Optimising Efficiency in Part-Load Transportation*

Srinivasa Ragavan Devanatthan, Stefan Glaser and **Klaus Dorer**
Hochschule Offenburg, Offenburg, Germany
sdevanat@stud.hs-offenburg.de
{Stefan.Glaser, Klaus.Dorer}@hs-offenburg.de

Abstract

Existing approaches solving multi-vehicle pickup and delivery problems with soft time windows typically use common benchmark sets to verify their performance. However, there is a gap from these benchmark sets to real world problems with respect to instance size and problem complexity. In this paper we show that a combination of existing approaches together with improved heuristics is able to deal with the instance sizes and complexity of real world problems. The cost savings potential of the heuristics is compared to human dispatching plans generated from the data of a European carrier.

1 Introduction

With an increase in transport business and many mergers between major logistics companies, it becomes increasingly difficult for the the dispatchers to have an overview of the orders relevant to their business. Consequently, opportunities to load orders together on the same vehicle are missed frequently, resulting in increased costs.

Many optimisation algorithms fail to improve the situation due to the size of the problem instances and the complexity of the constraints involved. In this paper we show that a combination of existing approaches together with improved heuristics is able to deal with the instance sizes of real world problems and reduce the costs of transport plans considerably. To do so we have used real data of a major logistics carrier and compared the results of our approach with the transport plan that has been created by human dispatchers and has been performed by the vehicle fleet.

The rest of the paper is organised as follows: Section 2 introduces the transport domain. Section 3 explains how real world problem instances can be addressed with results shown in section 4 before we conclude in Section 5.

2 Domain

The multi-vehicle pickup and delivery problem with soft time windows (m-PDPSTW) [Psaraftis, 1995; Dorer and Calisti, 2005] consists of finding optimal plans for serving transportation requests of customers. The problem is 'single-vehicle' if all transportation requests are served by a unique vehicle. Here, we deal with a 'multi-vehicle' problem where multiple vehicles can be used for transporting all orders. The vehicles may be of different type and have different capacities. As opposed to vehicle routing problems [Laporte and Osman, 1995], in *pickup and delivery problems* (PDP), vehicles do not necessarily start or end in the same location. Transportation requests may have the same, but usually different, pickup and delivery locations. The pickup and delivery of orders has to occur within a specific time window, even though time constraints can be possibly violated up to some tolerated degree. These kind of problems are called PDP with soft time windows.

Table 1 presents different approaches from literature that directly handle PDPTW problems. It is worthy of note, that the approaches defined in the table dealt with benchmark instances while we report on results on significantly larger instance size from real world.

A rich overview on different versions of the problem as well as a collection of solution methods and applied heuristics can be found in [Parragh *et al.*, 2008a; 2008b].

In the following, we specify the information and con-

Method	Author(s)	Characteristics
Insertion heuristic	[Jaw *et al.*, 1983]	300 orders, 24 vehicles
Clustering followed by decomposition	[Dumas *et al.*, 1991]	880 orders, 53 vehicles
Reactive tabu search	[Nanry and Barnes, 2000]	100 orders, 10 vehicles, based on VRPTW instances of [Solomon, 2005]
Branch and cut algorithm	[Ropke *et al.*, 2007]	40 instances of [Savelsbergh and Solomon, 1998]
Insertion heuristic with k-opt	this paper	2137 orders with 1736 available vehicles

Table 1: Methods for m-PDPTW

*This work is supported by the *IngenieurNachwuchs* program of the German BMBF grant number 17 N25 09.

straints of the domain relevant for our work. The term *node* is used to indicate the combination of a stop location and the corresponding time (arrival and departure time) for a given vehicle. A *leg* is the path between two nodes. A *route* is the sequence of nodes a vehicle visits. The vehicle is assumed to be empty at the beginning and at the end of a route. The sum of all routes is called the *delivery plan* representing the schedule of each vehicle. The quality of the solution is the cost of the delivery plan (see Section 2.2).

2.1 Initial Information

The information required to solve transport optimisation is, a set of transport requests or orders and the set of vehicles that are available. Also information for the distance and drive time required for driving any possible leg has to be available.

Every *order* specifies: order type, capacity demand (loading meters), weight, pickup location, pickup time window, pickup service time, delivery location, delivery time window, delivery service time and the time at which the order is known to the system.

The *vehicle* definitions include: vehicle type, capacity (in loading meters and weight), availability location and time. A mathematical analysis of the specific data used is presented in chapter 4.

