PERGAMON

Transportation Research Part B 34 (2000) 107–121

TRANSPORTATION RESEARCH PART B

www.elsevier.com/locate/trb

Solving the pickup and delivery problem with time windows using reactive tabu search

William P. Nanry [*], J. Wesley Barnes

Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, Austin, TX 78712, USA

Received 13 July 1998; accepted 24 April 1999

Abstract

The pickup and delivery problem with time windows requires that a group of vehicles satisfy a collection of customer requests. Each customer request requires the use of a single vehicle both to load a specified amount of goods at one location and to deliver them to another location. All requests must be performed without violating either the vehicle capacity or the customer time window stipulated at each location. In this paper, we present a reactive tabu search approach to solve the pickup and delivery problem with time windows using three distinct move neighborhoods that capitalize on the dominance of the precedence and coupling constraints. A hierarchical search methodology is used to dynamically alternate between neighborhoods in order to negotiate different regions of the solution space and adjust search trajectories. In order to validate our technique's effectiveness, we have constructed a new set of benchmark problems for the pickup and delivery problem with time windows based on Solomon's benchmark vehicle routing problem with time windows data sets. Computational results substantiate the solution quality and efficiency of our reactive tabu search approach. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Tabu search; Coupling constraints; Neighborhood selection; Benchmark problems

1. Introduction

This paper presents a reactive tabu search approach, RTS-PDPTW, for solving the pickup and delivery problem with time windows (PDPTW). The specific problem that we address requires the satisfaction of a set of transportation requests by a homogeneous vehicle fleet housed at one depot. Each transportation request requires picking up material at a predetermined location during its associated time window and delivering it to a "paired" destination during its time

[*] Corresponding author. Tel.: +512-471-3083; fax: +512-232-1489; e-mail: wbarnes@mail.utexas.edu

window. Loading and unloading times are incurred at each location. In addition to the above precedence constraints, each route must satisfy *coupling* constraints since paired pickup and delivery locations must be serviced by the same vehicle. Precedence and coupling constraints are strictly enforced at all times.

Economic incidents such as the oil crisis of the early 1970s, deregulation of the US airline and trucking industry in the 1980s, and a rapidly declining military budget have motivated both private companies, government entities, and academic researchers to vigorously pursue new methods to improve the efficiency of logistics, distribution and transportation. This rekindled interest has fueled the recent development in metaheuristic procedures. New developments in adaptive memory strategies of tabu search (TS) have been especially productive in solving difficult combinatorial optimization problems with greater effectiveness than ever before. Time windows constrained routing problems form a large segment of this class of combinatorial optimization problems.

PDPTW routing problems arise under a variety of circumstances. They describe situations in which vehicles must travel to a variety of places to deliver and/or pick up goods and provide services. The enormous costs of acquisition or leasing of additional vehicles often compel managers to exhaust the capacity of the existing fleet. Simultaneously, increasing fuel, maintenance, and overtime costs strongly favor minimizing the usage of those vehicles. Practical applications of the PDPTW include the dial-a-ride problem, airline scheduling, bus routing, tractor–trailer problems, helicopter support of offshore oil field platforms and logistics and maintenance support. They also arise in less obvious situations such as VLSI circuit design, flexible manufacturing systems, and evacuating casualties.

We assume that all parameters of the PDPTW are known with certainty. Each problem has a set P of customers requiring service, either a pickup, P^+, or a delivery, P^- and $P^+ \cup P^- = P, |P| = n$, and $|P^+| = |P^-| = n/2$. Each member of P^+ is uniquely paired with its successor in P^-, forming $n/2$ predecessor–successor pairs (ps-pairs). The set of n customers is indexed by subscripts, i and j. Other parameters required to define a specific problem instance are travel times between all pairs of locations, t_{ij}; the service duration time, s_i; demand, d_i; earliest service start time, e_i; and latest service start time, l_i, for each of the customers. $d_i > 0$ for pickup customers and $d_i < 0$ for delivery customers. If $s_i > 0$, it is added to all t_{ij} for customer i. Thus, nonzero s_i need not be considered explicitly by the solution algorithm after their effects are accounted for in the t_{ij} and l_i. If customer j is the paired successor of customer i, t_{ji} will be set to an arbitrarily large value, M, to make its selection undesirable. There are two other cases where the travel time between locations will be set arbitrarily large. Vehicles leave the depot empty and must travel to a supplier first, not a delivery location. Since all vehicles are assumed to return to the depot empty, the last stop a vehicle makes is to deliver any remaining supplies on the vehicle. Travel times from the depot to a delivery location and from the supplier returning to the depot will be set to M. All distances are assumed to satisfy the triangle inequality, unless otherwise indicated.

