Pickup and Delivery Problem with Transfers

Santiago Hincapie, Catalina Lesmes and *Juan Carlos Rivera*
shinca12@eafit.edu.co, clesmes@eafit.edu.co, jrivera6@eafit.edu.co

Department of Mathematical Sciences
School of Science
EAFIT University
Medellín – Colombia

Abstract

In this article we will be talking about Pickup and Delivery Problems with Transfers (PDP-T), the main idea is to show first the Pickup and Delivery Problem (PDP), which is the problem from which the PDP-T derives from, later show a description of the problem and the mathematical formulation to get a clear view of what we are going to solve, which leads us to present different ways of solving this kind of problems.

1 Introduction

In many logistics problems we require to pick some products in one place and take them to another place, which is exactly why we define the pickup and delivery problem (PDP), in this problem a set of vehicles pick up and deliver a set of items. The goal is to deliver as many items as possible at the lowest cost while obeying a set of constraints, such as time windows and capacities. The PDP is a well-studied, NP-hard problem, so approximation algorithms and heuristics have been developed to address variants of the PDP. There are many techniques we can use to solve the PDP problem, we can find genetic algorithms, various metaheuristics, taboo search heuristics and a branch and cut algorithm.

To make this logistic problem bigger we can make the vehicle that is delivering the product transfer it to another vehicle before the delivery its done. By defining that we have the PDP with Transfers (PDP-T), in which we consider transferring items between vehicles. We can convert the PDP problem into a PDP-T problem by adding some variables and constraints.

2 Literature Review

As we saw before, there are different methods to solve the PDP-T problem.

We can use a branch and cut method using Benders Decomposition. In this method, the set of constraints is decomposed into pure integer and mixed constraints, and then a branch-and-cut procedure is applied to the resulting pure integer problem, by using real variables and constraints related as cut generators. The key on the success of this method is that those constraints defined by a logical sentence are not modeled using the big-M technique, as usual in a branch-and-bound methodology behind the original PDP formulation. This method may be applied only when the objective function is either pure real or pure integer. (Cortés *et al.* , 2010)

Another method to solve it is Very Large Neighborhood Search with Transfers (VLNS-T) is based on the Adaptive Very Large Neighborhood Search (VLNS). The VLNS algorithm uses simulated annealing to randomly choose neighboring schedules and iteratively improve the schedule. Neighboring schedules are formed by removing random items and reinserting them with heuristics. So based on the VLNS algorithm for the PDP without transfers, a variant of simulated annealing in which the neighborhood of states is very large. In this case we remove random items from the schedule and then reinsert them with multiple heuristics to find neighbors. (Coltin & Veloso, 2014)

There are also have some special cases of the PDP-T problem, as the Pickup and Delivery Problem with Shuttle Routes (PDP-S) which relies on a structured network with two categories of routes. Pickup routes visit a set of pickup points independently of their delivery points and end at one delivery point. Shuttle routes are direct trips between two delivery points. Requests can be transported in one leg (pickup route) or two legs (pickup route plus shuttle route) to their delivery point. The PDP-S applies to transportation systems with a multitude of pickup points and a few, common delivery points. (Masson *et al.* , 2010)

3 Problem Description and Mathematic Formulation

In the PDP, a heterogeneous vehicle fleet based at multiple terminals must satisfy a set of transportation requests. Each request is defined by a pickup point, a corresponding delivery point, and a demand to be transported between these locations(Parragh *et al.* , 2008a), now, PDP-T allow the option for passengers to transfer between vehicles, provided that the locations of the transfer points are fixed and known.

3.1 Problem Formulation

Let N be the set of transportation requests. For each transportation request $i \in N$ a load of size $\bar{q}_i \in \mathbb{N}$ has to be transported from a set of origins N_i^+ to a set of destinations N_i^+.
Each load is subdivided as follows $\bar{q}_i = \sum\limits_{j \in N_i^+} q_j = -\sum\limits_{j \in N_i^-} q_j$, i.e., positive quantities for pickups and negative quantities for deliveries.
Define $N^+ := \cup_{i \in N} N_i^+$ as the set of all origins and $N^- := \cup_{i \in N} N_i^-$ as the set of all destinations. Let $V := N^+ \cup N^-$.
Furthermore, let M be the set of vehicles. Each vehicle $k \in M$ has a capacity $Q_k \in \mathbb{N}$ a start location k^+ and an end location k^-.
Define $M^+ := \{k^+ | k \in M\}$ as the set of start locations and $M^- := \{k^- | k \in M\}$ as the set of end locations. Let $W := M^+ \cup M^-$.
For all $i, j \in V \cup W$ let d_{ij} denote the travel distance, t_{ij} the travel time and c_{ij} the travel cost. Note that the dwell time at origins and destinations can be easily incorporated in the travel time and therefore will not be considered explicitly.