2.2 Cost Model

Cost reduction is a main driving factor for logistics companies. Cost is therefore used as the objective function for optimisation. The cost model has to make sure that solutions are preferred by the optimisation algorithm that are cheaper to perform in practice. It has therefore to reflect the real costs of the companies as close as possible.

Two types of cost models are typically distinguished: fix-variable for own vehicles and matrix-based for subcontracted vehicles. Costs for own vehicles of the fleet are calculated as

$$c_{fv} = c_{fix} + c_{var} \tag{1}$$

with $c_{fix} = k_{fix} * t$ and $c_{var} = d_{empty} * k_{empty} + d_{loaded} * k_{loaded}$. k_{fix} is a constant representing the fix costs per day. It may depend on the vehicle type in general, but did not in the context data of this paper. t is the number of days the route covers. d_{empty} is the sum of distance of all legs driven empty including a possible empty leg to the first pickup location and a possible empty leg to drive home at the end of the route. d_{loaded} is the sum of distance of all legs where at least one order is loaded. k_{empty} and k_{loaded} are costs per kilometre for empty or loaded legs respectively.

Costs for subcontracted or spot market vehicles are typically based on distance and load matrices.

$$c_{ma} = \sum_{i=1}^{n} d_i * l_i * k_{ma}(d, l) \tag{2}$$

where n is the number of legs, d_i is the distance of leg i, l_i is the load on leg i in loading meters and $k_{ma}(d, l)$ is a function defining the costs per kilometre and loading meter. The function is represented by a matrix defining different distance and load classes with linear interpolation between the specified values. It is usually retrieved from historic data. Note that this cost model does not account for fixed costs nor does it take empty legs to the first pickup or after the last delivery into account. Comparable entries of the cost constants in the matrix are therefore typically much higher than k_{empty} and k_{loaded}. In the context of this paper, two distance classes and thirteen load classes have been used.

2.3 Constraints

The optimisation heuristics have to obey certain constraints in order to create solutions that are drivable in reality.

- Load constraints:
  - Precedence (pickup has to be before delivery);
  - Pairing (pickup and delivery have to be done by the same vehicle);
  - Capacity limitation of a vehicle;
  - Weight limitation of a vehicle;
- Time constraints:
  - Earliest pickup and delivery;
  - Latest pickup and delivery;
  - Legal driving time regulations for drivers.
  - Service times at pickup/delivery locations

In practice pickup and delivery times are typically treated as soft constraints. This means that short delays are accepted if they allow for better delivery plans. A soft constraint is defined by a tuple $< s, e, c_f, c_v >$ where s is a start value above (below) which the condition is soft violated, e is an end value above (below) which the constraint is considered hard violated, c_f are the fix violation costs assigned if the constraint is (soft) violated and c_v are variable violation costs that grow proportional to the amount of soft violation. The fixed violation costs can be used to control the number of violations. The variable violation costs ensure that the amount of constraint violation is kept low and only accepted if the violation cost is less than the benefit of violating the constraint.

3 Implementation Strategies

The classification of the problem as NP-Hard and the size of the problem in reality, thousands of orders to be served with a fleet of hundreds of vehicles, impairs the application of exact methods. Most exact methods, which work well for small specific problem instances in the absence of many constraints, fail to work acceptably fast in practice. Heuristics find *good* solutions in reasonably short time, which is the major concern in the real world.

A straight-forward method to apply insertion heuristic to build an initial solution, followed by a tour improvement heuristics seems the first best tentative approach towards a problem of the size we have handled.

3.1 Insertion Heuristic

The insertion heuristic builds a set of routes by inserting one order at a time. The number of routes is freely determined while inserting. It is not expected that it produces the optimal set of routes for transporting the orders. The main idea is to build an initial feasible solution which is then optimised for the objective function. The quality of this initial solution depends on the sequence in which the orders are inserted. In our case orders have been sorted by earliest delivery time.

A new order is inserted at the best feasible insertion place over every route. It considers the objective function of the problem as the insertion cost. For TSP problems, the objective function is *distance* and an example would be the *smallest detour in distance* [Azi *et al.*, 2010], which may not find the optimal tour, but would certainly produce an acceptably short route.

For the m-PDPSTW that we handle, each route in the solution is a TSP but with constraints which impose precedence of the pickup node before the delivery node. This is referred to as the *pickup and delivery*-TSP or PDTSP, where the objective function is the total cost.Here, we require a simple permutation heuristic which would find the cheapest point of insertion of a new order on the route. This heuristic, *cheapest permutation*, is described below.

