Available online at www.sciencedirect.com

ScienceDirect

Procedia Social and Behavioral Sciences 2 (2010) 5931–5943

Procedia
Social and Behavioral Sciences

The Sixth International Conference on City Logistics

Exact solution for the vehicle routing problem with semi soft time windows and its application

Ali Gul Qureshi[a*], Eiichi Taniguchi[a], Tadashi Yamada[a]

[a]*Department of Urban Management, Kyoto University, Katsura campus C-1, Kyoto 615-8540, Japan*

Abstract

This paper reflects on a soft time windows variant of the Vehicle Routing Problem (VRP) that only considers penalties on late arrival while waiting on early arrival is allowed without cost, namely the Vehicle Routing and scheduling Problem with Semi Soft Time Windows (VRPSSTW). A column generation (Dantzig-Wolfe decomposition) based exact optimisation approach is presented to obtain exact solutions for the VRPSSTW. While the computation time for the exact approach is adequate for smaller instances, the computation time for large-sized problems is very large for the exact solution technique, which favours the use of heuristics for the soft time windows variants of the VRP, in city logistics-related research. Therefore, as an application, this paper shows that how these exact solutions can be used as benchmark solutions to evaluate the performance of heuristics on smaller instances before applying these heuristics to large VRPSSTW instances. Errors between the optimal solutions and approximate solutions were used to calibrate a genetic algorithm heuristic for the VRPSSTW. Large errors instigated a change in the initialization procedure in the genetic algorithm heuristic, which resulted in the improved performance in terms of cost and computation time.

Keywords: City logistics; vehicle routing; column generation; soft time windows

1. Introduction

The aim of city logistics is to optimise urban freight movement with respect to the public and private costs and benefits (Taniguchi et al., 2001). Traffic congestion, parking issues, accidents and environmental problems are also considered in the design and evaluation of city logistics-related measures and policies. The Vehicle Routing and scheduling Problem with Time Windows (VRPTW) is a typical route optimisation technique employed in city logistics. It is also used to evaluate many types of city logistics measures, for example, the optimal location of logistics terminals (Yamada et al., 2001) and cooperative delivery systems (Qureshi and Hanaoka, 2005), etc. City logistics focuses on practical logistics problems, which are often set in soft time windows environment where late

* Corresponding author. Tel.: +81-75-383-3420; fax: +81-75-950-3800.
E-mail address: aligul@kiban.kuciv.kyoto-u.ac.jp.

1877-0428 © 2010 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.sbspro.2010.04.008

deliveries (with respect to time windows $[a_i, b_i]$) are possible at some penalty cost; this scenario is modelled using the Vehicle Routing and scheduling Problem with Soft Time Windows (VRPSTW).

At present, no efficient exact optimisation approach is available for the VRPSTW due to its complex cost structure that includes the early and late arrival penalties based on the arrival time, and mostly heuristics (approximate) solutions have been used for the VRPSTW in city logistics-related research. Solutions for the large instances are required within a reasonable time in city logistics research, which has also favoured the predominant use of heuristics in this field.

However, heuristics are sometimes faster and more easily implemented, yet they do not guarantee to identify the exact solution or state how close to the exact solution a particular feasible solution is (Thompson and Duin, 2003). On the other hand, many exact optimisation approaches have been reported for the hard time windows variant (VRPHTW) in the literature such as column generation (Feillet et al., 2004) and Lagrangian relaxation (Kallehauge et al., 2006), even though it not so practical due to the stringent restriction on late deliveries. The simple cost structure of the VRPHTW that allows waiting without any penalty and follows the cost triangular inequality, makes it attractive to the researchers in the exact optimisation field.

This paper presents an exact optimisation approach for a variant of the VRPTW named, the Vehicle Routing and scheduling Problem with Semi Soft Time Windows (VRPSSTW). It retains the best features from both the VRPSTW and the VRPHTW, the cost triangular inequality is reserved by allowing waiting at no cost (similar to the VRPHTW), whereas the practicality of the VRPSTW is incorporated by allowing late deliveries with late arrival penalties. This type of time window is very important from a practical point of view for logistics managers, as it offers more economical use of resources at minimum delays in delivery due to the penalties for late arrival. A branch and price algorithm is presented for the VRPSSTW by extending the column generation solution of the VRPHTW. These exact VRPSSTW solutions can be used to calculate the errors (between optimal and approximate solutions) and to calibrate the heuristics on smaller instances before the heuristics are used to solve large scale instances.

Therefore, as an application, exact solutions of benchmark problems have also been compared with approximate solutions from a simple genetic algorithm heuristic (hereafter referred as GA-1) and the relative error in delivery costs and computation time requirements are reported. Large errors in the GA-1 results required some changes to improve the quality of heuristics solutions. Consequently, another genetic algorithm heuristic (hereafter referred as GA-2) is also evaluated, which uses a different initialization procedure than the GA-1. Such an evaluation of heuristics on the exact benchmark solutions will provide researchers in the city logistics field with more confidence and stronger foundations to appraise and support city logistics-related policies, which are assessed using the heuristics solutions of the VRPSSTW.

2. Literature Review

Column generation-based algorithms have been very popular in the exact optimisation field for the VRPHTW. Dantzig-Wolfe decomposition (commonly known as column generation) of the VRPHTW results in the set partitioning master problem and an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) as its sub-problem. While the master problem remains the same, many researchers have worked with various shortest path variations as sub-problems in their column generation schemes for the VRPHTW. For instance, Desrochers et al. (1992) presented the first column generation-based approach for the VRPHTW, using 2-cycle elimination while solving the relaxed shortest path sub-problem. Irnich and Villeneuve (2003) used a relaxed shortest path sub-problem with k-cycle elimination (with $k \geq 3$) in their column generation scheme for the VRPHTW. Recently, Feillet et al. (2004) and Chabrier (2006) have used the ESPPRC as the sub-problem in a very efficient Dantzig-Wolfe decomposition-based approach for the VRPHTW.

