AN ITERATIVE ROUTE CONSTRUCTION AND IMPROVEMENT ALGORITHM FOR THE VEHICLE ROUTING PROBLEM WITH SOFT AND HARD TIME WINDOWS

Miguel Andres Figliozzi
College of Engineering and Computer Science
Portland State University-CEE
Portland 97201-0751, USA
e-mail: figliozzi@pdx.edu

ABSTRACT

The joint solution of routing problems with soft and hard time windows has valuable practical applications. Simultaneous solution approaches to both types of problems are needed when: (a) the number of routes needed for hard time windows exceeds the number of available vehicles, (b) a study of cost-service tradeoffs is required or the dispatcher has qualitative information regarding the relative importance of hard time window constraints across customers. A new Iterative Route Construction and Improvement (IRCI) algorithm of average run time performance $O(n^2)$ is proposed to sequentially solve Vehicle Routing Problems with Soft Time Windows (VRPSTW) and Hard Time Windows (VRPHTW). Due to its modular and hierarchical design, the IRCI algorithm is intuitive, easy to code, and able to accommodate general cost and penalty functions. The solution quality and computational time of the new algorithm is compared against existing results on benchmark problems for the VRPHTW and VRPSTW. Furthermore, the algorithm can be used to obtain faster simultaneous solutions for both VRPHTW and VRPHTW problems using the soft time windows solutions as a lower bound for hard time window problems. Despite its simplicity and flexibility, the algorithm performs well in terms of solution quality and speed in instances with soft and hard time windows.

Keywords: vehicle routing, soft and hard time windows, route construction and improvement algorithms.

1. INTRODUCTION

The Vehicle Routing Problem with Hard Time Windows (VRPHTW) has a significant body of literature. Clearly, the VRPHTW is a problem with practical applications in distribution and logistics due to the rising importance of just-in-time (JIT) production systems and the increasingly tight coordination of supply chain operations. In comparison, the Vehicle Routing Problem with Soft Time Windows (VRPSTW) has received meager attention. The VRPSTW is a relaxation of the VRPHTW; in the former, time windows can be violated if a penalty is paid; in the latter violations are infeasible.

The VRPSTW also has many practical applications (Chiang and Russell, 2004): (1) relaxing time windows can result in lower total costs without hurting customer satisfaction significantly; (2) many applications do not require hard time windows – e.g. the delivery of fuel/gas to service stations, (3) travel times cannot be accurately known in many practical applications, and (4) VRPSTW approaches can be used to solve VPRHTW if the penalties are modified appropriately. In addition, VRPSTW solutions provide a workable alternative plan of action when the problem with hard time windows is infeasible.

The objective of this paper is to develop one algorithm that can be applied sequentially to solve VRPSTW and VRPHTW instances. It is useful for dispatchers to have the solutions to both VRPSTW and VRPHTW problems when: (a) the number of routes needed for the HTW case exceeds the number of available vehicles, (b) a study of cost-service tradeoffs is required, and (c) the dispatcher has qualitative information regarding the relative importance of service level across customers. For example, in many practical situations late deliveries have penalties that significantly exceed the penalties for early delivery. In addition, customers may be

incapable or unwilling to set precise time windows in advance and simply prefer the flexibility to alter their pickup or delivery requests (Powell et al., 2002).

This paper provides a solution approach that solves both soft and hard problems simultaneously. The rest of this paper is organized into five additional sections. Section two briefly reviews the relevant literature on VRPHTW and VRPSTW problems. Section three introduces the mathematical notation and describes the new iterative route construction and improvement (IRCI) algorithm. Section four compares IRCI computation time and solution quality against existing solutions available in the literature. Section five discusses IRCI algorithmical properties. Section six ends with conclusions.

2. LITERATURE REVIEW

Heuristics to solve the VRPHTW can be classified – in increasing order of solution quality – as construction heuristics, local search heuristics, and metaheuristics. Although metaheuristics generally produce solutions of higher quality this is usually at the expense of significantly longer computation times. There is a clear trade-off between computation time and solution quality.

Route construction algorithms work by inserting customers one at a time into partial routes until a feasible solution is obtained. Construction heuristics include the work of Solomon (1987), Potvin and Rousseau (1993) and Ioannou et al. (2001). Local search methods improve on feasible solutions performing exchanges within a neighborhood while maintaining the feasibility of the solutions. Some of the most successful local improvement methods include the algorithms proposed by Russel (1995), Caseau and Laburthe (1999), Cordone and Calvo (2001), and Braysy (2002).

Metaheuristics include a diverse set of methods such as simulated annealing, genetic algorithms, tabu search, ant-colony, and constraint programming. Some of the most successful methaheuristics include the algorithms proposed by Taillard et al. (1997), Liu and Shen (1999), Homberger and Gehring (1999), Berger et al. (2003), and Braysy (2003). For additional references and a review of the large body of VRPHTW research the reader is referred to a recent comprehensive survey by Braysy and Gendreau (2005a,b).

The body of work related to the VRPSTW is relatively scant. Early work on the topic includes the work of Sexton and Choi (1986) using Benders decomposition to solve a single-vehicle pickup and delivery routing problem. Ferland and Fortin (1989) solves a variations of the VRPSTW where customers' time windows are adjusted to lower service costs. Koskosidis et al. (1992) propose a generalized assignment problem of customers to vehicles and a series of traveling salesman problems with soft time windows constraints.

