Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 148 (2003) 559–569
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

Vehicle routing problem with time windows and a limited number of vehicles

Hoong Chuin Lau [a,*], Melvyn Sim [b], Kwong Meng Teo [c]

[a] Department of Computer Science, School of Computing, National University of Singapore, 3 Science Drive 2, 117543 Singapore
[b] Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[c] Savy Technology Asia Pte Ltd., Technology Park@Chai Chee, 469001 Singapore

Abstract

This paper introduces a variant of the vehicle routing problem with time windows where a limited number of vehicles is given (m-VRPTW). Under this scenario, a feasible solution is one that may contain either unserved customers and/or relaxed time windows.

We provide a computable upper bound to the problem. To solve the problem, we propose a tabu search approach characterized by a holding list and a mechanism to force dense packing within a route. We also allow time windows to be relaxed by introducing the notion of penalty for lateness. In our approach, customer jobs are inserted based on a hierarchical objective function that captures multiple objectives.

Computational results on benchmark problems show that our approach yields solutions that are competitive to best-published results on VRPTW. On m-VRPTW instances, experiments show that our approach produces solutions that are very close to computed upper bounds. Moreover, as the number of vehicles decreases, the routes become more densely packed monotically. This shows that our approach is good from both the optimality as well as stability point of view.

Keywords: Tabu search; Heuristics; Routing; Combinatorial optimization; Vehicle routing problem with time windows

1. Introduction

Many practical transport logistics and distribution problems can be formulated as a vehicle routing problem whose objective is to obtain a minimum-cost route plan serving a set of customers with known demands. Each customer is assigned to exactly one vehicle route and the total demand of any route must not exceed the vehicle capacity.

To date, most of the proposed algorithms assume that the number of vehicles is unlimited, and the objective is to obtain a solution that either minimizes the number of vehicles and/or total travel cost. However, transport operators in the real world face resource constraints such as a fixed fleet. The question we like to ask is, if the given problem is over-constrained in the sense of insufficient vehicles, what constitutes a good solution and how may we find one?

In this paper, we provide some insights to this question. In our view point, it is desirable to have an algorithm that not only performs well given a

* Corresponding author.
E-mail addresses: lauhc@comp.nus.edu.sg (H.C. Lau), melvyn@mit.edu (M. Sim), kmteo@ap.savy.com (K.M. Teo).

0377-2217/03/$ - see front matter
doi:10.1016/S0377-2217(02)00363-6

standard VRPTW problem, but also handles over-constrained problems well in the following sense:

1. *Optimality:* It returns solutions which serve (or pack) as many customers as possible as the primary objective, while optimizing standard criteria such as the number of vehicles and distance travelled.
2. *Stability:* It degrades gracefully under constrainedness, i.e. when the number of vehicles is reduced, the *customer packing density*, defined as the average number of customers per vehicle in service, must be monotically increasing, although the total number of customers served will become less.

This paper proceeds as follows. We first introduce the problem (m-VRPTW) and a computable upper bound to the problem. We then present a tabu search approach with the following characteristics: (a) a holding list to accommodate unserved customers; (b) a mechanism that introduces new vehicles in stages so as to force denser customer packing within a route. We then extend the algorithm to a generalization of the problem with relaxed time windows.

In terms of computational results, experiments on VRPTW benchmark problems show that our approach can produce solutions that are very close to previous best-published results. What is more interesting perhaps is the performance on m-VRPTW instances. Results show that our approach produces solutions that are very close to computed upper bounds. Moreover, as the number of vehicles is reduced, the average number of customers per route is monotically increasing. This shows that our approach is good from both the optimality as well as *stability* point of view.

2. Literature review

The primary objective of m-VRPTW is to maximize the number of customers served, which is NP-hard, since finding it is a generalization of the multiple constrained knapsack problem. Although the classical VRPTW has been the subject of intensive research since the 80s, to our knowledge, there has been little research work on m-VRPTW.

We give some research developments in VRPTW. Solomon's insertion heuristics [18] is the seminal work behind heuristic construction algorithms. Many efficient heuristic and meta-heuristic approaches have been proposed recently, including the works of Chiang and Russell [5], Potvin and Rousseau [14], Rochat and Taillard [15], Taillard et al. [20], and Thangiah et al. [21]. More recently, Schulze and Fahle [19], Gehring and Homberger [9] proposed new parallel tabu search heuristics that enable large-scale VRPTW instances to be solved.

