Open Vehicle Routing Problem with Time Deadlines: Solution Methods and an Application

Zeynep Özyurt[1], Deniz Aksen[2], Necati Aras[3]

[1,2] Koç University, College of Administrative Sciences and Economics,
Rumelifeneri Yolu, Sarıyer, 34450 İstanbul, Turkey
[3] Boğaziçi University, Industrial Engineering Dept., Bebek, 34342 İstanbul, Turkey

Abstract. In the open route version of the well-known vehicle routing problem, vehicles are not required to return to the depot; or if they are required, then they return by traveling the same route back. In this study, we present a modified Clarke-Wright parallel savings algorithm, a nearest insertion algorithm and a tabu search heuristic for the open vehicle routing problem with time deadlines. Some random test problems and a real-life school bus routing problem are solved by these heuristics, and results are compared.

1 Introduction

Capacitated vehicle routing problem (VRP) can be defined as determining a set of routes for a fleet of vehicles based at one or several depots. The objective of the VRP is to deliver a set of geographically dispersed sites or customers with known demands on minimum-cost vehicle tours originating and terminating at a depot. Open vehicle routing problem (OVRP) is a variant of VRP where vehicles are not required to return to the depot, or if they are required, then they return by traveling the same route back. Although OVRP received little attention from researchers until recent years, it has been commonly occurring in the transportation business.

In this study, OVRP with time deadlines (OVRP-TD) is solved with a modified Clarke-Wright parallel savings algorithm, with a greedy nearest insertion algorithm, and with a tabu search heuristic. In OVRP-TD, each customer must be visited before a certain time deadline. The timing of service delivery which arises as vehicle arrival time in routing problems is an important Quality of Service (QoS) guarantee given to meet customer expectations in service systems.

2 Literature Review and Problem Analysis

OVRP studies reported in the literature are not as abundant as the studies on VRP. First, Schrage (1981) mentions OVRP in an article mentioning real-life routing problems. Sariklis and Powell (2000) solve symmetric OVRP by a two phase algorithm which uses cluster-first-route-second mechanism.

Tarantilis and Kiranoudis (2002) solve a real-life instance of multi-depot OVRP for fresh meet distribution by a meta-heuristic they called “list based threshold ac-

cepting algorithm" (LBTA). A spatial decision support system (SDSS) is proposed by Tarantilis et al. (2004). A genetic solution procedure called BoneRoute is used for the OVRP. Tarantilis et al. (2004) propose a single parameter simulated annealing-based algorithm for the same problem. Brandão (2004) proposes a tabu search algorithm (TS) for OVRP with maximum route length constraint. Another TS algorithm is due Fu et al. (2005) again subject to maximum route length constraint. These two TS algorithms differ in their initial solutions, neighborhood structures, objective function and tabu definitions. Both algorithms seem to outperform Sariklis and Powell's solutions; however, CPU times of Sariklis and Powell are considerably better. Although Fu et al. improve the solutions for several of the problems in Brandão's paper, for some others they find worse solutions in terms of total traveling distance and the number of used vehicles.

Eliminating the constraint that all vehicles have to return to the depot does not make OVRP a simpler problem. Also, a good solution for VRP cannot be converted to a good OVRP solution by simply dropping incoming arcs of the depot. Thus, OVRP is to be studied separately. Our problem differs from the current OVRP literature in two points: First is the incorporation of time deadlines. Each customer must be visited before his time deadline. The second difference is the constraint that each route terminates at one of the driver nodes which are specified beforehand. Driver nodes practically correspond to parking lots or homes of drivers. The presence of such fixed driver nodes suits especially those situations in which deliveries to customers are outsourced to a shipping company, or drivers use the same vehicles also to commute between home and depot.

2.1 Clarke-Wright Parallel Savings Algorithm Modified for OVRP

This method (CW) is proposed by Clarke and Wright (1964) for the single depot VRP. Since the algorithm is efficient and simple to implement, it still remains popular to date. In order to adapt CW to OVRP-TD, we modify the distances between customers and depot, and drivers and depot. The modified distances are assigned as follows. Customer-Depot distance is set to infinity, because a vehicle is not to return to the depot directly from a customer node. Driver-Customer distance is also infinity, because a vehicle is not allowed to proceed from a driver to a customer. The same is true for the Driver-Driver distance as well. Finally, Driver-Depot distance is taken as zero to assure that a vehicle will return to the depot from a driver node without increasing the objective value.

As a result of these modified distances, a route is guaranteed to start from the depot, visit one or more customers and end at a driver node.