Each problem has a set V of available vehicles, indexed by k, such that $|V| = m$, and each vehicle has a known capacity C. We have used the transformation developed by Laporte (1992) to represent a solution to the PDPTW, i.e., a solution is given by a vector indicating the order in which customers are served by the unique vehicles to which they are assigned. The solution is denoted by

$$T = \{\tau_0, \tau_1, \tau_2 \ldots, \tau_{n+m-1}, \tau_{n+m}\}, \tag{1}$$

where τ_i is the index of the customer, vehicle, or depot "node" in the ith position of the route. By convention, the node in position 0 is the depot depicted as $\tau_0 = 0$. Customers and vehicles may occupy any position from 1 to $n + m$. Nodes indexed 1 through n are customer nodes. Nodes indexed $n + 1$ through $n + m$ are vehicle nodes. Customers lying to the left of a vehicle node, τ_k, and to the right of a depot node or another vehicle node are assigned to vehicle τ_k.

The m vehicle nodes are modeled after the depot such that $d_k = d_0 = 0$, $s_k = s_0 = 0$, $e_k = e_0 = 0$, and $l_k = l_0$. Hence, the travel times from customers to the vehicle nodes are:

$$t_{i,j} = t_{i,0} \quad \text{for } i \in P^-, \quad j \in V \quad \text{and} \quad j > n, \tag{2a}$$

$$t_{i,j} = t_{0,j} \quad \text{for } j \in P^+, \quad i \in V \quad \text{and} \quad i > n, \tag{2b}$$

$$t_{i,j} = M \quad \text{for } i \in P^+, \quad j \in V \quad \text{and} \quad j > n, \tag{2c}$$

$$t_{i,j} = M \quad \text{for } j \in P^-, \quad i \in V \quad \text{and} \quad i > n. \tag{2d}$$

We assume that the number of vehicles required will not impose a limitation. Hence, m can be initialized at $n/2$ and N is the set of all modeled nodes representing all customers, vehicles and the depot, $|N| = n + m + 1$.

Since waiting is allowed, define the arrival time, A_i, departure time, D_i and waiting time, W_i, at customer i as follows:

$$A_{\tau_i} = D_{\tau_{i-1}} + t_{\tau_{i-1},\tau_i} \quad \text{for } i \in N, \tag{3}$$

$$D_i = \max(A_i, e_i) \quad \text{for } i \in P \quad \text{and} \quad D_i = 0 \quad \text{for } i \in V, \tag{4}$$

$$W_i = \max(0, e_i - A_i) = D_i - A_i. \tag{5}$$

Note that $W_i > 0$ whenever a vehicle arrives at customer i prior to e_i

The early service start time window constraints will be treated as "soft" constraints allowing the vehicle to wait if it arrives before the earliest service start time. The late service start time window constraints must be strictly satisfied in any feasible solution. These constraints are represented:

$$A_i \leqslant l_i \quad \forall i \in N. \tag{6}$$

If i is a vehicle node, Eq. (6) determines if the route is completed during the normal working day.

All solutions considered during our search procedure are required to satisfy the precedence and coupling constraints, i.e., the same vehicle that picks up supplies will deliver them to their delivery site and the pickup site will be visited prior to the delivery site. Unlike the precedence and coupling constraints, we do allow the traversal of solutions that violate the customer time window constraints. This search capability is required by the fact that the feasibility region may be disjoint with regard to these constraints for any reasonable neighborhood (Van der Bruggen et al., 1993). The vehicle capacity constraints, which may also be violated during the search, are expressed:

$$\text{load}_{\tau_i} \leqslant C \quad \forall i \in m + n, \tag{7a}$$

where

$$\text{load}_{\tau_i} = \text{load}_{\tau_{i-1}} + d_i \quad \text{for } i \leqslant m + n \quad \text{and } \text{load}_{\tau_k} = 0 \text{ for } k \in V. \tag{7b}$$

A penalty structure for time window and load violations associated with each solution is provided below:

1. S_{TW} totals the amount of time windows violations for the solution and is computed by

$$S_{\mathrm{TW}} = \sum_{i=1}^{n+m} \max(0, A_i - l_i), \tag{8}$$

where the expression $A_i - l_i > 0$ indicates either that the vehicle arrived at customer i after the late service time or the vehicle returned to the depot late.

2. S_{ld} totals the amount of overload for all vehicles, $k = 1, \ldots, m$, used in a solution and is computed by

$$S_{\mathrm{ld}} = \sum_{\forall k} \max(0, \mathrm{load}_k - C), \tag{9}$$

S_{TW} and S_{ld} are computed for two purposes. First, the time windows and overload violations, appropriately weighted, will be added to the objective function for infeasible solutions. Additionally, the time windows and overload violations are used in a two-level open hashing structure to further discriminate between solutions (Horowitz et al., 1993; Carlton, 1995).