Several works have been carried out advocating the hybrid use of constraint programming and local search. For example, Pesant and Gendreau [13] applied constraint programming to evaluate the local neighborhood to find the best local moves. There is also constrained-directed local search proposed in [1,11,17]. In [17], for example, the author presented a method called large neighborhood search (LNS) for VRP in which a part of a given solution is extracted and then reinserted into the partial solution using a quasi-complete search process. If the reinsertion procedure generated a better solution, then the solution is kept. This process is repeated until certain stopping criterion is met. The result produced with this technique is competitive with other meta-heuristic approaches.

In terms of exact algorithms, Desrochers et al. [6] has applied column generation that was able to solve some 100-customer problems optimally. Based on this, Kohl et al. [12] developed a more efficient optimization algorithm by introducing a new valid inequality within a branch-and-cut algorithm, called k-path cuts, which solves 70 of the 87 Solomon benchmark problems to optimality. However, due to the exponential size of the solution space, it is unlikely that these optimization procedures can be used for larger-scale problems.

3. Problem definition and notation

The standard VRPTW problem is defined formally as: Given an undirected graph $G(V, A)$

where $V = \{v_0, v_1, \ldots, v_n\}$, v_0 is the depot, v_i, $i \neq 0$ is a customer with demand d_i, time windows (e_i, l_i) and service duration s_i; $A = \{(v_i, v_j) : i \neq j,\ v_i, v_j \in V\}$, each arc (v_i, v_j) having a travel distance (time) t_{ij}; and vehicle capacity Q, find a minimum set of vertex-disjoint routes starting and ending at depot v_0 such that each customer v_i is served by exactly one vehicle within its time windows, $\sum d_i$ for all customers v_i served by each vehicle is less than Q, and the total distance travelled is minimized (as the secondary objective function).

m-VRPTW is defined formally as: Given m (number of vehicles) and a VRPTW instance, find m or less routes with the primary objective function of maximizing total number of customers served, and the secondary objective function of minimizing the total distance travelled.

4. Upper bound for *m*-VRPTW

In this section, we determine an upper bound for the total number of customers that can be served by a given fixed number of vehicles. We propose an integer programming (IP) formulation. The IP formulation should be able to solve large-scale problems, yet not be overly simplified such that the gap of the bound from the optimum is too wide. We have adopted a formulation that accounts for the capacity constraints of the vehicles as well as the time constraints imposed by the latest return times of every vehicle to the depot. The upper bound is derived by solving a relaxation of m-VRPTW, formulated as follows.

Define $U = \{1, 2, \ldots, m\}$ to be the indices of the set of m serving vehicles and V^c to be the set of customer nodes (excluding the node at the depot). Define $r_i = \min_{j, j \neq i} t_{ij}$, $i, j \in V$ which is the traveling time from node i to its nearest neighbor. This quantity is used to lower bound the travel time from node i to any other node.

Let $w_i = l_i + s_i + t_{i0}$, $i \in V^c$ denote the time of return to the depot after serving node i as its last customer at its latest start time. Without loss of generality, assume that all w_is do not exceed the depot close time. Let $G = [g_1, g_2, \ldots, g_m]$ be a list of m unique customers in V such that $w_i \geqslant w_j$ for all $i \in G$ and $j \notin G$. Since vehicles are identical, the m elements of G represent the latest possible times of return to the depot for each of the m vehicles, for a solution to be feasible. The decision variables are denoted by $x_{ij} \in \{0, 1\}$, $i \in V^c$, $j \in U$ where $x_{ij} = 1$ if and only if the customer at node i is served by vehicle j. The following IP returns the upper bound of the total number of customers served by all the vehicles:

$$\begin{aligned} \max \quad & \sum_{i \in V^c} \sum_{j \in U} x_{ij} \\ \text{s.t.} \quad & \sum_{j \in U} x_{ij} \leqslant 1, \quad \forall i \in V^c \end{aligned} \tag{1}$$

$$\sum_{i \in V^c} x_{ij} d_i \leqslant Q, \quad \forall j \in U \tag{2}$$

$$\sum_{i \in V^c} x_{ij}(s_i + r_i) + r_0 \leqslant w_{g_j}, \quad \forall j \in U \tag{3}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in V^c,\ j \in U$$

Constraints (1) state that all customers must be assigned to at most 1 vehicle. Constraints (2) ensure that the vehicle capacity constraint is not violated. Constraints (3) impose some linear timing constraints: it says that for each vehicle, the earliest possible time of returning to the depot (induced by the assignment x) cannot exceed the latest possible return time (imposed by G). Note that in this formulation, we ignore the full considerations of time window constraints and the actual travel time between two nodes. The bound is thus expected to be less effective on test cases with tight time window constraints.