Using the same principle, this study decomposes the VRPSSTW into a set partitioning master problem and an Elementary Shortest Path Problem with Resource Constraint and Late Arrival Penalties (ESPPRCLAP) sub-problem. To our knowledge, no exact solution algorithm for the VRPSSTW is available; however, Taillard et al. (1997) and Gendreau et al. (1999) have used similar semi-soft time windows settings in their tabu search heuristics, arguing for their practicality and a trade-off between fleet size and service quality to customers (i.e., delivery within time windows).

Earlier research on exact solutions incorporating soft time windows have focused on the schedule optimisation of a given fixed path considering linear (Sexton and Bodin, 1985) and/or generalized convex penalty functions (Dumas et al., 1990). Recently, Tagmouti et al. (2007) presented an arc routing problem with soft time windows, where vehicles are not allowed to wait along their routes. In their column generation scheme, they have used a modified labelling algorithm for the Shortest Path Problem with Time Windows and Time Costs (SPPTWTC) sub-problem earlier given by Ioachim et al. (1998). The vehicle arrival pattern has been represented by a continuous variable resulting in very high computation times and limited the maximum size of problem solved to 40 customers. Unlike our column generation scheme that iteratively adds the feasible routes of marginal negative cost from the sub-problem to the set partitioning master problem, Calvete et al. (2007) exploited goal programming to enumerate all feasible routes in the first stage and then used the set partitioning problem to solve a VRP with soft time windows, heterogeneous fleet and multiple objectives. In a similar approach, Fagerholt (2001) solved a ship-scheduling problem with soft time windows. The Traveling Salesman Problem with Capacity, Hard Time Windows and Precedence Constraint (TSP-CHTWPC) was used to enumerate all feasible routes and then their schedules were optimised using soft time windows, before optimising a set partitioning problem based on these routes.

As said earlier, the bulk of the research targeting the soft time windows is in heuristics domain such as local search heuristics (Hashimoto et al., 2006), Tabu search (Duin et al., 2007) and heuristics presented by Balakrishnan (1993) based on the nearest neighbour, Clarke-Wright savings and space-time rules. Particularly, genetic algorithm (GA) based heuristics have been used abundantly in solving complex and close to real life VRPSTW instances in city logistics; for example, Taniguchi and Heijden (2000) used GA solutions of the VRPSTW to evaluate many city logistics measures such as cooperative delivery systems (CDS) and load factor controls. Taniguchi et al. (2001) used a GA to solve a Probabilistic VRPSTW (VRPSTW-P) that incorporates the uncertainties of travel times on a road network. Yamada et al. (2004) used a similar GA approach for VRPSTW-P to study the travel time reliability of a road network.

Utilising the VICS (Vehicle Identification and Communication System) data and the data from 66 days operation of probe pickup/delivery trucks, Ando and Taniguchi (2007) applied the VRPSTW-P and its GA solution to an actual delivery system in Osaka, Japan. Yamada et al. (2001) combined logistics terminal location, CDS and the VRPSTW into a single framework. The combined model was solved using a GA heuristic, and the results were compared to a base case that did not use the CDS. Qureshi et al. (2009) presented a hybrid genetic algorithm embedded in the flexible framework of the column generation scheme in an effort to reduce the computation time of the VRPSTW heuristics solution. A detailed review and discussion concerning the VRPSTW can be found in Taniguchi et al. (2001), while excellent reviews of the heuristic methods applied to the VRPTW are available in Braysy and Gendreau (2005a; 2005b).

3. Model Formulation

The VRPSSTW is defined on a directed graph $G = (V, A)$. The vertex set V includes the depot vertex 0 and set of customers $C = \{1, 2, \ldots, n\}$. The arc set A consists of all feasible arcs (i, j), $i, j \in V$. Both cost c_{ij} as well as time t_{ij} are associated with each arc $(i, j) \in A$. Time t_{ij} includes the travel time on arc (i, j) and the service time at vertex i, and a fixed vehicle utilisation cost is added to all outgoing arcs from the depot, i.e. in c_{0j}, $j \in C$. A set of identical vehicles (represented by K) with capacity q stationed at the depot, is available to service customers' demands. With every vertex of V there is an associated demand d_i, with $d_0 = 0$, and a time window $[a_i, b_i]$ representing the earliest and the latest possible service start times.

This study incorporates the semi-soft time windows constraint by extending the latest possible service start time b_i to b_i' as shown in Figure 1. To obtain the maximum limit b_i', a maximum late arrival penalty equivalent to the cost of a dedicated single vehicle route only serving the concerned vertex was used. Taking c_l as the unit late arrival penalty cost, the maximum limit of b_i' can be defined as equation (1).

$$b_i' = \min\left[b_0 - t_{i0},\ \ b_i + \frac{(c_{0i} + c_{i0})}{c_l} \right] \tag{1}$$

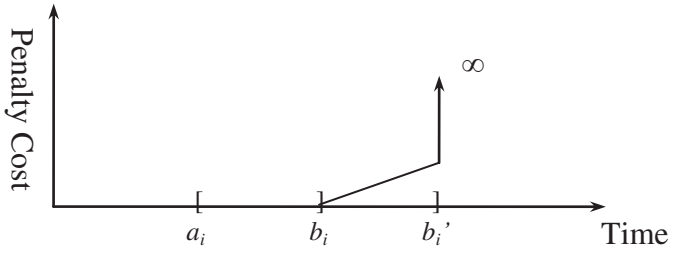

Figure 1 Penalty cost function for the VRPSSTW

Let s_{jk} define the service start time at a vertex $j \in C$ by a vehicle $k \in K$. For all arcs $(i, j) \in A$, except arcs which violate the inequality, $a_i + t_{ij} \le b_j'$, the modified time dependent travel cost, c'_{ijk} is defined as a function of s_{jk} (equation (2)). The VRPSSTW can be mathematically formulated as equations (3) to equations (11).