2.2 Tabu Search Algorithm

Tabu search (TS) is a meta-heuristic algorithm that guides a local search to prevent it from being trapped in premature local optima by prohibiting those moves that cause to return to previous solutions and cycling. TS starts with an initial solution. At each iteration, a neighborhood of solutions is generated, and the best one from this neighborhood is selected as the new solution. Certain attributes of

previous solutions are kept in a tabu list which is updated at the end of each iteration. The selection of the best solution in the neighborhood is done such that it does not adopt any of the tabu attributes. Best feasible solution so far (incumbent) is updated if the new current solution is better and feasible. The procedure continues until any of two stopping criteria is met, which are maximum number of iterations performed and maximum number of non-improving iterations during which the incumbent does not improve. Characteristics of TS heuristic proposed for OVRP-TD can be stated as follows.

Initial Solution

Two different methods of initial solution generation are used at the beginning of TS. First one is the well-known Clarke-Wright parallel savings algorithm (CW). If CW fails to create a feasible initial solution due to time deadline stringency, we try to correct or at least minimize this infeasibility by shifting customer nodes from their current positions to new positions on the same or on a different route.

Second one is a greedy constructive heuristic called nearest insertion method (NI). In this method, we start with as many routes as the number of drivers. Each route initially consists of the depot and a driver node. Customers are then inserted into these routes one by one. All feasible insertion positions are examined for all customers awaiting insertion. Each time, that particular customer is selected which has the least expensive insertion position. The procedure is repeated until all customers have been inserted. For n customers, NI creates the initial solution in $0(n^3)$ time. When a feasible insertion point cannot be found at all, then the least infeasible position with respect to vehicle capacity and time deadlines is chosen.

In both methods described above, when a feasible initial solution cannot be generated, it is hoped that feasibility will be restored during the TS iterations.

Evaluation of Solutions

In our TS heuristic, we apply strategic oscillation by admitting infeasible solutions into the procedure. The evaluation of such solutions is different from that of feasible solutions in that a penalty cost for violating capacity and time constraints will be added to their objective value. This penalty is added to prevent the algorithm from exploring the infeasible regions of the search space in excess.

Penalty costs rise and fall according to the number of feasible and infeasible solutions visited. Every 10 iterations, the numbers of visited feasible and infeasible solutions are compared. If more feasible solutions are visited than infeasible ones, penalty terms are divided by 1.5; otherwise penalty terms are multiplied by 1.5.

The objective value for a solution is obtained by $\sum_{r=1}^{\kappa} [D(r) + p_c V_c(r) + p_t V_t(r)]$

where $D(r)$ denotes total distance traveled on route r, κ denotes the total number of routes, V_c denotes overcapacity (total demand of customers in route r - vehicle capacity), V_t denotes total tardiness in route r, p_c and p_t denote penalty coefficients for overcapacity and for total tardiness in a route, respectively.

Neighborhood Structure

Three move operators are used to generate a neighborhood to the current solution. For each move two customers are selected randomly as pilot nodes:

i. 1-0 move: One of the selected nodes is taken from its current position and inserted after the other node. ii. 1-1 exchange: Two selected nodes are swapped by preserving their original positions. iii. 2-Opt move: For two pilot nodes in the same route, visiting order between these is reverted. If the pilot nodes are in different routes, then the route segments following them are swapped preserving the order of nodes on each segment.

Besides the neighborhood generation, local search with these moves is incorporated into TS as a tool of local post optimization (LPO). In the application of LPO, all customers are set one by one as the first pilot node. For a certain customer set as the first pilot node, second pilot node of the move is selected such that the move yields the highest improvement in total distance traveled without causing any infeasibility. At the end of every 100 iterations as well as when the incumbent solution is updated, a series of LPO is applied to the current solution. This LPO comprises 1-1 exchange, 2-Opt move, 1-0 move and one more time 2-Opt move.

Tabu Attributes and Tabu Tenure

Tabu attribute definitions for three move operators are as follows:

1. 1-0 move: If customer *i* is inserted after customer *j*, the position of customer *i* cannot be changed by the same move while it is tabu-active.
2. 1-1 exchange: If customers *i* and *j* are swapped, *i* and *j* cannot be swapped again while they are tabu-active.
3. 2-Opt move: If it is applied to customers *i* and *j*, it cannot be applied again to the same customers while they are tabu-active.

At each iteration tabu tenure is selected randomly between 5 and 15 iterations. In some cases, namely if aspiration criterion is satisfied, a move can be executed although its attributes are tabu-active. Aspiration criterion is satisfied if the total distance resulting from the move is better than the incumbent's objective value.

3 Computational Results

All codes in the study are written in ANSI C language, compiled and executed in Visual C++ 6.0 on a 3.40 GHz Pentium 4/HT PC with 2 GB RAM. Five random OVRP-TD instances and a real-life OVRP are solved with TS as well as CW and NI followed by rigorous LPO. The real-life instance is taken from a company that carries students of an elementary school in the metropolitan city of İstanbul. 22 vehicles pick 434 students from home in the morning and carry them back home in the afternoon.