The total travel time is the sum of all inter-node distances in the solution. A solution's total travel time is given by $\sum_{i=0}^{m+m-1} t_{\tau_i,\tau_{i+1}}$. In our experiments, the penalty for time windows, $\mathrm{PEN}_{\mathrm{TW}}$, is set equal to 1 to reflect the actual total time that the solution violates the l_i. The penalty for capacity violations, $\mathrm{PEN}_{\mathrm{ld}}$, is set to 100. The objective function is thus defined as

$$Z_t(T) = \sum_{i=0}^{n+m-1} t_{\tau_i,\tau_{i+1}} + \mathrm{PEN}_{\mathrm{TW}} \times S_{\mathrm{TW}} + \mathrm{PEN}_{\mathrm{ld}} \times S_{\mathrm{ld}}. \tag{10}$$

Reactive tabu search (RTS) developed by Battiti and Tecchiolli (1994) is a robust search technique that enhances classical tabu search (Glover, 1989, 1990; Glover and Laguna, 1997) by allowing the algorithm to automatically adjust the search parameters based on the state and quality of the search. Thus, rather than using constant parameter values or selecting randomly from a set of possible parameter values, RTS enables the algorithm to choose strategies and parameter values at each iteration. Further, RTS enables the search process not only to detect the presence of chaotic attractor basins in the solution topology but also provides a method to escape from such basins and continue the search in previously unvisited arenas. We assume that the reader is familiar with the basic principles of classical tabu search and RTS.

The remainder of the paper is organized as follows: Section 2 gives a brief review of the relevant literature associated with the PDPTW. Section 3 presents our RTS-PDPTW algorithm and gives computational results for our algorithm when applied to a newly constructed set of benchmark problems and shows RTS-PDPTW to be effective and efficient. Section 4 gives conclusions and recommendations for future research.

2. Literature review

Psaraftis (1980, 1983) developed the first exact dynamic programming algorithms to solve single vehicle dial-a-ride problems (DARP). Those algorithms could solve only small problem instances involving 10 or fewer customers. Sexton and Bodin (1985a, b) minimized customer inconvenience in single vehicle problems by applying Benders' decomposition procedure to a mixed binary nonlinear formulation that solves the routing and scheduling components individually. Sexton and Choi (1986) used a two-phase routing and scheduling procedure for single vehicle problems to minimize a linear combination of total vehicle operating time and total customer penalty due to missing any of the time windows. Their approaches were only capable of solving problems with up to 18 customers.

Desrosiers et al. (1986) solved the single vehicle DARPTW using a forward dynamic programming approach that increases algorithm efficiency by eliminating states that are incompatible with vehicle capacity, precedence and time window constraints. Dumas et al. (1991) employed a column generation scheme with a constrained shortest path as a subproblem. Dumas et al. (1991) approach worked well for single vehicle DARPTW problems with up to 55 customers in the presence of restrictive vehicle capacity constraints.

Since the problem sizes that exact methods can solve are relatively small, researchers have developed heuristic algorithms in an attempt to solve larger sized problems encountered in practice. Such heuristic algorithms do not seek global optimal solutions, but rather seek to quickly provide near-optimal solutions.

Van der Bruggen et al. (1993) employ a two-phase *local search* algorithm to determine near-optimal solutions for the single vehicle PDPTW. The construction phase starts with a time window infeasible solution and reduces infeasibility at each iteration by applying an objective function that penalizes the violation of time windows. Precedence and capacity constraints are never violated. The construction phase returns a feasible solution and the improvement phase continues to minimize the objective of route duration. Both phases use a variable-depth exchange procedure based on a lexicographic neighborhood search strategy and an embedded arc-exchange algorithm using seven variants of arc-exchanges. Van der Bruggen et al. (1993) also developed an alternative algorithm based on a penalized simulated annealing algorithm that provided the power to escape local optima by accepting inferior solutions and traversing infeasible regions in the state space to find other local optima.

With the exception of Dumas et al. (1991), no mention is made of the development of methods to solve the multiple vehicle version of the PDPTW. Although Dumas et al. (1991) describe an extension of their approach for solving multivehicle problems was described, it was not implemented.

3. A reactive tabu search approach to the PDPTW

In the construction of RTS-PDPTW, we have made extensive use of the findings of Carlton (1995) and Carlton and Barnes (1996).

3.1. The feasible initial solution

A simple predecessor–successor (ps) pair insertion algorithm was used to create the initial solution. This routine first attempts to insert a ps-pair onto the first vehicle's route. If more than one feasible ps-pair placement exists, the routine finds the feasible insert location for the ps-pair that adds the least amount to the partial solution's travel time. The routine then attempts to feasibly place subsequent ps-pairs on that same route. If there is no feasible placement, a new vehicle route is created. The routine continues until all $n/2$ ps-pairs are scheduled. This procedure provided a feasible initial solution for all the test bed problems.

3.2. Move neighborhoods

The following three move neighborhoods are hierarchically employed after the initial solution is constructed.

3.2.1. Single paired insertion (SPI)

The first move neighborhood attempts to move a ps-pair from its current vehicle route to another vehicle route in the solution (Fig. 1). SPI performs the following process for all $n/2$ predecessor nodes in the current solution. Once the predecessor node is identified, the method attempts to place it on all other vehicle routes. An admissible placement is one where the predecessor node satisfies both time window and capacity constraints for incorporation on the alternate route. If an admissible placement is found for the predecessor node, the associated successor node is inserted after the predecessor node in all later locations remaining on the route. Inserting the successor node subsequent to the predecessor node may violate load and/or time window constraints. Penalties for violating these constraints is included in computing the change to the objective function, or *move value*, for each of the subsequent location.

