Vehicle Routing: Methods and Studies
B.L. Golden and A.A. Assad (Editors)

Vehicle Routing with Time Windows: Optimization and Approximation

M. Desrochers
Centre for Mathematics and Computer Science, Amsterdam
Ecole des Hautes Etudes Commerciales, Montréal

J.K. Lenstra, M.W.P. Savelsbergh
Centre for Mathematics and Computer Science, Amsterdam
Erasmus University, Rotterdam

F. Soumis
Ecole Polytechnique, Montréal

This is a survey of solution methods for routing problems with time window constraints. Among the problems considered are the traveling salesman problem, the vehicle routing problem, the pickup and delivery problem, and the dial-a-ride problem. We present optimization algorithms that use branch and bound, dynamic programming and set partitioning, and approximation algorithms based on construction, iterative improvement and incomplete optimization.

1. Introduction

Over the past ten years, operations researchers interested in vehicle routing and scheduling have emphasized the development of algorithms for real-life problems. The size of the problems solved has increased and practical side constraints are no longer ignored.

Most of the existing algorithms have been designed to solve pure routing problems and hence only deal with spatial aspects. They are not capable of handling all kinds of constraints that frequently occur in practice. One such constraint is the specification of time windows at customers, i.e., time intervals during which they must be served. These lead to mixed routing and scheduling problems and ask for algorithms that also take temporal aspects into account.

In this survey, we consider two types of problems. One is the *vehicle routing problem with time windows* (VRPTW), which is defined as follows. A number of vehicles is located at a single depot and must serve a number of geographically dispersed customers. Each vehicle has a given capacity. Each customer has a given demand and must be served within a specified time window. The objective is to minimize the total cost of travel.

The special case in which the vehicle capacities are infinite is called the *multiple traveling salesman problem with time windows* (m-TSPTW). It arises in school bus

routing problems. The problem here is to determine routes that start at a single depot and cover a set of trips, each of which starts within a time window. Trips are considered as customers. There are no capacity constraints, since each trip satisfies those by definition and vehicles moving between trips are empty.

The second problem type is the *pickup and delivery problem with time windows* (PDPTW). Again, there is a single depot, a number of vehicles with given capacities, and a number of customers with given demands. Each customer must now be picked up at his origin during a specified time window, and delivered to his destination during another specified time window. The objective is to minimize total travel cost.

The special case in which all customer demands are equal is called the *dial-a-ride problem* (DARP). It arises in transportation systems for the handicapped and the elderly. In these situations, the temporal constraints imposed by the customers strongly restrict the total vehicle load at any point in time, and the capacity constraints are of secondary importance. The cost of a route is a combination of travel time and customer dissatisfaction.

We will denote the time window of an address i (whether it be a customer in the VRPTW or an origin or destination in the PDPTW) by $[e_i, l_i]$, the time of arrival at i by A_i, and the time of departure at i by D_i. It is assumed throughout this paper that the service time at i is included in the travel time t_{ij} from address i to address j. Since service must take place within the time windows, we require that $e_i \leqslant D_i \leqslant l_i$ for all i. If $A_i < e_i$, then a waiting time $W_i = e_i - A_i$ occurs before the opening of the window at i.

There are several ways to define the tightness of the time windows. One could say that the windows are tight when the underlying network with addresses as vertices contains no time-feasible cycles. This guarantees that all feasible routes are elementary paths. However, this condition is difficult to verify, and we do not get much information if it does not hold. The following two definitions may be more useful:

$$T_1 = \frac{\overline{(l_i - e_i)}}{\overline{t_{ij}}}, \quad T_2 = \frac{\overline{(l_i - e_i)}}{\max_i\{l_i\} - \min_i\{e_i\}}.$$

T_1 is the ratio between the average window width and the average travel time. If T_1 is at its minimum value 0, we have a pure scheduling problem. If T_1 is inbetween 0 and 2, we can expect that there are not many time-feasible cycles, and the temporal aspects are likely to dominate the spatial aspects. If T_1 is large, we have almost a pure routing problem. These are, of course, only rough indications.

T_2 is the ratio between the average window width and the time horizon. The value of T_2 is between 0 and 1, with 0 indicating a pure scheduling problem and 1 a problem with identical time windows.

In the following, VRP denotes the VRPTW without time windows. TSPTW is the m-TSPTW with a single salesman, and TSP is the TSPTW without time windows. Since the TSP is already NP-hard, one has to obtain solutions to the VRPTW and PDPTW by fast approximation or enumerative optimization. In

Section 2, we present mathematical programming formulations for these problems and some of their extensions. In Section 3, we survey optimization algorithms based on dynamic programming and set partitioning. In Section 4, we review various types of approximation algorithms.

There are more time-constrained routing problems and more solution approaches than we can cover in this survey. The interested reader is referred to a recent collection of papers on this topic [Golden and Assad 1986].

2. Formulation

In this section, the VRPTW and the PDPTW are defined and formulated as mathematical programs. We concentrate on the basic problems, with a single depot and a single vehicle type. We indicate generalizations involving multiple depots, multiple vehicle types, and constraints on the travel time of the vehicles.

2.1. *The vehicle routing problem with time windows*

Given is a graph $G = (V,A)$ with a set V of vertices and a set A of arcs. We have $V = \{0\} \cup N$, where 0 indicates the depot and $N = \{1,...,n\}$ is the set of customers, and $A = (\{0\} \times N) \cup I \cup (N \times \{0\})$, where $I \subset N \times N$ is the set of arcs connecting the customers, $\{0\} \times N$ contains the arcs from the depot to the customers, and $N \times \{0\}$ contains the arcs from the customers back to the depot. For each customer $i \in N$, there is a demand q_i and a time window $[e_i, l_i]$. For each arc $(i,j) \in A$, there is a cost c_{ij} and a travel time t_{ij}. Finally, the vehicle capacity is given by Q; we note that the number of vehicles is unbounded in the present formulation. We also note that an arc $(i,j) \in I$ may be eliminated by temporal constraints $(e_i + t_{ij} > l_j)$, by capacity constraints $(q_i + q_j > Q)$, or by other considerations. The objective is to minimize total travel cost.