$$c'_{ijk} = \begin{cases} c_{ij}, \text{if } s_{jk} \le b_j \\ c_{ij} + c_l(s_{jk} - b_j), \text{if } s_{jk} > b_j \end{cases} \tag{2}$$

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c'_{ijk} x_{ijk} \tag{3}$$

subject to

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \qquad \forall i \in C \tag{4}$$

$$\sum_{i \in C} d_i \sum_{j \in V} x_{ijk} \le q \qquad \forall k \in K \tag{5}$$

$$\sum_{j \in V} x_{0jk} = 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \qquad \forall h \in C, \qquad \forall k \in K \tag{7}$$

$$\sum_{i \in V} x_{i0k} = 1 \qquad \forall k \in K \tag{8}$$

$$s_{ik} + t_{ij} - s_{jk} \le (1 - x_{ijk})M \qquad \forall (i, j) \in A, \qquad \forall k \in K \tag{9}$$

$$a_i \le s_{ik} \le b_i' \qquad \forall i \in V, \qquad \forall k \in K \tag{10}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall (i, j) \in A, \qquad \forall k \in K \tag{11}$$

The model contains two decision variables: s_{jk} determines the service start time at customer j as well as the travel cost of arc (i, j), and x_{ijk} represents whether arc (i, j) is used in the solution ($x_{ijk} = 1$) or not ($x_{ijk} = 0$). The objective function (3) minimises the delivery cost; whereas, constraints (4)-(10) ensures that all routes must start and end at the central depot, serving every customers once within their relaxed time windows $[a_i, b_i']$, and that the vehicle routes do not violate the capacity constraint. Finally, constraint (11) defines the integration of the arc flow variables x_{ijk}.

4. Column Generation based Exact Solution of the VRPSSTW

Using the Dantzig-Wolfe decomposition, the VRPSSTW is decomposed into a set partitioning master problem and an ESPPRCLAP as its sub-problem. The ESPPRCLAP gives the feasible shortest path subject to constraints (4)–(10). The objective function (3) is a non-linear function; however, the same decomposition methodology can be adopted for the exact solution of the non-linear optimisation problem (3)–(11) as that used for a variant of the Capacitated Arc Routing Problem (CARP) (Tagmouti et al., 2007). The master problem remains linear while the non-linearity is handled at the sub-problem level using dynamic programming. The master problem, which consists of selecting a set of feasible paths of minimum cost, is described mathematically in equations (12)-(14), where P is the set of all feasible paths. The variable y_p takes value 1 if the path $p \in P$ is selected and 0 otherwise. The cost of path p is denoted by c_p, and a_{ip} represents the number of times path p serves customer i. As the set P can grow exponentially with the number of customers, a restricted master problem is optimised based on the available number of columns (routes) represented by set $\tilde{P}$.

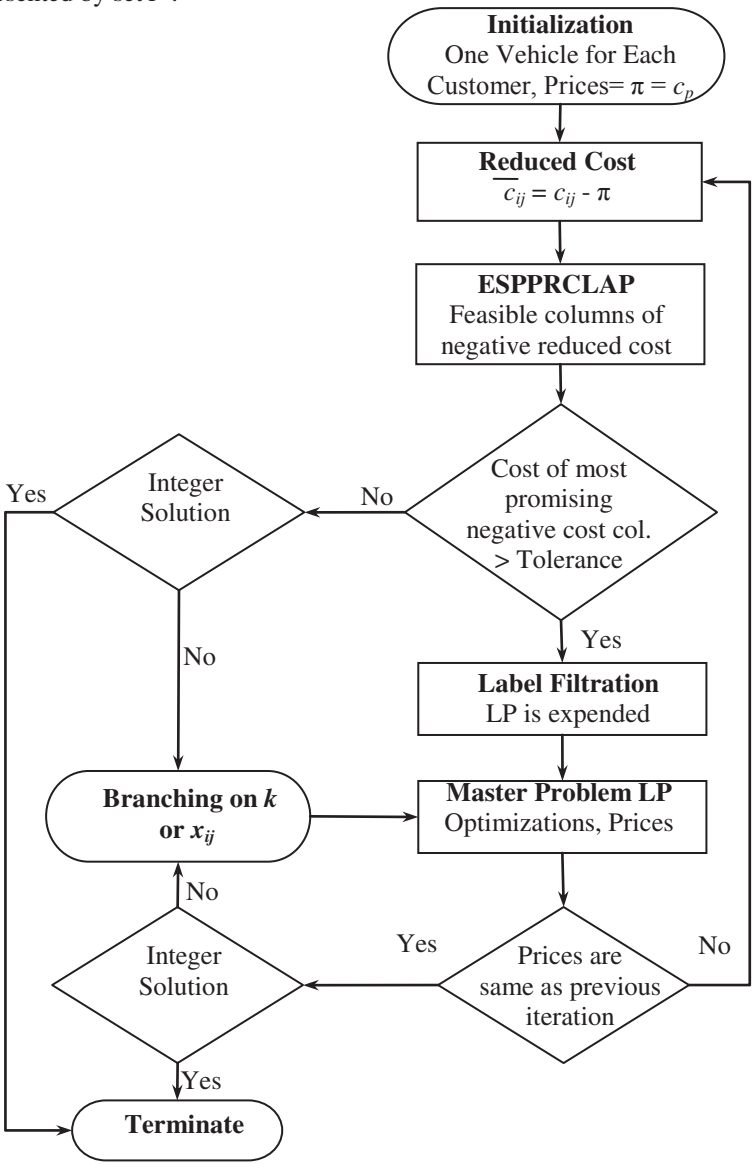

Figure 2 Flowchart of the exact solution algorithm for the VRPSSTW

During this optimisation, dual variables' values (prices) π_i, $i \in C$, are obtained as a by-product, which are used to find the reduced costs (marginal costs) $\overline{c'_{ij}}$ as per equation (15). The ESPPRCLAP sub-problem is called with these reduced costs and it returns columns with negative reduced costs, which are used to augment the partial set $\tilde{P}$. Finally, the column generation procedure was embedded in a branch and price scheme to obtain integer solutions. Figure 2 shows the complete exact solution algorithm for the VRPSSTW.

