December 2002

SOLVING THE PICKUP AND DELIVERY PROBLEM WITH TIME WINDOWS USING iSSQUEAKY WHEELi OPTIMIZATION WITH LOCAL SEARCH

Hongping Lim
Massachusetts Institute of Technology

Andrew Lim
National University of Singapore

Brian Rodrigues
Singapore Management University

SOLVING THE PICKUP AND DELIVERY PROBLEM WITH TIME WINDOWS USING "SQUEAKY WHEEL" OPTIMIZATION WITH LOCAL SEARCH

Hongping Lim
Massachusetts Institute of Technology
hongping99@yahoo.com

Andrew Lim
National University of Singapore
alim@comp.nus.edu.sg

Brian Rodrigues
Singapore Management University
br@smu.edu.sg

Abstract

The Pickup and Delivery Problem with Time Windows (PDPTW) is an important problem in fleet planning where decisions can involve not only dispatching company fleets but also the selection of carriers on certain routes. In this problem, vehicles travel to a variety of locations to deliver or pick up goods and to provide services. The increasing costs for additional vehicles motivate managers to optimize fleet usage. Managers also seek to achieve economical use of fuel, maintenance and overtime costs by minimizing travel distance and duration. As such, PDPTW impacts the interface of supplier-customer relationship management in the supply chain process and is essential for any linked decision support system. In this paper, we describe deployment of a relatively new optimization technique, known as "Squeaky Wheel" Optimization (SWO), to the PDPTW. Our objective is to minimize the fleet size, travel distances, schedule durations and waiting times.

We implement an SWO framework for the PDPTW, integrating Solomon's Insertion Heuristic and Local Search into a construction phase. In addition, we design a blame assignment and prioritizing schemes to facilitate problem solution. Our new method has been tested on the Solomon's 56 benchmark cases for which we have obtained encouraging results with this new technique.

Keywords: Vehicle routing, squeaky wheel, heuristics

Introduction and Research Objectives

A complete logistics system involves the transportation of raw materials from suppliers to factory plants for manufacturing and/or processing, as well as delivery of the manufactured products to depots and warehouses and customers. Supply and distribution procedures need to be efficient to minimise systemwide costs. In transport logistics, savings can be derived, for example, from lowered trucking costs resulting from reduced numbers of vehicles, shorter distances and less penalties incurred from untimely delivery. The optimization of vehicle routes is an important and integral component of supply chain management.

The Pickup and Delivery Problem with Time Windows (PDPTW) is a particularly important problem in transportation logistics management, particularly in fleet routing (see Bruggen et al. 1993, Chiang et al. 1997, Desrosiers et al. 1995, Dumas et al. 1986, Dumas et al. 1991, Psarafis 1980, Psarafis 1983, Savelsbergh et al. 1995 and Sexton et al. 1986). It is a problem that can arise in static as well as dynamic systems, and whose solution would be necessary for any decision support system underlying the supply chain management of the larger system. In this problem, we require a fleet of vehicles with capacity limits to service a set of transportation "requests", in which pickup customers have loads to be delivered to delivery customers. A pickup and delivery pair (PD-pair) consists of two customers. A vehicle can service a PD-pair and all vehicles begin and end their journeys at the same

depot. This problem arises from real-life situations in which vehicles must travel to a number of destinations to pickup and deliver goods and to provide services. Examples of this, and other closely related problems, include bank deliveries, postal deliveries, industrial refuse collection, over-night delivery service, dial-a-ride problem (see Dumas et al.1986, Psarafis 1980, Psarafis 1983 and Sexton et al. 1985), airline scheduling (see Erdmann et al. 1999), bus routing (see Forbes et al. 1994), tractor-trailer problems, helicopter support of offshore oil field platforms, logistics and maintenance support. Related applications include VLSI circuit design, flexible manufacturing systems (see Qiu and Hsu 1999), and casualty evacuation.

Our objective is to minimize the fleet size, travel distance, schedule duration and waiting time. Our purpose here is to describe the use of a relatively new optimization technique, known as the "Squeaky Wheel Optimization" (SWO) (see Joslin et al. 1999), on the PDPTW.