Balakrishnan (1993) proposes construction heuristics for the VRPSTW based on the nearest neighbor, Clarke and Wright savings, and space–time rules algorithms. The heuristics are tested on a subset of the Solomon set problems for hard time windows using linear penalty functions. Taillard et al. (1997) propose a tabu search heuristic to solve a VRPSTW as proposed by Balakrishnan, i.e. with linear penalty functions. The tabu search algorithm produced very good results on the Solomon set with hard time windows; however, no results are reported for the VRPSTW.

Ioannou et al. (2003) solves Solomon problems and extended Solomon problems of up to 400 customers with a nearest neighbor that generate and modify customer time windows to find lower cost solutions; no computation times are reported. Chiang and Russell (2004) uses a

tabu search approach with a mixed neighborhood structure and advance recovery to find some of the best solutions ever reported for Solomon VRPSTW instances. The algorithm designed by Ibaraki et al. (2005) is another metaheuristic that could handle soft time-window constraints and penalties using a local search based on a cyclic-exchange neighborhood to assign and sequence customers; only results for instances with hard time windows are reported . Calvete et al. (2007) propose a goal programming approach to the vehicle routing and solve medium size problems (less than 70 customers) with soft and hard time windows, a heterogeneous fleet of vehicles, and multiple objectives.

As indicated by Braysy and Gendreau (2005a,b), fair and meaningful comparisons of vehicle routing heuristics require standard benchmark problems and the full reporting of : (a) solution quality, (b) number of run needed and computation time per run, and (c) computing power or processor speed. From the survey of the VRPSTW only two journal publications comply with these prerequisites: Balakrishnan (1993) and Chiang and Russell (2004). Regarding VRPHTW, only Taillard et al. (1997) and Ibaraki et al. (2005) present algorithms that are designed to handle soft and hard time windows and also comply with the reporting of solution quality, computation time, and processor speed. Section 5 compares IRCI results with previous results found in the literature in terms of solution quality and computational time.

3. SOLUTION ALGORITHM

This section firstly introduces a precise mathematical definition of the VRPHTW and VRPSTW studied in this research. The remainder of this section is to describe the solution algorithm.

Problem Definition

The vehicle routing problem with hard time windows (VRPHTW) studied in this research can be described as follows. Let $G = (V, A)$ be a graph where $V = (v_0, ..., v_n)$ is a vertex set and $A = \{(v_i, v_j) : i \neq j \wedge i, j \in V\}$ is an arc set. Vertex v_0 denotes a depot at which the routes of m identical vehicles of capacity $q_{\max}$ start and end. The set of vertices $C = \{v_1, ..., v_n)$ specify the location of a set of n customers. Each vertex in V has an associated demand $q_i \geq 0$, a service time $s_i \geq 0$, and a service time window $[e_i, l_i]$. Each arc (v_i, v_j) has an associated constant distance $d_{ij} > 0$ and travel time $t_{ij} > 0$. The arrival time of a vehicle at customer $i, i \in C$ is denoted a_i and its departure time b_i ; the beginning of service time is denoted y_i . The primary objective function for the VRPHTW is the minimization of the number of routes. A secondary objective is the minimization of total time or distance. The solution to the VRPHTW must satisfy the following:

(a) the value of m is not specified initially, it is an output of the solution algorithm;

(b) a route cannot start before e_0 and cannot end after l_0 ;

(c) service to customer i cannot start before e_i and cannot start after l_i ;

(d) every route starts and ends at the depot v_0 ;

(e) every customer is visited exactly once by one vehicle; and

(f) the total demand of any vehicle route does not exceed the vehicle capacity.

The VRPSTW is a relaxation of the VRPHTW. With soft time windows, there is an allowable violation of time windows denoted $P_{\max} \geq 0$. The time window of each customer $i, i \in C$ can be enlarged to $[e_i - P_{\max}, l_i + P_{\max}] = [e_i^{\#}, l_i^{\#}]$. In addition, an early penalty $p_e(e_i - y_i)$ is applied if service time starts early, i.e. $y_i \in [e_i^{\#}, e_i]$. Similarly, a late penalty

$p_l(y_i - l_i)$ is applied if service starts late, i.e. $y_i \in [l_i, l_i^{\#}]$. The primary objective function for the VRPSTW is the minimization of the number of routes. A secondary objective is the minimization of the number of time window violations. A third objective is the minimization of total time or distance plus penalties for early or late deliveries. It is important to notice that the depot time windows as well as the maximum route duration are not changed as a result of the customers' time window relaxation.

It is commonly assumed in the literature that fix costs associated with each additional route (vehicle) outweigh travel time or distance related costs. As discussed in Section 5, the presented IRCI algorithm can be applied to any hard or soft time window problem with an objective function that is a combination of positive functions of fleet size, travel time, travel distance, and early/late penalties.

Solution Algorithms

The solution method is divided into two phases: route construction and route improvement. The route construction phase utilizes two algorithms: (a) an auxiliary route building algorithm and (b) a route construction algorithm. The route improvement phase also utilizes two algorithms: (c) a route improvement algorithm and (d) a service time improvement algorithm. Using a bottom up approach the algorithms are introduced in the following order: (a) the auxiliary algorithm, (b) the construction algorithm, (c) the route improvement algorithm, and (d) the start time improvement algorithm.

