Transportation Research Part C 18 (2010) 668–679

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows

Miguel Andres Figliozzi *

Department of Civil and Environmental Engineering, Portland State University-CEE, Portland 97201-0751, USA

ARTICLE INFO

Article history:
Received 14 December 2008
Received in revised form 15 August 2009
Accepted 24 August 2009

Keywords:
Vehicle routing
Soft time windows
Route construction and improvement algorithms

ABSTRACT

The solution of routing problems with soft time windows has valuable practical applications. Soft time window solutions 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 (c) the dispatcher has qualitative information regarding the relative importance of hard time-window constraints across customers. This paper proposes a new iterative route construction and improvement algorithm to solve vehicle routing problems with soft time windows. Due to its modular and hierarchical design, the solution algorithm is intuitive and able to accommodate general cost and penalty functions. Experimental results indicate that the average run time performance is of order $O(n^2)$. The solution quality and computational time of the new algorithm has been compared against existing results on benchmark problems. The presented algorithm has improved thirty benchmark problem solutions for the vehicle routing problems with soft time windows.

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 a flexible algorithm that can be applied to solve VRPSTW instances. It is useful for dispatchers to have these different solutions when: (a) the number of routes needed for the hard time window (HTW) problem 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. Besides, 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).

* Tel.: +1 503 725 2836; fax: +1 503 725 5950.
E-mail address: figliozzi@pdx.edu

0968-090X/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trc.2009.08.005

This paper provides a fast and high quality solution approach that solves soft time window (STW) problems and that can also be easily adopted to solve hard time window problems using the STW solution as a lower bound. The rest of this paper is organized into five additional sections. Section 2 reviews the relevant literature on VRPSTW problems. Section 3 introduces the mathematical notation and describes the new iterative route construction and improvement (IRCI) algorithm. Section 4 compares IRCI computation time and solution quality against available benchmarked solutions. Section 5 discusses IRCI algorithmical properties. Section 6 ends with conclusions.

2. Literature review

Heuristics to solve the VRP with time windows can be classified (in increasing order of solution quality) as construction heuristics, local search heuristics, and metaheuristics. Metaheuristics generally produce solutions of higher quality but this is usually at the expense of significantly longer computation times. There is a clear tradeoff 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 Russell (1995), Caseau and Laburthe (1999), Cordone and Calvo (2001), Braysy (2002), and Ibaraki et al. (2005). The method proposed by Cordone and Calvo (2001) is a relatively simple deterministic heuristic based on the iterative application of k-opt exchanges combined with an insertion (or simple ejection chain) procedure to reduce the number of routes. The IRCI is also based on an iterative application of algorithms, i.e. route construction and improvement algorithms. However, the IRCI is not a local search method but a *route* based method as detailed in Section 3.

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 metaheuristics include the algorithms proposed by Taillard et al. (1997), Liu and Shen (1999), Homberger and Gehring (1999), Berger et al. (2003), and Braysy (2003). Recent publications include the work of Hashimoto and Yagiura (2008) that proposed a path relinking approach with an adaptive mechanism to control parameters where the generated solutions in the path relinking are improved by a local search. Hsu et al. (2007) proposed an algorithm tailored to the problem of perishable food distribution. 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 variation of the VRPSTW where customers' time windows are adjusted to lower service costs. Koskosidis et al. (1992) proposes 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 generates and modifies 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. Hashimoto et al. (2006) proposes an algorithm for flexible time windows (hard and soft) and travel times using local search; soft time window and soft traveling time constraints are treated as part of the objective function and the authors deal with a generalized VRP. Fu et al. (2008) adapted a tabu search algorithm, previously used in the open vehicle routing problem (Fu et al., 2005), for the VRPSTW.

It is assumed in the literature that fixed costs associated with each additional route (vehicle) outweigh travel time or distance related costs, i.e. a multi-objective hierarchical approach to evaluate solutions. For VRPSTW problems, it is assumed that the primary objective is the minimization of vehicles, the secondary objective is the minimization of the number of soft time windows used, and the tertiary objective is the minimization of distance travel. Although this rigid hierarchy facilitates benchmarking, in real-world situations it is possible that only a *subset* of customers may allow soft time windows. Hence, practical problems may consist of a *mix* of hard and soft time windows. It is valuable that solution approaches can deal with both types of time windows.