Problem Statement

In this section we provide notations and a mathematical formulation of the PDPTW. The mathematical framework that we present here are built upon that presented in Solomon 1987.

Let:

P: Set of all customer locations; for some $n \in \mathrm{N}$.
0: The central depot.
K: Set of available vehicles, indexed by . indicates the maximum number of vehicles that are available.
V: Set of all nodes, i.e. customer locations and depots, .
A: Set of arcs linking the nodes.
G: Digraph of problem, $G = (V, A)$.
$G^k = (V^k, A^k)$: The sub-digraph associated with a specific vehicle k, with V^k the node subset that vehicle k traverses, while $A^k = V^k \times V^k$ contains all feasible arcs between the nodes in V^k.
P^P: Set of pickup customers, $P^P \subset P$.
P^D: Set of delivery customers, $P^D \subset P$.
Q^k: Maximal load of vehicle k.
P^+: Set of pickup locations of PD-pairs, $P^+ \subset P$.
P^-: Set of delivery locations of PD-pairs, $P^- \subset P$

From the above, we note that $P = P^+ \cup P^- |P^+| = |P^-| = n/2$.

Each node $i \in V$ has an associated customer demand $q_i (q_0 = 0)$, a service time $s_i (s_0 = 0)$, and a service-time window $[e_i, l_i]$. $q_i > 0$ for $i \in P^+ \cup P^P$ and $q_i < 0$ for $i \in P^- \cup P^D$. For each $i \in P^+$, we denote $i_d \in P^-$ as the delivery location of i. For each pair of nodes $(i,j) | (i \neq j, i,j \in V)$, a non-negative distance c_{ij} and a non-negative travel time t_{ij} are known.

Due to time window constraints, arcs may not exist between some node pairs. Therefore, the arc set can be defined as $A = \{(i,j) | i,j \in V, i \neq j, t_{0i} + s_i + t_{ij} l_j\}$. If a vehicle reaches a customer i before e_i, it needs to wait until e_i in order to service the customer.

The schedule duration of a route is the sum of waiting time, service time and travel time. Different contexts have different priorities for minimizing a number of objectives, subject to a variety of constraints.

For transportation of goods, the objective involves minimizing the number of vehicles, travel costs and schedule time. However, for dial-a-ride situations, it is preferable to minimize the inconvenience caused by pickups or deliveries performed earlier or later than desired time.

To formuate the pickup and/or delivery problems, there types of variables are used:

1. Binary variables $X^{k,ij}$, where $k \in K, i, j \in V, i \neq j$. $X^{k,ij}$ equal to 1 if vehicle k travels the arc (i, j) from node i to node j, equal to 0 otherwise.

2. Time variable T_i, where $i \in P$, represents the time to begin service at node i.

3. Load variable Y_i, where $i \in P$, represents the total load on the vehicle after it leaves node *i,* $i \in P$.

The mathematical formulation is as follows:

$$\min \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X_{ij}^k \tag{1}$$

subject to:

$$\sum_{k \in K} \sum_{j \in V} X_{ij}^k = 1, \forall i \in V, \tag{2}$$

$$\sum_{k \in K} \sum_{j \in P} X_{0j}^k \le |K|, \forall j \in V, \tag{3}$$

$$\sum_{j \in V} X_{ij}^k - \sum_{j \in V} X_{ji}^k = 0, i \in V, k \in V, \tag{4}$$

$$\sum_{j \in V} X_{ij}^k - \sum_{j \in V} X_{j,i_d}^k = 0, i \in P^+, k \in K, \tag{5}$$

$$T_i + s_i + t_{i,i_d} \le T_{i_d}, i \in P^+, \tag{6}$$

$$T_i \le T_{i_d}, i \in P^+, \tag{7}$$

$$X_{ij}^k = 1 \Rightarrow T_i + s_i + t_{ij} \le T_j, \quad \forall (i,j) \in A^k, k \in K, \tag{8}$$

