School of Mathematical and Computer Sciences

COMP SCI 2203 - Problem Solving & Software Development

Programming Assignment

Capacitated Vehicle Routing Problem with two-dimensional loading constraints

25th October 2017

Prepared by

Zhongfan Zhang
a1670764

Michael Bradley Vincent
a1670261

Contents

1 Abstract

This paper proposes a divided and conquer approach to solving the 2-Dimensional Capacitated Vehicle Routing Problem. Instead of applying a routing algorithm on all nodes, the proposed method divides the data set by clustering the customer nodes based on euclidean distance using a K-means algorithm. A greedy routing algorithm is then applied to all clusters. A recursive packing algorithm is then applied to the clusters to determine whether each cluster is feasible with respect to vehicle capacity and size.

2 Unrestricted Packing

The packing algorithm employs a binary tree structure to separate the truck into different rectangular regions, each of which can hold items. The algorithm starts with the entire truck area and recursively splits into two smaller regions, repeating until items fit perfectly into the new space, or an item cannot fit. This works for unrestricted packing because the items can be placed anywhere.

Figure 1 shows an example of using the algorithm to insert two items into a truck. Item 1 is inserted first. It does not fit perfectly into the root node, so it is split into the upper and lower sections. Again it does not fit perfectly, so the upper section is split into 2. Finally the item is inserted into the node. As this is happening, the tree structure is generated and can be reused for all of the other items to be placed in the truck.

This algorithm does not return optimal results; it produces different results based on the order in which the items are inserted into the truck. The benefit of this is its complexity. This is $O(n\log(n))$ for n items, as traversing a binary search tree requires $\log(n)$ operations and it needs to be traversed n times. This is quick enough that a large amount of possible solutions can be tested for feasibility.

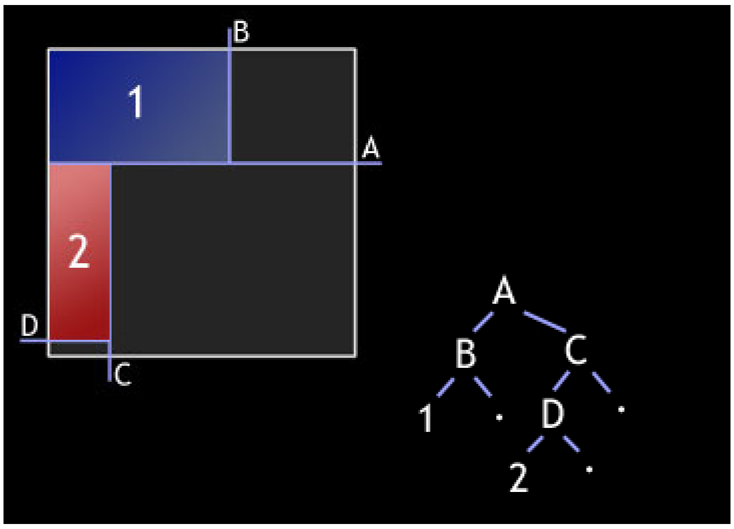


Figure 1: visualization of the packing algorithm and the binary search tree **source**: http://blackpawn.com/texts/lightmaps/