Of the three move neighborhoods, SPI has the greatest potential for improvement in the objective function and is the only move that can reduce the number of routes, i.e., the number of vehicles required.

While the selected SPI move can yield an infeasible solution, eliminating routes is permitted only if the new solution is feasible. SPI is an $O(n^3)$ search neighborhood and is the most expensive search mechanism used by the algorithm.

3.2.2. Swapping pairs between routes (SBR)

The second move neighborhood swaps (exchanges) ps-pairs between two different routes (Fig. 2). During the search process, the SPI move strategy can reach a barrier where no admissible SPI moves exist. This situation is extremely common when time windows are tight. SBR moves overcome this barrier and alter the search trajectory to find new areas of the feasible solution space.

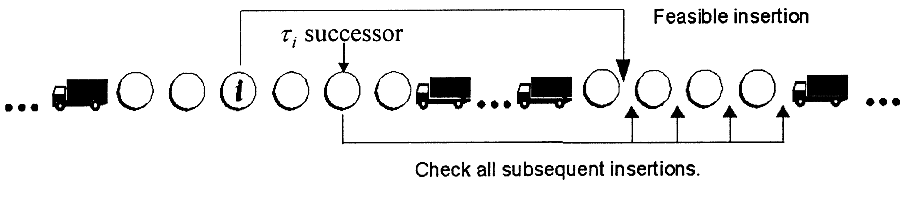

Fig. 1. The SPI neighborhood search.

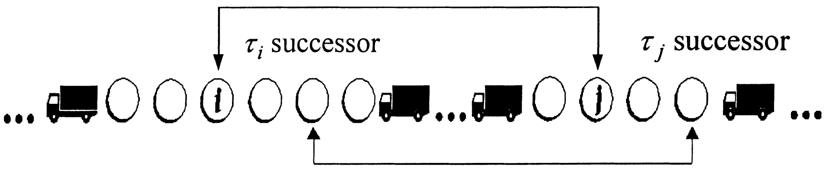

Fig. 2. The Swap pairs neighborhood search.

In using SBR moves, no attempt is made to find the best place to insert the successors after the predecessors are swapped. This type of polishing is entrusted to the WRI neighborhood discussed next.

The SBR move often selects a new solution that is infeasible. This is allowable because the purpose of this search is to alter the makeup of the routes and the search is not constrained to feasible solutions. Time window infeasible and/or capacity infeasible neighbor solutions are critical to the search. If the new solution is infeasible, the WRI neighborhood can often make modifications within a route to lessen or remove constraint violations.

3.2.3. Within route insertion (WRI)

The third move neighborhood is used to polish routes by moving individual predecessor or successor nodes forward or backward in their respective routes (Fig. 3). The WRI move neighborhood will attempt to reorder the customers to lessen or remove existing infeasibilities or to improve the objective function for feasible solutions.

This search neighborhood is especially helpful when large time windows are prevalent. Numerous feasible solutions are available when large time windows are present. Altering the customer arrangement within the routes will explore other possible orderings to determine the best possible order for the routes. The WRI search is further restricted by the precedence relations for all ps-pairs and the inclusion of strong time window infeasibility considerations, i.e., when one customer can never be feasibly serviced before (or after) another customer on the same vehicle route (Barnes and Carlton, 1995).

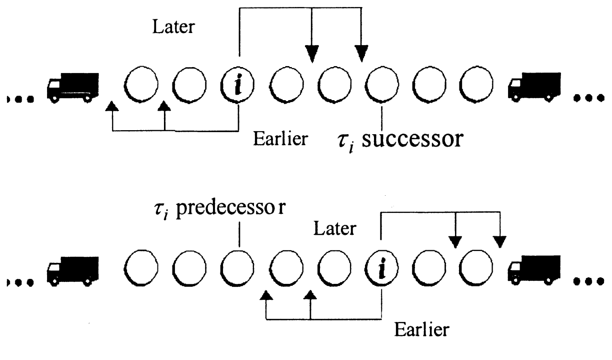

Fig. 3. The WRI neighborhood search.

3.3. An overview of RTS-PDPTW

As in Carlton and Barnes (1996), we blended:

1. a fine gauge screening to allow differentiation at the individual solution level and
2. a broader gauge screening by means of tabu attributes to allow differentiation between sets of solutions that do not share a selected "feature".

The fine gauge differentiation was achieved by means of the two-level hashing scheme implemented by Carlton (1995) which was a straightforward extension of the scheme described in Carlton and Barnes (1996). The broad gauge differentiation was accomplished using the tabu criteria and data structures detailed below.