In Tables 1 and 2, TS-NI denotes TS whose initial solution is generated by the nearest insertion method (NI), and TS-CW denotes TS whose initial solution is generated by CW. Their results are compared to see the initial solution effect. Best, average, and worst total distances and CPU seconds of 20 random runs are reported. Table 1 shows the results for five random test problems, and Table 2 shows the results for the school bus problem. TS is also compared against the pure CW and pure NI both of which are followed by LPO. CW+LPO denotes CW with

a series of LPO consisting of 1-1 exchange, 2-Opt move, 1-0 move and one more time 1-1 exchange. Similarly, NI+LPO denotes NI with the same LPO series.

Compared with the LPO-enhanced classical heuristics, TS finds better solutions with both initial solution generation methods. The only exception to this is the problem with 75 customers. Here, CW+LPO finds a slightly better total distance value than the best total distance of TS-NI. The test runs with the TS are inconclusive about the effect of the initial solution method as can be seen in the tables. Finally, the total Euclidean distance traveled by the school bus company in the real-life example amounts to 1,192,230 m according to their current routing plan. This distance in the best solution found by CW+LPO is 351,809 m, while average distance is computed as 354,987 m. TS-CW provides 70.5% improvement over the company's routing plan and 2.3% improvement over the CW+LPO heuristic.

4 Conclusion

In this paper, OVRP-TD is presented with the constraint that routes terminate at one of the driver nodes. The problem is solved with two classical heuristics enhanced by local post optimization, and with a tabu search meta-heuristic. In the latter, infeasible solutions are penalized dynamically, which distinguishes it from the previous meta-heuristics proposed in the OVRP literature. In test problems which range from 25 to 100 customers in size, tabu search with the Clarke-Wright initial solution performs better than the classical heuristics with local post optimization. Limited empirical evidence shows that the initial solution's effect on the quality of the final tabu search solution is problem-dependent. For the real-life school bus routing problem tabu search with Clarke-Wright initial solution improves the company's current routing plan by 70.5%.

5 References

Brandão J, (2004). A tabu search heuristic algorithm for open vehicle routing problem. European Journal of Operational Research 157: 552–564.

Clarke G, Wright JW (1964). Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12: 568–581.

Fu Z, Eglese R, Li LYO (2005). A new tabu search heuristic for the open vehicle routing problem. Journal of the Operational Research Society 56: 267–274.

Sariklis D, Powell S (2000). A heuristic method for the open vehicle routing problem. Journal of the Operational Research Society 51: 564–573.

Schrage L (1981). Formulation and structure of more complex/realistic routing and scheduling problems. Networks 11: 229–232.

Tarantilis CD, Diakoulaki D, Kiranoudis CT (2004). Combination of geographical information system and efficient routing algorithms for real life distribution operations. European Journal of Operational Research 152: 437–453.

Tarantilis CD, Ioannou G, Kiranoudis CT, Prasdacos GP (2004). Solving the open vehicle routing problem via single parameter meta-heuristic algorithm. Journal of the Operational Research Society: 1–9.

Tarantilis CD, Kiranoudis CT (2002). Distribution of fresh meat. Journal of Food Engineering 51: 85–91.

Table 5.1. Total distances and CPU times for random test problems

Problem Size	*TS-CW*			*TS-NI*			*Classical Heuristics*	
	minimum	**average**	**maximum**	**minimum**	**average**	**maximum**	**CW+LPO**	**NI+LPO**
25 customers	2,030.1[a]	2,080.0	2,193.9	1,782.9	2,071.9	2,193.8	2,030.2	2,033.2
5 drivers	5.09[b]	6.12	6.42	4.67	5.82	6.61	0.00	0.00
50 customers	1,104.9	1,116.8	1,140.6	1,081.1	1,108.5	1,157.6	1,148.2	1,142.5
10 drivers	1.59	13.19	22.13	1.75	11.05	19.70	0.0	0.00
75 customers	1,199.8	1,223.1	1,274.0	1,211.6	1,228.6	1,275.1	1,210.7	1,231.2
10 drivers	6.83	24.43	31.41	15.56	24.97	29.92	0.02	0.00
80 customers	1,467.8	1,501.9	1,530.6	1,476.3	1,521.7	1,558.2	1,562.9	1,622.1
8 drivers	6.91	29.18	50.09	6.94	22.80	43.42	0.00	0.00
100 customers	2,382.9	2,445.7	2,516.3	2,362.1	2,443.2	2,527.7	2,431.5	2,411.4
15 drivers	4.64	10.12	25.80	6.42	11.33	21.75	0.02	0.02

[a,b] In each cell, the first figure shows the total distance value while the figure below indicates the CPU time in seconds

Table 5.2. Total distances and CPU times for the real-life school bus routing problem

	TS-CW			*TS-NI*			*Classical Heuristics*	
	minimum	**average**	**maximum**	**minimum**	**average**	**maximum**	**CW+LPO**	**NI+LPO**
Total Distance [m]	351,809.4	354,986.7	358,381.6	355,695.4	366,647.7	386,925.8	359,917.4	397,202.0
CPU time [sec]	48.3	105.45	166.77	50.19	117.70	244.22	0.64	1.16