Note that the above formulation gives us a *constrained knapsack problem*, which is NP-hard. Fortunately, many variants and generalizations of the knapsack problem have been well-studied there exist exact algorithms which are computationally efficient, such as [22].

5. Standard two-phase method

Most of recently published VRPTW heuristics are two-phase approaches. First, a construction heuristic is used to generate a feasible and as good as possible initial solution. Then, an iterative improvement heuristic is applied to this solution.

It generates successive solutions by searching the neighborhood of the current solution. In the second phase, various methods are then used to prevent the algorithms from being trapped at local optimal and to explore a larger search space.

The construction phase involves insertion of all the customers into a set of feasible vehicles routes. The purpose of the construction phase is to provide an initial feasible solution. Usually, each customer will be inserted in turn to the route that gives the minimum additional cost or distance at that instance. The order of customer selection then defines the heuristics, some of which are listed follows:

- *Nearest insertion rule:* Next nearest unserved customer will be selected.
- *Earliest ready time rule:* Next unserved customer with the earliest ready time will be selected.
- *Window tightness rule:* Next unserved customer with the tightest time window will be selected.

Observe that these heuristics assume that there are enough vehicles to serve all the customers. As such, for over-constrained problems, these heuristics may fail to deliver satisfactory solutions.

Given the initial feasible solution from phase I, the phase II route improvement phase involves an iteration of moving from a feasible solution to its feasible neighborhood until certain terminating condition is met. In this phase, the heuristics are defined by the neighborhood structure, the choice of the next move, and the terminating condition.

The simplest approach of the steepest descent algorithm chooses the best and improved solution among all the neighboring solutions at every iteration. However, the algorithm would very quickly be trapped within a local minimum.

The neighborhood structure used is usually a k-opt local search procedure, where k refers to the number of customers/arcs that can be interchanged from the initial solution to its neighborhood solutions. Some of these interchanges are

- *Relocate:* Customer from one route transfers to another route.
- *Exchange:* Customer from one route exchanges position with another customer in another route.

The quality of a two-phase method depends on whether the choice of construction and improvement heuristics is a good fit to the nature of the search space. The construction heuristics should produce a good enough initial location such that the improvement algorithm starts in a region where good solutions can be achieved. Subsequently, the improvement heuristics would need to be able to bypass sufficiently many local minima to terminate at a good solution.

6. Proposed algorithm

The above-mentioned two-phase method would normally have to work differently for an over-constrained problem. One way is to use the insertion heuristic to determine whether the problem instance is feasible. Following which, we have two sets of heuristics to handle separately the infeasible case and the feasible case. Another approach is to increase enough vehicles so as to serve all customers, and then, through subsequent heuristics, try to obtain a subset of the solution that maximizes the number of customers that can be served.

However, there are pitfalls in both of these approaches. Though the first seems credible, it demands extensive use of heuristics and does not value-add in terms of algorithm development. The second approach is not addressing the issue of infeasibility directly, and is therefore unlikely to give a good solution consistently.

We therefore seek to have a generalized method to handle *both* feasible and infeasible problem instances. The approach to such an algorithm lies in the introduction of the holding list, the data structure that contains unserved customers. Although the idea of holding list is not new at first sight (for example, the ILOG Dispatcher product uses the same idea), our overall algorithmic strategy of transferring customers back and forth the holding list under our tabu search strategy (see below) is, to our knowledge, a novel idea.

6.1. Holding list

The holding list contains the list of the customers that are not served in the current solution. The idea of introducing a holding list is triggered by the role of artificial variables in the phase I of the simplex algorithm. A feasible solution of the VRPTW is found when all the customers are driven out of the holding list, which is analogous to driving out all the artificial variables from the simplex tableau.

The holding list will induce an extended neighborhood search space, which includes the following moves, in addition to the basic *relocate* and *exchange* moves discussed in Section 5:

- *Relocate from holding list:* Transferring a customer from holding list to an existing route.
- *Relocate to holding list:* Transferring a customer from an existing route to the holding list.
- *Exchange with holding list:* Exchanging a customer from an existing route with another customer in the holding list.

The holding list is similar to a ''phantom'' route which participates in the regular local search, with a variant that insertion of a customer to the holding list is always feasible and does not incur any cost. The customers of a selected route will be searched completely for possible of transfer to/from or exchange with customers in the holding list. The next accepted move is determined by using a best improvement strategy depicted in the hierarchical cost structure.

6.2. Hierarchical cost structure

m-VRPTW introduces the additional objective of maximizing the number of served customers and minimizing lateness (if time windows can be relaxed). This implies that the objective function becomes a composite function:

- maximize total number of customers served,
- minimize total number of customer served late (if allowed),
- minimize total lateness duration (if allowed),
- minimize total number of vehicle used,
- minimize total distance traveled.