RTS differs from classical tabu search in that RTS monitors the previously visited solutions and dynamically adjusts the algorithm's search parameters based on its assessment of the quality of that exploration. In this context, high quality search paths seldom revisit solutions previously encountered. If a solution is revisited within a specified number of iterations, RTS increases the short-term memory length by a specified multiplicative factor to discourage short-term cycles that would lead to further repetitions. If no solutions are repeated during a specified period, the short-term memory length is decreased by a specified multiplicative factor to encourage intensification of the search in the current locale. If too many solutions are repeated too often, this is an indication that increasing the short-term memory length is insufficient; something other than a simple cycling behavior is present. Battiti and Tecchiolli (1994) assume that such behavior is caused by a chaotic attractor basin and suggest that an "escape" procedure be invoked to move to a previously unvisited portion of the solution space. The escape procedure we employed is described below.

3.3.1. The tabu attributes, tabu length and data structures

Short-term memory functions are used to determine whether a solution with a characteristic attribute has been visited before. If the algorithm discerns that a candidate solution possesses attributes of a recently visited solution within a specified number of iterations, the "tabu_length", the move is disallowed and the next candidate move is entertained. The selection of the tabu attribute, the associated data structure and the tabu length are vital design features which contribute to the efficiency and success of the search. Whenever a move yields an objective function value lower than previously discovered, the aspiration criteria is invoked and the tabu status of such a move is overridden.

The algorithm uses an $(N + 1)$ by $(N + 1)$ array, **L**, to record and enforce the tabu status. The "row" index identifies the customer, τ_i. The "column" index depicts the solution position. The list elements store the iteration number *after* which the current τ_i can return to position i in subsequent iterations. Customer τ_i can be moved directly or τ_i might move indirectly due to direct moves of other customers. Failure to account for both possibilities may cause "indirect cycling" among two or more customers. Thus, when the search determines τ_i is to move to position j, the value iter + tabu_length, where iter is the current iteration count, is stored in two locations. Setting $\mathbf{L}\ [\tau_i, j] = \mathbf{L}\ [\tau_i, i] =$ iter + tabu_length prohibits subsequent moves of node τ_i to position j and position i, respectively, for tabu_length iterations. A modification is made for the WRI search neighborhood when the move is to an adjacent later location. Since a return move is based on node index τ_{i+1}, $\mathbf{L}\ [\tau_{i+1}, i + 1]$ is set equal to the value iter + tabu_length.

With these settings, if the current iteration number is less than or equal to $\mathbf{L}[\tau_i, j]$, then a proposed move of τ_i to position j is tabu unless it leads to a globally superior objective function value.

The SPI and SBR neighborhoods move ps-pairs between routes. It is only necessary to update the $\mathbf{L}\ [i, j]$ array for the predecessor nodes in the ps-pairs. Possible insertion positions of the successor node using SPI are based solely on finding a feasible insertion point for the predecessor. Similarly, SBR finds predecessor nodes on differing routes and evaluates the possibility of swapping the pairs between the routes. Setting $\mathbf{L}[\tau_i, j] = \mathbf{L}[\tau_i, i] = \mathbf{L}[\tau_j, i] = \mathbf{L}[\tau_j, j] = \text{iter} + \text{tabu\_length}$, where i and j are the positions of the predecessor nodes, imposes the appropriate tabu restrictions for the SBR neighborhood.

Based on experience with the vehicle routing problem with time windows, the initial tabu length was set to

$$\text{tabu\_length} = \max(30, \text{number of ps-pairs}). \tag{11}$$

3.3.2. *The hierarchy of neighborhood selection*

As stated above, the RTS-PDPTW algorithm first generates a feasible starting solution. Next, all SPI moves are examined and the most favorable non-tabu move is selected. A hierarchical multineighborhood search strategy, based on average time window length (atwl), is then used to direct the search. The atwl

$$\text{atwl} = \frac{\sum_{i=1}^{n} l_i - e_i}{n} \tag{12}$$

is directly related to how "tight" the time window constraints are, i.e., the number of time window feasible solutions available.

When atwl is greater than 25% of the average route duration length, numerous feasible solutions exist and we have observed that more adjustments within routes are required to polish the search. In this case, after performing an SPI move, the algorithm is directed to perform $n/10$ successive WRI moves.

If atwl is less than 25% of the average route duration length, more strong time windows infeasibilities will exist which will limit the number of candidate moves for within route insertions. In this case, after performing one SPI move, perform $n/25$ successive WRI moves.

The tighter the time windows, the more likely there will be no admissible moves. We address this eventuality with the following hierarchy of neighborhood selection. If WRI has no admissible moves, the algorithm switches to SPI moves. If SPI has no admissible moves, the algorithm switches to SBR moves. If SBR has no admissible moves, the algorithm simply "escapes" to the best move available.

3.3.3. *Detecting and escaping from a chaotic attractor basin*

Pure local search methods become trapped by local minima. This type of barrier is easily defeated with simple tabu search short-term memory structures. Simple cycles, i.e., an endless repetition of a finite sequence of solutions can also be overcome by a sufficiently large tabu_ length. Unfortunately, a search can be confined to a limited portion of the solution space characterized as a chaotic attractor basin (Battiti, 1995). Such confinements exhibit complex trajectories with no clear periodicity.