(a) The Auxiliary Algorithm

The auxiliary routing algorithm $\mathbf{H}_r$ can be any heuristic that given a starting vertex, a set of customers, and a depot location returns *a set of routes* that satisfy the constraints of the VRPHTW or VRPSTW.

In this research $\mathbf{H}_r$ is a generalized nearest neighbor heuristics (GNNH). The GNNH has four inputs: (a) the weights or parameters for "generalized cost" function denoted by $\Delta = \{\delta_0, \delta_1, ..., \delta_i\}$, (b) an initial vertex denoted by v_i, (c) a set of customers to route denoted by C, and (d) a depot location denoted by v_0. The GNNH starts every route by finding the unrouted customer with the least appending "generalized cost". At every subsequent iteration, the heuristics searches for the remaining unrouted customer with the least appending cost.

The "generalized cost" function used in this research accounts for geographical and temporal closeness among customers, the remaining capacity in the vehicle, and the cost of adding a new vehicle if the next customer is infeasible. Let i denote the initial vertex and let j denote the customer to append next. Let q_i denote the remaining capacity of the vehicle after serving customer i. The service at a customer $i, i \in V$ begins at time $y_i = \max(a_i, e_i)$. The generalized cost of going from customer i to customer j is estimated as:

$$g(\Delta, i, j) = \delta_1 d_{ij} + \delta_2 (y_j - (a_i + s_i)) + \delta_3 (l_j - (a_i + s_i + t_{ij})) + \delta_4 (q_i - d_j)$$

The parameter δ_2 takes into account the "slack" between the completion of service at i and earliest feasible beginning of service at j, i.e. $y_j = \max(y_i + s_i + t_{ij}, e_j)$. Following Solomon's approach (1987), the parameter δ_3 takes into account the "urgency" of serving customer j

expressed as the time remaining until the vehicle's last possible start. The parameter δ_4 is introduced in this research and takes into account the capacity slack of the vehicle after serving customer j.

If customer j is infeasible, i.e. it cannot be visited after serving customer i, the cost of ending customer i's route and starting a new one to serve customer j is estimated as:

$$g(\Delta, i, j) = \delta_0 + \delta_1 d_{0j} + \delta_2 y_j + \delta_3 (l_j - t_{0j}) + \delta_4 (q_{\max} - d_j)$$

The parameter δ_0 is the cost of adding a new vehicle. The same GNNH can be applied to VRPSTW with the addition of two terms. For feasible customers:

$$g(\Delta, i, j) = \delta_1 d_{ij} + \delta_2 (a_j - (a_i + s_i)) + \delta_3 (l_j - (a_i + s_i + t_{ij})) + \delta_4 (q_i - d_j) + \\ + \delta_5 [e_j - a_j]^+ + \delta_6 [a_j - l_j]^+$$

The parameters δ_5 and δ_6 are added to account for possible early or late service penalties respectively; for infeasible customers δ_0 is added. With soft time windows, the service at a customer $i, i \in V$ begins at time $y_i = \max(a_i, e_i^{\#})$. For problems with general time windows, i.e. two or more time window intervals, the generalized cost is calculated for each time interval and the least expensive interval provides the generalized cost for that particular customer.

The auxiliary route heuristic is defined as $\mathbf{H}_r(\Delta, v_i, C, v_0)$ where $\Delta = \{\delta_0, \delta_1, ..., \delta_6\}$ are the parameters of the generalized cost function, v_i is the vertex where the first route starts, C is the set of customers to route, and v_0 the depot where all routes end and all additional routes

start – with the exception of the first route that starts at v_i. In all cases, the deltas are positive weights that satisfy: $\delta_1 + \delta_2 + \delta_3 = 1$ and $\delta_i \geq 0 \; i \in \{0,1,...,6\}$.

(b) The Route Construction Algorithm

In this algorithm, denoted $\mathbf{H}_c$, routes are constructed sequentially. Given a partial solution and a set of unrouted customers, the algorithm uses the auxiliary heuristic $\mathbf{H}_r$ to search for the feasible least cost set of routes. The algorithm also uses an auxiliary function $\mathrm{w}(v_i, C, \mathrm{g}, W)$ that given a set of unrouted customers C, a vertex $v_i \notin C$, and a generalized cost function $\mathrm{g}(\Delta, v_i, v_j)$ returns a set of vertexes with the lowest generalized costs $\mathrm{g}(\Delta, v_i, v_j)$ for all $v_j \in C$.

Functions or Algorithms:

$\mathbf{H}_r$: Route building heuristic.

$\mathrm{w}(v_i, C, \mathrm{g}, W)$: returns set of vertexes with the lowest generalized costs

Data:

C : Set of customers to route (not including the depot v_0)

$LLimit$ = initial number of routes or best known lower bound

W : Width of the search, number of solutions to be built and compared before adding a customer to a route.

