A MULTI-VEHICLE PROFIT-MAXIMIZING PICKUP AND DELIVERY SELECTION PROBLEM WITH TIME WINDOWS

Research in Progress

Qiu, Xiaoqiu, University of Freiburg, Freiburg im Breisgau, Germany, xiaoqiu.qiu@is.uni-freiburg.de
Feuerriegel, Stefan, University of Freiburg, Freiburg im Breisgau, Germany, stefan.feuerriegel@is.uni-freiburg.de

Abstract

Today the transportation market is highly competitive yet asymmetric. Large carriers enjoy market power from economies of scale and better networking. Small carriers rely on horizontal cooperation between each other to compete with, usually by trading on freight exchanges. One major challenge for small carriers is how to select requests so that the total profit from the delivery can be maximized. Thus in this paper we model it as a Profitable Pickup and Delivery Selection Problem, which is a generalization of the classic Pickup and Delivery Problem, but with a goal of profit maximization and it does not require all requests to be delivered. For a solution, we propose a simple graph search that traverses through feasible routes to solve the problem optimally. Preliminary computer experiments show that random Euclidean instances with up to 500 requests can be solved optimally within seconds in a single-vehicle setup, and up to 50 requests and 4 vehicles in minutes for the multi-vehicle cases, both of which are faster than the solution by the GUROBI Optimizer. For multi-vehicle cases, we also devise a greedy heuristic that is able to find a 99% optimal solution in less than a second.

Keywords: Transportation, Vehicle Routing, Pickup and Delivery, Selective Pickup, Decision Support.
1 Introduction

Road freight transportation is a highly competitive market in which large carriers are more competitive because of their ability to leverage economies of scale, as well as better planning tools. Small carriers, in contrast, seek to establish horizontal cooperation to achieve a matching operation efficiency. The cooperation ranges from coupling the line haul and backhaul traffics in roundtrip transportations (Ihde, 2003), to “split-load” the transportation requests so that carriers can extend their reach beyond their own fleets (Krajewska et al., 2007, Gulczynski et al., 2010), and to utilize freight exchanges for trading redundant requests and capacities. Currently, there are more than 100 online transportation markets in Europe alone (Brummifreund, 2014), among which the largest exchanges trade over a million tons of freights daily and have over 70,000 registered users (e.g. Teleroute, 2014).

Traditionally, operational research has focused on optimizing routes for large carriers: to minimize the total costs of delivering all requests. The problem roots from the classic Traveling Salesman Problem (TSP) and evolves into many variations, e.g. the famous Vehicle Routing Problem (VRP) and the Pickup and Delivery Problem (PDP). Most variations can be framed as a routing problem, with the aim of identifying the best feasible route that meets the constraints. Most problems in this family have been studied extensively over the years, and the achievements are reflected in the decreasing proportion of empty trucks on European roads from 22.6% in 2007 to 21.4% in 2012 (Eurostat, 2013).

For carriers that form explicit coalitions, similar optimization problems to large carriers are present. Carriers trading on the freight exchanges, however, face very different challenges: since they can select which request to accept and deliver, the goal changes into maximizing the total profit of delivering a selective subset of the available requests. The problem is no longer one of pure routing, since increasing route length may also lead to additional gains.

Unfortunately, most transportation markets today remain as bulletin boards, where results are simply geographically filtered and users need to manually evaluate the profitability of each request. Thus in this paper, we aim to find a solution that combines request selection, routing and evaluation in a multi-vehicle setup for less-than-truckload (LTL) transportations. The result is a simple graph search method that does not require commercial solvers. We also develop a greedy heuristic based on the graph search to enhance solution speed for multi-vehicle problems. Preliminary experiments show that the graph search finds the optimal solution faster than the GUROBI Optimizer (Gurobi, 2014), especially in single-vehicle cases, while the greedy heuristic is highly efficient in multi-vehicle cases.

The remainder of this paper is structured as follows: Section 2 surveys the related work by comparing three aspects of the problem: pickup and delivery, routing, and selection. Section 3 defines the model and its assumptions. The solutions are presented in section 4, with computation experiments and evaluations in section 5. Section 6 concludes with an outlook for future research.

