IE716: Integer Programming **Jan-April 2023**

Single Vehicle Pickup and Delivery using Integer Programming

Team Name: Dantzig *Team Members:* Hritiz Gogoi, Jyoti, Tarun Bisht

Abstract

Pickup and Delivery Problem is a widely appearing class of problems in real life. It has many variants. We solve an example of a pickup and delivery problem taking 34 stations within Mumbai City and solve it efficiently for an exact optimal solution.

1 Introduction

The Traveling Salesman Problem is widely used in a variety of real-life problems in its augmented forms. A class of such problems is the Pickup and Delivery Problem. In this class of problems, similar to TSP, we are given a network (in the form of a directed or undirected Graph). The nodes are classified as Depot (typically a singleton), Pickup Nodes, and Delivery Nodes. These nodes are associated with quantities of a product. Each delivery node requires a given amount of the product, while each pickup node provides a given amount of the product. The vehicle, starting and ending its route at the depot, needs to collect the product from the pickup nodes and supply it to the delivery nodes. Typically, the vehicle has a fixed upper-limit capacity.

The problem is to find a minimum distance route for the vehicle to satisfy all the delivery requirements without ever exceeding its capacity. Pickup and delivery problems are also studied with multiple vehicles. In those cases, it can be seen as a generalization of the Vehicle Routing Problem in which the vehicles are required to service customer requests for picking up and delivering transport loads.

2 Literature Review

2.1 Variations of Pickup and Delivery Problem

Pickup and delivery problems are studied in three main variants:

2.1.1 One commodity Pick Up and Delivery Problem

In this class of problems, only one product needs to be transferred from the pickup nodes to the delivery nodes. No unit of the product has a precise pickup or delivery location. A real-life example is when a bank company must move money between branch offices, some of them providing money and others requiring money; the main office (i.e., the vehicle depot) provides or collects the remaining money. Another example occurs when milk must be distributed from farms to factories by a capacitated vehicle, assuming that each factory is only interested in receiving a stipulated demand of milk but not in the providers of this demand. This variant was introduced by Hipolito et. al.[1]

2.1.2 Two commodity TSP with Pick up and Delivery

In this class of problems, the product collected from pickup customers differs from the product supplied to delivery customers. Therefore, the total amount of items collected from pickup customers must be delivered only to the depot, and there are different items going from the depot to the delivery customers. An application of this class of problems is the collection of empty bottles from customers for delivery to a warehouse and full bottles being delivered from the warehouse to the customers. These problems were introduced by Mosheiov et. al. [4]

2.1.3 Dial A Ride TSP

In this class of problems, there is a one-to-one correspondence between pickup customers and delivery customers, and each delivery customer must be visited only after the corresponding pickup customer has been visited. This is a particular case of TSP with Precedence Constraints.

An example is post office cargo, where cargo from one PO must be sent to another. Food delivery services are another example. The delivery agent must visit multiple restaurants and pick up orders which are needed to deliver to assigned customers. Dial-a-ride TSP is extremely popular and was introduced by David M Stein et. al. [6].

All these problems can be of single or multiple trucks, single or multiple depots, and combined with the vehicle routing problem. In the case of multiple trucks, problems can be further complicated by allowing for transshipment, i.e., cargo can be transferred from one truck to another.

2.2 Computational Complexity

Informally a decision problem (problems with exclusive yes or no answers) is in the NP complexity class if they have proofs verifiable in polynomial time by a deterministic Turing machine. NP-hard problems are the class of problems that are at least hard as the hardest problems in NP. Unless P = NP, these problems don't have an algorithm to solve them in polynomial time. Strongly NP-hard is those decision problems that remain NP-Hard even when all of their numerical parameters are bounded by a polynomial in the length of the input.

Even if the distances are small, the traveling salesman problem (TSP) is still an NP-hard problem. We have to try out all possible tours, which increase exponentially with the number of nodes. The pickup and delivery problem is at least as hard as the TSP, with even more constraints, such as precedence constraints. Hence it is a strongly NP-hard problem.