As indicated by Braysy and Gendreau (2005a,b), fair and meaningful comparisons of vehicle routing heuristics require standard benchmark problems, common objective functions, and the full reporting of: (a) solution quality, (b) number of runs needed and computation time per run, and (c) computing power or processor speed. To the best of the author's knowledge, the only three journal publications that include results for VRPSTW benchmark problems and comply with prerequisites (a)–(c) are: Balakrishnan (1993), Chiang and Russell (2004), and Fu et al. (2008). Section 4 compares IRCI results with previous results found in the VRPSTW literature in terms of solution quality and computational time.

3. Problem definition and solution algorithm

This section begins with an introduction of a precise mathematical definition of the problems studied in this research. The remainder of this section describes the solution algorithm.

3.1. 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 $V = (v_0, \ldots, v_n)$ 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 $C = \{v_1, \ldots, v_n)$ specify the location of a set of n customers. Each vertex in V has an associated demand $q_i \geqslant 0$, a service time $s_i \geqslant 0$, 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.
(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 $P_{\max} \geqslant 0$. 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.

3.2. Solution algorithms

At its core the IRCI algorithm is a construction algorithm were routes are sequentially built and improved. The solution method is divided into two phases: route construction and route improvement. Unlike local search heuristics, where customers or groups of *customers* are exchanged and inserted, the IRCI improves the solution of VRP problems creating and improving groups or sets of *routes*.

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.

3.2.1. The auxiliary algorithm

The auxiliary routing algorithm $\mathbf{H}_r$ can be any heuristic that is given a starting vertex, a set of customers, and a depot location that 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 $\Delta = \{\delta_0, \delta_1, \ldots, \delta_i\}$, (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:

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

The parameter δ_1 takes into account the weight of the intercustomer distance. The parameter δ_2 takes into account the "slack" between the completion of service at i and the earliest feasible beginning of service at j, i.e. $a_j = \max(y_i + s_i + t_{ij},\ e_j)$. 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:

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

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 $\Delta = \{\delta_0, \delta_1, \ldots, \delta_6\}$ 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 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 $\delta_i \geqslant 0\ i \in \{0, 1, \ldots, 6\}$.

3.2.2. 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 $w(v_i,\ C,\ g,\ W)$ that given a set of unrouted customers C, a vertex $v_i \notin C$, and a generalized cost function $g(\Delta,\ v_i,\ v_j)$ returns a set of vertexes with the lowest generalized costs $g(\Delta,\ v_i,\ v_j)$ for all $v_j \in C$.

Route construction algorithm

Functions or algorithms
$\mathbf{H}_r$: route building heuristic
$w(v_i,\ C,\ g,\ W)$: returns set of vertexes with the lowest generalized costs
$c(R)$ = function that returns the cost of a route R

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* ← v_0
2 *start* ← v_0
3 *bestSequence* ← *v*
4 # *vehicles* ← min#veh ← *lowestCost* ← ∞
5 *Ccopy* ← *C*
6 **for** each $\Delta \in \mathbf{\Delta}$
7 **while** $C \neq \varnothing$ **AND** *LLimit* < #*vehicles* **AND** #*vehicles* ⩽ ***min***#*veh* **do**
8 $W \leftarrow \min(W, |C|)$
9 $C^* \leftarrow w(start, C, g, W)$
10 **for** each $v_i \in C^*$
11 **if** $c(bestSequence \cup v_i) + c(\mathbf{H}_r\ (\Delta, v_i, C, v_0)) < lowestCost$ **then**
12 $lowestCost \leftarrow c(bestSequence \cup v_i) + c(\mathbf{H}_r(\Delta,\ v_i,\ C,\ v_0))$
13 $lowestNext \leftarrow v_i$
14 **end if**
15 **end for**
16 *start* ← *lowestNext*

```
17    C ← C \ lowestNext
18    bestSequence ← bestSequence ∪ lowestNext
19    R ← bestSequence ∪ H_r (Δ, lowestNext, C, v_0)
20    #vehicles ← cardinality of the set of routes R
21    end while
22    C ← Ccopy
23    if min#veh > #vehicles
24      min#veh ← #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.

The running time is significantly improved if $\mathbf{H}_r$ is used to complete the current route with as many customers as possible and with the minimum distance cost, i.e. $\mathbf{H}_r$ returns a partial solution instead of a solution that routes all customers. Then, in line (11) of the $\mathbf{H}_c$ algorithm the best solution is the one that inserts the most costumers in the current route where v_i belongs; in case of ties they are broken by the distance of the partial solutions.

3.2.3. 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 $k_p(r_i, S, p)$ 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 $k_p(r_i, S, p)$.

The function $k_s(R, s)$ orders the set of routes R from smallest to largest based on a criteria such as the number of customers per route, route distance, or route duration and then returns a set of $s \geqslant 1$ routes with the least number of customers; e.g. $k_s(R, 1)$ will return the route with the least number of customers if this is the chosen criteria. The intuition is that routes with a small number of customers, distance traveled, duration, or volume utilization have a considerable "capacity slack" and can therefore be improved significantly. These four basic measures of capacity slack and their combinations are used to select subsets of routes in the improvement phase of IRCI. If two or more routes are tied in a given criteria, ties are solved drawing random numbers or by looking at a secondary criteria. To simplify notation the term $C(S)$ is the set of customers served by the set of routes S.

Route construction algorithm

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 $G' \leftarrow \mathbf{H}_c(\mathbf{H}_r, W, \Delta, s, p, C(G), LLimit)$

