Solution Report

Foundations of Operations Research

Prof. Malucelli Federico, Prof. Schettini Tommaso

3 January 2022

Ottavia Belotti **10657411**

10621546

Martin Bronzo
10619358

Alessio Braccini

The solution is computed by dividing the problem in two sequential steps:

1) Finding the shops to open based upon the minimum building total cost
2) Finding the plan for refurbishing the open shops found at step 1

This approach could mine the solution's optimality, since the periodical nature of the refurbishing cost could affect the shop's building decision. However, the complexity of the second problem makes it hard to merge the two problems into one, moreover trying to compare a recurring cost (refurbishing) and *una tantum* cost (building) without knowing the amortisation time-span it makes no sense.

Shop building:

We approached the problem with MIP library for Python (equivalent to AMPL) in order to find an optimal solution for the first subproblem.

Variable: $y_i = \begin{cases} 1 & \text{shop } i \text{ opened} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N$

The objective function is: $\min \sum_{i \in N} c_i y_i$

The constraints are:

$$y_1 = 1$$

$$y_i \leq \text{usable}[i]$$

$$\sum_{\substack{j \in N \\ \text{dist}[i,j] < \text{range}}} y_i \geq 1$$

Trucks' schedule planning:

Finding the paths that connect all the opened shop (nodes) to plan the refurbishing respecting the maximum capacity of shops visitable by one truck while trying to minimise the cost is not a trivial problem. It can be reduced to the Multiple Travelling Salesman problem (MTSP) and Multiple Vehicle Routing Problem (MVRP), which are notably NP-complete problems. So we had to

implement an heuristic that could find a reasonably good feasible solution, but suboptimal since we don't test every combination possible.

Multiple variables take place in this problem like how many trucks are needed to visit all the stores, how many shops each of them has to visit and which ones should a truck visit.
Less truck translates into less fixed cost, but it could increment the total variable cost because a truck might need to visit very far away shops as well if the graph is strongly sparse.

We implemented different algorithms based on cluster analysis: the ones which gave us better results are derived by Clarke-Wright algorithm and the nearest neighbour algorithm.

Since both approaches are extremely fast we decided to implement an ensemble algorithm that provides the best result of the given problem, it first runs with NN and then with the CW to find the best solution.

The CW is a well known VRP algorithm based on the concept of saving.
Our interpretation works as described: the "saving" quantity is defined as $s_{ij} = d_{1i} + d_{1j} - d_{ij}$ and it measures the advantages of combining the shops i and j in a single route, where d stands for distance and 1 refers to the depot. We iterate in savings and if neither i nor j have already been assigned to a route, a new route is created. If both are at the start or end of different routes (extremity points), the respective routes are merged. If just one is an extremity point in a route, the route is extended with the other element.
We apply constraints capacity and number of trucks in order to find the lowest number of routes having the maximum length.

The NN uses a slightly different approach: the algorithm tries out different combinations of different lengths. For example, if the max truck capacity is 10, it tests chosen paths of size 10 (excluding shop 1 that is the headquarter), removing a node from the list of opened stores once it has been included in a path. When the truck has used up all its capacity, we find a complete route for it and, if we still have some shops left to visit, we iteratively start from the headquarter again with a second truck until we use up all its capacity or we finish the shops.
Once the shops are all settled, we have built the solution of trucks having routes length of 10. Then we repeat the same computation but with a maximum length of 9, 8 down to 1.
The strategy to choose the next shop to visit is exploring the nodes in a greedy way. The closer node to the current one will be the next visited one (since the variable cost directly depends on the mileage).

Since our solutions try to find out the shops closer to each other, we're creating a sort of neighbourhood or cluster.

These algorithms are very fast and compared to the optimal solution on small datasets (the largest dataset we tested with the optimal procedure was the minimart-I-50.dat file that took over 30 minutes to complete) is not so far behind, the optimal solution has cost 280 while our algorithm has cost 284 that is an acceptable cost.

Conclusion:

In the end, we deem this algorithm as a reasonable one, it is very fast and finds solutions that are not too far away from the optimal ones, for what we have tested.
We immediately excluded the model implementation that explored all route combinations because its computation would take too much time (we tried it anyway for small datasets to test the other approaches).
We also tried other ways to solve this task like genetic algorithms but because of the poor results we abandoned this way.