3 Problem Description

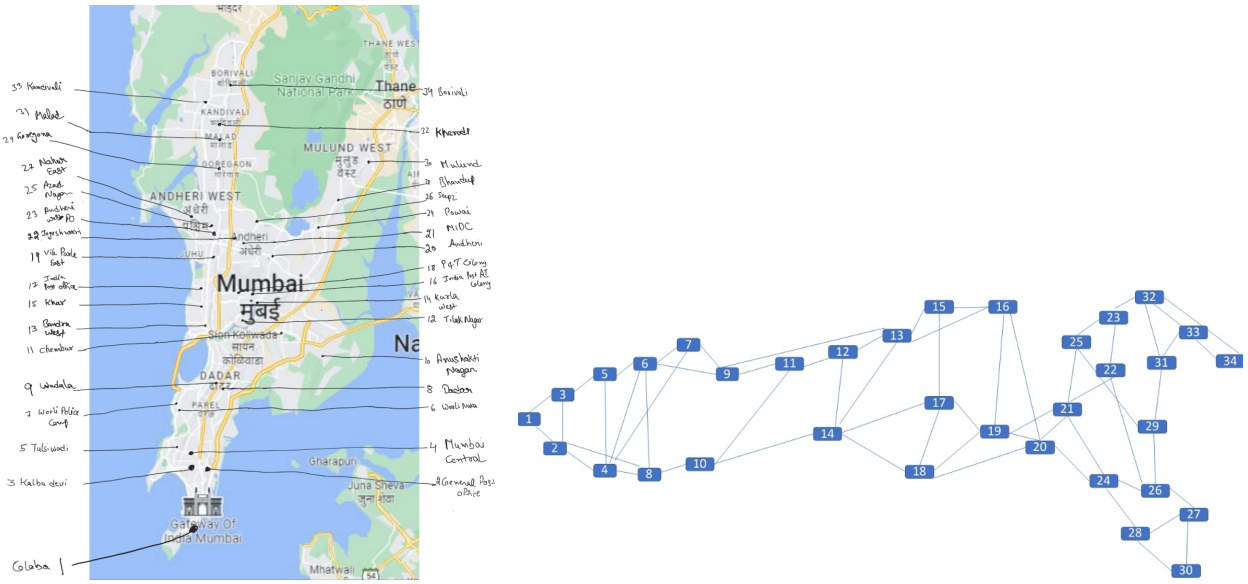
3.1 Dantzig Pickup and Delivery Services

We create a toy example for the application of the Pickup and Delivery Problem. Dantzig Pickup and Delivery Services is a pickup and delivery company situated at Colaba Mumbai. One of its regular orders is to distribute cargo between 34 post office locations in Mumbai as shown in fig (1. Cargo from pickup points needs to be offloaded to delivery points. We assign a mini-truck with a 600 kg payload capacity for the task. This is an instance of a single commodity pick-up and delivery problem. We solve this problem using Solver for three instances and compare the results with the standard TSP solution.

3.2 Formulation

For formulation, we take the following assumptions on our model:

- The truck has adequate fuel
- The product doesn't deteriorate
- It doesn't matter which time of the day the delivery depot receives the order
- Pickup and delivery at each node is less than the capacity of the truck (taken as 600 kg)



(a) Selected locations from Map (b) Generated Road Network from Map

Figure 1: Problem Map

Let variable $x_{i,j}$ be a binary variable denoting if the edge connecting i, j is part of the optimal solution or not. The number of these variables is equal to the total number of edges.

$$x = \begin{cases} 1 & \text{if edge } (i,j) \text{ is part of optimal solution} \\ 0 & \text{if edge } (i,j) \text{ is not part of optimal solution} \end{cases}$$

Another variable u will track the edge number. It starts from 1 and goes to the number of nodes. The number of these variables is equal to the total number of nodes.