One possible way of dealing with multi-criteria objective is to define a composite cost function with different weights for the different cost parameters. However, setting the proper weights becomes a tricky (or almost impossible) mission, and the resulting function becomes meaningless to interpret.

Our approach is to define a *hierarchical cost structure*. For example, serving more customers is always better regardless of the number of vehicles used. Although one can argue that the 'big M' approach on the composite cost function can be used to enforce the hierarchy, we believe our approach is a cleaner way. We propose a hierarchical cost structure in decreasing order of priority of the above list of objectives.

In hierarchical cost comparison between two states (during local search), the state with greater preference down the hierarchy of importance assumes a 'lower cost'. As opposed to the composite cost function, where the total composite cost is computed, hierarchical cost is never computed. This is because in local search techniques, computation of the absolute cost is not needed. Rather, the hierarchical cost is used for comparing the current solution state with the previous best solution state.

6.3. Tabu search strategy

Although many construction and local improvement techniques have been reported to solve VRPTW problems, it is unclear how these heuristics will behave when constrained by a limited vehicle fleet. Particularly, it is unclear how the two-phase approach can work co-operatively in ensuring the ultimate solution to have good customer packing while not exceeding the prescribed vehicle limit. For instance, if we use too few vehicles in the construction phase, it may limit the search space of the local improvement phase; on the other hand, if we use up to the maximum allowable number of vehicles, then it leaves little room for the local search phase to add more unserved customers, since all routes have been used up. The outcome

may be a solution where some of the routes are relatively loosely packed, but no more customers can be added to any one of them unless drastic changes are made to the solution, which local search, by its nature, is incapable of realizing.

To deal with limited vehicle fleet, our strategy is to meld the two-phase approach into a nested approach. The idea is to increase the number of vehicles *in stages* and at each stage, apply standard tabu search to maximize the number of customers to be inserted onto those vehicles. Within each stage, the number of vehicles is fixed and hence the search will not consider adding a vehicle to serve unassigned customers. In other words, we steer the tabu search to favor packing of customers within the existing routes. With certain abuse of terminology to draw parallelism with duality, the dual of the vehicle minimization problem is the maximization of customer packing problem. Empirical results have shown that this strategy has tremendous improvement of packing density on Solomon test cases, without having to rely on a set of good construction heuristics.

With the incorporation of the holding list, it is easy to implement the tabu search strategy discussed above. The hierarchical cost structure favors transfer of customers from the holding list to the routes. Unlike many tabu search algorithms, where the search space is always feasible, the holding list is a neat way of incorporating local search towards a path of feasibility that favors customer packing.

Let **TS** denote one iteration of a standard tabu search procedure with the search neighborhood and hierarchical cost structure discussed in Sections 6.1 and 6.2 respectively. The tabu list stores customers that have been moved within the preceding number of iterations defined by the tabu length. A move is tabu if and if only the customers are in the tabu list *and* it is not aspired by being better than the best solution so far, in the sense of the hierarchical cost function defined in Section 6.2.

Let *StepSize* denote the additional number of vehicles introduced in each stage. Let *numVeh* denote the current number of vehicles used, initialized to *StepSize*. Let *CountLimit* denote the maximum number of non-improving moves using *numVeh* vehicles.

The algorithm proceeds as follows (refer to Algorithm A). The holding list initially contains all customers of the given instance. We introduce *StepSize* additional number of vehicles (i.e. empty routes) in each stage. Each stage is implemented by the **while** loop (i.e. steps (3) to (6)). When **TS** is called in step (4), it will return a solution which differs from the previous solution by a local move made with respect to the neighborhood and the hierarchical cost structure. In step (5), a better solution means that the hierarchical cost objective value is better than that of the best solution found so far. If a better solution has not been found after *CountLimit* consecutive tries, then the stage ends, and the algorithm proceeds to the next stage by adding more vehicles.

Algorithm A

(1) **until** holding list is empty **or** $numVeh = m$
(2) set $Count = 0$
(3) **while** $Count \leqslant CountLimit$
(4) call **TS** based on $numVeh$ vehicles
(5) **if** better solution found **then** set $Count = 0$
else set $Count = Count + 1$
(6) **endwhile**
(7) set $numVeh = \min(numVeh + StepSize, m)$
(8) **end until**

6.4. m-VRPTW with relaxed TWs

The logical extension to solving m-VRPTW problems is to relax the time window constraints. In other words, we allow late arrivals after the intended due time to increase the number of customers that could be served. Although one could perceive this as a separate objective that may entail a separate algorithm, our challenge is to incorporate this feature into one seamless generalized algorithm.