3 Solving 2L-CVRP

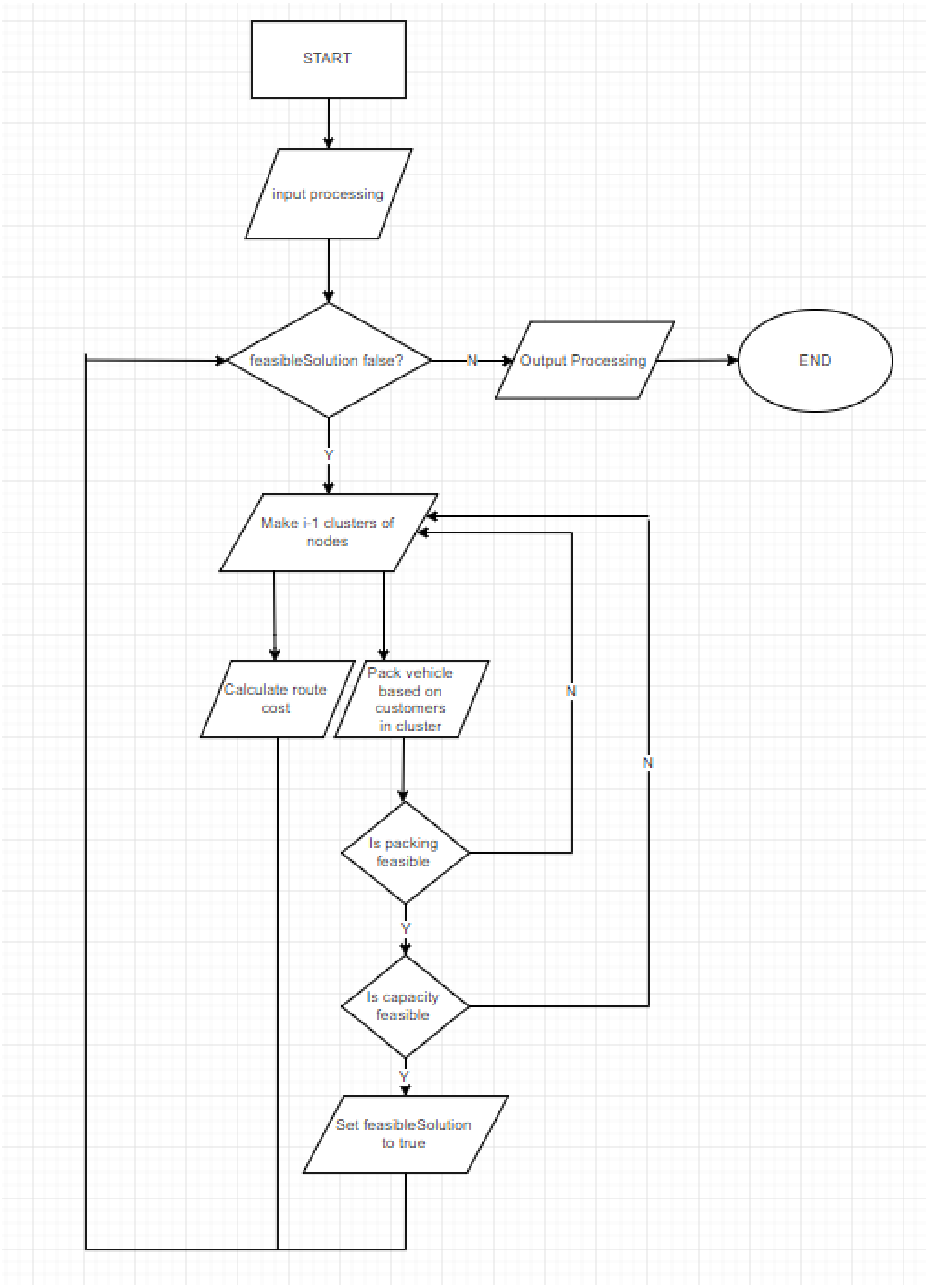


Figure 2: Algorithm flowchart (See section 5.1 for full page diagram)

3.1 Data structures

3.1.1 Node

To represent the locations and demands of the customers, a **Node** data structure was used to store all the information that was needed. This data structure contains the coordinates of the customer, weight demand, and a vector of items that needs to be delivered to that node. Furthermore it contains a pointer to another node, the index and group of the current node as well as the distance to the next node. The index was used to debug the route function and output the route in the format required. The group of the nodes are determined by the K-means function which groups customer nodes in clusters based on proximity in euclidean distance. The pointer to the next node is used to set up a linked list of customer nodes and the distance to next node variable is used to calculate the final cost of the routes.

3.1.2 Route

To store the routes that the program produced, the **Route** data structure was created. This data structure stores the head(the depot node) of a linked list of customer nodes and is responsible for producing the routes and calculating the total distance of the route. A route is produced by extracting all node of the same group and then applying a greedy routing algorithm on the extracted nodes.

3.1.3 Item

To solve the two-dimensional packing problem, an **Item** data structure was created. This data structure stores the width and length of an item and its coordinates within a vehicle.

3.2 Input processing

During the input processing section of the program, the location, index and weight attributes of the nodes are filled. All nodes are stored in a vector of node structures and the first node(the depot node) is separated.