$$\min \sum_{p \in P} c_p y_p \tag{12}$$

subject to

$$\sum_{p \in P} a_{ip} y_p = 1, \qquad \forall i \in C \tag{13}$$

$$y_p \in \{0,1\} \qquad \forall p \in P \tag{14}$$

$$\overline{c'_{ij}} = c'_{ij} - \pi_i \qquad \forall\, i \in V \tag{15}$$

5. The ESPPRCLAP Sub-problem

In the VRPSSTW, all vehicles are considered identical; therefore, the ESPPRCLAP sub-problem is solved on the same network as the VRPSSTW, with the arc costs being reduced by the corresponding dual variables (prices) generated in the master problem using equation (15). A new labelling algorithm was developed for the ESPPRCLAP, which is based on the template-labelling algorithm for shortest path problem described in Irnich and Villenuve (2003). Every path from origin depot to a customer vertex is represented by a label that contains some information about that path. In the labelling algorithm, a label is selected based on some criteria and subjected to a usefulness test (label dominance). If found useful, the path associated with this label is extended to all possible successor vertices (path extension). The main features of the labelling algorithm for the ESPPRCLAP are described next.

5.1. Label representation

A label was represented in the form of a row of a matrix, containing $|V|$+7 entries, which show the important aspects of the associated path such as the last vertex on the path (the resident vertex), the service start time at the resident vertex (the time resource), capacity of the vehicle consumed so far along the path (the capacity resource) and the reduced cost of the path. A vector (*vis*(*L*)) is also a part of the label *L* with $|V|$ entries, where an entry of 1 represents the already visited or unreachable vertex, and possible successors are marked with zero entries. The sum of these entries is also kept to show the total number of unreachable vertices. A unique label number is associated with every label (*Label no.*), which is used to link them with their predecessor labels using the label number of its predecessor label (*pred*(*L*)). Figure 3 shows an example of a label created at customer 6 for an instance with $V = \{0, 1, 2, \ldots, 7\}$.

5.2. Dominance rule

A labelling algorithm generates new states or labels from previously generated labels, dominance rules are implemented to avoid proliferation of labels. The role of dominance rules is to identify labels (thus associated paths), extension of which will never generate an optimum path so that such labels shall not be considered in the path extension step. As long as the cost function and consumption of resources follow a non-decreasing function such as in the case of ESPPRCLAP, the dominance rules developed for any ESPPRC variant remain the same (Larsen, 1999). Next, the dominance rules are described which are same as ESPPRC dominance rules described in

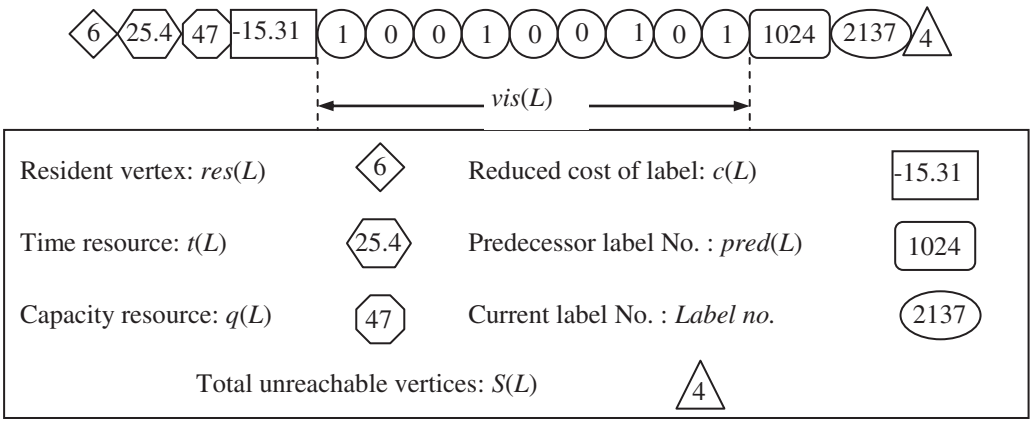

Figure 3 Label representation in the ESPPRCLAP labeling algorithm

Feillet et al. (2004). Consider two labels L_1 and L_2 both having same resident vertex, $res(L_1) = res(L_2)$. To verify whether the label L_1 dominates the other label L_2, the following dominance criteria (equation (16)-equation (20)) are validated in the given sequence. For example, the label L_1 shown in the Figure 4 dominates the Label L_2, because it satisfies all of the dominance criteria sequentially.

$$S(L_1) \le S(L_2) \tag{16}$$
$$vis(L_1) \le vis(L_2) \tag{17}$$
$$t(L_1) \le t(L_2) \tag{18}$$
$$q(L_1) \le q(L_2) \tag{19}$$
$$c(L_1) \le c(L_2) \tag{20}$$

5.3. Path extension

In labelling algorithms for various variants of shortest path problem, dominance rules and path extension steps form the two main building blocks (Irnich and Villenuve, 2003). During the path extension step, an existing label at vertex i is extended to all new possible labels, one at every possible successor of i, by updating the resource consumptions and cost. In the labelling algorithm for the ESPPRCLAP, if a label L_i with $res(L_i) = i$ is to be extended to L_j with $res(L_j) = j$ using arc (i, j), the following rules were used to update the associated resources and cost.

