Minimart Project

Davide Canali (10674880), Matteo Cordioli (10611332)

Our solution to the problem is structured into two parts:

A. The first part aims at choosing which cities will become markets. More specifically it identifies 80 possible solutions: 40 trying to minimize the cost of building the markets and 40 trying to minimize the number of markets.
B. The second part aims at finding the optimal route for each solution computed in the first part, to refurbish all markets using different number of trucks.

Designing A:

If the cost of building a market is considerably higher than the cost of refurbishing it, the optimal solution will likely to be in the first 40 results (number chosen to restrict the computation time). Otherwise, if refurbishing costs a lot more than building, the optimal solution will likely to be in the second group of 40 results given by A as few markets should help to minimize the objective.

The first 40 solutions are computed as followed:

$$\min \sum_{i \epsilon Nodes} chosen_i * costToBuilt_i$$

$$chosen_1 = 1$$

$$\forall i \in Nodes \mid usable_i = 0 \rightarrow chosen_i = 0$$

$$\forall i, \forall j \mid i \neq j, dist[i,j] < range \rightarrow numOfNodeReach_i \geq 0$$

The last 40 solutions are computed taking in consideration the amount of nodes selected and not the cost to build them:

$$\min \sum_{i \epsilon Nodes} chosen_i$$

$$chosen_1 = 1$$

$$\forall i \in Nodes \mid usable_i = 0 \rightarrow chosen_i = 0$$

$$\forall i, \forall j \mid i \neq j, dist[i,j] < range \rightarrow numOfNodeReach_i \geq 0$$

To be able to solve these 2 models multiple times, at each new iteration an extra constrain is added to remove the last solution from the possible future results.

At last, duplicated solutions in the 2 computed sets are identified and one of them is removed to make sure that each result is unique.

The library used to compute these solutions is or-tools.

Designing B:

For each solution previously identified, the algorithm now finds the optimal paths to refurbish all markets using k number of trucks. This is done for each k between:

- the minimum number of trucks required to reach all the markets, calculated as *Ceiling(NumOfMarket/PossibleRefurbishForTruck)* and

- the number of markets, considering the worst-case scenario that requires the acquisition of a truck for each market.

To optimize the computation time, the algorithm stops at the iteration when the number of trucks needed in that path is less than the number of trucks considered (k). The library used to compute the paths is or-tools.

To find each path, the algorithm identifies the cheapest route segment between the initial node and one of the markets, using a precomputed distance matrix. After finding the first route segment, the algorithm recursively finds all the next ones.

The heuristic to compute each path is the Guided Local Search. Its goal is avoiding that the process ends in a local minimum by modifying the objective function and applying penalties on those features that occur more often when a local minimum has been reached.

At the end of the process, the algorithm analyses all the computed solutions and returns the one with the minimum total cost.