Δ : space of the route heuristic generalized cost function parameters

START $\mathbf{H}_c$

1 $start \leftarrow v_0$
2 $start \leftarrow v_0$
3 $bestSequence \leftarrow v_0$
4 $\#vehicles \leftarrow min\#veh \leftarrow lowestCost \leftarrow \infty$
5 $Ccopy \leftarrow C$
6 **for** each $\Delta \in \mathbf{\Delta}$
7 **while** $C \neq \varnothing$ **AND** $LLimit < \#vehicles$ **AND** $\#vehicles \leq min\#veh$ **do**
8 $W \leftarrow \min(W, |C|)$
9 $C^* \leftarrow \mathrm{w}(start, C, \mathrm{g}, W)$
10 **for** each $v_i \in C^*$
11 **if** $\mathrm{c}(bestSequence \cup \mathbf{H}_r(\Delta, v_i, C, v_0)) <$ $lowestCost$ **then**
12 $lowestCost \leftarrow \mathrm{c}(bestSequence \cup \mathbf{H}_r(\Delta, v_i, C, v_0))$
13 $lowestNext \leftarrow v_i$
14 **end if**
15 **end for**
16 $start \leftarrow lowestNext$
17 $C \leftarrow C \setminus lowestNext$
18 $R \leftarrow bestSequence \cup \mathbf{H}_r(\Delta, lowestNext, C, v_0)$
19 $bestSequence \leftarrow bestSequence \cup lowestNext$
20 $\#vehicles \leftarrow$ cardinality of the set of routes R
21 **end while**
22 $C \leftarrow Ccopy$
23 **if** $min\#veh > \#vehicles$
24 $min\#veh \leftarrow \#vehicles$
25 **end if**
26 **end for**

Output:
Best set of routes R that serve all C customers

END $\mathbf{H}_c$

The conditions in the while-loop that starts in line 7 reduce the number of unnecessary computations after a lower bound have been reached or when a particular instance of the cost parameters $\Delta \in \mathbf{\Delta}$ are producing a solution with a larger number of routes. The generalized cost function g that is used in $\mathbf{H}_r$ must not be confused with the objective cost function c

that is used in $\mathbf{H}_c$ or the improvement heuristic $\mathbf{H}_i$; the latter cost function is the sum of the accrued vehicle, distance, time, or penalty costs as indicated in the objective function.

(c) The Route Improvement Algorithm

After the construction is finished, routing costs can be reduced using a *route* improvement algorithm. The improvement algorithm works on a subset of routes S. In this algorithm two functions are introduced. The function $\mathrm{k}_p(r_i, S, p)$ returns a set of p routes that belong to S and are located in the proximity of route r_i. In this research, the distance between routes' centers of gravity was used as a measure of geographic proximity. By definition, the distance of route r_i to itself is zero. Hence, the route r_i is always included in the output of the set function $\mathrm{k}_p(r_i, S, p)$.

The function $\mathrm{k}_s(R, s)$ orders the set of routes R from smallest to largest based on the number of customers per route and then returns a set of $s \geq 1$ routes with the least number of customers; e.g. $\mathrm{k}_s(R, 1)$ will return the route with the least number of customers. If two or more routes have the same number of customers, ties are solved drawing random numbers. To simplify notation the term $C(S)$ is the set of customers served by the set of routes S.

Functions or Algorithms:

$\mathbf{H}_c$: Route building heuristic

k_s and k_g: route selection functions

Data:

W: Number of solutions to be built and compared in the construction heuristic

Δ : Generalized cost parameters of the auxiliary route heuristic

s: Number of routes potentially considered for improvement

p : Number of neighboring routes to r_i that are reconstructed

R : Set of routes

$LLimit$ = lowest number of vehicles or stop condition for the $\mathbf{H}_c$ heuristic

START $\mathbf{H}_i$

1 $s \leftarrow \min(s, |R| - 1)$
2 $p \leftarrow \min(s, p)$
3 $S \leftarrow k_s(R, s) \subseteq R$
4 $S' \leftarrow R \setminus S$
5 **while** $|S| > 1$ **do**
6 $r^* \leftarrow k_s(S, 1)$
7 $G \leftarrow k_p(r^*, S, p)$
8 $G' \leftarrow \mathbf{H}_c(\mathbf{H}_r, W, \Delta, s, p, C(G), LLimit)$
9 **if** $c(G') < c(G)$ **then**
10 $R \leftarrow R \setminus G$
11 $R \leftarrow R \cup G'$
12 $S \leftarrow S \setminus G$
13 $S \leftarrow S \cup G'$
14 **end if**
15 $r = k_s(S, 1)$
16 $S = S \setminus r$
17 **if** $|S'| > 0$ **then**
18 $r' = k_s(S', 1)$
19 $S \leftarrow S \cup r'$
20 $S' \leftarrow S' / r'$
21 $s \leftarrow \min(s, |S|)$
22 $p \leftarrow \min(s, p)$
23 **end while**

Output:

R set of improved routes

END $\mathbf{H}_i$

(d) Start time improvement algorithm

With soft time windows, to reduce the number of roads during the construction and improvement algorithms, the service at a customer $i, i \in V$ begins at time $y_i = \max(a_i, e_i^{\#})$. However, once the algorithm $\mathbf{H}_i$ finishes, the sequence of customers per route is defined and some early time windows may be unnecessary.

This algorithm eliminates unnecessary usage of early time windows. The algorithm operates backwards, starting from the last customer, the algorithm verifies if a service time $y_i < e_i$ can be moved to $y_i = e_i$ without violating the following customer time window. Assuming that customer j follows customer i, then the service time can be moved later if two conditions are met: (1) $e_i + s_i + d_{ij} \le l_j$ if customer j is not using a soft time windows or (2) $e_i + s_i + d_{ij} \le l_j^{\#}$ if customer j is using a late soft time window. In the former case, the service time for customer i is set to $y_i = \min(l_j - (s_i + d_{ij}), l_i)$; in the latter case, the service time for customer i is set to $y_i = \min(l_j^{\#} - (s_i + d_{ij}), l_i)$.