2 Related Work

Transportation optimization problems are usually modelled as graph problems, in which routes are represented by edges and customers are represented by nodes. Many transportation problems originate from the Traveling Salesman Problem (TSP), whose aim is to find the shortest path that visits each city (node) exactly once. The Vehicle Routing Problem (VRP) generalizes the TSP by introducing a common depot and demands at each customer node (Toth and Vigo, 2002). Besides practical constraints like capacity and time limits, the key constraint in VRP (and TSP) is route feasibility where the final solution must be interconnected routes that visit each node exactly once, and the vehicles must start and end at the central depot. The goal is to find a route that satisfies the connectivity constraint and minimizes route costs.
The Pickup and Delivery Problem (PDP) extends the VRP with pickup and delivery requirements (Parragh et al., 2008). PDP is a vast family with many different setups, some focus on the cases of redistributing certain commodities between locations, in which any pickup can serve any deliveries (Ting and Liao, 2013). In this paper, we focus on the one-to-one case (Berbeglia et al., 2007), where each transportation request is unique and specified by a pickup location and delivery location. The key constraint is thus the precedence requirement that all requests must be picked-up before delivery, and each pickups must be delivered to its corresponding destination. This makes certain TSP techniques less powerful, e.g. the k-opt local search which swaps k sub-tours in a route, because swapping sub-tours easily breaks the precedence of the nodes. However, precedence is necessary for modelling applications like less-than-truckload (LTL) transportation or human transportation (e.g. Dial-a-Ride Problem, Cordeau and Laporte, 2003). The goal continues to be cost minimization, often with the additional requirement of delivering all requests.

Computationally, both VRP and PDP are NP-hard. A common approach is to model a mixed-integer problem, solve its Linear Programming (LP) relaxations, and then use techniques like Branch-and-Price or Branch-and-Cut to identify feasible integer solutions, i.e. feasible routes. The LP relaxation is often solved by commercial solvers such as CPLEX (IBM), and accelerated through valid inequalities (Parragh et al., 2008). The state-of-the-art solution for PDP is capable of solving instances of up to 500 requests optimally (Ropke and Cordeau, 2009).

Profit maximization with selective pickup resembles the classic Knapsack problem, where the goal is to find not all but a selective subset of requests under capacity and time constraints, so that the total values are maximized.

The classification of the problems is summarized in Figure 1. The intersection of selective pickup and precedence in delivery models the bid optimization problem faced by carriers operating on fixed lanes, where no routing is involved but carriers still need to consider matching requests’ precedence such that the round trip yields maximized profit. The intersection of selective pickup and routing includes selective VRP (Gribkovskaia et al., 2008), the Price-collecting Rural Postman Problem (Araoz et al., 2009), the Orienteering Problem (Vansteenwegen et al., 2011), the Attractive TSP (Erdoğan et al., 2010), and many more, among which Erdoğan et al. (2010) solve instances of up to 400 nodes optimally. Note that precedence in delivery is not required in these problems.

Our Profitable Pickup and Delivery Selection Problem (PPDSP) lies in the intersection of the three challenges: feasible routing, precedence in pickup and delivery, and profit-maximizing selection. Within the domain of supporting carriers trading on freight exchanges, several studies have looked into similar problems: Song and Regan (2003) propose an approximation algorithm which searches for all operationally feasible routes, and then uses branch-and-bound to form optimal bids. A similar problem with up to 100 requests is solved by Schönberger and Kopfer (2005) using a memetic algorithm.
In this paper, we elaborate on the idea of searching through operationally feasible routes and develop a simple graph search method for addressing PPDSP. As a comparison, we also implement the model with the GUROBI Optimizer (Gurobi, 2014). The graph search is able to obtain the same optimal solutions as GUROBI, but over 100 times faster for single-vehicle instances. Furthermore, we develop a graph-search-based greedy heuristic for the multi-vehicle case due to the increased difficulty.

3 The Profitable Pickup and Delivery Selection Problem

In this section, we first list the assumptions of the Profitable Pickup and Delivery Selection Problem (PPDSP), and then introduce the notation to define PPDSP in a mixed-integer model.

3.1 Assumptions

The following assumptions are applied throughout the model:

- **Independent transportation requests**: Every request’s revenue depends solely on its own attributes irrespective of the attributes of others. There is no precedence requirement between requests.

- **Profit maximizing carriers with selective pickup**: The carrier only takes a transportation request if it generates additional profit, and there is no obligation to deliver all requests, which is often a requirement in the classic VRP/PDP studies.

- **Flexible routing**: The requests are not route-dependent. The carriers are free to choose any route as long as the requests are delivered to their destinations within the time windows. As a result, the solution must not only specify which requests to take, but must also identify a cost-efficient route.

- **Less-than-truckload (LTL) shipping with capacitated vehicles**: LTL carriers mix freights in the same vehicle to optimize the utilization of the vehicle capacity. In comparison, truckload shipping considers the whole container or trailer as a basic unit. LTL is more complicated because capacity constraints must be taken into account when consolidating different freights into the same route.