Let $|N|$ denote the total number of nodes and $|E|$ denote the total number of edges.

$$u \in 1, 2, 3, ..., |N|$$

We take another variable f_a = load of the vehicle going through arc a.

$$f_a \in \mathbb{R}$$

Let $d_{i,j}$ denote distance between cities i and j

$$min \sum_{i=1}^{|E|} \sum_{j=1}^{|E|} d_{i,j} x_{i,j}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{i,k} = 1 \quad \forall k \in \{1, 2, 3...., |E|\}, i \neq j \tag{1}$$

$$\sum_{j=1}^{n} x_{k,j} = 1 \quad \forall k \in \{1, 2, 3...., |E|\}, i \neq j \tag{2}$$

$$u_1 = 1 \tag{3}$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{i,j}) \quad \forall (i,j) \in \{1, 2, 3...., |E|\}^2, i \neq j \tag{4}$$

$$u_i \geq 2 \quad \forall i \in \{2, 3...., |E|\} \tag{5}$$

$$u_i \leq |N| \quad \forall i \in \{2, 3...., |E|\} \tag{6}$$

$$\sum_{i=1}^{n} f_{i,k} - \sum_{j=1}^{n} f_{k,j} = q_i \quad \forall k \in \{1, 2, 3...., |E|\}, i \neq j \tag{7}$$

$$0 \leq f_a \leq Q x_{ij}, \quad \forall i, j; i \neq j \tag{8}$$

- *constraint*(1) denotes that from each node, there is one incoming edge.
- *constraint*(2) denotes that from each node, there is one outgoing edge.
- *constraint*(3) denotes when $x_{i,j} = 1$ then $u_j = u_i + 1$ ie. j^{th} edge will be labelled 1 more than previous label (i^{th}) label if edge $x_{i,j}$ is selected. This constraint denotes the ordering of each node in the optimal solution. This constraint is not considered for the last edge of the path, which connects the last node with the first node to create a cycle.
- *constraint*(4) denotes first node ordering is always 1
- *constraint*(5) and *constraint*(6) are bound constraints on u.
- *constraint*(7) and *constraint*(8) are demand constraints and bound constraints on f respectively.

4 Experiments

We solved the same toy problem as described 3. We have 34 locations across Mumbai, of which 17 are delivery points, and 16 are pickup points. We added the distances between the nodes by taking the length of direct routes between nodes using Google Map. We added a large number (100000) to denote edges corresponding to non-existing routes. If a solution includes such an edge, we consider solution to be infeasible.

This problem has 2346 variables and 4728 constraints. We use the formulation described in 3.2. We run the problem on a test road network as described in 1 with different demands of each node.

To model the problem, we use a Python package called pyomo [5]. It is an open-source Python-based optimization modeling language and allows users to model and solve complex optimization problems. After writing the abstract representation of the optimization model in pyomo it is passed to a solver interface that communicates with the solver. The solver interface in pyomo is responsible for translating the abstract model representation into a format the solver can understand and return the solution.

To solve the problem, we use two open-source solvers CBC (Coin-or branch and cut) [2] and GLPK (GNU Linear Programming Kit) [3]. These two solvers are supported by pyomo solver interface. Both GLPK and CBC are open-source mixed integer linear programming solvers written in C++. It uses other parts of the COIN-OR repository, mainly Linear Programming (LP) solver (CLP) and Cut Generation Library (CGL) for generating cuts. CBC uses Branch and Cut Algorithm and some heuristics to obtain valid solutions quickly. GLPK is written in C and is intended for solving large-scale LP and Mixed integer programs (MIP). It includes the branch-and-cut method, primal and dual simplex methods, and primal-dual interior-point methods.

For some chosen demand values of a node, our solution matches the optimal TSP path. Also, the time to solve the problem also depends on demands. It takes less than 1 min for some demand values to get to the optimal solution, and for some other demand values, it takes around 30 min.

4.1 Comparison of Solver Time

Table 1 shows the result of time taken and the number of sub-problems explored by the Branch and bound tree produced by the CBC and GLPK, respectively. Both solvers explored the same number of sub-problems, but GLPK beats CBC by 5 sec.