$$t(L_j) = \max.[t(L_i) + t_{ij}, a_j] \tag{21}$$
$$q(L_j) = q(L_i) + d_j \tag{22}$$
$$c(L_j) = \begin{cases} c(L_i) + c_{ij}, \text{if } t(L_j) \le b_j \\ c(L_i) + c_{ij} + c_l(t(L_j) - b_j), \text{if } t(L_j) > b_j \end{cases} \tag{23}$$

A newly generated label inherits the vector of visited vertices from its processor label. This vector is further updated by assigning a value of 1 for the vertices h having a 0 entry in $vis(L_i)$, which becomes unreachable from new resident vertex j due to violation of any of the conditions given by equation (24) and equation (25). Finally, the total number of unreachable nodes $S(L_j)$ in the new state is calculated as the sum of $vis(L_j)$, and the label number of

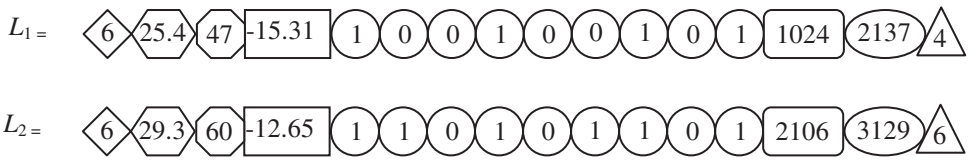

Figure 4 Dominance of label L_1 on label L_2

L_i is set as the immediate predecessor in $pred(L_j)$. New path extension rules were defined for the ESPPRCLAP to incorporate the variable costs (due to possible late arrival penalty) and in update rules for $vis(L_i)$ to consider relaxed time windows $[a_i, b_i']$ as shown in equation (23) and equation (24), respectively.

$$t(L_j) + t_{jh} < b'_h \tag{24}$$

$$q(L_j) + d_h < q \tag{25}$$

6. Genetic Algorithm (GA) Heuristics

As discussed earlier, meta-heuristics solution techniques have been applied to obtain good solutions for the VRPSTW. It is very difficult to evaluate a heuristics approach in absolute terms as they do not state the gap between their best feasible solution and any lower bound. In this study, as an application, the exact VRPSSTW solutions are used to evaluate the relative error and computation time requirement for a simple GA (GA-1). It was found that the GA-1 results contain considerably large errors; therefore in order to improve its quality, some of the procedures and parameters of GA-1 were changed to formulate another GA heuristic (GA-2). Some important procedures and parameters are described below for both the GAs.

6.1. Chromosome representation and population

GA-1: The population is composed of 1000 integer valued individuals or chromosomes, each representing a complete feasible VRPSSTW solution. A greedy look-ahead insertion heuristic based on time windows matching is used to generate a feasible solution. Half of the initial population is obtained by randomly swapping one to four customers in this feasible solution, while the remaining half is generated randomly. Figure 5 shows a chromosome for a twelve customer instance and its interpretation in the GA-1 for new vehicles due to the presence of a depot gene or due to the violation of capacity or time window constraints. Two continuous variables (qsum0 = 0 and twroute0) for each vehicle are initiated (as per equation (26)) and updated every time that vehicle travels from i to j according to the equation (27) and equation (28).

$$twroute_0 = \max[a_0, a_i - t_{0i}] \tag{26}$$

$$qsum_j = qsum_i + d_j \tag{27}$$

$$twroute_j = twroute_i + t_{ij} \tag{28}$$

GA-2: The chromosome representation and interpretation remain the same as the GA-1. The number of chromosomes is reduced to 200. But all of these chromosomes are obtained using a Stochastic Push Forward Insertion Heuristics (SPFIH) (Alvarenga et al., 2007), modified to incorporate soft time windows. Let (i_0, i_1, i_2, . . ., i_m) be a partial route in the SPIFH that starts and ends at the depot (i.e. $i_0 = i_m = 0$). The service start time s_{i_r}, waiting time w_{i_r} and late arrival time l_{i_r} are known for $0 \le r \le m$. Insertion of a customer vertex u between i_{p-1} and i_p, causes

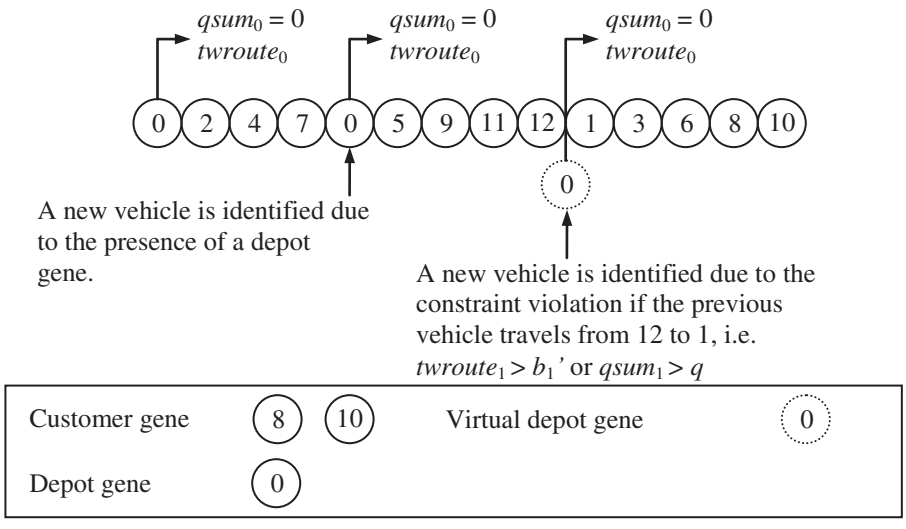

Figure 5 VRPSSTW chromosome coding and interpretation in GA

a *push forward* (PF_{i_p}) in the schedule at the customer i_p, which may change the values of s_{i_r}, w_{i_r} and l_{i_r}, $p \le r \le m$. For the VRPSSTW, the conditions $s_u \le b'_u$ and $s_{i_r} + PF_{i_r} \le b'_{i_r}$ provide the feasibility criteria for a feasible insertion position of customer u. Similar to Solomon (1987), the best feasible insertion place is determined using equation (29) for each un-routed customer u; however, an additional term is added to consider the changes in late arrival penalties for the customers i_r, $p+1 \le r \le m-1$ in order to find the insertion cost (equation (30)) of each un-routed customer u. Finally, the best customer u^* to be inserted in the route, is obtained using equation (31).