The mathematical programming formulation has three types of variables: x_{ij} $((i,j) \in A)$, equal to 1 if arc (i,j) is used by a vehicle and 0 otherwise; D_i $(i \in N)$, specifying the departure time at customer i; and y_i $(i \in N)$, specifying the load of the vehicle arriving at i. The problem is now to minimize

$$\sum_{(i,j) \in A} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{j \in N} x_{ij} = 1 \qquad \text{for } i \in N, \tag{2}$$

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0 \qquad \text{for } i \in N, \tag{3}$$

$$x_{ij} = 1 \Rightarrow D_i + t_{ij} \leqslant D_j \qquad \text{for } (i,j) \in I, \tag{4}$$

$$e_i \leqslant D_i \leqslant l_i \qquad \text{for } i \in N, \tag{5}$$

$$x_{ij} = 1 \Rightarrow y_i + q_i \leqslant y_j \qquad \text{for } (i,j) \in I, \tag{6}$$

$$0 \leqslant y_i \leqslant Q \qquad \text{for } i \in N, \tag{7}$$

$$x_{ij} \in \{0,1\} \qquad \text{for } (i,j) \in A. \tag{8}$$

The objective function (1) represents total travel cost; it is possible to include the fixed charge of using a vehicle by adding it to all c_{0j}. Minimizing (1) subject to (2),

(3) and (8) is a minimum cost flow problem, which has an integral solution. Constraints (4) and (5) ensure feasibility of the schedule, and constraints (6) and (7) guarantee feasibility of the loads.

This VRPTW formulation is more compact than the VRP formulation due to Bodin and Golden [1981]. The latter formulation has $O(n^3)$ variables and an exponential number of subtour elimination constraints. The above formulation has $O(n^2)$ variables, while the subtours are eliminated by (4), as well as by (6). These constraints can be rewritten as follows, where M is a large constant:

$$D_i + t_{ij} - D_j \leqslant (1-x_{ij})M \qquad \text{for } (i,j) \in I, \tag{4a}$$

$$y_i + q_i - y_j \leqslant (1-x_{ij})M \qquad \text{for } (i,j) \in I. \tag{6a}$$

In their TSP formulation, Miller, Tucker and Zemlin [1960] propose the following subtour elimination constraints:

$$D_i - D_j + nx_{ij} \leqslant n - 1 \qquad \text{for } (i,j) \in I.$$

These appear as a special case of (4a) when all $t_{ij} = 1$ and $M = n$, and as a special case of (6a) when all $q_i = 1$ and $M = n$.

The above single-depot formulation is based on a single-commodity flow. There is no explicit flow conservation constraint for the depot, as this is implied by the flow conservation constraints (3) for the customers. Let us now consider the multi-depot case. The single depot 0 is replaced by a set M of depots. In the graph $G = (V,A)$, we now have $V = M \cup N$ and $A = (M \times N) \cup I \cup (N \times M)$, where N and I are as before. There are two variants. In case each vehicle must return to its home depot, we need a multi-commodity flow formulation with a separate commodity for each depot. Each variable x_{ij} is replaced by variables x_{ij}^k ($k \in M$), where $x_{ij}^k = 1$ if arc (i,j) is used by a vehicle from depot k, and 0 otherwise. In case vehicles do not have to return to their points of origin, all we have to do is to add a flow conservation constraint for each depot.

The case of multiple vehicle types is modeled with fictitious depots. For each type of vehicle at a given depot, we create a fictitious depot with a separate commodity to ensure that the number of vehicles of each type at each depot is balanced. The case that the vehicles have upper bounds on their total travel time is handled by the specification of a time window for the depot. The case that the vehicles have different periods of availability is obviously dealt with by the introduction of fictitious depots with time windows.

2.2. *The pickup and delivery problem with time windows*

As in the previous section, there is a set N of customers. In the current situation, however, each customer $i \in N$ requests the transportation from an origin i^+ to a destination i^-. We write $N^+ = \{i^+ \mid i \in N\}$ for the set of origins and $N^- = \{i^- \mid i \in N\}$ for the set of destinations. The graph $G = (V,A)$ is now defined as follows. The vertex set is given by $V = \{0\} \cup N^+ \cup N^-$, where 0 denotes the depot. The arc set is given by $A = (\{0\} \times N^+) \cup I \cup (N^- \times \{0\})$, where $I \subset (N^+ \cup N^-) \times (N^+ \cup N^-)$ is the set of arcs corresponding to feasible

trips between origins and destinations. For each customer $i \in N$, there are a demand q_i and two time windows $[e_{i^+}, l_{i^+}]$ and $[e_{i^-}, l_{i^-}]$. For each arc $(i,j) \in A$, there is a cost c_{ij} and a travel time t_{ij}. Finally, there is a set M of vehicles, each with capacity Q. The objective is to minimize total travel cost.

The mathematical programming formulation has the same three types of variables as in the case of the VRPTW: x_{ij}^k ($(i,j) \in A$, $k \in M$), equal to 1 if arc (i,j) is used by vehicle k and 0 otherwise; D_i ($i \in N^+ \cup N^-$), specifying the departure time at vertex i; and y_i ($i \in N^+ \cup N^-$), specifying the load of the vehicle arriving at i. We note that the flow variables have now a third index in order to ensure that the pickup at i^+ and delivery to i^- are done by the same vehicle. The problem is to minimize

$$\sum_{(i,j)\in A, k\in M} c_{ij} x_{ij}^k \tag{9}$$

subject to

$$\sum_{k\in M} \sum_{j\in V} x_{ij}^k = 1 \qquad \text{for } i \in N^+, \tag{10}$$

$$\sum_{j\in V} x_{ij}^k - \sum_{j\in V} x_{ji}^k = 0 \qquad \text{for } i \in N^+ \cup N^-,\ k \in M, \tag{11}$$

$$\sum_{j\in V} x_{i^+j}^k - \sum_{j\in V} x_{ji^-}^k = 0 \qquad \text{for } i \in N,\ k \in M, \tag{12}$$

$$D_{i^+} + t_{i^+i^-} \leqslant D_{i^-} \qquad \text{for } i \in N, \tag{13}$$

$$x_{ij}^k = 1 \Rightarrow D_i + t_{ij} \leqslant D_j \qquad \text{for } (i,j) \in I,\ k \in M, \tag{14}$$

$$e_i \leqslant D_i \leqslant l_i \qquad \text{for } i \in N^+ \cup N^-, \tag{15}$$

$$x_{ij}^k = 1 \Rightarrow y_i + q_i \leqslant y_j \qquad \text{for } (i,j) \in I,\ k \in M, \tag{16}$$

$$0 \leqslant y_i \leqslant Q \qquad \text{for } i \in N^+, \tag{17}$$

$$x_{ij}^k \in \{0,1\} \qquad \text{for } (i,j) \in A,\ k \in M. \tag{18}$$

Minimizing (9) subject to (10), (11) and (18) is a multi-commodity minimum cost flow problem of a more complex structure than in the case of the VRPTW. Constraints (12) ensure that each i^+ and i^- are visited by the same vehicle. Constraints (13) represent the precedence relation between pickup and delivery points. Constraints (14) and (15) ensure feasibility of the schedule, and constraints (16) and (17) guarantee feasibility of the loads; we note that capacity constraints are only specified for origins because a vehicle reaches its maximum load after a pickup. We also note that all model extensions presented for the VRPTW can be applied to the PDPTW.