Solver	Time Taken	Number of sub-problems
CBC	1min 6s	8889
GLPK	1min 1s	8889

Table 1: Comparison of time taken by CBC and GLPK

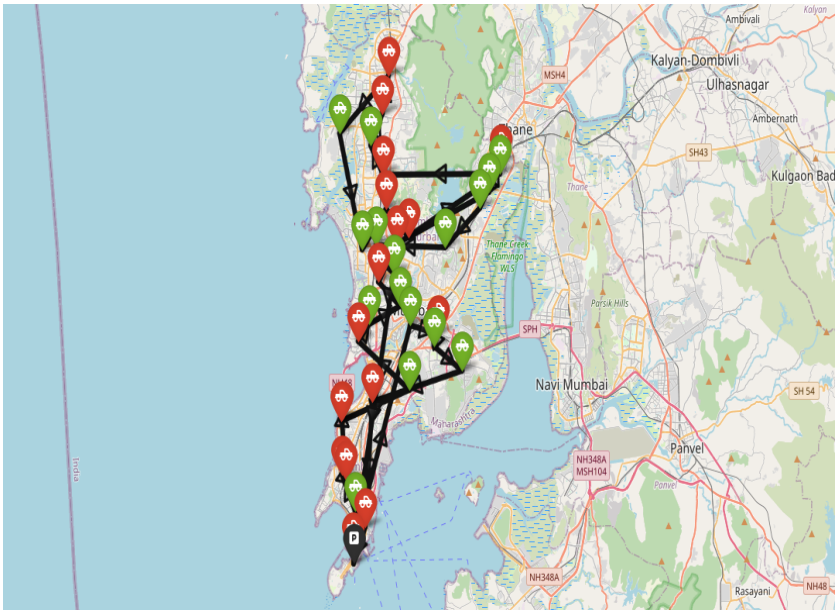


Figure 2: Solution Image of Instance 1

4.2 Problem Solution

Fig (2) shows the solution for one of the problem instances.

Optimal Path Sequence:
Colaba ⟶ Kalbadevi ⟶ Tulsiwadi ⟶ Mumbai Central ⟶ Worli Naka ⟶ Worli Police Camp ⟶ Wadala ⟶ Bandra West ⟶ Khar West Mumbai ⟶ Air India Colony Mumbai ⟶ Vile Parle East Mumbai ⟶ MIDC Mumbai ⟶ Azad Nagar Mumbai ⟶ Goregaon Mumbai ⟶ Malad West Mumbai ⟶ Kandivali Mumbai ⟶ Borivali Mumbai ⟶ Kharodi Mumbai ⟶ Andheri West ⟶ Jogeshwari East Mumbai ⟶ Seepz Mumbai ⟶ Nahur East ⟶ Mulund ⟶ Bhandup ⟶ Powai ⟶ Andheri Mumbai ⟶ P&T Colony Mumbai ⟶ India Post Office Mumbai ⟶ Kurla West ⟶ Tilak Nagar ⟶ Chembur ⟶ Anushakti Nagar ⟶ Dadar ⟶ Mumbai GPO ⟶ Colaba

Similarly (3, 4) shows the solution for instances 2 and 3.

We have also provided the above results as interactive maps, and they can be accessed by clicking on the following links:

- Result 1
- Result 2
- Result 3

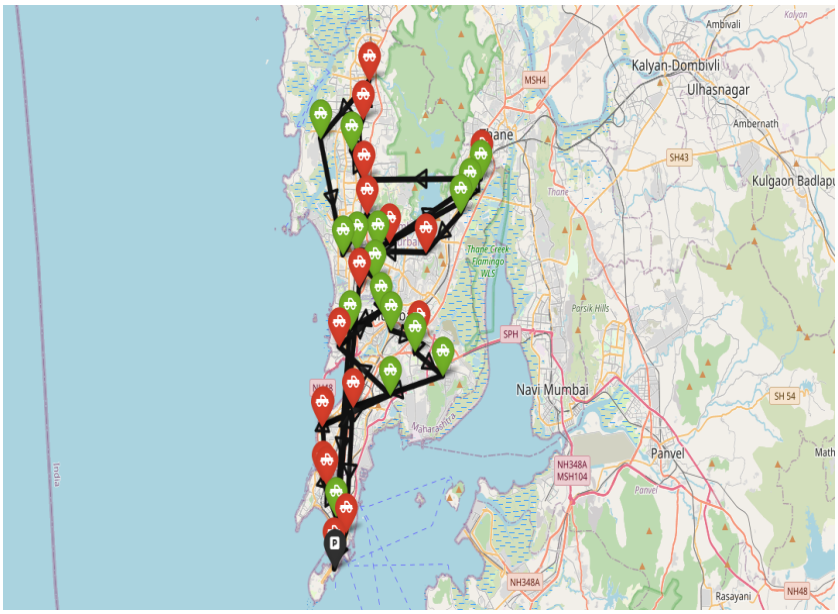


Figure 3: Solution Image of Instance 2

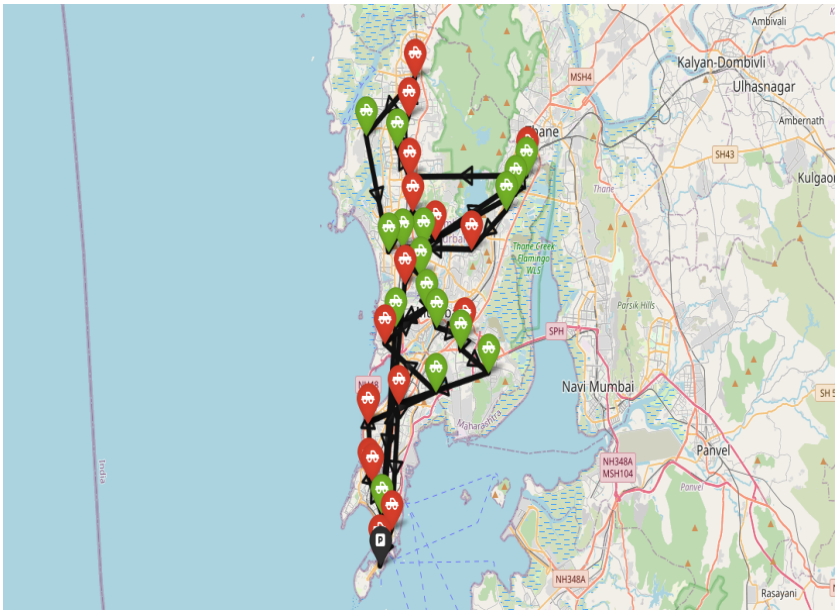


Figure 4: Solution Image of Instance 3

5 Conclusion

For three of the four data sets: cities_demands, cities_demands_2, cities_demands_3 ;the problem is efficiently solved with exact optimal solution. For cities_demands dataset, optimal solution coincided with optimal TSP solution. For the data set cities_demands_4, there was no feasible pickup and delivery route.

For infeasible solution, it included an edge corresponding non-existing routes (i.e. the large number). A practical solution can be interpreted this edge as taking the shortest path between the adjacent nodes of the edge. Few paths are visited more than once. Another solution can be by replace the distances between non-adjacent nodes with the length of the shortest path between those nodes.

6 Appendix

6.1 Description of Files uploaded

We have uploaded a **.zip** file containing all the code files and all necessary information on installation and running the code in a file named **README.pdf**. All data files used in the experiments are inside the **data** folder. A Python notebook named **pdtsp-1.ipynb** is also provided. The code for this project can be found in github.

References

[1] Hipólito Hernández-Pérez and Juan-José Salazar-González. The one-commodity pickup-and-delivery travelling salesman problem. In *Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 89–104. Springer, 2003.

[2] Robin Lougee-Heimer John Forrest. Cbc user guide. *https://www.coin-or.org/Cbc/ch01s04.html*.

[3] Andrew Makhorin. About glpk. *https://www.gnu.org/software/glpk/*.

[4] Gur Mosheiov. The traveling salesman problem with pick-up and delivery. *European Journal of Operational Research*, 79(2):299–310, 1994.

[5] pyomo org. About pyomo. *http://www.pyomo.org/about*.

[6] David M Stein. Scheduling dial-a-ride transportation systems. *Transportation Science*, 12(3):232–249, 1978.