Let $(i_0, \dots i_n)$ be the nodes on the route r. Let i_p and i_d be the pickup and delivery nodes of a new order that has to be inserted on the route. The pickup node i_p is inserted as (i_{k-1}, i_p, i_k), $1 \leq k \leq n$, i_0 is the start node and i_{k-1} and i_k are two adjacent pickup or delivery nodes on the route. For each partially inserted route, $(i_0, \dots, i_p, i_k, \dots, i_n)$, the delivery node i_d is inserted as (i_{l-1}, i_d, i_l), $k \leq l \leq n$. If the pickup-insert fails, the following permutations of delivery-inserts are not made.

For each order, the insertion heuristic is run on every route and the cheapest route is chosen. If no existing vehicle is able to transport the order, a new vehicle with a new route is created to handle this order. The set of all routes serviced by individual vehicles constitutes a delivery plan and is an initial solution.

The way in which precedence constraints are incorporated during the solution process is of particular importance to the effectiveness of this heuristic for this problem. It implicitly eliminates some of the infeasible PDTSP solutions. The heuristic is fairly quick mainly because it does not permute the existing nodes on a route when an insertion is made. The constraints are enforced at different levels of the heuristic.

- It is possible that the load constraints of the vehicle are hard violated at node i_p. These are physical restrictions of the vehicle and cannot be soft violated. In such cases, the corresponding permutations of delivery-inserts are not made, reducing the feasible states.
- A pickup-insert could alter the time parameters of the subsequent nodes after i_p, which may produce a violation of the time constraints up to a certain limit. In these cases, the subsequent nodes are one of the permutations of the delivery-inserts. The heuristic first constructs the route and then schedules. If scheduling fails, the route is thrown away.

Soft constraint violations produce costs which are included in the objective function of the problem. This ensures that an order is allocated on a route with a soft violation only when allocating the same order on all other routes is impossible or produces a higher cost.

The driving plan may either be improved by applying restrictive constraints which enforce route quality at the time of construction or by using tour improvement heuristics. Improvement may be achieved by a re-arrangement of existing nodes in a route or the re-assignment of an order to another route.

3.2 Tour Improvement Heuristic

The tour improvement heuristic aims to improve the quality of the entire delivery plan by employing a local search with k-change neighbourhood, simply referred as k-opt [Papadimitriou and Steiglitz, 1982; Helsgaun, 2006]. It has been applied for the travelling salesman problem and has been shown to produce high quality solutions in polynomial time [Lawler *et al.*, 1985]. As far as we have seen, the k-opt has not been applied for the m-PDPSTW. In this Section, we define a single *"change"* for the m-PDPSTW and brief on the k-opt.

A new feasible solution can be obtained by performing a single change to an existing solution. If k' number of changes are applied to the current solution, the new solution is then described to be in the k'-neighbourhood. Depending on the type of the problem, the parameter of change can be varied. This is explained as follows.

Let $R = \{r_1, r_2, \dots r_n\}$ be the set of all routes. Each route is serviced by a single unique vehicle. Let $O_i = \{o_1, o_2, \dots o_k\}$ be the orders transported by route r_i. It should be noted that $O_{all} = \sum_{i=1}^{n} \mid O_i \mid$ and $O_i \cap O_j = \phi$ for distinct $i, j \in \{1, \dots n\}$. As can be seen, here we assume that each order is transported on a unique route. Additionally, each route is assumed to be serviced by a unique vehicle.

A single *change*, $k' = 1$, is then, removing an order from the route it is transported on and inserting the order on another route. This remove-insert pair is together considered as a single change. Therefore, a removal is always followed by an insert. It is possible that a route in the current solution might violate constraints beyond their hard limit, either at the time of removal or at the time of insertion of the order. In either case, the new solution is not accepted as an improvement. This ensures that all the orders transported in the current solution are also transported in the successor solution.

In the case of TSP, a k-opt move changes a tour by replacing k edges between existing nodes, with k other edges such that a shorter tour can be obtained. For the problem instance handled in this paper, a m-PDPSTW, k orders transported on one route is replaced with k other orders from a different route, such that the total cost is reduced. These k-changes can be sequential as well as non-sequential. If a k-change produces a better solution in terms of the objective function, then this new solution is accepted as the current solution for further improvement.

The pseudo-code for tour improvement is shown in Algorithm 1. Picking routes in lines 4 and 5 is done in a brute force approach iterating over all routes. As stop-criterion a time limit as well as a maximal number of iterations was used. Lines 7 to 13 perform a hill-climbing in the k'-neighbourhood (not including the $k' - 1$ neighbourhood).