If 10 unique solutions are visited twice, the RTS-PDPTW algorithm concludes that the search is trapped in a chaotic attractor basin and the algorithm escapes from the basin by performing successive SBR moves to drastically alter the makeup of the solution and move the search into a different region of the solution space. Empirical studies indicated that the number of SBR moves had to be based on the size of the problem and the number of routes in the solution. Too few SBR moves did not sufficiently alter the makeup of the route to escape the basin; too many SBR moves generates too much infeasibility. The number of successive swap pairs iterations performed is

$$\min(\text{number of vehicles used}, n/10). \tag{13}$$

If the search is confined in another attractor basin, the algorithm reinitializes the search and restarts from the best solution found to that point.

The RTS-PDPTW algorithm also uses the above restart mechanism when few, if any, repeat solutions have been identified during the first half of the search. Empirical results indicate that, in general, the best solutions are often found early in the search process. The results also reveal that if the optimal solution was missed by the initial search trajectory, soon after restarting the search at the best solution, the optimal solution was found.

3.4. Computational results

This section presents the RTS-PDPTW computational results for the new benchmark problems when compared to the optimal VRPTW approaches of Desrochers et al. (1992) and Kohl (1995) and to the VRPTW reactive tabu search approach of Carlton (1995).

3.4.1. The benchmark data sets

The Solomon (1987) problems have long existed as a benchmark data set for the vehicle routing problem with time windows (VRPTW). Unfortunately, there are no clearly defined benchmark problems available to test algorithms for PDPTW. For this reason, the Solomon problems, with appropriate modifications, were used as the test bed for our method. The data sets are available by *anonymous* ftp at www.me.utexas.edu in directory/pub/barnes.

Carlton's (1995) RTS heuristic was run on Solomon's VRPTW benchmark problems to generate the optimal solution schedules used for the PDPTW problem instances. The exact VRPTW solutions of Desrosiers et al. (1986) and Kohl (1995) were used to validate the results from Carlton's RTS heuristic. Given the optimal solution schedules, customers were randomly paired within each solution while assuring that feasibility was maintained. If an odd number of customers was present on a vehicle route, the customer without a successor was paired with a dummy node modeled after itself. The service time for the predecessor (pickup) node is set to zero and the dummy successor (delivery) node gains the service time.

3.4.2. Selected parameter settings for RTS-PDPTW

The RTS-PDPTW algorithm is coded in C and the runs were performed on an IBM RISC 6000 workstation. The code was compiled with the standard C compiler using the -O3 optimization flag. The algorithm was run using the following parameter settings. These settings were chosen based on the values used so successfully by Barnes and Carlton (1995), Carlton (1995) and Carlton and Barnes (1996) on the related TSPTW and VRPTW problems:

1. $PEN_{TW} = 1.0$ is the factor used to weight the total amount of infeasibility with respect to time windows.
2. $PEN_{ld} = 100.0$ is the factor used to weight the total amount of infeasibility with respect to load capacity violations.
3. tabu_length = max(30, number of ps-pairs).
4. The tabu_length increase factor, set to 1.2, is the factor by which the tabu_length is increased if a solution is revisited within the designated cycle length.
5. The tabu_length decrease factor, set to 0.9, is the factor by which the tabu_length is decreased, when no solution has been repeated for a stated period.
6. Cycle length = 50. If a solution is revisited within 50 iterations, the tabu_length is increased by the multiplicative factor of 1.2.

The exact algorithm by Desrochers et al. (1992) was coded in FORTRAN and executed on a SUN SPARC 1 workstation. Kohl used a HP 9000-735 computer and coded the algorithm in PASCAL. The VRPTW RTS (Carlton, 1995) is coded in C and executed on an IBM RISC 6000 workstation.

Despite the differences in computing platforms and codes, the tables in this section demonstrate the significant decrease in computation effort the RTS-PDPTW algorithm achieves over the others. This decrease occurs despite the added data structure required for the modified PDPTW data sets and the additional information that must be processed resulting from this added structure.

3.4.3. 25-customer problems

Table 1 records the results for the clustered (c1), radially dispersed (r1) and mixed (rc1) 25-customer problem instances from Solomon. Column one identifies the problem instance where, for example, nc101 is the modification to Solomon clustered problem c101. Columns two and three display the minimum travel time and the number of vehicles required to achieve the travel time. Columns four and five reflect the number of iterations and the seconds of computation time required to locate the RTS solution, respectively. Column six shows the deviation of the RTS travel time from the optimum expressed as percentage. Columns seven through nine show the optimal results obtained by Carlton's (1995) VRPTW code. 750 iterations were used for the runs except where noted.

On average, 48 iterations were required to find the best solution in a time of 0.68 s for the RTS-PDPTW algorithm. This is an average savings of 89.7% for number of iterations and 82.7% in time to find the best solution when compared to the optimal VRPTW solutions of Carlton (1995). The overall difference in solution quality from the optima was 0.0%. All 29 optimal solutions were found.