3. Optimization

Optimization algorithms for routing problems with time windows employ the two standard principles of implicit enumeration: dynamic programming and branch and bound. Among the branch and bound methods, two approaches stand out. One is the set partitioning approach, which uses column generation to solve a continuous relaxation of the problem and branch and bound to obtain integrality. The other approach uses state space relaxation to compute lower bounds.

Dynamic programming is mainly applied to solve single-vehicle problems. Those problems arise in the context of column generation and state space relaxation, so that dynamic programming algorithms appear as subroutines in branch and bound methods.

In Section 3.1, we collect the applications of dynamic programming, including state space relaxation. In Section 3.2, we discuss the set partitioning approach. A variety of other branch and bound methods is reviewed below.

Baker [1983] presents a branch and bound method for the TSPTW, in which bounds are derived from longest path problems. He solves small problems with this method.

The most widely studied routing problem with time windows is the school bus routing problem [Orloff 1976], which is essentially an m-TSPTW. Two objectives are distinguished: minimizing fleet size and minimizing a weighted combination of fleet size and total travel time.

As to the first objective, Swersey and Ballard [1984] discretize the time windows and solve the linear programming relaxation of the resulting integer programming problem. For most instances, the solution is integral; otherwise, they are able to modify the solution so as to obtain integrality without increasing the fleet size. Desrosiers, Sauvé and Soumis [1985] study the Lagrangean relaxation which is obtained by relaxing constraints (2). As one visit to each customer is no longer required, the Lagrangean problem is a shortest path problem with time windows. Although the lower bound is often equal to the optimal fleet size, this dual method does not necessarily produce a feasible solution, in which case branch and bound has to be applied.

For the m-TSPTW with the second objective function, Desrosiers, Soumis, Desrochers and Sauvé [1985] study the network relaxation which is obtained by removing the scheduling constraints (4) and (5). If $e_i = l_i$ for all $i \in N$, then this relaxation produces an optimal solution in view of the definition of I. The quality of the bounds deteriorates with an increasing number of customers and an increasing width of the time windows. Two branching rules are proposed: branching on the flow variables and branching by splitting time windows. In the case of very tight time windows, Soumis, Desrosiers and Desrochers [1985] apply the first rule to solve problems with up to 150 customers; as the time windows become wider, the tree grows rapidly in size. The second branching rule can handle wider time windows, but it is concluded that the network relaxation is inferior to the set partitioning relaxation considered in Section 3.2.

Sörensen [1986] suggests the use of Lagrangean decomposition [Guignard 1984; Jörnsten, Nasberg and Smeds 1985] for the VRPTW. The two resulting subproblems are the shortest path problem with time windows and the generalized assignment problem. No computational results have been reported.

3.1. *Dynamic programming*

Dynamic programming is a traditional solution method for constrained shortest path problems. The constituents of a dynamic programming algorithm are states,

transitions between states, and recurrence equations that determine the value of the objective function at each state. Let us consider the standard shortest path problem on a graph $G = (V,A)$ with vertex set V, arc set A, a source $0 \in V$, and a travel time t_{ij} for each $(i,j) \in A$. Each vertex represents a state, each arc represents a transition between two states, and the value $d(j)$ associated with state j is the shortest path duration from the source 0 to vertex j. The recurrence equations to compute these values are:

$$d(0) = 0,$$

$$d(j) = \min_{(i,j)\in A}\{d(i)+t_{ij}\} \text{ for } j \in V \setminus \{0\}.$$

This algorithm has a running time that is polynomially bounded in the size of G.

Constraints are treated by expansion of the state space and modification of the recurrence equations. Such a dynamic programming approach can be useful for several NP-hard routing problems. However, the cardinality of the state space is usually exponential in the problem size. The practical use of dynamic programming in this context is restriced to state spaces of at most pseudopolynomial size and relatively small problem instances.

3.1.1. *Single-vehicle problems with time windows*

We will consider four problems in this section: the traveling salesman problem with time windows, the single-vehicle dial-a-ride problem, and two constrained shortest path problems.

The TSPTW can be viewed as the problem of finding a shortest path from an origin 0 to a destination $n+1$ that visits all vertices in the set N and respects the time window of each vertex. Christofides, Mingozzi and Toth [1981c] propose the following dynamic programming algorithm. There are states of the form (S,j) with $S \subset N$ and $j \in S$, and $d(S,j)$ denotes the shortest duration of a feasible path starting at 0, visiting all vertices in S, and finishing at j. The optimal solution value $d(N \cup \{n+1\},n+1)$ is determined by the following recurrence equations:

$$d(\{0\},0) = e_0,$$

$$d(S,j) = \min_{i\in S-\{j\},(i,j)\in A}\{d(S-\{j\},i)+t_{ij}\} \text{ for } j \in N \cup \{n+1\},$$

where we redefine $d(S,j) = e_j$ in case $d(S,j) < e_j$ and $d(S,j) = \infty$ in case $d(S,j) > l_j$.

Psaraftis [1983a] uses dynamic programming to solve the single-vehicle DARP. The states are of the form $(j,k_1,...,k_n)$, where j is the vertex presently visited and each k_i can assume three values that denote the status of customer i: not yet picked up, picked up but not yet delivered, and delivered. It is now straightforward to define the feasible transitions between states. The algorithm has $2n$ stages, each of which extends the paths constructed so far with one arc. The total time requirement is $O(n^2 3^n)$. Psaraftis estimates that this approach is able to solve problems with up to ten customers.