We define *late period* to be the amount of lateness between the time windows upper bound and the actual arrival time (and 0 if the arrival time lies within the time window). We impose the late period as a *soft* time constraint which can be violated. This is done by incorporating the number of late arrivals and the total late period into the hierarchical cost function.

Intuitively, by relaxing the time window constraints, we would expect to have improved solutions (in terms of its objective value). However, from experimentation with our proposed local search technique, we discovered an anomaly that the solution for some cases become worse if we relax the constraints as it is. This is illustrated in the sample run in solving an over-constrained Solomon benchmark problem R103 with 13 vehicles. Without relaxing the time windows, the total late period is 0, while with relaxed time windows, the late period became 77.8 units! This contradicts with the intuition that with relaxed time windows, the solution obtained from local search will always be an improvement. As local search techniques do not guarantee global optimality, it is likely that even with relaxed constraints the reported solution may be worse off.

To deal with this anomaly, we propose Algorithm B. Here, we first solve the problem *without* relaxing the time windows. The solution then becomes the initial feasible solution for the problem with relaxed time windows. In the latter case, more customers may be inserted from the holding list, albeit at the expense of relaxed time windows. In this manner, we can always guarantee improvement, if any, on the relaxed problem. Table 1 presents the new result obtained for R103 under this new scheme.

Algorithm B
(1) call Algorithm A
(2) **if** holding list not empty
(3) relax time windows constraints
(3) set $Count = 0$
(4) repeat steps (2) to (7) of Algorithm A

Table 1
Consistency of solution with relax TWs

	Without relaxed time windows	With relaxed time windows
# Vehicles	13	13
# Customers inserted	99	100
# Customers served late	0	2
Total late period	0	15.792

7. Results and analysis

In this section, we present experimental results of applying our algorithm to solve both the standard VRPTW as well as *m*-VRPTW problems. In our experiments, we set the tabu length to be 100. We set the values of *CountLimit* and *StepSize* in Algorithm A to be 500 and 1 respectively. We refer to our implementation as the OV method.

7.1. Performance on VRPTW problems

Here, we test the performance of Algorithm A on the set of 56 Solomon's test cases with 100 customers. The run time of the algorithm tested on a Pentium II 433 machine is about 1 min on average.

We first compare our results with the overall best-published heuristics results. [1] The comparison of solutions is presented in Table 2, where the Best and OV columns contain the best published and our results respectively.

Next, we also compare our results with the specific results of (a) Rochat and Taillard [15] (RT), (b) Chiang and Russell [3] (CR), (c) Taillard et al. [20] (TBGGP), (d) Homberger and Gehring [10] (HG), and (e) Cordeau et al. [4] (CLM). A summary of the comparison is given in Table 3. We observe that although our results are in general inferior compared to these results, they are within only several percent worse on average. We believe this is justifiable, in two sense. First, our goal is in obtaining results within 1 min on average and hence set the maximum number of iterations to 500. This is in contrast to the other methods which typically require hours of CPU time on compatible

[1] Contributions to the best-published results are taken from: Cordeau et al. [4]—R107, R108, RC104, RC106, R204, RC201, RC207, Chiang and Russell [3]—R207, Homberger and Gehring [10]—R103, R109, R112, R201, R203, R208, R210, R211, C202, RC203, RC204, RC205, R110, Rousseau et al. [16]—R111, RC105, R202, R205, R209, RC206, RC208, Rochat and Taillard [15]—R101, R104, R105, R106, C101, C102, C103, C104, C105, C106, C107, C108, C109, RC103, R206, C201, C202, C203, C204, C205, C206, C207, C208, R102, Taillard et al. [20]—RC101, RC102, RC107, RC108.