$$c_1(i(u),u,j(u)) = \min[c_1(i_{p-1},u,i_p)],\ p = 1,...,m \tag{29}$$

$$c_1(i_{p-1},u,i_p) = \overline{c'_{i_{lp-1},u}} + \overline{c'_{u,i_p}} - \overline{c'_{i_{lp-1},i_p}} + \sum_{r=p+1}^{m-1} c_l(l_{i_r}^{new} - l_{i_r}) \tag{30}$$

$$c_2(i(u^*),u^*,j(u^*)) = \min_u[c_1(i(u),u,j(u))] \tag{31}$$

6.2. Crossover, mutation and elitism

Both the GA-1 and GA-2 share the same settings for crossover, mutation and elitism. To generate the population for the next generation, individuals from the present population are selected using the Stochastic Universal Selection (SUS) method (Chipperfield et al., 1994) based on their fitness value. To maintain the feasibility of the chromosomes, i.e., to avoid duplication of the same customer gene, an ordered-based two-point crossover is used with a crossover rate of 98%. A simple swap mutation is used to stirrup the search pattern at a mutation rate of 10%. To ensure that each iteration of the GA always finds a new or maintains the best solution found so far, elitism is adopted thereby keeping the best 2% individuals of the current population in the population of the next generation.

6.3. Number of generations and population re-generation

In order to relate the optimisation effort with the instance size, the maximum number of iterations (generations) is kept as 250 times the number of customers in the instance. The population is regenerated after every 500 generations. During this step, a new population is generated by keeping 2% of the elite individuals of the current population and

the remaining 98% is generated using the same procedures as used in the initialization. However, in GA-1, the first half of the population was based on the neighbourhood of the current best solution instead of the greedy insertion solution (used in the initialisation).

7. Results and Discussions

The algorithms were implemented in MATLAB, and were run on a computer with 2.4 GHz AMD Athlon with 64 x 2 dual core processors and 2 GB of RAM. The test instances are based on the Solomon's R101 benchmark instance (Solomon, 1987). The R101 contains 100 randomly located customers, and the smaller instances are obtained by considering first 25, 50 and 75 customers. The nomenclature "R101-50-10" shows an instance derived from R101 with 50 customers and with a time windows relaxation (b_i'- b_i) by 10 minutes. In the maximum time windows relaxation the limit b_i' is found as per equation (1). The vehicle operating cost (VOC) was taken as 14.02 Japanese yen (JPY)/minute and a fixed cost of 10417.50 JPY/vehicle. The unit late arrival penalty cost (c_l) was set to five times that of the VOC. These parameter settings are based on an interview survey of logistics firms in Japan. A scaled cost matrix was used in programs taking VOC = 1 (travel cost = travel time), and all costs were also scaled to the same level. Figure 6 shows the convergence of such a scaled objective function (upper bound) in the exact solution of the R101-50-20 instance along with the lower bound.

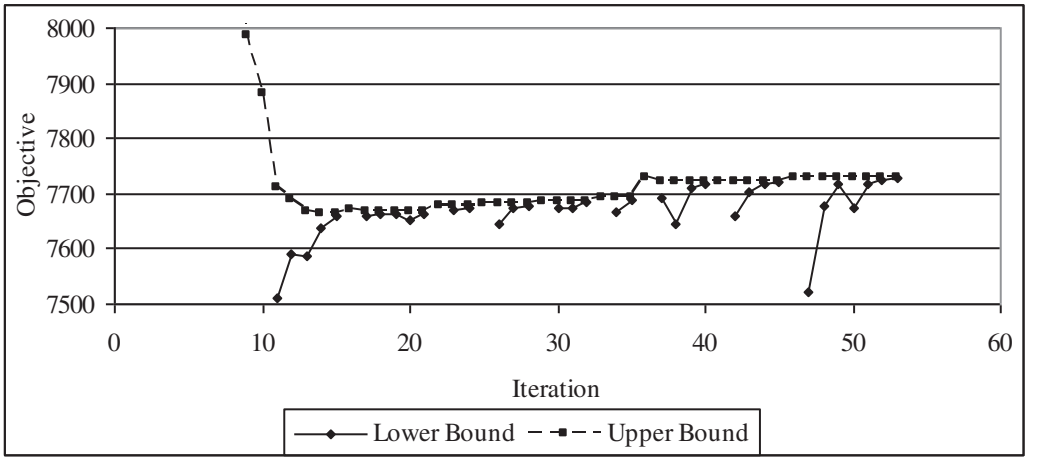

Figure 6 Upper and lower bound convergence in exact solution of R101-50-20

The operation time of a vehicle is composed of the time required for starting from the depot, serving all the customers along its route, and returning back to the depot. Table 1 provides the details of the exact solutions such as the number of required vehicles (routes) (Col. 2) their cumulative operation time (Col. 3) and late arrival time (Col. 4). Columns 5-7 and 8-10 provide the corresponding data for the best solutions obtained using GA-1 and GA-2, respectively. As compared to the exact solutions, the late arrival penalties were less in the GA-1 solutions, but it failed to simultaneously reduce the number of required vehicles. This resulted in substantially higher delivery cost, which is composed of the fixed vehicle utilisation cost, operation cost and the late arrival penalties. On the other hand, GA-2 was able to produce solutions with almost the same number of vehicles as in the exact solution. Table 2 provides the comparison of delivery cost and computation time for all the three approaches, i.e., the exact approach, GA-1 and GA-2.