Desrosiers, Dumas and Soumis [1986b] give a similar $2n$-stage algorithm for the

capacitated single-vehicle PDPTW. They propose a number of state elimination rules to reduce the computational effort. In addition to Psaraftis' feasibility tests which eliminate states on the basis of information about customers picked up so far, they also have feasibility tests which use information about customers not yet delivered. The algorithm can solve real-life problems with up to 40 customers.

Two types of constrained shortest path problems have been considered: the shortest path problem with time windows (SPPTW) and the capacitated shortest path problem with pickups, deliveries and time windows (SPPPDTW). The main difference between these problems and the single-vehicle DARP is that the path is no longer required to visit all customers. For the SPPTW, which is defined by (1), (3)-(5) and (8), Desrosiers, Pelletier and Soumis [1984] propose a label correcting method. Desrochers and Soumis [1985a, 1985b] give two pseudopolynomial algorithms. One is a label setting method, the other a primal-dual method. Desrochers [1986] generalizes the latter algorithm to the case of multidimensional time windows. For the SPPPDTW ((9) and (11)-(18)), Dumas [1985] and Dumas and Desrosiers [1986] present an algorithm which is similar to the one for the capacitated single-vehicle PDPTW.

As we have mentioned before, dynamic programming algorithms are mostly used as subroutines in other solution methods. This is because the problems considered in this section occur as subproblems in multi-vehicle problems. The TSPTW and the single-vehicle DARP arise in the second phase of cluster-first route-second approaches, where the first phase allocates customers to vehicles and the second phase asks for single-vehicle routes. The SPPTW occurs as a subproblem in the set partitioning algorithm for the m-TSPTW due to Desrosiers, Soumis and Desrochers [1984], in the Lagrangean relaxation algorithm for the fleet size problem due to Desrosiers, Sauvé and Soumis [1985], and in the Lagrangean decomposition algorithm for the VRPTW due to Sörensen [1986]. The SPPPDTW is a subproblem in the set partitioning algorithm for the PDPTW due to Desrosiers, Dumas and Soumis [1987].

3.1.2. *State space relaxation*

For a number of problems, Christofides, Mingozzi and Toth [1981a, 1981b, 1981c] have developed branch and bound algorithms that obtain lower bounds by dynamic programming on a relaxed state space. They take a dynamic programming algorithm for the problem under consideration as a starting point and replace its state space by a smaller space in such a way that the recursion over the new state space requires only polynomial time and yields a lower bound on the optimal solution value of the original problem.

State space relaxation is based on a mapping g from the original state space to a space of smaller cardinality. If there is a transition from S_1 to S_2 in the original state space, then there must be a transition from $g(S_1)$ to $g(S_2)$ in the new state space. We illustrate this idea on the TSPTW [Christofides, Mingozzi and Toth 1981c].

With each vertex i, an arbitrary integer β_i is associated, with $\beta_0 = \beta_{n+1} = 0$.

The mapping is defined by $g(S,j) = (k,\beta,j)$, where $k = |S|$ and $\beta = \sum_{i \in S} \beta_i$. The new recurrence equations are:

$$d(0,\beta,0) = \begin{cases} 0 & \text{if } \beta = 0, \\ \infty & \text{if } \beta \neq 0, \end{cases}$$

$$d(k,\beta,j) = \min_{i \neq j,(i,j) \in A} \{d(k-1,\beta-\beta_j,i) + t_{ij}\} \text{ for } j \in N \cup \{n+1\},$$

where we redefine $d(k,\beta,j) = e_j$ in case $d(k,\beta,j) < e_j$ and $d(k,\beta,j) = \infty$ in case $d(k,\beta,j) > l_j$. The lower bound is now given by

$$\min_{j \in N,(j,n+1) \in A} \{d(n, \sum_{i \in N} \beta_i, j) + t_{j,n+1}\}.$$

This lower bound can be improved by the use of vertex penalties and state space modifications. Vertex penalties serve to decrease the travel times of arcs incident to undercovered vertices and to increase the travel times of arcs incident to overcovered vertices; these penalties are adjusted by subgradient optimization. Similarly, the weights β_i can be modified by subgradient optimization. The resulting branch and bound method is able to solve problems with up to 50 vertices.

Kolen, Rinnooy Kan and Trienekens [1987] extend this approach to the VRPTW. They use a two-level state space relaxation. At the first level, a lower bound on the costs of a time-constrained path from the depot to vertex j with load q is computed. This is done with an adaptation of the above method for the TSPTW. The states are of the form (t,q,j), where q is the load of a shortest path arriving at vertex j no later than time t. We have $0 \leq t \leq T$ where T is the scheduling horizon, $0 \leq q \leq Q$ where Q is the vehicle capacity, and $j \in N$. At the second level, a lower bound on the costs of m routes with total load $\Sigma_{i \in N} q_i$ and different destination vertices is computed. The states are now of the form (k,q,j), where q is the total load of the first k routes and j is the destination vertex of route k. Vertex penalties are used to improve the lower bounds. Problems with up to fifteen customers are solved.

3.2. *Set partitioning*

Vehicle routing problems and in particular the VRPTW and the PDPTW can be reformulated as set partitioning problems, with variables (columns) corresponding to feasible routes.

Let R be the set of feasible routes of the problem under consideration. For each route $r \in R$, we define γ_r as the sum of the costs of its arcs and δ_{ri} $(i \in N)$ as a binary constant, equal to 1 if route r visits customer i and 0 otherwise. If x_r $(r \in R)$ is equal to 1 if route r is used and 0 otherwise, the set partitioning problem is to minimize

$$\sum_{r \in R} \gamma_r x_r \tag{19}$$

subject to

$$\sum_{r \in R} \delta_{ri} x_r = 1 \qquad \text{for } i \in N, \tag{20}$$

$$x_r \in \{0,1\} \qquad \text{for } r \in R. \tag{21}$$

Although problems (9)-(18) and (19)-(21) are equivalent, their continuous relaxations are not. This is because the variables in the latter problem are restricted to feasible paths in which each customer is included or not. Any solution to the relaxed version of (19)-(21) is a feasible solution to the relaxation of (9)-(18), but not vice versa. We can therefore expect to obtain better lower bounds on the basis of the set partitioning formulation.

Because of the cardinality of R, the relaxed set partitioning problem cannot be solved directly and column generation is used. That is, a new column of minimum marginal cost is generated by solving an appropriate subproblem. If its marginal cost is negative, then it is added to the linear program, the problem is reoptimized and column generation is applied again; otherwise, the current solution to the linear program is optimal. Before discussing results for specific vehicle routing problems, we first describe some general aspects of this approach.