Table 2
OV solutions against best-published results

	Best		OV			Best		OV	
	Vehicles	Distance	Vehicles	Distance		Vehicles	Distance	Vehicles	Distance
C101	10	828.94	10	828.94	R112	9	1003.73	10	1021.95
C102	10	828.94	10	834.64	R201	4	1252.37	4	1292.53
C103	10	828.06	10	834.56	R202	3	1191.70	3	1158.98
C104	10	824.78	10	846.32	R203	3	942.64	3	980.70
C105	10	828.94	10	828.94	R204	2	849.62	3	847.74
C106	10	828.94	10	828.94	R205	3	994.42	3	1146.80
C107	10	828.94	10	828.94	R206	3	912.97	3	1007.00
C108	10	828.94	10	828.94	R207	2	914.39	3	869.94
C109	10	828.94	10	828.94	R208	2	731.23	2	790.46
C201	3	591.56	3	591.56	R209	3	909.86	3	1020.06
C202	3	591.56	3	619.36	R210	3	955.39	3	1032.65
C203	3	591.17	3	604.01	R211	2	910.09	3	866.10
C204	3	590.6	3	644.23	RC101	14	1696.94	15	1657.46
C205	3	588.88	3	601.43	RC102	12	1554.75	13	1535.79
C206	3	588.49	3	588.88	RC103	11	1262.02	12	1386.03
C207	3	588.29	3	608.94	RC104	10	1135.48	10	1213.25
C208	3	588.32	3	591.83	RC105	13	1633.72	15	1625.13
R101	19	1650.8	20	1765.00	RC106	11	1427.13	12	1426.07
R102	17	1486.12	18	1548.61	RC107	11	1230.54	11	1330.59
R103	13	1292.85	14	1258.34	RC108	10	1139.82	10	1175.88
R104	10	982.01	10	1018.48	RC201	4	1406.94	4	1468.46
R105	14	1377.11	15	1462.69	RC202	3	1389.57	4	1222.69
R106	12	1252.03	12	1328.66	RC203	3	1060.45	3	1171.88
R107	10	1113.69	12	1160.08	RC204	3	799.12	3	839.32
R108	9	964.38	10	1045.83	RC205	4	1302.42	4	1338.70
R109	11	1194.73	13	1259.09	RC206	3	1153.93	3	1201.27
R110	10	1124.4	11	1127.70	RC207	3	1062.05	3	1139.48
R111	10	1096.72	11	1097.10	RC208	3	829.69	3	985.60

Table 3
Summary of comparison

	RT	CR	TBGGP	HG	CLM	OV
C1 mean no of vehicles	10.00	10.00	10.00	10.00	10.00	10.00
C1 mean distance	832.59	828.38	828.38	828.38	828.38	832.13
C2 mean no of vehicles	3.00	3.00	3.00	3.00	3.00	3.00
C2 mean distance	595.38	591.42	589.86	589.86	589. 86	589.86
R1 mean no of vehicles	12.83	12.17	12.17	11.92	12.08	12.16
R1 mean distance	1208.43	1204.19	1209.35	1220.97	1210.14	1211.55
R2 mean no of vehicles	3.18	2.73	2.82	2.73	2.73	3.00
R2 mean distance	999.63	986.32	980.27	968.55	969.58	1001.12
RC1 mean no of vehicles	12.75	11.88	11.50	11.50	11.50	12.25
RC1 mean distance	1381.33	1397.44	1389.22	1388.24	1389.78	1418.77
RC2 mean no of vehicles	3.62	3.25	3.38	3.25	3.25	3.37
RC2 mean distance	1207.37	1229.54	1117.44	1140.43	1134.52	1170.93

machines. Second, this work is not aimed at beating the best VRPTW results, but rather on proposing an algorithm that works well when limited to a number of vehicles.

7.2. Performance on m-VRPTW

To assess the performance of our algorithm on *m*-VRPTW, we will measure the total number of

Table 4
Number of customers served using our algorithm

Number of vehicles	10	9	8	7	6	5	4
C101	100	92	84	75	66	57	47
C102	100	99	92	84	72	60	49
C103	100	99	93	84	72	60	48
C104	100	99	94	84	72	60	48
C105	100	92	84	75	66	57	47
C106	100	93	86	77	68	58	47
C107	100	93	85	76	66	57	47
C108	100	96	87	78	69	59	48
C109	100	99	92	82	72	60	48

Table 5
Computed upper bound of the number of customers served

Number of vehicles	10	9	8	7	6	5	4
C101	100	99	91	81	70	59	48
C102	100	99	95	90	78	65	52
C103	100	99	95	90	78	65	52
C104	100	99	95	90	78	65	52
C105	100	99	95	86	75	63	51
C106	100	99	95	86	75	63	51
C107	100	99	95	87	75	63	51
C108	100	99	95	87	75	63	51
C109	100	99	95	90	78	65	52

customers that can be served per unit vehicle against a computable upper bound. To compute upper bounds, we use a software developed in-house (based on MATLAB) which can compute optimal solutions for the IP formulation given in Section 4. We will demonstrate that our algorithm is optimized in the sense that the solutions produced are close to the upper bound, and is stable in that the customer packing density [2] increases monotonically when the number of vehicles is reduced.