$$\sum_{j \in P^-} X_{0j}^k = 0, k \in K, \tag{9}$$

$$\sum_{j \in P^+} X_{i0}^k = 0, k \in K, \tag{10}$$

$$e_i \le T_i < l_i, i \in P, \tag{11}$$

$$X_{ij}^k = 1 \Rightarrow Y_j + q_j, \forall (i,j) \in A^k, k \in K, \tag{12}$$

$$0 \le Y_i \le Q^k, i \in V, \tag{13}$$

$$X_{ij}^k = 0/1, \forall (i,j) \in A^k, k \in K, \tag{14}$$

The formulation seeks to minimize travel cost. Constraints (2) and (4) ensure that each is exactly visited once. Constraints (5) are coupling contraints that ensure *PD*-pairs to be in the same route. Constraints (3), (9) and (10) impose contraints on the vehicle fleet, in which each vehicle should originate from, and return, to the depot. In addition, when a vehicle leaves the depot, the first customer location should not be a delivery location in a *PD*-pair. On the other hand, before a vehicle returns to the depot, the last customer location should not be a pickup location in a *PD*-pair. Constraints (6) and (7) are precedence constraints for *PD*-pairs. Constraints (8) and (11) are the time window constraints. Constraints (12) and (13) are capacity constraints.

We aim to service all customers without violating any of the vehicle capacities, time windows, precedence and coupling constraints. The order of our objectives is as follows

I. Minimize number of vehicles

II. Minimize total traveling distance
III. Minimize total schedule duration
IV. Minimize total waiting time

Research Methodology

We employ a "Squeaky Wheel Optimization" technique (SWO) (see Joslin et al. 1999) which has as its core a Construct – Analyze – Prioritize cycle. In this cycle, a greedy algorithm constructs an initial solution, making decisions in an order determined by priorities assigned to the elements of the problem. This solution is then analyzed to find the elements of the problem that are causing "trouble", and "blame" is assigned to them. According to the magnitude of the blame, the priorities of trouble causing elements are then increased, causing the greedy constructor to deal with them sooner in the next iteration. This cycle repeats until a termination condition occurs.

The construction, analysis and prioritization are all in terms of the elements that define a problem domain. In our case, the elements are the customers. In a scheduling domain, for example, those elements might be tasks, while in graph coloring they might be nodes to be colored.

The three main components of SWO are:

- **Constructor.** Given a sequence of problem elements, the constructor generates a solution using a greedy algorithm. The sequence determines the order in which decisions are made.
- **Analyzer.** The analyzer assigns a numeric "blame" factor to elements that contribute to flaws in the current solution.
- **Prioritizer.** The prioritizer uses the blame factors assigned by the analyzer to modify the previous sequence of problem elements. Elements that received blame are moved towards the front of the priority sequence. The higher the blame, the further the element is moved.

In this manner, difficult elements are assigned more blame and they move forward in the priority sequence so that they will be handled earlier by the constructor. By taking care of the difficult elements before the easier elements, the chances of obtaining a better solution are increased. The difficult elements, being dealt with earlier, will have more options to choose from, and are more likely to be placed in better positions. The less difficult elements will then fit into the remaining positions.

Implementation

Our framework consists of the three main components of SWO: the Constructor, the Analyzer and the Prioritizer. The priority sequence used for the first iteration is randomly generated. In addition, we use restarts after every given number of iterations to generate a new random priority sequence. To prevent cycles, we store eigenvalues of solutions found, and initiate a restart if a new eigenvalue is already present in the stored set of eigenvalues.

Eigenvalue Structure

Our eigenvalue structure consists of the following four values of the solution: Number of Vehicles, Total Travel Cost, Total Schedule Duration and Total Waiting Time.

Since the probability that two different solutions have the same eigenvalue is very small, it is reasonable to regard two solutions as the same if they share the same eigenvalue. We record the set of eigenvalues and check new eigenvalues against this set to prevent cycling.

Constructor

The constructor builds a solution by adding customers into the routes pair by pair, in the order they occur in the priority sequence.

For the first iteration, and after each restart, the priority sequence is randomly generated. When either of a pickup and delivery pair is selected, the other is selected together. Here, we use a modified version of Solomon's Insertion Heuristic (see Solomon 1987). We retain the evaluation criteria for selecting the best insertion positions, and omit the selection of the best customers for insertion. Each time the constructor selects a PD-Pair, it finds the best feasible insert positions among the current routes and inserts them. If no feasible positions are found, a new route is created for the pair.