```
    if c(G') < c(G) then
      R ← R \ G
      R ← R ∪ G'
      S ← S \ G
      S ← S ∪ G'
    end if
    s ← |S|
    r = k_s(S, s)
    S = S \ r
    if |S'| > 0 then
      r' = k_s(S', 1)
      S ← S ∪ r'
      S' ← S'/r'
      s ← min(s, |S|)
      p ← min(s, p)
    end if
  end while
Output
  R set of improved routes
END H_i
```

3.2.4. 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) $e_i + s_i + d_{ij} \leqslant l_j$ if customer j is not using a soft time windows or (2) $e_i + s_i + d_{ij} \leqslant l_j^{\#}$ 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 $y_i = \min(l_j^{\#} - (s_i + d_{ij}),\ l_i)$.

This next section compares the IRCI against other solution approaches using standard benchmark problems for the VRPSTW.

4. Computational results

As seen in the previous section, at its core the IRCI algorithm is a construction algorithm where 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 benchmark problems for the VRPSTW. These benchmark problems are variations of the well known Solomon benchmark problems for the VRPHTW.

The 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.

The first set of benchmark problems was originally proposed by Balakrishnan (1993). Balakrishnan (1993), Chiang and Russell (2004), Fu et al. (2008) are the only references with time and cost results for this set of VRPSTW problems. Balakrishnan (1993) worked on 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), Chiang and Russell (2004), and Fu et al. (2008) 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 $(e_i - P_{\max} - W_{\max})$ and waiting at a customer after service is not allowed. Since the VRPSTW is a relaxation of the VRPHTW, a new constraint that limits the maximum waiting time is clearly opposed to the spirit of the VRPSTW. Further, a maximum waiting time constraint $W_{\max}$ is completely unrelated to time-window constraints.[1] Despite these shortfalls, a $W_{\max}$ = 10% constraint is added, mainly to facilitate comparisons on a level playing field.

[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.

The solution results presented by Balakrishnan (1993) are denoted BAL. The solution results presented by Chiang and Russell (2004) which use two solution methods, tabu search (TB) and advance recovery (AR), are denoted by the initials TB and AR respectively. The solution results presented by Fu et al. (2008) using the unified tabu search method are denoted (UTS). This paper presents two solutions that utilize the same IRCI algorithms and different running times. These solutions are labeled "IRCI*s*" and "IRCI*e*", the final letter of each label stands for "short" and "extended" computational times respectively. IRCI*e* explores are larger number of generalized cost parameters Δ and runs more instances of the improvement algorithm $\mathbf{H}_i$.

Table 1 presents the solution results for R1 problems P_{max} = 5% and W_{max} = 10%. The penalty coefficients are set equal to 1, i.e. a unit of time of time window violation is equal to a unit of distance traveled. Table 1 presents the results in order of hierarchical value: number of routes, percentage of customers without time window violations, and distance traveled. The result is that the IRCI*e* outperforms other heuristics and has improved four of the existing benchmark solutions. The asterisk "$*$" is used to indicate that a solution has been improved.

Table 2 presents the solution results for RC1 problems with P_{max} = 10% and W_{max} = 10%. Here we see that the IRCI*e* has improved two of the existing benchmark solutions. It is clear that there is a tradeoff between the number of vehicles and time window violations. Consistently, for all methods, the number of routes is negatively correlated with time window violations.