- **Time windows and bounded total ride time**: Each request has a pickup node and a delivery node, both of which are constrained by time windows. Thus, in addition to selection and routing, the carrier must also consider the duration waiting at each stop on the route. Since most countries have regulations on the maximum working hours for drivers, the total ride time is bounded as well.

3.2 Mixed-Integer Formulation of PPDSP

Our model is based on the PDPTW in Ropke and Cordeau (2009) for its clarity and similarity. The objective function is changed to profit maximization, and selective pickup is incorporated by relaxing the constraint on full delivery of all requests.

The decision variable \( x_{ij}^k \) equals 1 if vehicle \( k \) goes from node \( i \) to node \( j \). A total of \( n \) available requests are formed on a directed graph \( G = (V, E) \) whose node set \( V \) is divided into pickup nodes \( P = \{1, 2, ..., n\} \), delivery nodes \( D = \{n + 1, n + 2, ..., 2n\} \), and depots \( \{0, 2n + 1\} \). All routes start from depot 0 and end at \( 2n + 1 \). Each request is specified by its pickup node \( i \) and delivery node \( i + n \). Note that nodes refer to requests rather than geographic locations. Thus different requests must have different nodes, albeit they can share common pickup and delivery locations.

Delivering request \( i \) generates revenue \( \pi_i \) for \( i \in P \). For \( i, j \in V \), let \( c_{ij}^k \) denote the travel cost from node \( i \) to node \( j \), and \( t_{ij}^k \) the corresponding travel time. Let \( u_i^k \) denote the exact time when vehicle \( k \) leaves \( i \), and \( d_i \) the time required for loading the freight. Let \( w_i^k \) be the capacity usage after \( i \), and \( q_i \) the change in capacity at \( i \). Finally, each vehicle \( k \) has a capacity of \( Q^k \), and a total ride-time constraint of \( T \). The optimization model is as follows:
\[
\max_{x_{ij}} \sum_{k \in K} \left( \sum_{i \in P} \sum_{j \in V} x_{ij}^k \right) - \sum_{k \in K} \sum_{i \in P} \sum_{j \in V} c_{ij}^k x_{ij}^k
\]

subject to

\[
\sum_{i \in V} x_{oi}^k = \sum_{i \in V} x_{i,2n+1}^k = 1 \quad \forall k \in K
\]

\[
\sum_{k \in K} \sum_{j \in V} x_{ij}^k \leq 1 \quad \forall i \in P, k \in K
\]

\[
\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{i+n,j}^k = 0 \quad \forall i \in P, k \in K
\]

\[
\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{ji}^k = 0 \quad \forall k \in K
\]

\[
x_{ij}^k \in \{0,1\} \quad \forall i,j \in V \quad \forall k \in K
\]

\[
u_j^k > (u_i^k + t_i^j + d_i^k)x_{ij}^k \quad \forall i,j \in V, k \in K
\]

\[e_i \leq u_i^k \leq l_i \quad \forall i \in V, k \in K
\]

\[u_i^k + t_i^j \leq u_i^{k+n} \quad \forall i,j \in V, k \in K
\]

\[u_{2n+1}^k - u_i^k \leq T \quad \forall k \in K
\]

\[w_j^k \geq (w_i^k + q_j)x_{ij}^k \quad \forall i,j \in V
\]

\[
\max\{0, q_i\} \leq w_i^k \leq \min\{Q^k, Q^k + q_i\} \quad \forall i \in V, k \in K
\]

The objective function (1) consists of two parts. The first half is the revenue: when vehicle \(k\) picks up request \(i\), the summation \(\sum_{j \in V} x_{ij}^k\) equals one, and yields revenue \(\pi_i\). As for the second half, route cost \(c_{ij}^k\) is incurred when edge \((i,j)\) is traversed by vehicle \(k\).

Constraints (2) to (6) are route connectivity and precedence constraints: constraint (2) ensures the start and end at the depots. Constraint (3) forbids repeated delivery of the same request. Constraint (4) ensures all pickups are delivered and by the same vehicle, and constraint (5) ensures all vehicles leave all non-depot nodes. Constraint (6) is the binary constraint for the decision variable \(x_{ij}^k\).