3.3 Main loop

The main loop is an iterative algorithm used to find the first feasible route, the loop is controlled by a boolean variable and continues to run as long as the variable is false. A iterator variable is used to reduce the amount of groups that the K-means function create each iteration of the main loop.

3.3.1 Routing

The routing algorithm is a recursive implementation of a greedy shortest path algorithm of complexity $O(n^2)$. This is because it calculates the distance from the current node to all other nodes, picks the node that is the closest to the current node and then links the nodes and calls the route function again but with the next node as the parameter. This means that the first call of the function will result in n operations, n being the amount of distance calculations that are done and the second $n-1$ and so on. Since $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, the growth of the route function is bounded above by some function $f(n) = cn^2, c \in \mathbb{R}^+$. This process is completed for as many times as there are groups of nodes in the current iteration of the main loop, however this does not affect the complexity order of this function.

Calculating routes for smaller groups has also the benefit of reducing the amount of distance calculations done. For example, if there are 15 customer nodes and 5 vehicles, the algorithm will first try to find 5 clusters of 3. For each cluster, there will be 6 distance calculation operations resulting in a total of 30 distance calculations for all 5 clusters. However, undivided 120 distance calculation operations will have to be completed.

3.3.2 Packing

The packing algorithm uses both recursion and iteration. The first part of the algorithm in which the program searches for the best place to pack an item is a recursive implementation of a binary search tree. This packing algorithm is then called n times(n being the amount of nodes in the group) to pack all loads demanded by the nodes in the current group.

4 Results from computational Experiments

4.1 Results

The algorithm produced 0 feasible solutions, this was caused mainly by the focus of the clustering algorithm.The routing algorithm was able to produce routes, however, the total weight for all routes would either exceed the maximum vehicle capacity or the algorithm would fail to deliver all items to the customer nodes. The greedy clustering heuristic causes the algorithm to behave differently to all classes.

In class 1 problems, all items are size 1 and the main cause of failure is the weight feasibility. However in class 2 and 3 cases, the rate of failed cases would be increased as some clusters will return failure for packing.

4.2 Euclidean Distance Clustering

The cause of the problem stated above was rooted in the clustering algorithm. The K-means clustering algorithm that was used was focused on optimizing the euclidean distance between the nodes in each cluster without taking into account the total weight and item size for each cluster. This would perhaps produce more optimal solutions than the opposite order, however it rarely produced a feasible solution due to the weight constraint not met. For example, test case $2l_cvrp0101$ had 15 customers and 3 vehicles. Using the aforementioned algorithm, the customers were clustered into 3 groups of 5. Since the clustering only considered the euclidean distance, the total weight demand of one of the clusters was greater than the vehicle capacity. This would cause the packing function to return infeasible for this route and the amount of clusters would be decremented for the next iteration of the loop. However, splitting the customer nodes into two clusters caused both clusters to have a greater weight demand than the total vehicle capacity and the packing function would return infeasible for this division as well. This continues until the amount of clusters reaches 1 and since a single cluster would mean that a single vehicle is assigned to all nodes, it resulted in an infeasible route.

A possible solution to this problem could be to apply the principle of finding feasible solutions first, and optimising after. If the node groups were generated by, for example, randomly adding nodes to a group until the truck capacity was reached, feasible solutions would have been obtained. The clustering needed to be a secondary consideration.

The advantage of clustering is the easy division of the nodes. As a single vehicle will most likely not be able to deliver all the goods that every node demands, the problem must be divided into smaller sub-problems. Using a clustering algorithm to divide the customer nodes based on euclidean distance is an efficient way of dividing the loads between the vehicles.