One of the objectives of this paper is to introduce the exact VRPSSTW solutions as benchmark solutions, used to evaluate and calibrate the heuristics approaches. The relative error in delivery cost and the relative computational burden between the heuristics and the exact solutions of the VRPSSTW have been used in this regard. The relative cost error shows the percentage error in the delivery cost of a heuristics solution to the exact solution, found by using the data of Table 2. For example, in the case of GA-1, it was found by ((Col.(4)-Col.(2))*100/Col.(2)). Similarly, the relative computational burden for GA-1 was found using ((Col.(5)-Col.(3))*100/Col.(3)). Table 3 reports these factors for both GAs.

Table 1 Details of exact solutions and best solutions obtained in GA-1 and GA-2

	Exact Solution			GA-1			GA-2		
Instance	Veh.	OT	LAT	Veh.	OT	LAT	Veh.	OT	LAT
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
R101-25-10	7	855.4	7	7	855.4	7	7	855.4	7
R101-50-10	9	1566	39.9	11	1529.8	3.2	10	1508.5	17.8
R101-75-10	13	2139.9	55.7	16	2164.4	3.4	14	2157	31.1
R101-25-20	5	801.8	118.5	6	835.8	33	5	798.3	126.4
R101-50-20	8	1482.9	160.3	10	1541.4	25.5	8	1488.7	180.3
R101-75-20	11	1985.4	212.2	14	2199.7	47.7	12	2155.3	261.9
R101-25-max	5	782.3	98.3	5	782.3	98.3	5	782.3	98.3
R101-50-max*	8	1425.5	184.5	9	1561.8	47.2	8	1458.2	193.2

Veh. = Required number of vehicles, OT = Operation time (minute), LAT = Late arrival time (minute)

Table 2 Comparison of delivery cost and computation time

	Exact Solution		GA-1		GA-2	
Instance	Delivery Cost	Computation Time	Delivery Cost	Computation Time	Delivery Cost	Computation Time
	(JPY)	(Seconds)	(JPY)	(Seconds)	(JPY)	(Seconds)
(1)	(2)	(3)	(4)	(5)	(6)	(7)
R101-25-10	85405.9	1.639	85405.9	766.378	85405.9	172.482
R101-50-10	118509.8	22.785	136264.6	1997.999	126572.0	1652.603
R101-75-10	169333.5	1132.870	197263.2	5554.983	178266.3	3068.613
R101-25-20	71635.6	3.465	76536.2	599.943	72140.3	321.580
R101-50-20	115367.3	128.966	127573.0	2295.015	116850.6	651.805
R101-75-20	157303.0	1385.912	180028.6	10428.610	173586.5	3375.351
R101-25-max	69946.2	70.941	69946.2	3439.813	69946.2	165.909
R101-50-max†	116259.0	45983.000	118962.7	9849.979	117327.3	299.588

As far as the cost is concerned, GA-1 performed better for the smaller instances with 25 customers, but for the larger instances (50 and 75 customers) its performance significantly suffered, which is the typical case with most of the heuristics. For a particular size of problem, a marked trend is that the relative computational burden goes on decreasing in GA-1 as the time windows relaxation increases. For example it started up with a very high value (8669.06%) for R101-50-10, reduced to a moderate value (1679.55%) for R101-50-20 and eventually became negative for R101-50-max, which means that the computation time of the GA-1 was less than the exact approach; in fact the R101-50-max was not solved to optimality and only the best integer solution is reported in the tables. This is due to the fact that the time complexity of the shortest path sub-problem in the exact approach is a function of the width of time windows (Desrochers et al., 1992), and most of the computation time in exact approach is consumed to solve the sub-problem. This also elaborates the significance of heuristics as viable solution approaches for larger instances with large time windows. In order to control the high relative cost error for larger instances the initialization procedure was changed in GA-1 to formulate GA-2. Table 3 shows the positive effects of this change in terms of lower relative cost errors for GA-2 as compared to GA-1. In addition, the change in the initialization also worked well with the relative computational burden, which was also reduced noticeably for most of the cases.

† Best integer solution found in more than 12 hours of computation time, is reported under the exact solution column.

Table 3 Performance evaluation of GA-1 and GA-2

	GA-1		GA-2	
Instance	Relative Cost Error	Relative Computational Burden	Relative Cost Error	Relative Computational Burden
(1)	(2)	(3)	(4)	(5)
	(%)	(%)	(%)	(%)
R101-25-10	0	46650.35	0	10421.67
R101-50-10	14.98	8669.06	6.80	7153.15
R101-75-10	16.49	390.35	5.28	170.87
R101-25-20	6.84	17216.88	0.70	9182.14
R101-50-20	10.58	1679.55	1.29	405.41
R101-75-20	14.45	652.47	10.35	143.55
R101-25-max	0	4748.86	0	133.87
R101-50-max‡	2.33	-78.58	0.92	-99.35

8. Conclusion

An exact solution approach using column generation framework is presented for the VRPSSTW problem. As the size of an instance (number of customers) and the size of time windows limit the viability (as per computation burden) of the exact solution approach for the VRPSSTW, the use of heuristics for larger instances with wider time windows width is inevitable. However, the exact solutions can be used as benchmark solutions to evaluate and calibrate the heuristics approaches on small sized instances before the heuristics are applied to large-sized instances. This idea was also elaborated with an exemplary evaluation of two heuristics based on genetic algorithms (GA), in this paper.