Tables 3 and 4 present the solution results for R1 and RC1 problems with P_{max} = 10% and W_{max} = 10% respectively. In these benchmark problems the IRCI*e* has improved three solutions and presents the best average performance for R1 problems. Consistently, the number of routes is negatively correlated with the number of time window violations. Previous methods are outperformed in a total of 9 problems. In addition, the IRCI outperforms, on average, previous results in two of the four sets of problems.

The range of computational times for each problem is detailed in Table 5. There is a clear tradeoff between computational times and solution quality as shown by the results provided by the AR and TS methods of Chiang and Russell (2004) and by the IRCI results with short and extended exploration and improvement phases. However, comparisons regarding computation times should be performed with caution because in general, computation times are difficult to compare due to the differences in processing power and hardware. The interested reader is referred to Dongarra's work (2007) which includes the results of a set of standard programs that measure and compare processing power of different machines. Unfortunately, comparisons are usually not straightforward because not all processors are included in Dongarra's work. In addition, it is difficult to account for potential differences in codes, compilers, and implementation computational efficiency.

The second set of VRPSTW benchmarks was proposed by Fu et al. (2008) assuming that the late delivery time window can be violated with a penalty coefficient for late time windows that is equal to 100, i.e. a unit of time of time window violation is equal to 100 units of distance traveled. The soft time window for late deliveries can be extended up to the depot closing time. The results for the 39 instances of R1, R2, RC1, and RC2 problems are presented in Table 6; the results are in order of hierarchical value: number of routes, percentage of customers without time window violations, and distance traveled. Results for problems type C1 and C2 are omitted because in these instances the number of routes is bounded by capacity constraints. Hence, even relaxing both early and late time windows does not reduce the number of routes.

Table 1
VRPSTW results for R1 problems, W = 10%, P = 5%.

W_{max}	10%						
P_{max}	5%						
Method	(1) BAL	(2) TS	(3) AR	(4) UTS	(5) IRCI*s*	(6) IRCI*e*	
R101							
# Veh.	17	16	14	14	15	14	
% HTW	72	65	24	45	71	68	*
Distance	1885	1491	1370	1438	1703	1633	
R102							
# Veh.	15	13	12	12	13	12	
% HTW	83	1322	47	61	84	63	*
Distance	1636	69	1265	1339	1629	1404	
R103							
# Veh.	13	11	11	11	11	11	
% HTW	86	77	59	73	84	93	*
Distance	1452	1184	1066	1168	1357	1374	
R109							
# Veh.	13	12	11	11	11	11	
% HTW	95	84	60	75	85	93	*
Distance	1445	1154	1084	1168	1336	1393	
Aver.							
# Veh.	14.5	13.0	12.0	12.0	12.5	12.0	
% HTW	84	74	48	64	81	79	*
Distance	1605	1288	1196	1278	1506	1451	

Table 2
Results for RC1 problems, $W = 10\%$, $P = 5\%$.

W_{max}	10%						
P_{max}	5%						
Method	(1) BAL	(2) TS	(3) AR	(4) UTS	(5) IRCI*s*	(6) IRCI*e*	
RC101							
# Veh.	14	14	13	13	14	13	
% HTW	56	71	39	64	94	93	*
Distance	1839	1521	1424	1529	1776	1778	
RC102							
# Veh.	13	13	11	12	13	12	
% HTW	88	76	58	81	96	98	*
Distance	1850	1384	1375	1413	1653	1635	
RC103							
# Veh.	12	11	10	11	11	10	
% HTW	82	92	69	86	98	83	
Distance	1469	1243	1183	1254	1456	1256	
RC106							
# Veh.	12	12	11	11	12	11	
% HTW	71	81	61	81	96	80	
Distance	1496	1338	1223	1336	1507	1522	
Aver.							
# Veh.	12.8	12.5	11.3	11.8	12.5	11.5	
% HTW	74	80	57	78	96	89	
Distance	1664	1372	1301	1383	1598	1548	

Table 3
VRPSTW results for R1 problems, $W = 10\%$, $P = 10\%$.

W_{max}	10%						
P_{max}	5%						
Method	(1) BAL	(2) TS	(3) AR	(4) UTS	(5) IRCI*s*	(6) IRCI*e*	
R101							