In cases where the neighbourhood does not have feasible and cheaper solutions, the neighbourhood is enlarged i.e., $k' = k' + 1$ changes are made to search for an improvement (line 6). A k-opt heuristic checks all k' neighbourhoods before termination. Here, $1 \leq k' \leq k$. It can be observed

Algorithm 1 tour improvement

```
1: procedure IMPROVE($s, k$)                ▷ initial solution, max neighbourhood size
2:   while true do
3:     $d \leftarrow dimensionOfSearchSpace(s)$
4:     pick route1 from $d$
5:     pick route2 from $d$
6:     for $k' \leftarrow 1, k$ do
7:       $o \leftarrow neighbourhood(route1, route2, k')$
8:       for all $o' \in o$ do
9:         $s' \leftarrow getNeighbour(o')$
10:        if $f(s') < f(s_{best})$ then
11:          $s_{best} \leftarrow s'$
12:        end if
13:      end for
14:      if stop-criterion met then
15:        return $s_{best}$
16:      end if
17:      $s \leftarrow s_{best}$
18:      if improvement then
19:        break                              ▷ repeat with $k = 1$
20:      end if
21:    end for
22:  end while
23: end procedure
```

that the k-opt may take exponential number of iterations to evaluate all possible k'-changes. The performance is hence sensitive to the number of orders on a route.

A similar k-opt algorithm for the TSP was studied for theoretical performance guarantee and results have been shown for the proof of quality of the 2-opt as a heuristic for random TSP instances in unit square [Chandra *et al.*, 1994]. The advantage of the k-opt heuristic is its scalability in the choice of the neighbourhood size. We show the results obtained from using the 2-opt as the *tour improvement heuristic*. Though k can take any integer value less than $\mid O_i \mid$ for that route r_i involved in the change, we applied $k = 2$. The restriction was not just to simplify implementation, but to see the first results of performance of the 2-opt algorithm on real world data for the m-PDPSTW, on an inexpensive hardware. The results are discussed in the following Section.

4 Results

The heuristics described in the previous section were evaluated on real world data of an international carrier. The availability of the human delivery plan allows us to compare the performance of the heuristics and demonstrate their applicability on real world problems. However, to have an idea on the gap of the used heuristics to optimal solutions, we ran them on the biggest available benchmark problems listed in [Lim, 2010]. In the following we start with a discussion of the benchmark problem data sets and the differences to real world scenarios, before we characterize the specific real world data set followed by a comparison of our results to the human delivery plan.

4.1 Benchmarks

In order to run benchmark problems, a couple of changes to the system are necessary. The expensive calculation of legal drive time regulations could be switched off. In general the drive time/distance lookup is much cheaper in the benchmark case just calculating Euclidean distances instead of looking up real road drive times. To avoid too heavy changes on the system, distance and drive time calculation is done using integers (rounded, not truncated) which is precise enough in real world. This is why the results marked with a * in Table 2 can not be counted as best known. Considering soft time windows could be switched off making time windows shorter and easier to prune. Having all trucks available in one depot and having identical trucks simplifies the decision on which truck to take. However some pruning is then not possible like not assigning orders to trucks that are too heavy or too big.

Given this, the comparison to benchmarks can only be an indicator. This is why we only added one instance for each of the six classes available in [Lim, 2010]. Anyhow, the main focus here is on optimizing real world data. As can be seen, the optimization approach is highly sensitive to the amount of orders on a single route as has been stated in Section 3.2. The corresponding amount of orders per truck is 3.1 in the optimized real world scenario.

4.2 Characterization of the real world data set

The real world problem data set consists of overall 1736 vehicle definitions starting at 248 different locations, and 2137 orders with pickup and delivery locations in six different countries across Europe. The orders define overall 921 different locations. Since this statistic is not taking the corresponding time windows into account, we decided to provide a more accurate statistic by combining a location with its corresponding date, to a so called location-date. The combination of all pickup locations with their earliest pickup date and respectively all delivery locations with their earliest delivery date results in 2113 different location-dates. Real world data also includes missing and/or implausible values, which have to be handled by the system. For this reason, carrier specific processing rules are used to clarify such situations most likely to what a human dispatcher would do.