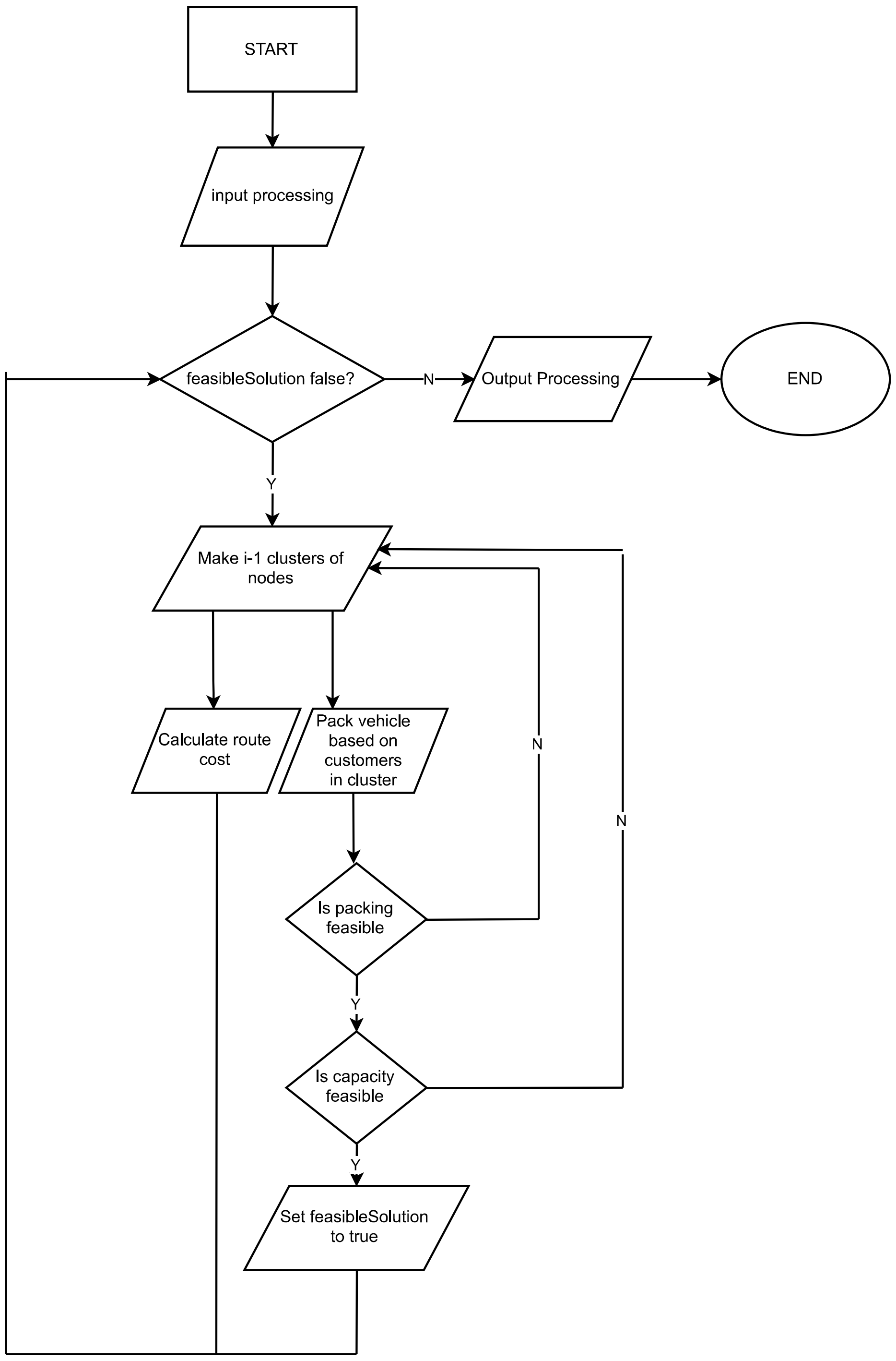
4.3 Conclusions

The proposed algorithm failed to produce feasible routes due to the nature of the clustering algorithm. The root of the problem was in the greedy clustering heuristic as the algorithm attempts to naively optimise the solution instead of producing feasible solutions before optimisation.

5 Appendices

5.1 Algorithm flow chart

See next page.


START
input processing
feasibleSolution false?
N
Output Processing
END
Y
Make i-1 clusters of nodes
Calculate route cost
Pack vehicle based on customers in cluster
N
N
Is packing feasible
Y
Is capacity feasible
Y
Set feasibleSolution to true

5.2 References

1. Jim Scott. 2010. *Packing Lightmaps*. [ONLINE] Available at: http://blackpawn.com/texts/lightmaps/. [Accessed 1 November 2017].