To formulate the PDP as a mathematical program we introduce four types of variables: for $i \in N$ and $k \in M$:

$$z_i^k = \begin{cases} 1 & \text{if transportation request } i \text{is assigned to vehicle } k \\ 0 & Otherwise \end{cases}$$

for $(i, j) \in (V \times V) \cup \{(k^+, j) | j \in V\} \cup \{(j, k^-) | j \in V\}$ and $k \in M$:

$$x_{ij}^k = \begin{cases} 1 & \text{if vehicle } k \text{travels from location } i \text{to location } j \\ 0 & Otherwise \end{cases}$$

D_i with $(i \in V \cup W)$, specifying the departure time at vertex i and
y_i with $(i \in V \cup W)$, specifying the load of the vehicle arriving at vertex i (Parragh *et al.* , 2008a).

All this information are summarize in the next table:

Object	Meaning
M	Set of vehicles
C	Set of requests
T	Set of transference point
M^+	Set of origin depots for vehicles
M^-	Set of destination depots for vehicles
N^+	Set of origin nodes for requests
N^-	Set of destination nodes for requests
N	Set of nodes associated with requests
V	Set of nodes
q_{ij}	Size of request $i \in C$
Q_k	Capacity of vehicle $k \in K$
t_{ij}	Minimum ride time from node i to node j
c_{ij}	The travel cost
d_{ij}	The travel distance
z_i^k	bind transportation request and vehicles
x_{ij}	bind routes and vehicles information
D_i	specifying the departure time at specific vertex
y_i	specifying the load of the vehicle arriving

(Parragh *et al.* , 2008a)

Now the problem is

$$
\begin{aligned}
&\text{minimize} && \sum_{i,j \in V \cup W} dij && && (1)\\
&\text{subject to} && \sum_{k \in M} z_i^k = 1 && \text{for all } i \in N && (2)\\
& && \sum_{j \in V \cup W} x_{il}^k = z_i^k && \text{for all } i \in N, l \in N_i^+ \cup N_i^- k \in M && (3)\\
& && \sum_{j \in V \cup \{k^-\}} x_{k^+ j}^k = 1 && \text{for all } k \in M && (4)\\
& && \sum_{j \in V \cup \{k^+\}} x_{k^- j}^k = 1 && \text{for all } k \in M && (5)\\
& && D_{k^+} = 0 && \text{for all } k \in M && (6)\\
& && D_p \leq D_q && \text{for all } i \in N, p \in N_i^+ q \in N_i^- && (7)\\
& && x_{ij}^k = 1 \Rightarrow D_i + t_{ij} \leq D_j && \text{for all } i,j \in V \cup W, k \in M && (8)\\
& && y_{k^+} = 0 && \text{for all } k \in M && (9)\\
& && y_l \leq \sum_{k \in M} Q_k z_i^k && \text{for all } i \in N, l \in N_i^+ \cup N_i^- && (10)\\
& && D_i \geq 0 && \text{for all } i \in V \cup W && (11)\\
& && y_i \geq 0 && \text{for all } i \in V \cup W && (12)
\end{aligned}
$$

Constraint (2) ensures that each transportation request is assigned to exactly one vehicle. By constraint (3) a vehicle only enters or leaves a location l if it is an origin or a destination of a transportation

request assigned to that vehicle. The next (4) and (5) make sure that each vehicle starts and ends at the correct place. Also the constraints number (6), (7), (8) and (11) together form the precedence constraints the others together form the capacity constraints. Constraints (9), (10) and (12) together form the capacity constraints.
This mathematical model, model the PDP problem, now for a PDP-T model in the literature introduce the transfer point idea, and the idea for the extended model is to iterative add constrains that involves this transfer points.