To get a general impression of the specific data set, the average and standard deviation to all significant attributes are listed in Table 3. While the pickup time windows are often defined more precise, most delivery time windows are either missing one limit in data, or sometimes both, which leads to a high average (towards the default value of one week) with a relatively small standard deviation compared to the pickup time windows. It also seems hard to find a good indicator for reasonable pruning of the search space. According to the average and standard deviation of the capacity demand of the orders (loading meters and weight), in relation to the capacity of the available vehicles, around 95% of the orders could be served by any arbitrary vehicle. In fact 253 orders (11.8%) have a bigger capacity demand than the smallest available vehicle. But since the smallest vehicles contribute by just 0.12% to the total vehicle fleet, this theoretical potential gets again negligible. The orders cover a 20 day period (from the earliest earliest-pickup to the latest latest-delivery).

Instance	vehicles			distance			orders	avg. orders per	time (s)
	own	best	off	own	best	off		vehicle (best)	
lc1101	100	100	0%	42 460	42 488.66	-0.1%*	527	5.27	65
lr1101	95	100	-5%*	70 242	56 903.88	+19%	527	5.27	95
lrc1101	104	84	+24%	62 887	49 315.30	+27%	527	6.27	170
lc2101	39	30	+30%	34 282	16 879.24	+103%	507	16.8	589
lr2101	30	19	+58%	89 454	45 422.58	+97%	503	26.5	1 021
lrc2101	43	22	+95%	66 943	35 073.70	+91%	507	23.0	717

Table 2: Application to benchmark instances

Orders	avg	stddev
Distance (km)	666.52	334.45
Loading meters	5.65	5.45
Weight (kg)	7 428.26	8 196.43
Pickup time window (h)	102.30	81.70
Delivery time window (h)	155.35	43.46
Service time (min)	90.00	0.00
Vehicles		
Loading meters	14.93	0.26
Weight (kg)	26 850.81	537.67

Table 3: Analysis of the real world data set

4.3 Optimization results

The availability of the human dispatcher's delivery plan allows a validation with respect to real world scenario. Table 4 shows the result statistics for insertion heuristic and tour improvement heuristic with respect to the human delivery plan. While the insertion heuristic itself is just applicable on start up to build an initial solution, the tour improvement heuristic can be applied upon both plans. As mentioned before, the objective function during optimisation was the total cost (transportation and constraint costs). In a real world scenario, apart from the total cost, other parameters like the average utilisation of the vehicles, the overall driving km, the empty km and the number of time window violations are additionally used to measure the quality of a delivery plan. In Table 4 the average utilisation of a vehicle was calculated as capacity utilisation per driven km. All results were computed single threaded on an ordinary PC (Intel Core 2 Duo @ 2.8 GHz). Runtime measurements correspond to this hardware.

The insertion heuristics was able to save more than 25% of the costs compared to the plan created by human dispatchers. In more detail, since the transportation costs are directly related to the load of a vehicle together with the driven distance, most of the cost savings have to be reached by a higher utilisation. In this case, the insertion heuristic was able to rise the average utilisation from initially 45.1% up to 76.2%. In terms of costs of the matrix cost model we used, this results in a cost difference around 25% to 30% - depending on the driving distance. This and the reduction of overall kilometres are the most significant cost reduction factors. The main reason why human dispatchers fail to achieve the same result quality is most likely that not all orders are visible to them. Typically dispatchers are organised in regional business centres and have limited insight into orders of other regions to keep the assignment problem tractable to humans.

The relatively low number of violations in the human plan was reduced by another 50%. The reduction in number of used vehicles is due to the higher utilisation as well as using the same vehicles more often. A runtime of 18 minutes is definitely acceptable to logistic companies.

The tour improvement heuristics was applied to the human delivery plan, in order to see its potential upon a hand made solution. As shown in Table 4 it was able to improve in all of the previously listed comparison criteria.

The best results are achieved by running tour improvement heuristics on the solution created with insertion heuristics saving 26.9% of the costs. Three hours of runtime could be problematic in dynamic situations, but the heuristics is incremental and can be interrupted at any time.

5 Conclusions and Future Work

We were able to show, that our heuristics are able to deal with the instance size and complexity of a real world m-PDPSTW. The reported results indicate that the insertion heuristic is efficient in building initial solutions compared to the delivery plan created by human dispatchers. It can be observed that the sequential application of both heuristics significantly reduce the overall transportation cost.

It is our strong believe that it is of big value to compare optimization results with human performance. It boils down to the question if optimization should address theoretically uninteresting but practically important issues like drive time regulations or real street distances. Only by doing so, we will practically benefit from work done in this area. In the absence of exact methods to solve such instance sizes, the first ones able to solve these problem instances to compare with are human dispatchers.