In addition to this heuristic, we integrate a local search in attempt to improve the solution found by the constructor. As suggested by Joslin et al 1999, SWO may be poor at making small "tuning" moves, and could be coupled with local search to look for good solutions in the vicinity. Our local search uses two operators to generate new solutions: PD-Shift Operator and PD-Exchange Operator.

PD-Shift Operator

The PD-Shift operator moves PD-pairs from one route to another, subject to all the constraints imposed on PDPTW. For each pair of selected routes, say, *Route1* to *Route2*, the PD-Shift operator is used twice to generate two different solutions. The operator first shifts PD-pairs from *Route1* to *Route2* and then from *Route2* to *Route1*. (See Figure 1)

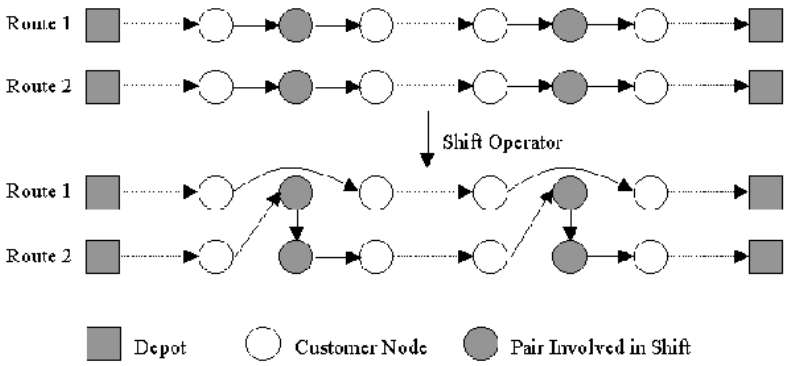

Figure 1. PD Shift Operator

PD-Exchange Operator

The PD-Exchange operator swaps PD-pairs from two routes, subject to all the constraints imposed on PDPTW. For each pair of selected routes, the PD-Exchange operator simultaneously moves a PD-Pair from *Route1* to *Route2* and another PD-Pair from *Route2* to *Route1*. (See Figure 2)

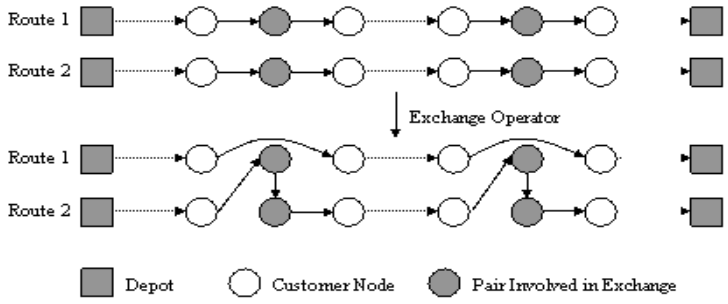

Figure 2. PD Exchange Operator

Analyzer

For each customer, the analyzer assigns four components of blame, each representing the criteria we have listed previously:

I. Vehicle number blame
II. Traveling distance blame
III. Schedule duration blame
IV. Waiting time blame

For traveling distance blame, schedule duration blame, and waiting time blame, we simply assign the contribution the customer makes to the total traveling distance, total schedule duration, and total waiting time respectively.

For vehicle number blame, we first sort all the routes according to non-decreasing length. We then assign the largest blame to the shortest route, decreasing the magnitude of the blame for each subsequent route in the ordered sequence. This blame-assigning approach is aimed at finding fault with vehicles which may be under-ultilized. If very few customers are serviced in a route by a vehicle, it may be possible to schedule them in other routes, hence reducing the number of vehicles used.

Prioritizer

The prioritizer will reorder the customers in the priority sequence by applying a multi-level stable sort, ordering them first by vehicle number blame, then traveling distance blame, schedule duration blame, and waiting time blame.

Experimental Results

Benchmark Problem Instances

We use the same problem instances used in Li et al 2001 (see [8]). These are generated from Solomon's benchmark problem instances (see Solomon 1987), by pairing up customer locations within routes in solutions obtained by their heuristic approach for the vehicle routing problem with time windows in Li et al 2001 (see [7]).