Next section compares the IRCI against other solution approaches using standard benchmark problems for the VRPHTW and VRPSTW.

4. COMPUTATIONAL RESULTS

As seen in the previous section, at its core the IRCI algorithm is a construction algorithm were routes are sequentially built and improved. This section compares the results of the IRCI algorithm against other solution methods that report solution quality and computation time on Salomon benchmark problems for the VRPHTW and VRPSTW. The comparison only

includes other construction algorithms or solution approaches that were designed for both hard and soft time windows.

The well-known 56 Solomon benchmark problems for the VRPHTW are based on six groups of problem instances with 100 customers. The six problem classes are named C1, C2, R1, R2, RC1, and RC2. Customer locations were randomly generated (problem sets R1 and R2), clustered (problem sets C1 and C2), or mixed with randomly generated and clustered customers (problem sets RC1 and RC2). Problem sets R1, C1, and RC1 have a shorter scheduling horizon, tighter time windows, and fewer customers per route than problem sets R2, C2, and RC2 respectively.

<< INSERT TABLE 1 HERE>>

Table 1 presents the summary of the results when construction heuristics for the VRPHTW are compared. Against the three construction heuristics proposed by Solomon, Potvin et al. and Ioannou et al., the IRCI algorithm outperform them all in classes R1, C2, RC1, and RC2 while ties with the best in classes R2 and C1. Distance-wise, the performance of the IRCI algorithm is superior in all six classes of problems. The IRCI produces results in a relatively short time, less than12 seconds per 100 customer problems on average; however the other simpler algorithms have shorter running times. The IRCI results presented in Table 1 and 2 were obtained first running a VRPSTW version of the Solomon instances to obtain a set of lower bounds and STW results, and then using these bounds the VRPHTW was solved afterwards. The reported time for the IRCI corresponds to the total time to solve *both* types of problems for all 56 Solomon instances. The other references solve *only* the VRPHTW type.

<< INSERT TABLE 2 HERE>>

Table 2 presents the summary of the results when the IRCI algorithm is compared against two metaheuristics presented in the literature review that were explicitly designed to solve both soft and hard time windows: the tabu search heuristics of Taillard et al. (1997) and the composite metaheuristic of Ibaraki et al. (2005). As in Table 1, the reported time for the IRCI corresponds to the total time to solve *both* types of problems for all 56 Solomon instances. The other references solve *only* the VRPHTW type.

When compared to the Tabu heuristic of Taillard et al., with its 20 iterations, the results are similar, though the IRCI is faster in computation time even accounting for the different processing speed. The solution method proposed by Ibaraki et al. has a very good solution quality but at the expense of lengthy computation times.

In the soft time window benchmark problems, the results of the IRCI are compared against the results of prerequisites: Balakrishnan (1993) – denoted BAL in Tables 3 and 4 – and Chiang and Russell (2004). The latter has two solution methods: tabu search and advance recovery which are denoted Tables 3 and 4 by the initials TB and AR respectively.

<< INSERT TABLE 3 HERE>>

Balakrishnan (1993) and Chiang and Russell (2004), the only references with time and cost results for a standardized set of problems, solve a subset of Solomon problems setting a $P_{\max}$ that can be either 10, 5, or 0 % of the total route duration $(l_0 - e_0)$. Balakrishnan (1993) and Chiang and Russell (2004) also set a maximum vehicle waiting time limit $W_{\max}$. The maximum waiting time limits the amount of time that a vehicle can wait at a customer location before starting service, i.e. a vehicle can arrive to customer i only after $(e_i - P_{\max} - W_{\max})$. Since the VRPSTW is a relaxation of the VRPHTW, a maximum waiting time constraint $W_{\max}$ is clearly opposed to the spirit of the VRPSTW since a new constraint completely unrelated to time windows is added[1]. Despite these shortfalls, a $W_{\max} = 10\%$ constraint is added, mainly to facilitate comparisons in a level playing field.

Table 3 shows the results for the R1 benchmark problems with soft time windows; results for $P_{\max} = 10\%$ and $P_{\max} = 0\%$ are shown. The latter is equivalent to the VRPHTW problem but with the addition of the $W_{\max} = 10\%$ constraint. In addition to the number of vehicles and distance, Tables 3 and 4 also show the number of customers where the time windows have NOT been relaxed (%HTW); a higher %HTW indicates a better solution quality. As expected, when $P_{\max} = 0\%$ the corresponding % HTW are all equal to 100 because there is no room to relax the customers' time windows.

It can be observed that the IRCI algorithms perform very well against Balakrishnan's heuristic. Against tabu search (TS) the IRCI is almost tied but it performs betters in terms of

[1] Further, if there are carrier's costs associated with waiting time, e.g. parking, these costs can be incorporated into the routing cost function c rather than imposing a hard time waiting constraint which is not usually found in practical problems.

customers that do not have time window violations. The IRCI solutions are not as good as the advance recovery (AR) method. However, regarding computation times, the IRCI is undoubtedly faster than the TS and without a doubt much faster than the AR method.

<< INSERT TABLE 4 HERE>>

The same trends are repeated in the RC1 benchmark problems with soft time windows. The IRCI outperforms Balakrishnan's and is competitive with the tabu search (TS) and advance recovery (AR) approach but at significantly faster running times.