3.2.1. *The subproblem*

The objective function of the subproblem has coefficients that depend on the values of the dual variables π_i ($i \in N$) of the continuous relaxation of the set partitioning problem. The constraints define a path subject to side constraints but not necessarily visiting all customers. They include (3)-(8) for the VRPTW, (3)-(5) and (8) for the m-TSPTW, and (11)-(18) for the PDPTW.

As we have seen in Section 3.1.1, dynamic programming is a suitable method to solve these subproblems to optimality, because the state spaces are relatively small.

3.2.2. *The master problem*

The continuous relaxation of the set partitioning problem is solved by the simplex algorithm. This method produces the dual values π_i that are needed for column generation and enables easy reoptimization each time new columns are generated.

To obtain an integral solution to the master problem, we add cutting planes or we use branch and bound. Each time a new constraint is added, another round of column generation is applied in order to solve the modified master problem. We must restrict ourselves to types of constraints that are compatible with the column generation method. For any cutting plane, the method must be able to compute its coefficients in order to evaluate the marginal cost of new columns. For any branching rule, the method must be able to exclude the columns that have become infeasible by branching

In case the c_{ij} are integral, a compatible type of cut is the one that rounds the objective up to the next integer. In the particular case that we minimize fleet size, this cut has the same coefficient 1 in each column; if it has a dual value π, a new column is generated by minimizing the reduced cost

$$\sum_{(i,j)\in A}(c_{ij}-\pi_i)x_{ij}-\pi.$$

However, we cannot use Gomory cuts or other types of cuts whose coefficients are not known before the new column is generated.

As to branching, the usual rule to fix a fractional variable x_r to 0 or 1 is not compatible. We can fix $x_r = 1$ by simply deleting the customers on route r from the subproblem. But we cannot fix $x_r = 0$: there is no way to prevent route r from being generated again. Four types of compatible branching rules have been proposed: branching on the flow variables of route r; branching on the position of a customer in route r; branching by splitting time windows; and branching on the number of vehicles of a given type in problems with multiple vehicle types. These rules have been listed here in order of increasing effectiveness.

3.2.3. *Acceleration techniques*

There are various ways to improve the performance of the set partitioning approach.

First of all, the set partitioning problems that arise in the context of vehicle routing are highly degenerate. It is an obvious idea to improve the convergence of the simplex method by a perturbation strategy.

Secondly, the solution of the relaxed master problem can be accelerated by the simultaneous generation of columns. The solution of a subproblem by dynamic programming produces not only a column of minimum reduced cost, but also many other columns of negative reduced cost. Several of these can be added.

In the third place, the solution of most of the subproblems can be greatly sped up by the heuristic elimination of vertices, arcs, and states. The first columns are generated in subnetworks, which only consist of customers with large dual values and inexpensive arcs; in addition, less promising states are ruled out during the recursion. At later stages, the elimination rules are gradually relaxed, until at the final stage the full network and state space are used in order to prove optimality.

3.2.4. *The multi-salesman and vehicle routing problem with time windows*

Desrosiers, Soumis and Desrochers [1984] propose a set partitioning approach to the m-TSPTW. The column generation problem is the SPPTW, which was reviewed in Section 3.1.1. In their algorithm, two cuts are added to the master problem: one to round up the number of vehicles and one to round up the total costs. After that, branching on flow variables is applied. With this rule, it is time consuming to achieve optimality, even if the integrality gap is small. They solve problems with up to 151 customers; the solution time on a CDC Cyber 173 ranges from 100 to 1000 seconds, depending on the width of the time windows. A recent improvement of the algorithm is able to solve problems with 223 customers within 600 seconds. A branching rule based on time window splitting is under development.

Desrosiers, Dumas and Soumis [1986a] extend this algorithm to the case of multiple vehicle types. Several SPPTW's are now to be solved, one for each type of vehicle. Branching is first done on the number of vehicles of a given type; when

this number is integral for each type, the usual branching on flow variables is applied.

No set partitioning algorithm for the VRPTW has been proposed so far. However, Desrochers [1986] presents a dynamic programming algorithm for the shortest path problem with a variety of constraints. This method is suitable for solving the subproblems that occur in this context.

3.2.5. *The pickup and delivery problem with time windows*

Dumas [1985] develops a set partitioning approach for the PDPTW. He solves problems with 30 customers (60 vertices) within 100 seconds on a CDC Cyber 173. These problems have tight capacity constraints ($q_i \geqslant Q/3$) and loose time window constraints. Narrowing the time windows significantly decreases the cardinality of the state space and thereby the computation time.

The subproblem in this case is the SPPPDTW, which was reviewed in Section 3.1.1. The algorithm of Dumas [1985] first branches on the number of vehicles per type and then on flow variables. Desrosiers, Dumas and Soumis [1987] replace the latter branching rule by branching on the position of customers in routes and obtain improved results.

4. Approximation

In spite of the recent success of optimization algorithms for vehicle routing with time windows, it is unlikely that they will be able to solve large-scale problems. In many situations one has to settle for algorithms that run fast but may produce suboptimal solutions. In this section, we review three types of approximation algorithms. *Construction* methods try to build a feasible solution starting from the raw data. *Iterative improvement* methods start from a feasible solution and seek to improve it through a sequence of local modifications. *Incomplete optimization* methods use a combination of enumeration of the solution space and heuristic rules to truncate the search. These types of methods have been widely applied to unconstrained routing problems. Their extension to constrained problems has only recently become a subject of investigation. In presenting this work, we will concentrate on feasibility rather than optimality aspects.

As in Section 3.1, we split the depot (vertex 0) in an 'origin' (vertex 0) and a 'destination' (vertex $n+1$). In the sequel, when we refer to a route, we assume that it is given by $(0,1,...,i,...,n,n+1)$, where i is the ith customer visited by the vehicle. There are two quantities associated with a subpath $(h,...,k)$ that play a dominant role in the algorithms below. The *possible forward shift* S_{hk}^{+} is the largest increase in the departure time D_h at h which causes no violation of the time windows along the path $(h,...,k)$:

$$S_{hk}^{+} = \min_{h \leqslant j \leqslant k}\{l_j - (D_h + \textstyle\sum_{h \leqslant i < j} t_{i,i+1})\}.$$

The *possible backward shift* S_{hk}^{-} is the largest decrease in the departure time D_h at h which causes no waiting time along the path $(h,...,k)$:

$$S_{hk}^{-} = \min_{h \leqslant j \leqslant k}\{D_j - e_j\}.$$

These quantities express the flexibility we have when we want to push customers forward or backward in time. It is not hard to see that all values S_{jk}^{+} and S_{jk}^{-} for $j = h,...,k$ can be computed in $O(n)$ time [Savelsbergh 1985].