It was found that the relative cost error (as compared to the exact solution) in a GA heuristics with a larger initial population size (1000 individuals) based on greedy algorithm can be decreased if a modified stochastic push forward insertion heuristic is used instead of the greedy algorithm even with a smaller population size (200 individuals). This change in the initialization policy also worked well to decrease the relative computational burden (as compared to the exact solution) of the GA heuristics. Use of exact benchmark solutions and evaluation of heuristics on these benchmarks will provide the researchers in the city logistics field with more confidence and they will be in better position to appraise and support city logistics-related policies, which are assessed using the heuristics solutions of the VRPSSTW. Future research includes the use of the developed exact method for the VRPSSTW in evaluating city logistics policies for practical logistics instances based on real road network data.

References

Ando, N., & Taniguchi, E. (2006). Travel time reliability in vehicle routing and scheduling with time windows. *Networks and Spatial Economics*, 6, 293-311.

Alvarenga, G. B., Mateus, G. R., & Tomi, G. de. (2007). A genetic and set partitioning two-phased approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34, 1561-1584.

Balakrishnan, N. (1993). Simple heuristics for the vehicle routing problem with soft time windows. *Journal of the Operational Research Society*, 44, 279-287.

Braysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, 39, 104-118.

Braysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39, 119-139.

‡ Based on best integer solution reported in the exact solution column in Table (2).

Calvete, H. I., Gale, C., Oliveros, M. J., & Valverde, B. S. (2007). A goal programming approach to vehicle routing problem with soft time windows. *European Journal of Operational Research*, 177, 1720-1733.

Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33, 2972-2990.

Chipperfield, A., Fleming, P., Pohlhiem, H., & Fnoseca, C. (1994). *User guide: Genetic algorithm toolbox for use with MATLAB*. Department of Automatic Control and Systems Engineering, University of Sheffield. Available at: http://www.shef .ac.uk/uni/projects/gaipp/gatbx.html [accessed on November 10, 2004].

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A new optimisation algorithm for the vehicle routing problem with time windows. *Operations Research*, 40, 342-354.

Duin, van. J. H. R., Tavasszy, L. A., & Taniguchi, E. (2007). Real time simulation of auctioning and re-scheduling process in hybrid freight markets. *Transportation Research Part B*, 41, 1050-1066.

Dumas, Y., Somis, F., & Desrosiers, J. (1990). Optimising the schedule for a fixed vehicle path with convex inconvenience costs. *Transportation Science*, 24, 145-152.

Fagerholt, K. (2001). Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, 131, 559-571.

Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 216-229.

Gendreau, M., Guertin, F., Potvin, J., & Taillard, E. (1999). Parallel tabu search for real-time vehicle and dispatching. *Transportation Science*, 33, 381-390.

Hashimoto, H., Ibaraki, T., Imahori, S., & Yagiura, M. (2006). The vehicle routing problem with flexible time windows and travelling times. *Discrete Applied Mathematics*, 154, 2271-2290.

Ioachim, I., Gelinas, S., Soumis, F., & Desrosiers, J. (1998). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31, 193-204.

Irnich, S., & Villeneuve, D. (2003). The shortest path problem with K-cycle elimination (K≥3): Improving a branch and price algorithm for the VRPTW. Technical Report G-2003-55, GERAD.

Kallehauge, B., Larsen, J., & Madsen, O. B. G. (2006). Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33, 1464-1487.

Larsen, J. (1999). *Parallelization of the vehicle routing problem with time windows*. PhD. Thesis No. 62, Department of Mathematical Modeling (IMM) at the Technical University of Denmark (DTU).

Qureshi, A. G., & Hanaoka, S. (2005). Analysis of the effects of cooperative delivery system in bangkok. In E. Taniguchi, and R. G. Thompson (Eds.), *Recent advances in city logistics* (pp. 59-73). Elsevier, Oxford.

Qureshi, A. G., Taniguchi, E., & Yamada, T. (2009). A hybrid genetic algorithm for VRPSTW using column generation. In E. Taniguchi, and R. G. Thompson (Eds.), *Innovations in city logistics* (pp. 153-170). Nova Publishers, New York.

Sexton, R. T., & Bodin, L. D. (1985). Optimising single vehicle many to many operations with desired delivery times: I. Scheduling. *Transportation Science*, 19, 378-410.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problem with time windows constraints. *Operations Research*, 35, 254-265.

Tagmouti, M., Gendreau, M., & Potvin, J. Y. (2007). Arc routing problem with time-dependent service costs. *European Journal of Operational Research*, 181, 30-39.

Taillard, E., Badeau, P., Guertin, F., Gendreau, M., & Potvin, J. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31, 170-186.

Taniguchi, E., Thompson, R. G., Yamada, T., & Duin, R. V. (2001). *City logistics: Network modeling and intelligent transport systems*. Pergamon, Oxford.

Taniguchi, E., & Heijden, R. E. C. M. van der. (2000). An evaluation methodology for city logistics. *Transport Reviews*, 20, 65-90.

Thompson, R. G., & Van Duin, J. H. R. (2003). Vehicle routing and scheduling. In E. Taniguchi, & R. G. Thompson (Eds.), *Innovations in freight transportation* (pp. 47-63). WIT Press, Southampton.

Yamada, T., Yoshimura, Y., & Mori, K. (2004). Road network reliability analysis using vehicle routing and scheduling procedures. In E. Taniguchi, & R. G Thompson (Eds.), *Logistics systems for sustainable cities, proceedings of the 3rd international conference on city logistics* (pp. 97-110). Elsevier, UK.

Yamada, T., Taniguchi, E., & Itoh, Y. (2001). Co-operative vehicle routing model with optimal location of logistics terminals. In E. Taniguchi, & R. G. Thompson (Eds.), *City logistics II* (pp. 139-153). Institute for City Logistics, Japan.