It can be observed that on average the IRCI performs well in benchmark instances against simpler and more complex algorithms for hard and soft time windows. The average CPU times are more than reasonable given the relatively modest processing capabilities of a 1.6 Mhz Pentium M laptop. In general, computation times are difficult to compare due to the differences in processing power. The interested reader is referred to Dongarra's work (2007) which includes the results of a set of standard programs to measure processing power and to compare the processing power of different machines. However, comparisons are not straightforward because not all the processors are included and there always differences in codes, compilers, and implementation computational efficiency.

5. DISCUSSION

The relative simplicity of the IRCI allows for a straightforward algorithmic analysis. The auxiliary heuristic $\mathbf{H}_r$ is called by the construction algorithm no more than $nW|\Delta|$ times; where n is the number of customers. Hence, the asymptotic number of operations of the

construction algorithm is of order $(nW|\Delta|O(\mathbf{H}_r(n)))$ where $O(\mathbf{H}_r(n))$ denotes the computational complexity of the auxiliary algorithm to route $n$customers.

The improvement procedure calls the construction procedure a finite number of times. The number of calls is bounded by the number of routes $|R|$. Further, the called computational time of the construction algorithm is $(mW|\Delta|O(\mathbf{H}_r(m)))$ where $m<n$ because only a subset of routes is iteratively improved.

It is clear that the complexity and running time of the auxiliary heuristic $\mathbf{H}_r$ will have a substantial impact on the overall running time. Hence, a generalized nearest neighbor heuristics of (GNNH) is used due to its reduced number of operations and computation time. In particular, if the GNNH has $O(n^2)$ and $W < n$, then the *worst case complexity* for the IRCI algorithm is of order $O(n^3)$.

To test the *average complexity*, instances with different numbers of customers are run. Firstly, the first 25 and 50 customers of each Solomon problem are taken to create instances with $n=$ 25 and $n=$50 respectively. Secondly, to create and instance with $n=$200 customer, for each customer in the original Solomon problem a "clone" is created but with new coordinates but still keeping the characteristics of the problem as clustered, random, or random-clustered. The summary results for the 56 Solomon problems are shown in Table 5. The results are expressed as the ratio between each average running time and the running time for $n=$25. To facilitate comparisons, the corresponding increases in running time ratios for $O(n^2)$ and

$O(n^3)$ are also presented. The results indicate that the average running time is increasing by a factor of $O(n^2)$ as expected from the complexity analysis and the last column of Table 5.

<< INSERT TABLE 5 HERE>>

The proposed IRCI approach can accommodate cost functions that cover most practical applications. The cost functions must be positive functions of fleet size, distance, time, or penalties. Cost functions can be asymmetrical, e.g. $p_e(t) \neq p_l(t)$ where t accounts for the early or late time. Additionally, cost functions are not required to be linear or identical. Similarly, symmetry is not required and $d_{ij} \neq d_{ji}$ or $t_{ij} \neq t_{ji}$ does not affect the complexity of the algorithm. That is, the corresponding penalty function can be non-convex and discontinuous as long as it is piecewise linear. In addition, customers with two or more time windows can be easily included in the auxiliary route construction algorithm. In addition, the number of routes m is not specified initially and it is an output of the solution algorithm. The bounds for the VRPHTW can be generated endogenously solving a relaxed VRPSTW beforehand.

The relatively simplicity and generality of the IRCI are important factors in real-world applications. Although solution quality and computation times are two key factors to evaluate vehicle routing heuristics, for practical implementations it is also crucial that algorithms are relatively simple and flexible (Cordeau et al., 2002). According to Cordeau et al (2002) the majority of the commercial software and in-house routing programs are still based on somewhat simple and unsophisticated methodologies dating back to the 1960s. Some of the reasons that explain this status quo are: (a) dispatchers preference for algorithms/programs that are highly interactive and allow for manual improvements and the manipulation of

constraints and customer priorities, (b) better results on benchmark problems are usually obtained at the expense of too many parameters or complicated coding that lacks flexibility to accommodate real-life constraints, (c) dispatcher may find algorithms with too many parameters difficult to calibrate or even understand, and (d) solution approaches that are markedly tailored to perform well on the benchmark problems may lack generality and robustness in real-life problems. As indicated by Golden et al. (1998), algorithms should also be compared not only by the number of parameters but also by how intuitive and reasonable these parameters are from a users perspective.

6. CONCLUSIONS

The main contribution of this paper is to propose an efficient, simple, and flexible algorithm to deal with general time window constraints and objective functions. The proposed IRCI algorithm provides high quality solutions and requires small computation times when compared with existing algorithms that can handle both hard and soft time window constraints. For a problem with 100 customers, the joint solution of soft and hard time window problems requires a few seconds The developed IRCI algorithm is based on a modular and hierarchical algorithmic approach. Its average running time is of order $O(n^2)$ and the worst case running time is of order $O(n^3)$.

The flexibility of the IRCI algorithm allows for a sequential and integrated solution of routing problems with soft and hard time windows. With both types of solutions, dispatchers can easily identify the customers and time windows that are increasing the number of routes. Fast solution times allow for cost-service tradeoff studies. In addition, soft time window solutions provide a workable and realistic alternative plan of action when the problem with hard time windows is infeasible.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the Oregon Transportation, Research and Education Consortium (OTREC) and the Department of Civil & Environmental Engineering in the Maseeh College of Engineering & Computer Science at Portland State University for sponsoring this research. The author would like to thank Stuart Bain, at the University of Sydney, for his assistance coding during the early stages of this research and Myeonwoo Lim , Computer Science Department at Portland State University, for assistance coding during the final stages of this research. Any errors are the sole responsibility of the author.