4.1. *Construction*

In the design of construction methods, there are two key questions:

(1) Selection criterion: *which* customer is selected next to be inserted into the current solution?

(2) Insertion criterion: *where* will it be inserted?

While such decisions may be made at the same time, several of the algorithms in this section employ different criteria for selection and insertion.

4.1.1. *The vehicle routing problem with time windows*

Solomon [1983] was one of the first who attempted to adapt the existing approximation algorithms for the VRP to the VRPTW. Much of the material in this section is based on his work.

Savings. The savings method of Clarke and Wright [1964] is probably the first and certainly the best known heuristic proposed to solve the VRP. It is a sequential procedure. Initially, each customer has its own route. At each iteration, an arc is selected so as to combine two routes into one, on the basis of some measure of cost savings and subject to vehicle capacity constraints. Note that in this case the selection criterion applies to arcs rather than customers and that the insertion question does not occur.

In order to adapt this procedure for the VRPTW, we must be able to test the time feasibility of an arc. While in pure routing problems the direction in which a route is traversed is usually immaterial, this is not the case anymore in the presence of time windows. Hence, we only consider arcs from the last customer on one route to the first customer on another.

If two routes are combined, the departure times on the first route do not change. As to the second route, one necessary condition for feasibility is that the departure time at the first customer is no more than his latest service time, but that is not all. The other departure times on the route could be pushed forward, and one of them could become infeasible. This is where the possible forward shift enters the picture. For any path $(1,...,n+1)$, a change in the departure time at 1 is feasible if and only if it is no more than $S_{1,n+1}^{+}$.

By selecting of a cost effective and time feasible arc, the modified heuristic could link two customers whose windows are far apart in time. This suggests a further modification which selects arcs on the basis of both spatial and temporal closeness of customers, e.g., by adding a waiting time penalty to the cost savings.

Nearest neighbor. Initially, a route consists of the depot only. At each iteration, an unvisited customer who is closest to the current end point of the route is selected and added to the route to become its new end point. The selection is restricted to those customers whose addition is feasible with respect to capacity and time win-

dow constraints. A new route is started any time the search fails, unless there are no more customers to schedule.

The measure of closeness should include spatial as well as temporal aspects. Solomon [1983] proposes the following:

$$\alpha_1 t_{ij} + \alpha_2(\max\{e_j, D_i + t_{ij}\} - D_i) + \alpha_3(l_j - (D_i + t_{ij})), \text{ with } \alpha_1 + \alpha_2 + \alpha_3 = 1.$$

This measures the travel time between customers i and j, the difference between their respective delivery times, and the urgency of a delivery to j.

Insertion. Insertion methods treat the selection and insertion decisions seperately. We distinguish sequential and parallel insertion rules. The former construct the routes one by one, whereas the latter build them up simultaneously. All methods considered here are of the sequential type.

The general scheme of an insertion method is simple. Let U be the set of unrouted customers. For each customer $u \in U$, we first determine the best feasible point i_u after which it could be inserted into the emerging route:

$$\iota(u, i_u) = \min_{0 \leqslant i \leqslant n}\{\iota(u, i)\} \text{ for } u \in U.$$

We next select the customer u^* to be inserted into the route:

$$\sigma(u^*, i_{u^*}) = \min_{u \in U}\{\sigma(u, i_u)\}.$$

The insertion criterion ι and the selection criterion σ are still to be specified; we refer to Solomon [1983] and Savelsbergh [1985] for a number of possible definitions which take both spatial and temporal aspects into account. When no more customers can be inserted, a new route is started, unless all customers have been routed.

Insertion of u between i and $i+1$ could change all departure times on the path $(i+1,\ldots,n+1)$. Again, the insertion is feasible if and only if the change in departure time at $i+1$ is no more than $S^+_{i+1,n+1}$.

Solomon [1983] concludes on the basis of extensive computational experiments that insertion methods outperform other types of construction methods.

4.1.2. *The pickup and delivery problem with time windows*

Jaw, Odoni, Psaraftis and Wilson [1986] consider a variant of the DARP. Their approach seems to be applicable to the proper DARP as well.

The customers that are to be picked up and delivered have the following types of service constraints. Each customer i specifies either a desired pickup time $\overline{D}_{i^+}$ or a desired delivery time $\overline{A}_{i^-}$, and a maximum travel time $\overline{T}_i$; in addition, there is a tolerance $\overline{U}$. If customer i has specified a desired pickup time, the actual pickup time D_{i^+} should fall within the time window $[\overline{D}_{i^+}, \overline{D}_{i^+} + \overline{U}]$; if he has specified a desired delivery time, the actual delivery time A_{i^-} should fall within the window $[\overline{A}_{i^-} - \overline{U}, \overline{A}_{i^-}]$. Moreover, his actual travel time should not exceed his maximum travel time: $A_{i^-} - D_{i^+} \leqslant \overline{T}_i$. Note that this information suffices to determine two time windows $[e_{i^+}, l_{i^+}]$ and $[e_{i^-}, l_{i^-}]$ for each customer i. Finally, waiting time is not allowed when the vehicle is carrying passengers.

The selection criterion is simple: customers are selected for insertion in order of increasing e_i^+. The insertion criterion is as follows: among all feasible points of insertion of the customer into the vehicle schedules, choose the cheapest; if no feasible point exists, introduce an additional vehicle.