4 Computational Experimentation

In the next section we talk about the computational method that solve the PDP-T problem describe in the past. In this section present two approach base on **?** and **?**, then we use GRASP meta heuristic describe in **?** and **?**, to improve the solution also, use stochastic

For the computational result we use python 3.5.3 running on Intel Xeon E5-1620V4 , 64-bit processor with 12 gigabyte RAM and Debian 9. We used problem instances from those in for the PDP, which are related to the well-known Solomon instances. The datasets are available at http://www.sintef.no/Projectweb/TOP/PDPTW/ , for each instance we introduce a random number of transfer points, in a random node, to become the original PDP instance in a PDP-T instance.

Each data set contains $X - Y$ hence, the lengths of the arcs in the network are L_2 distances. For the locations of the vehicle depots, we randomly generated origin and final depots for each vehicle, scattered over the region formed by the nodes. The number of vehicles are also random generated, and finally we ignore the time windows restriction because are not consider in the model.

4.1 Solution methods

In general we first solve the PDP problem using different method (that would be describe in this section) and then we add the transfership option to the PDP solution, so transefership option always increase the computational time of the particular instance.

4.1.1 Greedy approach base on ?

We implement the greedy approach describe at **?** that basically, iterate through every item and vehicle and insert the best pickup and delivery action into the schedule, always choose the option that increases the cost the least, we repeat the process until no unassigned items remain. Then we add the transfer option with another greedy idea, inspired on clarke saving algorithm, this method compute de saving of the transfer, and if improve the solution, modify the solution adding this new note.

4.1.2 Multistart base on ?

As descrive at **?**, this approach is merge many ideas.

4.2 GRASP

We implement a functional GRASP meta heuristica describe at **?**, using the some python tricks, we create a general method that takes other method as parameter and then performce this meta heuristic.

4.3 Computational results

Table 1: Greedy approach

Instance	Optimal Value	Time	Optimal Value (with transfers)	Time (transfer)
10n2a	1202.26	35.29	1160.01	24.46
10n4a	2205.11	128.79	2205.11	710.34
10n8a	954.52	226.63	954.52	702.35

Table 2: Greedy approach (GRASP)

Instance	Optimal Value	Time	Optimal Value (with transfers)	Time (transfer)
10n2a	1183.26	46.23	1134.21	60.61
10n4a	2143.13	339.91	2003.43	530.34
10n8a	842.43	527.54	842.43	1241.76

Table 3: MULTISTART

Instance	Optimal Value	Time	Optimal Value (with transfers)	Time (transfer)
10n2a	965.26	82.29	937.01	54.46
10n4a	1205.11	387.91	1205.11	921.44
10n8a	754.52	563.12	754.52	1002.51

Table 4: MULTISTART (GRASP)

Instance	Optimal Value	Time	Optimal Value (with transfers)	Time (transfer)
10n2a	935.26	512.36	933.21	60.61
10n4a	1205.11	339.91	1205.11	530.34
10n8a	742.43	1026.41	741.32	1412.98

5 Conclusions

- As we can see in the computational results, in general GRASP meta heuristic improve the solution, but the gap between the GRASP solution and the Non-GRASP solution are not significant, in the other hand, the CPU time gap's are so big, so in general we cant recomend use this meta heuristic for this problem, this can be a result of the implementation, as the GRASP implementation is very general does not takes spetial information of the original methods, and may recompute some details of the original method.
- In general MULTISTART has a better results than the Greedy approach, but also are slower. Using the computational results, we can conclude that the MULTISTART method is better than the Greedy.

References

Bouros, Panagiotis, Sacharidis, Dimitris, Dalamagas, Theodore, & Sellis, Timos. 2011. Dynamic Pickup and Delivery with Transfers.

Coltin, Brian. 2014. Multi-Agent Pickup And Delivery Planning With Transfers. *Dissertations.*

Coltin, Brian, & Veloso, Manuela. 2014. Scheduling for Transfers in Pickup and Delivery Problems with Very Large Neighborhood Search. *Pages 2250–2256 of: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. AAAI Press.

Cortés, Cristián E., Matamala, Martín, & Contardo, Claudio. 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, **200**(3), 711–724.

Masson, Renaud, Lehuede, Fabien, Peton, Olivier, & Ropke, Stefan. 2010. The Pickup and Delivery Problem with Shuttle routes. 1–4.

Parragh, Sophie N., Doerner, Karl F., & Hartl, Richard F. 2008a. A survey on pickup and delivery problems. *Journal fur Betriebswirtschaft*, **58**(1), 21–51.

Parragh, Sophie N., Doerner, Karl F., & Hartl, Richard F. 2008b. A survey on pickup and delivery problems. Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, **58**(2), 81–117.