Computational Results

Our experimental environment in which we ran our algorithms was the Linux Kernel 2.2.14-5.0 smp on i686. The algorithms were coded in C++ language, with the use of the C++ Standard Template Library, so as to achieve code efficiency and reusuability. For each problem instance, SWO runs a maximum of 500 iterations. It will terminate after 100 consecutive iterations if there is no improvement in the global best solution, and will restart after every 50 normal iterations.

Results for the 56 Problem Instances

Table 1 to Table 3 shows the best results obtained by our algorithms for the LC, LR and LRC cases.

Comparison of Results

We have used the results obtained by Li et al 2001 (see [8]) as a basis for comparison. The analysis is shown in Table 4 to Table 8. Table 4 shows how our SWO technique fares against Li et al's Tabu-Embedded Simulated Annealing (SA-Tabu), with regard to the mean number of vehicles obtained for the problem instances. Our technique achieves on the average 0.536 vehicle more than SA-Tabu.

Table 5 compares the mean traveling cost. SWO's mean traveling cost is on the average 3.262 percent higher than SA-Tabu's. Table 6 shows the comparison of mean schedule duration. On the average, the mean schedule cost obtained by SWO is 3.809 percent higher than that of SA-Tabu.

Table 7 displays the comparison of mean waiting time. Our mean waiting time is on the average 56.288 percent higher than SA-Tabu's.

Finally, Table 8 compares the mean CPU time taken by the algorithms to calculate their solutions. Our mean CPU time is, on the average, 48.687 percent lower than SA-Tabu (running on i686 as well).

Except for waiting time, the other criteria are within a reasonable percentage of the values obtained by SA-Tabu. Our solution also runs faster, taking almost half the time required by SA-Tabu.

Table 1. Solutions for LC Instances

Problem Instance	*Vehicle Number*	*Travel Costs*	*Schedule Duration*	*Waiting Time*	*CPU Time(S)*
LC101	10	828.937	9828.937	0.000	15.597
LC102	10	828.937	9828.937	0.000	29.523
LC103	10	827.865	10058.030	230.166	60.247
LC104	9	914.443	10122.459	208.016	160.982
LC105	10	828.937	9828.937	0.000	21.260
LC106	10	828.937	9828.937	0.000	22.614
LC107	10	828.937	9828.937	0.000	30.317
LC108	11	866.637	10563.869	697.232	43.340
LC109	10	827.817	9831.779	3.962	71.177
LC201	3	591.557	9591.557	0.000	135.410
LC202	3	591.557	9591.557	0.000	462.967
LC203	3	591.173	9601.715	10.542	996.456
LC204	3	638.181	9752.817	114.636	1072.537
LC205	3	588.876	9588.876	0.000	338.005
LC206	3	588.493	9588.493	0.000	312.259
LC207	3	588.286	9660.404	72.117	434.269
LC208	3	588.324	9744.235	155.911	786.990

Table 2. Solutions for LR Instances

Problem Instances	*Vehicle Numbers*	*Travel Costs*	*Schedule Duration*	*Waiting Time*	*CPU Time(S)*
LR101	19	1650.799	3599.449	948.650	10.209
LR102	17	1491.970	3202.463	710.493	23.287
LR103	13	1293.144	2729.623	436.479	21.842
LR104	11	1074.517	2297.476	222.959	37.689
LR105	14	1390.555	2642.386	251.831	16.904
LR106	12	1272.785	2421.639	148.854	19.931
LR107	11	1150.881	2348.351	197.470	31.718
LR108	10	1028.628	2202.079	173.451	37.888
LR109	13	1257.682	2502.294	244.611	21.963
LR110	12	1244.164	2416.101	171.936	37.146
LR111	11	1136.871	2264.105	127.234	55.459
LR112	11	1060.514	2184.905	124.391	47.386
LR201	4	1267.968	3495.807	1227.839	265.461
LR202	4	1262.166	3557.722	1295.556	1021.936
LR203	3	1093.082	2770.699	677.617	1219.920
LR204	3	934.155	2768.201	834.047	2837.348
LR205	4	1138.710	3262.329	1123.620	306.579
LR206	3	942.298	2502.462	560.164	637.255
LR207	3	967.169	2726.991	759.822	1290.766
LR208	2	833.480	1920.336	86.856	2060.523
LR209	4	1021.119	2933.868	912.749	578.900
LR210	3	980.119	2782.057	801.939	880.742
LR211	3	920.509	2319.829	399.320	801.760