Capacity and time constraints are captured in constraints (7) to (12): constraint (7) defines the service time from \(i\) to \(j\): departure time at \(j\) must be later than the departure time at \(i\), plus the travel and loading time. Waiting time is implicitly given by the difference between the left hand side and right hand side of the inequality, since the right hand side is essentially the arrival time at \(j\), if \(x_{ij}^k = 1\). Constraint (8) ensures pickup and delivery within the time windows, (9) guarantees pickup before delivery, and (10) enforces a maximum ride-time. Constraints (8) — (10) together eliminate sub-tours. Constraint (11) defines the load change from \(i\) to \(j\), and (12) is the capacity constraint so that if \(i\) is a pickup node, then the load must be at least the capacity of request \(i\), and if \(i\) is a delivery node then the final load must be less than \(Q + q_i\) (note \(q_i < 0\) for delivery nodes).

4 Solution Methods

In LTL pickup and delivery problems, there are two common approaches to represent routes. The first is the hierarchical method, which splits the problem into routing geographic locations and optimizing the pickup and delivery overlay. Since routing and selection are intertwined aspects of the PPDSP, we select the sequential method, which labels each request with two nodes (Moura and Oliveira, 2009). Now delivery plans, as well as routing, can be specified in a single sequence with the drawback of a large number of decision variables. For \(n\) requests and \(k\) vehicles, it requires \(k(2n + 2)^2\) decision variables \(x_{ij}^k\), most of which are infeasible routes.

Following the idea of enumerating feasible routes in Song and Regan (2003), we develop a simple graph search method for searching feasible routes in the LTL pickup and delivery problems. The basic
idea of the method is introduced in Section 4.1, with extensions to the multi-vehicle case and the accompanying greedy heuristic in Section 4.2.

### 4.1 Simple Graph Search for Single Vehicle PPDSP

Most combinations of values in the $k(2n + 2)^2$ decision variables are infeasible. In the LTL setup, a feasible route must not only visit each node only once and match the time windows, but also adhere to the following requirements:

- Routes must contain no disconnected sub-tours, and must start and end at depots.
- Pickup nodes must be visited before the corresponding deliveries (subject to capacity limit).
- Pickup nodes without deliveries or delivery nodes without pickups are excluded.

These restrict the vehicle driver’s choice at each node to one of the following three:

- Pick up a new request, and prepare for its delivery.
- Deliver an already picked-up request.
- Vehicle empty and return to depot.

We program the choices in a depth-first search algorithm that traverses recursively on each of the three moves. Before each branching, the feasibility of the potential route with respect to capacity and time constraints is evaluated. In addition, the unvisited new pickup nodes are evaluated for their revenues, and compared to the current best route. If the profit from the visited nodes plus all the revenues, minus a reasonable cost to reach the unvisited nodes, is less than that of the current best route, we can infer that the potential route cannot be optimal, even if all the remaining requests were delivered. In this case, we stop and return to the last node. This provides a simple bounding to the graph search.

Let $P$ be the set of new requests to pickup, $D$ be the set of onboard requests to deliver, and $R$ the visited nodes. The sets $P$, $D$ and $R$ are implemented as sets by default. Set $D$ can be changed into a stack or list implementation to incorporate specific delivery requirements, such as Last-In-First-Out (LIFO) or First-In-First-Out (FIFO) loading (Carrabs et al., 2007). The graph search can be described in pseudo codes as follows:

---

<table>
<thead>
<tr>
<th>Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \leftarrow {1, \ldots , n}$</td>
</tr>
<tr>
<td>$R \leftarrow {0}, D \leftarrow \emptyset$</td>
</tr>
</tbody>
</table>

**GraphSearch** ($P, R, D$)

if time or capacity constraint violated in $R$

or potential revenue gain too small in $P$ then

return

end

for each $i \in P \cup D$

if $i \in P$ then

$P \leftarrow P \setminus \{i\}$

$R \leftarrow R \cup \{i\}$

$D \leftarrow D \cup \{i + n\}$

Call **GraphSearch** ($P, R, D$)

else

$D \leftarrow D \setminus \{i\}$

$R \leftarrow R \cup \{i\}$

Call **GraphSearch** ($P, R, D$)

$D \leftarrow D \cup \{i\}$

$R \leftarrow R \setminus \{i\}$

end

if $D$ is empty then

$R \leftarrow R \cup \{2n + 1\}$

Evaluate $R$, update the current best route

$R \leftarrow R \setminus \{2n + 1\}$

end

return

---
4.2 Extensions to the Multi-Vehicle PPDSP

A solution with $k$ vehicles can be modelled as a route that starts and returns to the depot $k$ times. Thus by simply adding an additional branching after updating the current best route, the problem can be extended to the multi-vehicle case. The performance of the method is currently unsatisfactory, and an issue for future improvement.

Since the single-vehicle version performs well in simulations, a natural extension is to use it greedily for the multi-vehicle case:

- First, initialize $P$ to contain all pickup nodes, and use the graph search to find first-best solutions.
- Remove the delivered requests from $P$, and search for the best route with the remaining requests.
- Repeat the previous step successively, until all vehicles are used up, or no more profitable routes are available.