REFERENCES

BALAKRISHNAN, N. (1993) Simple Heuristics for the Vehicle Routeing Problem with Soft Time Windows. *The Journal of the Operational Research Society,* 44**,** 279-287.

BERGER, J., BARKAOUI, M. & BRAYSY, O. (2003) A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *INFOR,* 41**,** 179–194.

BRAYSY, O. (2002) Fast local searches for the vehicle routing problem with time windows. *Infor,* 40**,** 319-330.

BRAYSY, O. (2003) A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows. *INFORMS Journal on Computing,* 15**,** 347-368.

CALVETE, H. I., GALÉ, C., OLIVEROS, M. J. & SÁNCHEZ-VALVERDE, B. (2007) A goal programming approach to vehicle routing problems with soft time windows star, open. *European Journal of Operational Research,* 177**,** 1720-1733.

CASEAU, Y. & LABURTHE, F. (1999) Heuristics for Large Constrained Vehicle Routing Problems. *Journal of Heuristics,* 5, 281-303.

CHIANG, W. C. & RUSSELL, R. A. (2004) A metaheuristic for the vehicle-routeing problem with soft time windows. Journal of the Operational Research Society, 55, 1298-1310.

CORDEAU, J. F., GENDREAU, M., LAPORTE, G., POTVIN, J. Y. & SEMET, F. (2002) A guide to vehicle routing heuristics. Journal Of The Operational Research Society, 53, 512-522.

CORDONE, R. & CALVO, R. W. (2001) A Heuristic for the Vehicle Routing Problem with Time Windows. Journal of Heuristics, 7, 107-129.

DONGARRA, J. J. (2007) Performance of various computers using standard linear equations software. Technical Report CS-89-85 - University of Tennessee Nov 20, 2007, Accessed Nov 23, 2007, http://www.netlib.org/utk/people/JackDongarra/papers.htm.

FERLAND, J. A. & FORTIN, L. (1989) Vehicles Scheduling with Sliding Time Windows. European Journal of Operational Research, 38, 213-226.

GOLDEN, B., WASIL, E., KELLY, J. & CHAO, I. (1998) Metaheuristics in Vehicle Routing. IN CRAIGNIC, T. & LAPORTE, G. (Eds.) Fleet Management and Logistics. Boston, Kluwer.

HOMBERGER, J. & GEHRING, H. (1999) Two evolutionary metaheuristics for the vehicle routing problem with time windows. INFOR, 37, 297–318.

IBARAKI, T., IMAHORI, S., KUBO, M., MASUDA, T., UNO, T. & YAGIURA, M. (2005) Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints. Transportation Science, 39, 206-232.

IOANNOU, G., KRITIKOS, M. & PRASTACOS, G. (2001) A greedy look-ahead heuristic for the vehicle routing problem with time windows. Journal Of The Operational Research Society, 52, 523-537.

IOANNOU, G., KRITIKOS, M. & PRASTACOS, G. (2003) A problem generator-solver heuristic for vehicle routing with soft time windows. Omega, 31, 41-53.

KOSKOSIDIS, Y. A., POWELL, W. B. & SOLOMON, M. M. (1992) An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints. Transportation science, 26, 69-85.

LIU, F. H. F. & SHEN, S. Y. (1999) A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. European Journal of Operational Research, 118, 485-504.

POTVIN, J. & ROUSSEAU, J. (1993) A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research, 66, 331–340.

POWELL, W., MARAR, A., GELFAND, J. & BOWERS, S. (2002) Implementing Real-Time Optimization Models: A Case Application form the Motor Carrier Industry. Operations Research, Vol50, N4, pp. 571-581.

RUSSELL, R. A. (1995) Hybrid heuristics for the vehicle routing problem with time windows. Transportation science, 29, 156-166.

SEXTON, T. R. & CHOI, Y. M. (1986) Pickup and delivery of partial loads with" soft" time windows. American Journal of Mathematical and Management Sciences, 6, 369-398.

SOLOMON, M. M. (1987) Algorithms For The Vehicle-Routing And Scheduling Problems With Time Window Constraints. Operations Research, 35, 254-265.

TAILLARD, E., BADEAU, P., GENDREAU, M., GUERTIN, F. & POTVIN, J. Y. (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science, 31, 170-186.