For the identification of feasible insertions, the notion of an *active period* is introduced. This is a period of time a vehicle is active between two successive periods of slack time. For convenience, we drop the superscript indicating pickup or delivery. For each visit to an address i during an active period, we define the following variants of possible backward and forward shifts:

$$\Sigma_i^- = \min\{\min_{j \leqslant i}\{A_i - e_i\}, \Lambda\},$$
$$\Sigma_i^+ = \min_{j \leqslant i}\{l_i - A_i\},$$
$$S_i^- = \min_{j \geqslant i}\{A_i - e_i\},$$
$$S_i^+ = \min\{\min_{j \geqslant i}\{l_i - A_i\}, L\},$$

where Λ and L are the durations of the slack periods immediately preceding and following the active period in question. Σ_i^- (Σ_i^+) denotes the maximum amount of time by which every stop preceding but not including i can be advanced (delayed) without violating the time windows, and S_i^- (S_i^+) denotes the maximum amount of time by which every stop following but not including i can be advanced (delayed). These quantities thus indicate how much each segment of an active period can be displaced to accommodate an additional customer. Once it is established that some way of inserting the pickup and delivery of customer i satisfies the time window constraints, it must be ascertained that it satisfies the maximum travel time constraints.

The cost measure that is used to choose among feasible insertions is a weighted combination of customer dissatisfaction and resource usage.

Sexton and Bodin [1985a, 1985b] consider a variant of the single-vehicle DARP in which only deadlines for the deliveries are specified. Their solution approach is to apply Benders decomposition to a mixed 0-1 nonlinear programming formulation, which separates the routing and scheduling component.

4.2. *Iterative improvement*

Croes [1958] and Lin [1965] introduced the notion of k-exchanges to improve solutions to the TSP. Lin and Kernighan [1973] generalized the approach, and many authors reported on its application to related problems. In the context of vehicle routing, Christofides and Eilon [1969] and Russell [1977] adapted the approach to the basic VRP, and Psaraftis [1983b] used it for the DARP.

In this section, we will show how time windows can be handled in k-exchange procedures for the TSPTW [Savelsbergh 1985]. It is not hard, however, to extend the techniques to multi-vehicle problems with various types of side constraints [Savelsbergh 1987, 1988]. The issue is also addressed in another contribution to this volume [Solomon, Baker and Schaffer 1988].

A k-exchange is a substitution of k arcs of a route with k other arcs. In the TSP, the processing of a single k-exchange takes constant time for any fixed value of k. One only has to test whether the exchange is profitable and does not have to bother about feasibility. In the case of the TSPTW, the processing of a k-exchange may take $O(n)$ time. This is because a modification at one point may affect the departure times on the entire route, so that feasibility questions arise. It will be indicated below that, even in the presence of time windows, constant time suffices for the processing of a single k-exchange.

4.2.1. *The traveling salesman problem with time windows*

The number of possible k-exchanges in a given route is $O(n^k)$. The computational requirement of k-exchange procedures thus increases rapidly with k, and one usually only considers the cases $k = 2$ and $k = 3$. A 2-exchange replaces two arcs $(i,i+1)$ and $(j,j+1)$ by (i,j) and $(i+1,j+1)$, thereby reversing the path $(i+1,...,j)$. In a 3-exchange, three arcs are deleted and there are seven possibilities to construct a new route from the remaining segments. Or [1976] proposes to restrict attention to those 3-exchanges in which a string of one, two or three consecutive customers is relocated between two others. Note that no paths are reversed in this case and that there are only $O(n^2)$ exchanges of this kind. We will illustrate the combination of the k-exchange concept and time windows on those Or-exchanges in which one customer is relocated.

The basic idea of the approach is the use of a search strategy and of a number of global variables such that, for each considered exchange, testing its feasibility and updating the global variables require no more than constant time.

The search strategy is as follows. Suppose that customer i is relocated between j and $j+1$; this means that the arcs $(i-1,i)$, $(i,i+1)$ and $(j,j+1)$ are substituted by $(i-1,i+1)$, (j,i) and $(i,j+1)$. The cases of backward relocation $(j < i)$ and forward relocation $(j > i)$ are handled separately. In the former case, j is successively chosen to be equal to $i-1$, $i-2$, ..., 0; note that in each exchange the path $(j+1,...,i-1)$ of the previous exchange is expanded with the arc $(j,j+1)$. In the latter case, j assumes the values $i+1$, $i+2$, ..., n in that order; in each exchange the path $(i+1,...,j-1)$ of the previous exchange is expanded with the arc $(j-1,j)$.

The global variables we need are:

(1) the possible forward shift S^+, which is equal to $S^+_{j+1,i-1}$ as defined above;
(2) the possible backward shift S^-, which is equal to $S^-_{i+1,j-1}$ as defined above;
(3) the gain G made by going directly from $i-1$ to $i+1$:

$$G = A_{i+1} - (D_{i-1} + t_{i-1,i+1});$$

(4) the loss L incurred by going from j through i to $j+1$:

$$L = \max\{D_j + t_{ji}, e_i\} + t_{i,j+1} - A_{j+1};$$

(5) the waiting time W on the path $(j+1,...,i-1)$:

$$W = \sum_{j+1 \leq k \leq i-1} W_k.$$

During the backward search, an exchange is feasible if $D_k^{\text{new}} \leqslant l_k$ for $k = j+1,\ldots,i-1$, and potentially profitable if $D_{i+1}^{\text{new}} < D_{i+1}$. The superscript 'new' indicates the value if the exchange were carried out. Note that a decrease in the departure time at $i+1$ does not guarantee an earlier arrival at the depot, but 'potential profitability' is still a suitable criterion for accepting an exchange. In terms of global variables, feasibility and potential profitability are equivalent to

$$L < \min\{S^+, G + W\}.$$

The global variables are updated by

$$S^+ := W_{j+1} + \min\{l_{j+1} - D_{j+1}, S^+\};$$
$$W := W + W_{j+1}.$$

During the forward search, an exchange feasible if $D_i^{\text{new}} \leqslant l_i$ and potentially profitable if $D_{j+1}^{\text{new}} < D_{j+1}$. This equivalent to

$$L < \min\{S^-, G\}.$$

The only update is

$$S^- := \min\{D_j - e_j, S^-\}.$$

It follows that a single exchange of this type can be handled in constant time. The adaptation of other types of exchange procedures to time window constraints is conceptually similar but technically more complicated.

4.3. *Incomplete optimization*

Fast approximation algorithms can also be derived from the optimization algorithms presented in Section 3.2. The two principal ideas are the heuristic generation of columns and the partial exploration of the branch and bound tree.

Heuristic generation of columns is based on the third type of acceleration technique mentioned in Section 3.2.3. While solving the relaxed master problem, we eliminate vertices, arcs and states in a heuristic fashion. The elimination rules are not relaxed, so that an approximate solution to the linear program is obtained.