3.4.4. 50-customer problems

Table 2 records the overall results for problems r1, c1 and rc1 for the 50-customer problem instances for the RTS-PDPTW algorithm. Desrochers et al. (1992) claimed they were able to solve 14 of the 29 problems to optimality. Carlton (1995) used their results for comparison with his RTS heuristic solutions. Kohl (1995) claimed that he was able to solve 27 of the 29 problems to optimality. However, the detailed schedules for the optimal solutions are not available. For com-

Table 1
RTS-PDPTW results, 25-customers

Problem instance	$Z_t(T)$	Vehs used	Iter to best	Time to best (s)	% dev	Optim $Z_t(T)$	Iter to best	Comp time (s)
nc101	2441.30	3	0	0.01	0	2441.30	2	0.07
nc102	2440.30	3	4	0.12	0	2440.30	82	0.67
nc103	2440.30	3	192	4.38	0	2440.30	671	5.93
nc104	2436.90	3	21	0.69	0	2436.90	1038	9.92
nc105	2441.30	3	0	0.01	0	2441.30	3	0.07
nc106	2441.30	3	0	0.01	0	2441.30	2	0.06
nc107	2441.30	3	0	0.02	0	2441.30	3	0.07
nc108	2441.30	3	0	0.06	0	2441.30	5	0.09
nc109	2441.30	3	0	0.10	0	2441.30	6	0.10
nr101	867.10	8	5	0.16	0	867.10	54	0.40
nr102	797.10	7	374	4.43	0	797.10	1039	8.23
nr103	704.60	5	21	0.37	0	704.60	963	8.20
nr104	666.90	4	10	0.33	0	666.90	195	1.81
nr105	780.50	6	23	0.36	0	780.50	56	0.43
nr106	715.40	5	8	0.14	0	715.40	962	7.51
nr107	674.30	4	0	0.03	0	674.30	194	1.69
nr108	647.30	4	56	1.44	0	647.30	67	0.66
nr109	691.30	5	236	2.28	0	691.30	722	5.88
nr110	694.10	5	21	0.31	0	694.10	919	8.27
nr111	678.80	4	18	0.28	0	678.80	181	1.59
nr112	643	4	110	2.86	0	643	63	0.73
nrc101	711.10	4	252	0.43	0	711.10	1077	7.26
nrc102	601.80	3	0	0.02	0	601.80	1994	16.53
nrc103	582.80	3	4	0.10	0	582.80	890	7.38
nrc104	556.60	3	0	0.05	0	556.60	659	5.55
nrc105	661.30	4	7	0.10	0	661.30	180	1.31
nrc106	595.50	3	17	0.14	0	595.50	39	0.31
nrc107	548.30	3	10	0.10	0	548.30	1412	12.37
nrc108	544.50	3	16	0.30	0	544.50	63	0.67
Avg			48	0.68	0.0		466	3.92

parison purposes, where possible, the VRPTW code of Carlton (1995) was used to identify the detailed schedules for the optimal solutions. Those schedules were used to create the PDPTW benchmark problem sets. Table 3 displays the results for the 50-customer problems and follows the same format used for the 25-customer problems except where noted. 750 iterations were used for the runs.

On average, 104 iterations were required to find the best solution in a time of 7.31 s for the RTS-PDPTW algorithm. The overall difference in solution quality from the optima was 0.02%. 14 of 15 optimal solutions were found.

3.4.5. 100-customers

The table indicates the nine problems investigated. Desrochers et al. (1992) claim to have solved seven problems to optimality. Carlton (1995) used their results for comparison with his RTS

Table 2
RTS-PDPTW results, 50-customers

Problem instance	$Z_t(T)$	Vehs used(#)	Iter to best	Time to best (s)	% dev	Results from Kohl (1995)		
						$Z_t(T)$	Iter to best	Comp time (s)
nc101	4862.40	5	0	0.04	0	4862.40	24	2.07
nc102	4861.40	5	11	1.02	0	4861.40	38	14.89
nc103	4861.40	5	7	1.61	0	4861.40	34	26.45
nc104	4855.60	5	50	11.87	0	4855.60	53	153.56
nc105	4862.40	5	0	0.07	0	4862.40	17	1.50
nc106	4862.40	5	0	0.06	0	4862.40	19	1.34
nc107	4862.40	5	0	0.10	0	4862.40	26	5.43
nc108	4862.40	5	0	0.23	0	4862.40	23	3.53
nc109	4862.40	5	8	2.71	0	4862.40	18	3.29
nr102	1409	11	63	25.04	0	1409	15	3.50
nr106	1296.30	8	93	3.37	0.26	1293	30	14.90
nr107	1211.10	7	138	10.86	0	1211.10	255	324.50
nr110	1197	7	392	31.79	0	1197	38	27.13
nrc105	1355.30	8	538	15.97	0	1355.30	90	70.23
nrc108	1098.10	6	62	4.86	0	1098.10	82	598.90
Avg			104	7.31	0.02		50	83.41