To analyze the performance of the algorithm in packing density against the upper bound, we select the C1 set of the Solomon's test cases. For this set, it is easy to verify from the vehicle capacity constraint that the minimum number of vehicles required to serve all the customers in all the cases is 10. With the uniformity in the number of vehicles in service, we compute the unbiased mean of the packing density for all the test cases in the C1 set. By starting from the fleet size of 10 vehicles for every test case in C1, we gradually reduce the number of vehicles in service and determine the unbiased mean of the packing density for all the test cases. The comparison is given in the Tables 4 and 5. Experiments on other Solomon test sets yield similar behavior, but since their vehicle requirements are less uniform, we do not present their results in this paper.

The results of comparison are summarized in Figs. 1 and 2. The two curves represent respectively the values obtained by our algorithm (i.e.

[2] Recall that this refers to the total number of served customers in the solution divided by the total number of vehicles used.

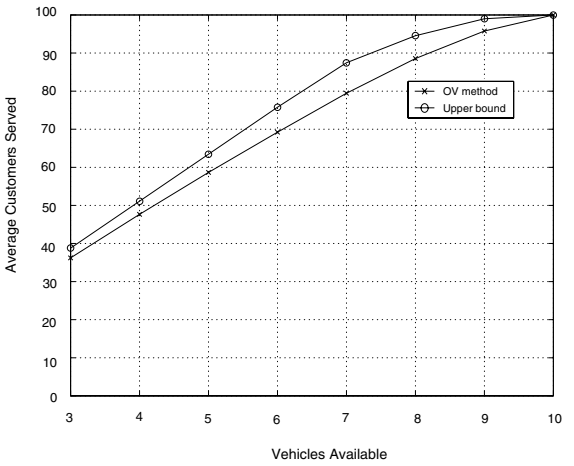

Fig. 1. Average customers served vs vehicle availability.

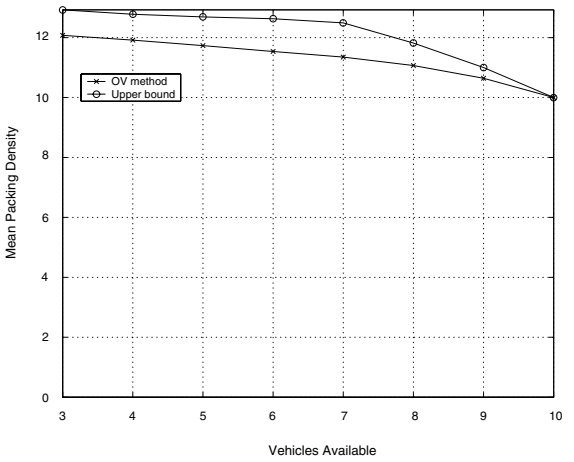

Fig. 2. Customer packing density vs vehicle availability.

OV method) and the computed upper bounds. From Fig. 1, we observe that the proposed algorithm generates solutions that are substantially close to the upper bound. In term of packing density illustrated in Fig. 2, the average gap from the upper bound is less than one customer per vehicle. Moreover, the solution quality in response to tighter vehicle constraints is consistent with the upper bound. This demonstrates the stability of our algorithm.

7.3. Performance on m-VRPTW with relaxed time windows

For *m*-VRPTW problems with relaxed time windows, we apply Algorithm B. For stability analysis, we again use the set of C1 test cases. We set the relaxed time window of each customer to be its early start time and the closing time of the depot, i.e. we do not allow any customer to be served earlier than its prescribed early start time, but allow every customer to be served late potentially, but not later than the closing time of the depot. The results are summarized in Figs. 3 and 4, which contain one additional curve in between the previous two curves. Our algorithm again generates consistent results with more customers served and improved packing density when compared with the case with unrelaxed time windows.

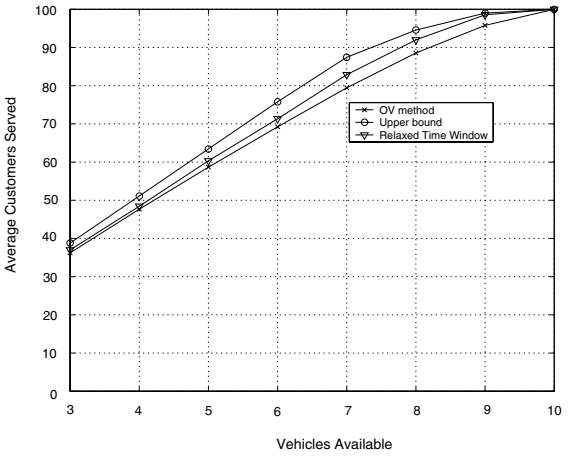

Fig. 3. Average customers served under relaxed TWs.