The greedy heuristic cannot guarantee optimality of the routes, as it can happen that the global optimal assignments do not contain the best route for the single vehicle solution. Furthermore, given a fleet of $m$ vehicles, the optimal route may utilize $k < m$ vehicles. Thus approaches that first partition the pickup nodes, and then optimize the routes within the node subsets, cannot guarantee full efficiency.

5 Computational Experiments

For preliminary testing, we generate random Euclidean instances following uniform distributions with lower and upper bounds listed in Table 1. Travel time is proportional to the route costs with random variations between 0 and 10 hours. The vehicles have maximal capacity of 7.5 tons. The total ride time is constrained at 56 hours (weekly hour constraint. EU Driving Regulation, 2006). Request revenue is generated based on route cost and capacity usage plus random variation: $\pi_i = 0.2 c_{i,i+n} q_i + \delta_i$, where the variation term $\delta$ follows uniform distribution between 20€ and 100€.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Revenue Variation $\delta_i$</th>
<th>Route Cost</th>
<th>Travel Time</th>
<th>Time Window</th>
<th>Loading Time</th>
<th>Capacity Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>20€</td>
<td>50€</td>
<td>0.5h</td>
<td>1h</td>
<td>0h</td>
<td>1t</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>100€</td>
<td>400€</td>
<td>20h</td>
<td>10h</td>
<td>20h</td>
<td>7.5t</td>
</tr>
</tbody>
</table>

Table 1 Parameters for the random instances in simulation, all follow Uniform Distribution.

The simple graph search and the greedy heuristic are implemented in C# with multi-threading. For comparison, the model is also implemented and solved using the GUROBI Optimizer (Gurobi, 2014). The experiments are performed on a PC with Intel Core i5-2400 3.10 GHz CPU and 2GB RAM. We first test the single-vehicle setup with different numbers of requests. For each setup, 10 random instances are tested and the averages are plotted in Figure 2.

![Figure 2](image-url) Simulation results of the single-vehicle case with 5 to 500 requests across 10 runs.
The results of the single-vehicle cases confirm that the graph search is able to find the same optimal solution in less runtime: for the same number of requests, the graph search is over 100 times faster. We test the GUROBI Optimizer with up to only 100 requests due to memory constraints. GUROBI can address larger instances with virtual memory, but often requires several hours to solve. In comparison, the graph search is able to solve instances with up to 500 requests in seconds without exhausting memory. Both the number of delivered requests and total profit increase with an increased number of available requests. The capacity utilization stays relatively stable at 30 — 40%, which can be explained by unfavorable time windows. Hence, we can conclude that the increase in revenue is not due to delivering many more requests, but instead by selecting more profitable requests that fit into the same constrained delivery plan (7.5t, 56h).

Figure 3  
Simulation results of the multi-vehicle case with 5 to 50 requests across 10 runs.  

Figure 3 shows the results of the multi-vehicle case. For up to 20 requests, the optimum is delivered using only 2 vehicles. For larger instances, using more vehicles can achieve higher profits, but with a much longer solution time. For up to 4 vehicles, the graph search is quicker than the GUROBI solver, but the speed advantage diminishes with an increased number of requests; both need minutes or even longer for certain instances. The greedy heuristic is able to attain 99% of the optimal profit in less than a second, and the runtime increases only slightly with an increased number of requests.

6 Conclusion and Research Outlook

In this paper, we introduced a simple graph search method and an accompanying greedy heuristic for solving the Profitable Pickup and Delivery Selection Problem with both capacity and time window constraints. The method uses depth-first-search to directly enumerate feasible routes that satisfy both connectivity and precedence constraints, and the search process is bounded by the time and capacity constraints and the potential gains from undelivered requests. Preliminary experiments show that the graph search is able to solve single-vehicle instances of up to 500 requests optimally in seconds, while the greedy heuristic is able to solve the multi-vehicle instances quickly with high efficiency, both of which are quicker than the standard solution by the GUROBI Optimizer, and matches the performance of previous studies. We estimate that in practice, our solution is sufficient to provide real-time decision support for carriers trading in freight exchanges.

We are currently working on improving the performance of the graph search method for the multi-vehicle case. As Arda et al. (2008) indicate, pre-processing the instances with respect to the time windows and the precedence between requests can greatly accelerate solution speed. We are currently testing the pre-processing technique to obtain tighter bounds and potentially better traversing for the graph search method. We also plan to expand the simulations to more setups and instances from previous studies.
References