# Veh.	15	14	12	12	13	12	
% HTW	62	49	8	31	43	25	
Distance	1832	1388	1212	1376	1493	1314	
R102							
# Veh.	14	13	10	11	12	10	
% HTW	81	59	33	51	63	25	
Distance	1569	1266	1173	1287	1463	1238	
R103							
# Veh.	13	11	10	10	11	10	
% HTW	83	65	58	76	76	66	
Distance	1657	1063	1013	1185	1274	1138	
R109							
# Veh.	12	11	10	11	11	10	
% HTW	90	72	47	82	83	53	*
Distance	1431	1102	1005	1183	1280	1116	
AVER.							
# Veh.	13.5	12.3	10.5	11.0	11.8	10.5	
% HTW	79	61	37	60	66	42	*
Distance	1622	1205	1101	1258	1378	1201	

The IRCI*e* improves the results of 21 instances by reducing the number of routes. Yet again, the number of routes is negatively correlated with the number of time window violations. The IRCI solutions utilize fewer vehicles but also violate more time windows. The asterisk "$*$" is used to indicate that the number of routes has been reduced. The average computation times per problem class are presented in Table 7.

The focus of this research is on STW problems. Hard time window problems have been extensively studied and the literature review briefly mentioned some of the most successful approaches. However, to provide an indication of the flexibility of the IRCI*s*, Table 8 presents the performance of the IRCI against other construction heuristics for the VRPHTW proposed by Solomon (1987), Potvin and Rousseau (1993), and Ioannou et al. (2001). Table 9 shows a summary of the results when the IRCI algorithm is compared against two metaheuristics presented in the literature review that were explicitly designed to

Table 4
Results for RC1 problems, $W = 10\%$, $P = 10\%$.

W_{max}	10%						
P_{max}	5%						
Method	(1) BAL	(2) TS	(3) AR	(4) UTS	(5) IRCI*s*	(6) IRCI*e*	
RC101							
# Veh.	14	15	11	12	14	11	*
% HTW	61	62	27	54	73	43	
Distance	1795	1569	1275	1457	1839	1322	
RC102							
# Veh.	13	12	11	11	13	11	
% HTW	83	68	56	74	81	63	
Distance	1719	1307	1222	1367	1632	1288	
RC103							
# Veh.	12	10	10	11	11	10	*
% HTW	92	85	65	90	92	79	
Distance	1530	1228	1119	1275	1400	1194	
RC106							
# Veh.	13	12	10	11	12	11	
% HTW	97	77	49	81	92	66	
Distance	1620	1262	1160	1337	1487	1210	
Aver.							
# Veh.	13.0	12.3	10.5	11.3	12.5	10.8	
% HTW	83	73	49	75	85	63	
Distance	1666	1342	1194	1359	1590	1253	

Table 5
Computation time for each STW problem.

Method	CPU	Running time for each problem
(1) BAL	25 MHz 80386	17–73 s
(2) TS	2.25 GHz Athlon	52–82 s
(3) AR	2.25 GHz Athlon	448–692 s
(4) UTS	600 MHz Pentium-II	193–1900 s
(5) IRCI*s*	Intel Pentium-M 1.6 MHz	4.5–4.9 s
(6) IRCI*e*	Intel Pentium-M 1.6 MHz	537–653 s

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). The reported time for the IRCI*s* corresponds to the total time needed to solve *both* types of problems (soft and hard) for all 56 Solomon instances. The result of the STW problem is used as a lower bound for the HTW problem. The other references solve only the VRPHTW type because this is the standard way of testing. Hence, results presented in Tables 8 and 9 are used to highlight that the IRCI can be used as a tool to quickly evaluate tradeoffs between HTW and STW solutions and to identify "difficult" customers that increase the number of vehicles needed.

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 $nW|\mathbf{\Delta}|$ times; where n is the number of customers. Hence, the asymptotic number of operations of the construction algorithm is of order $(nW|\mathbf{\Delta}|O(\mathbf{H}_r(n)))$ 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 $(mW|\mathbf{\Delta}|O(\mathbf{H}_r(m)))$ 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 were run. The instances were based on the four R1 problems proposed by Balakrishnan. Firstly, the first 25 and 50 customers of each of problem are taken to create instances with $n = 25$ and $n = 50$, respectively. Secondly, in order to create an instance with $n = 200$ customer, a "clone" is