Table 3. Solutions for LRC Instances

Problem Instance	*Vehicle Number*	*Travel Costs*	*Schedule Duration*	*Waiting Time*	*CPU Time(S)*
LRC101	15	1713.984	3096.228	382.245	23.382
LRC102	13	1599.435	2872.113	272.678	25.645
LRC103	11	1275.817	2434.323	158.506	43.516
LRC104	10	1134.489	2241.798	107.309	37.259
LRC105	14	1679.656	2936.557	256.901	17.846
LRC106	14	1604.901	2885.686	280.784	22.130
LRC107	11	1235.553	2350.348	114.795	17.928
LRC108	12	1206.029	2404.474	198.445	29.455
LRC201	4	1858.738	3426.018	567.279	291.004
LRC202	4	1395.244	3348.195	952.951	320.678
LRC203	4	1181.532	3376.840	1195.307	413.442
LRC204	3	821.561	2511.277	689.716	1801.043
LRC205	4	1311.359	3464.606	1153.247	139.101
LRC206	4	1221.384	2987.730	766.346	406.651
LRC207	4	1142.869	2881.529	738.660	387.984
LRC208	4	939.373	2733.041	793.668	868.296

Table 4. Comparison of Mean Number of Vehicles Obtained by SWO and SA-Tabu

Problem Instance	*SWO*	*SA-Tabu*	*Absolute Difference*
LC1	10.000	9.889	0.111
LC2	3.000	3.000	0.000
LR1	12.833	11.917	0.917
LR2	3.273	2.727	0.545
LRC1	12.500	11.625	0.875
LRC2	3.875	3.250	0.625
Average	7.768	7.232	0.536

Table 5. Comparison of Mean Total Travel Costs Obtained by SWO and SA-Tabu

Problem Instance	*SWO*	*SA-Tabu*	*Percentage Difference*
LC1	842.383	832.084	1.238
LC2	595.806	589.229	1.116
LR1	1254.376	1222.201	2.633
LR2	1032.798	973.870	6.051
LRC1	1431.233	1387.594	3.145
LRC2	1234.008	1187.817	3.889
Average	1072.913	1039.015	3.262

Table 6. Comparison of Mean Schedule Duration Obtained by SWO and SA-Tabu

Problem Instance	*SWO*	*SA-Tabu*	*Percentage Difference*
LC1	9968.980	9874.204	0.960
LC2	9639.957	9609.744	0.314
LR1	2567.573	2467.738	4.046
LR2	2821.846	2455.745	14.908
LRC1	2652.691	2537.215	4.551
LRC2	3091.154	2736.971	12.941
Average	4904.329	4724.381	3.809

Table 7. Comparison of Mean Total Waiting Time Obtained by SWO and SA-Tabu

Problem Instance	*SWO*	*SA-Tabu*	*Percentage Difference*
LC1	126.597	42.119	200.568
LC2	44.151	31.765	38.990
LR1	313.197	245.537	27.556
LR2	789.048	478.602	64.865
LRC1	221.458	149.621	48.013
LRC2	857.147	549.154	56.085
Average	402.845	257.758	56.288

Table 8. Comparison of Mean CPU Time Taken by SWO and SA-Tabu

Problem Instance	*SWO*	*SA-Tabu*	*Percentage Difference*
LC1	50.562	225.610	-77.589
LC2	567.362	196.317	189.003
LR1	30.119	371.025	-91.882
LR2	1081.926	1876.384	-42.340
LRC1	27.145	261.218	-89.608
LRC2	578.525	1536.175	-62.340
Average	394.677	769.155	-48.687

Conclusion and Future Research

In this paper, we have successfully implemented SWO in a new domain to the PDPTW, obtaining good results. Our implementation includes the integration of Solomon's Insertion Heuristic and local search in the constructor. It uses a simple blaming system based on the objectives we seek, and works well. For the prioritizer, we have used a multi-level stable sort.