The results presented in this paper are non-dynamic, i.e. they do not take the time into account at which the data was available to the system (see [Azi *et al.*, 2010]). The optimization algorithms used, however, can easily be adjusted to deal with dynamic versions of the problem. The time calculation has to be adjusted to take the current time into account. Nodes already in the past are skipped during calculation. Orders have to be inserted by the date they are known to the system. The insertion heuristics works unchanged and produces 985,756 cost on the data presented in Section 4 which is 1.8% off the non-dynamic result and still more than 20%

	Human plan	Insertion heuristic		Impr. heuristic on Human		Impr. heuristic on Insertion	
Transportation Cost	1 300 233	968 658	(-25.5%)	1 030 162	(-20.8%)	950 916	(-26.9%)
Driving km	1 246 771	885 063	(-29.0%)	951 390	(-23.7%)	873 027	(-30.0%)
Empty km	26 338	9 435	(-64.2%)	5 545	(-79.0%)	8 579	(-67.4%)
Utilisation (%)	45.1	76.2		61.2		76.1	
Violations	47	23	(-51.1%)	7	(-85.1%)	19	(-60.0%)
Vehicles	1 736	699	(-59.7%)	1 111	(-36.0%)	697	(-59.9%)
Transported Orders	2 137	2 137		2 137		2 137	
Runtime(min)	-	18		180		180	

Table 4: Comparison of human and optimised delivery plans

better than the human dispatchers. We are currently working on changing the high level workflows to include tour improvement heuristics. The main changes affect when tour improvement heuristics are triggered and what routes to select.

Currently we assume that every order is transported only on one route serviced by only one vehicle. In reality, an order on a certain vehicle might be exchanged with another vehicle while in transit. The mode of transport could also differ. In such cases, for example, it would be cheaper to transport large number of orders by rail than individually transporting them by road. The orders in the example would have to be checked for partial transport on adjacent routes. We refer to the problem as “inter-modal heterogeneous m-PDPSTW”. It can be observed that the already large combinatorial space enlarges further. The analysis of performance and behaviour of known methods would allow us to understand the problem space better and improve our techniques.

References

[Azi *et al.*, 2010] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. A dynamic vehicle routing problem with multiple delivery routes. *CIRRELT-2010-44*, (44), 2010.

[Chandra *et al.*, 1994] Barun Chandra, Howard karloff, and Craig Tovey. New results on the old k-opt algorithm for the tsp. *5th ACM-SIAM Symposium on Discrete Algorithms*, pages 150–159, 1994.

[Dorer and Calisti, 2005] K. Dorer and M. Calisti. An adaptive solution to dynamic transport optimization. In Michael Pechoucek, Donald Steiner, and Simon Thompson, editors, *AAMAS 2005 proceedings*, Utrecht, 2005.

[Dumas *et al.*, 1991] Y. Dumas, J. Desrosiers, and F. Soumnis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

[Helsgaun, 2006] Keld Helsgaun. An effective implementation of k-opt moves for the lin-kernighan tsp heuristic. *Writings on Computer Science*, (109), 2006.

[Jaw *et al.*, 1983] J. Jaw, A. Odoni, H. Psaraftis, and N. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Research B*, 20B(3):243–257, 1983.

[Laporte and Osman, 1995] G. Laporte and I. H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61:227–262, 1995.

[Lawler *et al.*, 1985] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York, 1985.

[Lim, 2010] Li & Lim. Li & lim benchmark. Website, 2010. `http://www.sintef.no/Projectweb/TOP/Problems/PDPTW/Li--Lim-benchmark/`.

[Nanry and Barnes, 2000] William P. Nanry and J. Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research*, Part B 34:107–121, 2000.

[Papadimitriou and Steiglitz, 1982] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[Parragh *et al.*, 2008a] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems: Part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[Parragh *et al.*, 2008b] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems: Part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008.

[Psaraftis, 1995] H. Psaraftis. Dynamic vehicle routing: status and prospects. *Annals of Operations Research*, 61:143–164, 1995.

[Ropke *et al.*, 2007] Stefan Ropke, Jean-Francois Cordeau, and Gilbert Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272, 2007.

[Savelsbergh and Solomon, 1998] MWP. Savelsbergh and M. Solomon. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46:474–490, 1998.

[Solomon, 2005] M. Solomon. Vrptw benchmark problems. Website, 2005. `http://w.cba.neu.edu/~msolomon/problems.htm`.