Table 6
VRPSTW results for type 1 problems.

Problem	Fu et al. (UTS)			IRCI*e*			
	Routes	HTW (%)	Distance	Routes	HTW (%)	Distance	
R101	14	75	1535.2	12	44	1128.7	*
R102	13	89	1416.8	11	54	1058.7	*
R103	11	96	1267.3	10	66	1027.4	*
R104	9	99	983.5	9	82	947.3	
R105	13	98	1441.2	11	58	1073.5	*
R106	11	97	1355.3	10	67	1047.4	*
R107	10	100	1147.6	10	76	987.6	
R108	9	100	978.7	9	86	947.2	
R109	11	100	1264.2	10	72	1001.4	*
R110	11	100	1084.0	9	71	1013.4	*
R111	10	100	1138.5	10	74	983.3	
R112	10	100	963.2	9	83	940.9	*
R201	3	89	1500.4	3	44	984.0	
R202	3	100	1205.8	3	60	943.5	
R203	3	100	950.4	2	70	901.8	*
R204	2	100	854.3	2	81	836.3	
R205	3	100	1001.8	3	64	911.9	
R206	3	100	917.9	2	75	956.9	*
R207	2	100	903.0	2	82	876.6	
R208	2	100	738.3	2	89	833.4	
R209	3	100	909.9	2	74	950.5	*
R210	3	100	948.2	2	71	963.8	*
R211	2	100	953.2	2	86	906.8	
RC101	13	92	1654.3	11	56	1255.3	*
RC102	12	100	1593.7	10	68	1230.1	*
RC103	11	100	1321.7	10	75	1154.6	*
RC104	10	100	1175.2	10	88	1083.9	
RC105	12	92	1654.1	11	62	1219.7	*
RC106	11	99	1422.7	10	73	1150.3	*
RC107	11	100	1237.6	10	72	1123.0	*
RC108	10	100	1184.6	10	90	1071.6	
RC201	4	100	1409.9	3	52	1147.4	*
RC202	3	100	1435.6	3	65	1073.5	
RC203	3	100	1062.4	3	71	906.3	
RC204	3	100	800.0	2	86	850.7	*
RC205	3	93	1656.8	3	60	1158.4	
RC206	3	100	1186.8	3	60	978.4	
RC207	3	100	1127.8	3	67	986.4	
RC208	3	100	846.1	2	79	885.5	*

Table 7
Running times for type 1 problems.

Average CPU time in seconds		
	UTS	IRCI*e*
R1	786.3	724.1
R2	528.3	402.3
RC1	697.3	715.6
RC2	548.7	478.1

UTS: 600 MHz Pentium-II; IRCI*e*: Pentium-M 1.6 MHz.

created for each customer in the original Solomon problem. The clone is created with new coordinates while still keeping the characteristics of the problem as random. The summary results for the eight problems with $W = 10$ and $P = 10$ are shown in Table 10.

The results are expressed as the ratio between each average running time and the running time for $n = 25$ for the IRCIe. 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 10.

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,

Table 8
VRPHTW results for construction algorithms vs. IRCI.

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles by problem class						
(1) Solomon (1987)	13.58	3.27	10.00	3.13	13.50	3.88
(2) Potvin and Rousseau (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) Cordone and Calvo (2001)	12.50	2.91	10.00	3.00	12.38	3.38
(5) IRCI*s*	12.50	3.09	10.00	3.00	12.00	3.38
Average distance						
(1) Solomon (1987)	1437	1402	952	693	1597	1682
(2) Potvin and Rousseau (1993)	1509	1387	1344	798	1724	1651
(3) Ioannou et al. (2003)	1370	1310	865	662	1512	1483
(4) Cordone and Calvo (2001)[a]	1242	995	834	592	1409	1140
(5) IRCI*s*	1262	1171	872	656	1420	1342

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) Pentium 166 MHz, 1 run, 15.7 min; (5) Intel Pentium-M 1.6 MHz, 10.9 min.

[a] The work of Cordone and Calvo (2001) is an iterative local search heuristic not a construction heuristic. However, it is included in this table because it has a common element with IRCI: it is based on the iterative application of simple heuristics.

Table 9
VRPHTW results for selected metaheuristic algorithms vs. IRCI.

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles by problem class						
(1) Taillard et al. (1997)	12.64	3.00	10.00	3.00	12.08	3.38
(2) Ibaraki et al. (2005)	11.92	2.73	10.00	3.00	11.50	3.25
(3) IRCI*s*	12.50	3.09	10.00	3.00	12.00	3.38
Average distance by problem class						
(1) Taillard et al. (1997)	1220.4	1013.4	828.5	590.9	1381.3	1198.6
(2) Ibaraki et al. (2005)	1217.4	959.1	828.4	589.9	1391.0	1122.8
(3) IRCI*s*	1261.6	1170.8	871.8	655.6	1419.8	1342.4

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 10
VRPSTW average run time ratios.