TABLES

Table 1. **VRPHTW Results for Construction Algorithms vs. IRCI**

Average Number of Vehicles by Problem Class

Method	R1	R2	C1	C2	RC1	RC2
(1) Solomon (1987)	13.58	3.27	10.00	3.13	13.50	3.88
(2) Potvin et al. (1993)	13.33	3.09	10.67	3.38	13.38	3.63
(3) Ioannou et al. (2003)	12.67	3.09	10.00	3.13	12.50	3.50
(4) IRCI	12.50	3.09	10.00	3.00	12.00	3.38

Average Distance

Method	R1	R2	C1	C2	RC1	RC2
(1) Solomon (1987)	1,437	1,402	952	693	1,597	1,682
(2) Potvin et al. (1993)	1,509	1,387	1,344	798	1,724	1,651
(3) Ioannou et al. (2003)	1,370	1,310	865	662	1,512	1,483
(4) IRCI	1,262	1,171	872	656	1,420	1,342

Computation time for all 56 problems: (1) DEC 10, 1 run, 0.6 min.; (2) IBM PC, 1 run, 19.6 min.; (3) Intel Pentium 133 MHz, 1 run, 4.0 min. (4) Intel Pentium M 1.6 Mz, 10.9 min

Table 2. **VRPHTW Results for Metaheuristic Algorithms vs. IRCI**

Average Number of Vehicles by Problem Class

Method	R1	R2	C1	C2	RC1	RC2
(1) Taillard et al. (1997)	12.64	3.00	10.00	3.00	12.08	3.38
(2) Ibaraki et al. (2002)	11.92	2.73	10.00	3.00	11.50	3.25
(3) IRCI	12.50	3.09	10.00	3.00	12.00	3.38

Average Distance *by Problem Class*

Method	R1	R2	C1	C2	RC1	RC2
(1) Taillard et al. (1997)	1,220.4	1,013.4	828.5	590.9	1,381.3	1,198.6
(2) Ibaraki et al. (2002)	1,217.4	959.1	828.4	589.9	1,391.0	1,122.8
(3) IRCI	1,261.6	1,170.8	871.8	655.6	1,419.8	1,342.4

Computation time for all 56 problems: (1) Sun Sparc 10, 261 min.; (2) Pentium III 1 GHz, 250 min.; (3) Intel Pentium-M 1.6 Mhz 10.9 min

Table 3. **VRPSTW Results for R1 Problems.**

Wmax		10%				10%			
Pmax		0%				10%			
Method		(1) BAL	(2) TS	(3) AR	(4) IRCI	(1) BAL	(2) TS	(3) AR	(4) IRCI
R101	# Veh.	19	19	19	19	15	14	12	13
	Distance	1,915	1,710	1,692	1,639	1,832	1,388	1,212	1,493
	% HTW	100	100	100	100	62	49	8	39
R102	# Veh.	19	17	17	17	14	13	10	12
	Distance	1,890	1,520	1,511	1,481	1,569	1,266	1,173	1,463
	% HTW	100	100	100	100	81	59	33	60
R103	# Veh.		14	13	13	13	11	10	11
	Distance		1,225	1,304	1,284	1,657	1,063	1,013	1,274
	% HTW		100	100	100	83	65	58	73
R109	# Veh.	13	13	12	12	12	11	10	11
	Distance	1,492	1,280	1,165	1,240	1,431	1,102	1,005	1,280
	% HTW	100	100	100	100	90	72	47	82
AVERAGE	# Veh.	17.0	15.8	15.3	15.3	13.5	12.3	10.5	12.3
	Distance	1,766	1,434	1,418	1,411	1,622	1,205	1,101	1,467
	% HTW	100	100	100	100	79.0	61.2	36.5	66.3

Computation time for each STW problem: (1) 25Mhz 80386, 17 to 73 seconds; (2) 2.25 Ghz Athlon, 52 to 82 seconds; (3) 2.25 Ghz Athlon, 448 to 692 seconds; (4) 1.6 Ghz Pentium-M, 4.5 to 4.9 seconds

Table 4. **VRPSTW Results for RC1 Problems.**

Wmax		10%				10%			
Pmax		0%				10%			
Method		(1) BAL	(2) TS	(3) AR	(4) IRCI	(1) BAL	(2) TS	(3) AR	(4) IRCI
RC101	# Veh.	16	15	15	15	14	15	11	14
	Distance	2,012	1,719	1,651	1,644	1,795	1,569	1,275	1,839
	% HTW	100	100	100	100	61	62	27	73
RC102	# Veh.	14	13	13	13	13	12	11	13
	Distance	1,808	1,519	1,530	1,575	1,719	1,307	1,222	1,632
	% HTW	100	100	100	100	83	68	56	81
RC103	# Veh.	12	11	11	11	12	10	10	11
	Distance	1,679	1,293	1,284	1,318	1,530	1,228	1,119	1,400
	% HTW	100	100	100	100	92	85	65	92
RC106	# Veh.		12	12	12	13	12	10	12
	Distance		1,445	1,409	1,412	1,620	1,262	1,160	1,487
	% HTW	100	100	100	100	97	77	49	92
AVERAGE	# Veh.	14.0	12.8	12.8	12.8	13.0	12.3	10.5	12.5
	Distance	1,833	1,494	1,469	1,488	1,666	1,342	1,194	1,590
	% HTW	100	100	100	100	83.3	73.0	49.2	84.5

Computation time for each STW problem: (1) 25Mhz 80386, 17 to 73 seconds; (2) 2.25 Ghz Athlon, 52 to 82 seconds; (3) 2.25 Ghz Athlon, 448 to 692 seconds; (4) Intel Pentium-M 1.6 Mhz 4.5 to 4.9 seconds

Table 5. **VRPTW Average Run Time Ratios – VRPHTW**

(1) n	(2) $O(n^2)$	(3) $O(n^3)$	(4) Run Time Ratio*	(5)= (4)/(3)*100 % $O(n^3)$
25	1	1	1.0	100%
50	4	8	2.9	36%
100	16	64	15.0	23%
200	64	512	86.3	17%

* The ratio of running times is taking the run time for n=25 as a base.