t solve all given orders in limited lingo version, it takes much longer to find the best routes as the number of locations grow. Large versions of this model will be tackled using heuristics algorithm in next section of this report.



# Model part 2 - Heuristic Routific optimization algorithm

To solve the whole problem, we apply routific optimization engine, which is using Bee Algorithm for optimization. (Refer to- Bee Algorithm section)

The algorithm requires applying data in json (easy to understand JSON (JavaScript Object Notation) is a lightweight data-interchange format) and output the optimal routes in json format.

First, we run algorithm with same 5 orders inputs in Lingo. We can see the both models give same route.

File -> *fiveOrder.json*

Solution: 5 orders route

|  |  |
| --- | --- |
| Lingo | Routific |
|  |  |

**Solution 100 orders route**

Number of drivers: 15

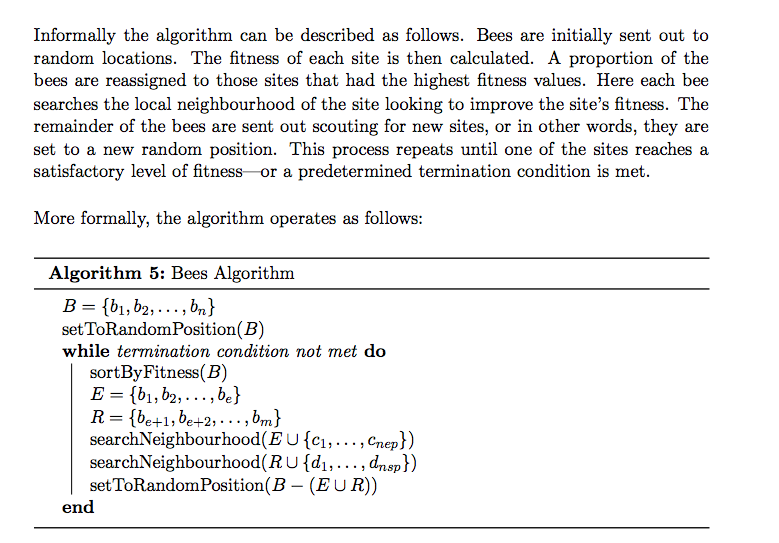
Total distance: 791.6km

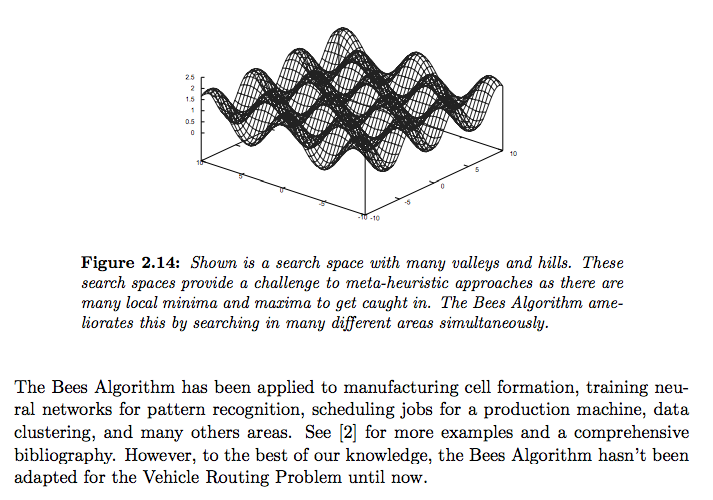
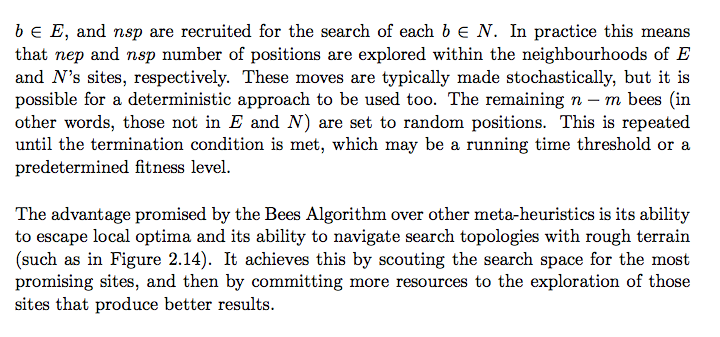
|  |  |
| --- | --- |
| **Route** | **Distance (km)** |
| Route 1 | 42.4 |
| Route 2 | 35.8 |
| Route 3 | 35.2 |
| Route 4 | 35.9 |
| Route 5 | 81.5 |
| Route 6 | 66.9 |
| Route 7 | 40.3 |
| Route 8 | 68.8 |
| Route 9 | 61.4 |
| Route 10 | 57.7 |
| Route 11 | 61.9 |
| Route 12 | 61.3 |
| Route 13 | 53.4 |
| Route 14 | 43.6 |
| Route 15 | 45.5 |
| Total distance | 791.6 |

# Summaries

|  |  |  |
| --- | --- | --- |
| Constraints | Distance (km) | Vehicles |
| * With 20 mins time window * Starts and ends at depo * Service time 10 mins | 791.6 | 15 |
| * With 20 mins time window * Starts at depo and do not return to depo * Service time 10 mins | 561.1 | 15 |
| * Without Time window * Starts and ends at depo * Service time 10 mins |  |  |

# Bee algorithm





Reference to lingo model - <http://www.lindo.com/lsmodels/Controller/doSelect.php?action=2&name=VRouteWindow.lng>

Routific Engine API - <https://docs.routific.com/docs/api-reference>

Bee Algorithm for VRPTW problem - <https://arxiv.org/abs/1605.05448>