(1) n	(2) $O(n^2)$	(3) $O(n^3)$	(4) Run time ratio IRCI*e*[a]	(5) = (4)/(3) * 100 IRCI*e* % $O(n^3)$
25	1	1	1.0	100
50	4	8	2.2	28
100	16	64	12.2	19
200	64	512	48.8	10

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

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. The number of routes m is not specified initially and it is an output of the solution algorithm.

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 simple and unsophisticated methodologies dating back to the 1960s. Some of the reasons that explain this preference 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 user's perspective.

The relative simplicity and generality of the IRCI are important factors in real-world applications. The IRCI algorithms have also been adapted to solve time-dependent vehicle routing problems (TDVRP) as detailed in (Figliozzi, 2009).

6. Conclusions

The main contribution of this paper is the introduction of an efficient, simple, and flexible algorithm for the vehicle routing problem with soft time windows. Thirty soft time window benchmark problems have been improved. The proposed IRCI algorithm can provide reasonable solutions with small computation times or high quality solutions with extended computation times. 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 also allows for a sequential solution of routing problems with soft and hard time windows.

Acknowledgements

The author gratefully acknowledges the Oregon Transportation, Research and Education Consortium (OTREC) and the Department of Civil and Environmental Engineering in the Maseeh College of Engineering and 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 routing 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, I., Gendreau, M., 2005a. Vehicle routing problem with time windows, part 1: route construction and local search algorithms. Transportation Science 39, 104–118.

Braysy, I., Gendreau, M., 2005b. Vehicle routing problem with time windows, part II: metaheuristics. Transportation Science 39, 119–139.

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-routing 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, November 20, 2007. <http://www.netlib.org/utk/people/JackDongarra/papers.htm> (accessed 23.11.07).

Ferland, J.A., Fortin, L., 1989. Vehicles scheduling with sliding time windows. European Journal of Operational Research 38, 213–226.

Figliozzi, M., 2009. A route improvement algorithm for the vehicle routing problem with time dependent travel times. In: Proceeding of the 88th Transportation Research Board Annual Meeting CD ROM, Washington, DC, January 2009. <http://web.cecs.pdx.edu/~maf/publications.html>.

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

Fu, Z., Eglese, R., Li, L.Y.O., 2008. A unified tabu search algorithm for vehicle routing problems with soft time windows. Journal of the Operational Research Society 59, 663–673.

Golden, B., Wasil, E., Kelly, J., Chao, I., 1998. Metaheuristics in vehicle routing. In: Craignic, T., Laporte, G. (Eds.), Fleet Management and Logistics. Kluwer, Boston.

Hashimoto, H., Ibaraki, T., Imahori, S., Yagiura, M., 2006. The vehicle routing problem with flexible time windows and traveling times. Discrete Applied Mathematics 154, 2271–2290.

Hashimoto, H., Yagiura, M., 2008. A path relinking approach with an adaptive mechanism to control parameters for the vehicle routing problem with time windows. Lecture Notes in Computer Science 4972, 254.

Homberger, J., Gehring, H., 1999. Two evolutionary metaheuristics for the vehicle routing problem with time windows. INFOR 37, 297–318.

Hsu, C.I., Hung, S.F., Li, H.C., 2007. Vehicle routing problem with time-windows for perishable food delivery. Journal of Food Engineering 80, 465–475.

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 50 (4), 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.