Table 3
RTS-PDPTW results, 100-customers

Problem instance	$Z_t(T)$	Vehs used(#)	Iter to best	Time to best (s)	% dev	$Z_t(T)$ by Kohl	Iter to best	Comp time (s)
nc101	9827.30	10	0	0.25	0	9827.30	26	6.40
nc102	9827.30	10	0	1.91	0	9827.30	66	98.43
nc103	9829.90	10	25	16.50	0	9826.30	70	339.72
nc104	9834.70	10	300	328.87	0.12	9822.90	57	1150.80
nc105	9827.30	10	0	0.51	0	9827.30	24	6.68
nc106	9827.30	10	0	0.54	0	9827.30	33	12.37
nc107	9826.10	10	75	9.36	0	9826.10	35	12.40
nc108	9826.10	10	83	31.49	0	9826.10	39	26.77
nc109	9827.30	10	291	184.76	0	9827.30	35	32.95
Avg			86	63.80	0.001		42.78	187.39

heuristic solutions. Kohl (1995) reported to solve all twelve of the investigated problems to optimality and 14 of the 29 Solomon benchmark problems to optimality. However, the optimal schedules for these solutions were not provided except that Kohl (1995) indicated that c105 has the same optimal solution as five of the other clustered problems; c101, c102, c106, c107 and c108. Carlton's code was used to identify the detailed schedules for the optimal solutions for those six clustered problems. These schedules were used to create the PDPTW benchmark problems. Table 3 uses the same format as described previously for the 50-customer problems. 500 iterations were run for the algorithm. On average, 86 iterations were required to find the best solution in a time of 63.8 s for the RTS-PDPTW algorithm. The overall difference in solution quality from the optima was 0.0013%. 8 of 9 optimal solutions were found.

4. Conclusions

The results presented above mark the first application of RTS to the PDPTW. Indeed, this is the first fully implemented method to be effectively applied to a set of practical sized multiple vehicle instances of the PDPTW.

Limiting the search to precedence viable solutions makes practical the use of an expensive SPI neighborhood search scheme. The dominance of the precedence and coupling constraints is critical to developing appropriate search strategies and is a major factor in the marked efficiency exhibited by the RTS-PDPTW algorithm. The RTS-PDPTW algorithm is compared to the current best known optimal and heuristic approaches to the VRPTW. The RTS-PDPTW algorithm consistently returns solutions within one percent, on average, in a fraction of the computational effort required by the other algorithms.

Optimal or near-optimal solutions are obtained for the modified problem with relatively small computational effort despite the addition of increased information to transform the VRPTW data sets into PDPTW data sets.

References

Barnes, J.W., Carlton, W.B., 1995. Solving the vehicle routing problem with time windows using reactive tabu search. Presented at the Fall INFORMS Conference in New Orleans, Louisiana, October 31, 1995.

Battiti, R., Tecchiolli, G., 1994. The reactive tabu search. ORSA Journal of Computing 6, 126–140.

Battiti, R., 1995. Reactive search: toward self-tuning heuristics. Keynote talk at Applied Decision Technologies, 3–4 April 1995, Brunel, UK.

Carlton, W.B., 1995. A tabu search approach to the general vehicle routing problem. Ph.D. Dissertation, University of Texas at Austin.

Carlton, W.B., Barnes, J.W., 1996. Solving the traveling salesman problem with time windows using tabu search. IIE Transactions 28, 617–629.

Desrochers, M., Desrosiers, J., Solomon, M.M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40, 342–354.

Desrosiers, J., Dumas, Y., Soumis, F., 1986. A dynamic programming solution of the large-scale single vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences 16, 301–325.

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7–22.

Glover, F., 1989. Tabu search part I. ORSA Journal of Computing 1, 190–206.

Glover, F., 1990. Tabu search: a tutorial. Interfaces 20, 74–94.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Norwell, MA.

Horowitz, E., Sahni, S., Anderson-Freed, S., 1993. Fundamentals of Data Structures in C, Freeman, New York.

Kohl, N., 1995. Exact method for time constrained routing and scheduling problems. Ph.D. Dissertation, Department of Mathematics University of Copenhagen, Denmark.

Laporte, G., 1992. The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 59, 345–358.

Psaraftis, H., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science 14, 130–154.

Psaraftis, H., 1983. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. Transportation Science 17 (3), 351–361.

Sexton, T.R., Lawrence, D.B., 1985a. Optimizing single vehicle many-to-many operations with desired delivery times: I Scheduling. Transportation Science 19, 378–410.

Sexton, T.R., Lawrence, D.B., 1985b. Optimizing single vehicle many-to-many operations with desired delivery times: II Routing. Transportation Science 19, 411–435.

Sexton, T.R., Choi, Y.-Y., 1986. Pickup and delivery of partial loads with "soft" time windows. American Journal of Mathematical and Management Sciences 6, 369–398.

Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265.

Van der Bruggen, L.J.J., Lenstra, J.K., Schuur, P.C., 1993. Variable-depth search for the single vehicle pickup and delivery problem with time windows. Transportation Science 27, 298–311.