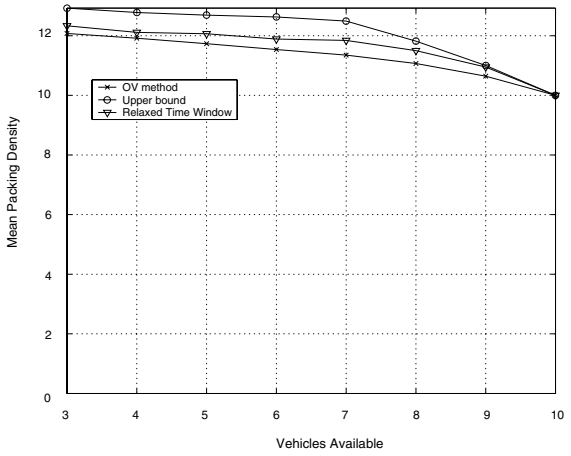

Fig. 4. Customer packing density under relaxed TWs.

8. Conclusion

In this paper, we considered a variant of VRPTW constrained by a limited vehicle fleet, which we believe is a more realistic problem in logistics. We presented an analytical upper bound for that formulation, and showed that our tabu search approach came fairly close to the upper bound. This algorithm is also good from the stability point of view. We also showed that the same algorithm could be used to give reasonably good results for the standard VRPTW problem. Hence, in terms of achievements, we have opened a new variant of a well-studied research problem, and provided the first step in tackling this problem. Future works include closing the optimality gap, by perhaps a tighter upper bound or more elegant algorithms; experiments on larger-scale problem instances.

Acknowledgement

We like to thank Prof Sun Jie for his guidance and comments on early drafts of this paper.

References

[1] B.D. Backer, P. Kilby, P. Prosser, P. Shaw, Solving vehicle routing problems using constraint programming and metaheuristics, Journal of Heuristics 6 (4) (2000) 501–525.

[3] W.C. Chiang, R.A. Russell, Simulated annealing metaheuristics for the vehicle routing problem with time windows, Annals of Operations Research 63 (1996) 3–27.
[4] J.F. Cordeau, G. Laporte, A. Mercier, A unified tabu search heuristic for vehicle routing problems with time windows, Working Paper CRT-00-03, Centre for Research on Transportation, Montreal, Canada, 2000.
[5] A.W. Chiang, R. Russell, A reactive tabu search metaheuristic for the vehicle routing problem with time windows, INFORMS Journal on Computing 9 (4) (1997).
[6] M. Desrochers, J. Desrosiers, M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, Operations Research 40 (1992) 340–354.
[9] H. Gehring, J. Homberger, Parallelization of a two-phase metaheuristic for routing problems with time windows, Presented at 5th APORS Conference, Singapore, 2000.
[10] J. Homberger, H. Gehring, Two evolutionary metaheuristics for the vehicle routing problem with time windows, INFOR 37 (1999) 297–318.
[11] P. Kilby, P. Prosser, P. Shaw, A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints, Journal of Constraints 5 (4) (2000) 389–414.
[12] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, F. Soumis, *k*-path cuts for the vehicle routing problem with time windows, Transportation Science 33 (1999) 101–116.
[13] G. Pesant, M. Gendreau, A constraint programming framework for local search methods, Journal of Heuristics 5 (3) (1999) 255–279.
[14] J.Y. Potvin, J.M. Rosseau, A parrallel route building algorihtm for the vehicle routing and scheduling problem with time windows, European Journal of Operational Research 66 (1993) 331–340.
[15] Y. Rochat, E. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1 (1995) 147–167.
[16] L.M. Rousseau, M. Gendreau, G. Pesant, Using constraint-based operators to solve the vehicle routing problem with time windows, Journal of Heuristics 8 (1) (2002) 43–58.
[17] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: Proceedings of CP'98, Springer-Verlag, 1998, pp. 417–431.
[18] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Operations Research 35 (2) (1987) 254–265.
[19] J. Schulze, T. Fahle, A parallel algorithm for the vehicle routing problem with time window constraints, Annals of Operations Research 86 (1999) 585–607 (special issue).
[20] E. Taillard, P. Badeau, M. Gendreau, F. Geurtin, J.Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows, Transportation Science 31 (1997) 170–186.
[21] S.R. Thangiah, I.H. Osman, T. Sun, Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows, Working Paper UKC/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury.
[22] R.L.M.J. van de Leensel, C.P.M. van Hoesel, J.J. van de Klundert, Lifting valid inequalities for the precedence constrained knapsack problem, Mathematical Programming 86 (1) (1999) 161–185.