With this relatively primitive approach, we have managed to obtain encouraging results on the 56 Solomon's benchmark instances. In future work, it is possible to attempt using SWO on PDPTW to explore more complex blame and prioritizing systems. Characteristics specific to the PDPTW problem could be analyzed and incorporated into the SWO framework to further improve its performance. A study on how various heuristics and searches would perform within the SWO framework will shed more light on the effectiveness of various combinations and hybridizations of heuristics and SWO.

It is very likely that with these additional enhancements and modifications, our straightforward implementation of SWO can be greatly improved. We hope to achieve results and performance on par with those obtained by SA-Tabu, and be able to even surpass these.

References

Bruggen, L. J. J.Van der, Lenstra, J. K., and Schuur, P. C. "Variable-depth search for the single vehicle pickup and delivery problem with time windows", Transportation Science (27), 1993, pp. 298–311.

Chiang, W. -C., and Russel, R. "A reactive tabu search metaheuristic for the vehicle routing problem with time windows", INFORMS Journal on Computing (9), 1997, pp. 417–430.

Desrosiers, J., Dumas, Y., Solomon, M. M., and Sormis, F. "Time Constrained Routing and Scheduling," in Handbooks in Operations Research and Management Science: Network Routing, M.O.Ball, T.L.Magnanti, C.L.Monma and G.L.Nemhauser Elsevier(eds), Elsevier, Amsterdam, 1995, pp. 35–139.

Dumas, Y., Desrosiers, J., and Soumis, F. "A dynamic programming solution of the large-scale single vehicle dial-a-ride problem with time windows", American Journal of Mathematical and Management Science (16), 1986, pp. 301–325.

Dumas, Y., Desrosiers, J., and Soumis, F. "The pick-up and delivery problem with time windows", European Journal of Operational Research (54), 1991, pp. 7–22.

Erdmann, A., Nolte, A., Noltemeier, A., and Schrader, R. "Modeling and Solving the Airline Schedule Generation Problem", Submitted to Math. in Industrial Systems, 1999

Forbes, M. A., Holt, J. N., and Watts, A. M. "An exact algorithm for multiple depot bus scheduling", European Journal of Operational Research, (72:1), 1994, pp. 115–124.

Joslin, D. E., and Clements, D. P. "Squeaky Wheel Optimization" in Journal of Artificial Intelligence Research (10), 1999, pp. 353–373.

Li, H. B., Lim, A., and Huang, J. H. "Local Search with Annealing-like Restarts to solve the VRPTW", ACM symposium on Applied Computing, SAC-2002, Madrid, Spain

Li, H. B., Lim, A., and Huang, J. H. "Metaheuristics for solving the pickup and delivery problem with time windows", The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001, Dallas, USA

Psarafis, H. "A dyanmic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem", Transportation Science (14), 1980, pp. 130–154.

Psarafis, H. "An exact algorithm for the single vehicle many-to-many immediate request dial-a-ride problem", Transportation Science (17:4), 1983, pp. 351–361.

Qiu L. and Hsu W. J. "Scheduling and Routing Algorithms for AGVs: a Survey" Technical report: CAIS-TR-99-26, Centre for Advanced Information Systems, School of Applied Science, Nanyang Technological University, Singapore, October 1999.

Savelsbergh, M. W. P., and Sol, M. "The general pickup and delivery problem", Transportation Science 29 (1), 1995, pp. 107–121.

Sexton, T. R., and Choi, Y.-Y. "Pickup and delivery partial loads with "soft" time windows", American Journal of Mathematical and Management Science (6), 1986, pp. 369–398.

Sexton, T. R., and Lawrence, D. B. "Optimizing single vehicle many-to-many dial-a-ride problem with desired delivery time: I Scheduling", Transportation Science (19), 1985, pp. 378–410.

Sexton, T. R., and Lawrence, D.B. "Optimizing single vehicle many-to-many dial-a-ride problem with desired delivery time: II Routing", Transportation Science (19), 1985, pp. 411–435.

Solomon, M. M. "Algorithms for the vehicle routing and scheduling problems with time window constrains", Operations Research (35), 1987, pp. 254–264.

William, P. N., and Barnes, J. W. "Solving the pickup and delivery problem with time windows using tabu search", Transportation Research Part B (34), 2000, pp. 107–121.