Partial exploration of the search tree can take place in several ways. One is to obtain an integral solution by depth-first search and then to explore the tree for the remaining available time. Another way is to use an invalid branching rule, i.e., to eliminate branches on heuristic grounds.

A combination of these ideas has been used to obtain feasible integral solutions within two percent from the optimum with highly reduced running times.

5. Conclusion

If one conclusion emerges from the preceding survey, it is that algorithm designers have turned their attention to the development of efficient methods that are capable of solving large-scale routing problems subject to real-life constraints.

A striking example is the set partitioning approach, which appears to be particularly efficient for strongly constrained problems. The continuous relaxation of

the set partitioning formulation can be solved by the use of a column generation scheme and provides for better bounds than the relaxation of other formulations. Dynamic programming turns out to be a powerful tool to generate columns. This family of algorithms is well designed to produce approximate solutions to problems of a realistic size. Optimization algorithms of this type are being used for school bus scheduling.

The construction and iterative improvement algorithms that have received so much attention in the context of the TSP and the VRP have now been adapted to incorporate time windows and other constraints, such as precedence constraints and mixed collections and deliveries. These types of algorithms are all familiar, but their modification to handle practical problems is nontrivial. Although the worst-case performance of these methods is very bad [Solomon 1986], they have been successfully incorporated in distribution management software [Anthonisse, Lenstra and Savelsbergh 1987].

Our survey is no more than an interim report. The developments in the area of constrained routing problems have just started. The practical need for effective routing algorithms will continue to stimulate further advances.

References

J.M. Anthonisse, J.K. Lenstra, M.W.P. Savelsbergh (1987). *Functional Description of CAR, an Interactive System for Computer Aided Routing*, Report OS-R8716, Centre for Mathematics and Computer Science, Amsterdam.

E.K. Baker (1983). An exact algorithm for the time-constrained traveling salesman problem. *Oper. Res. 31*, 65-73.

L. Bodin, B. Golden (1981). Classification in vehicle routing and scheduling. *Networks 11*, 97-108.

N. Christofides, S. Eilon (1969). An algorithm for the vehicle dispatching problem. *Oper. Res. Quart. 20*, 309-318.

N. Christofides, A. Mingozzi, P. Toth (1981a). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Math. Programming 20*, 255-282.

N. Christofides, A. Mingozzi, P. Toth (1981b). State space relaxation procedures for the computation of bounds to routing problems. *Networks 11*, 145-164.

N. Christofides, A. Mingozzi, P. Toth (1981c). *Exact Algorithms for the Travelling Salesman Problem with Time Constraints, Based on State-Space Relaxation*, Unpublished manuscript.

G. Clarke, J.W. Wright (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res. 12*, 568-581.

A. Croes (1958). A method for solving traveling salesman problems. *Oper. Res. 5*, 791-812.

M. Desrochers (1986). *An Algorithm for the Shortest Path Problem with Resource Constraints*, Publication 421A, Centre de recherche sur les transports, Université de Montréal.

M. Desrochers, F. Soumis (1985a). *A Generalized Permanent Labelling Algorithm for the Shortest Path Problem with Time Windows*, Publication 394A, Cen-

tre de recherche sur les transports, Université de Montréal.

M. DESROCHERS, F. SOUMIS (1985b). *A Reoptimization Algorithm for the Shortest Path Problem with Time Windows*, Publication 397A, Centre de recherche sur les transports, Université de Montréal.

J. DESROSIERS, Y. DUMAS, F. SOUMIS (1986a). *The Multiple Vehicle Many to Many Routing and Scheduling Problem with Time Windows*, Cahiers du GERAD G-84-13, Ecole des Hautes Etudes Commerciales de Montréal.

J. DESROSIERS, Y. DUMAS, F. SOUMIS (1986b). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *Amer. J. Math. Management Sci. 6*, 301-326.

J. DESROSIERS, Y. DUMAS, F. SOUMIS (1987). *Vehicle Routing Problem with Pick-up, Delivery and Time Windows*, Cahiers du GERAD, Ecole des Hautes Etudes Commerciales de Montréal.

J. DESROSIERS, P. PELLETIER, F. SOUMIS (1984). Plus court chemin avec contraintes d'horaires. *RAIRO Rech. Opér. 17(4)*, 357-377.

J. DESROSIERS, M. SAUVÉ, F. SOUMIS (1985). *Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows*, Publication 396, Centre de recherche sur les transports, Université de Montréal.

J. DESROSIERS, F. SOUMIS, M. DESROCHERS (1984). Routing with time windows by column generation. *Networks 14*, 545-565.

J. DESROSIERS, F. SOUMIS, M. DESROCHERS, M. SAUVÉ (1985). Routing and scheduling with time windows solved by network relaxation and branch-and-bound on time variables. J.-M. Rousseau (ed.). *Computer Scheduling of Public Transport 2*, North-Holland, Amsterdam, 451-469

Y. DUMAS (1985). *Confection d'itinéraires de véhicules en vue du transport de plusieurs origines à plusieurs destinations*, Publication 434, Centre de recherche sur les transports, Université de Montréal.

Y. DUMAS, J. DESROSIERS (1986). *A Shortest Path Problem for Vehicle Routing with Pick-up, Delivery and Time Windows*, Cahiers du GERAD G-86-09, Ecole des Hautes Etudes Commerciales de Montréal.

B.L. GOLDEN, A.A. ASSAD (1986). *Vehicle Routing with Time-Window Constraints: Algorithmic Solutions, Amer. J. Math. Management Sci. 6*, 251-428 (special issue).

M. GUIGNARD (1984). *Lagrangean Decomposition: an Improvement over Lagrangean and Surrogate Duals*, Report 62, Department of Statistics, University of Pennsylvania, Philadelphia.

J.-J. JAW, A.R. ODONI, H.N. PSARAFTIS, N.H.M. WILSON (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Res. Part B 20B*, 243-257.

K.O. JÖRNSTEN, M. NASBERG, P.A. SMEDS (1985). *Variable Splitting - a New Lagrangean Relaxation Approach to Some Mathematical Programming Models*, Report LITH-MAT-R-85-04, Department of Mathematics, Linköping Institute of Technology.

A.W.J. KOLEN, A.H.G. RINNOOY KAN, H.W.J.M. TRIENEKENS (1987). Vehicle routing with time windows. *Oper. Res.*, to appear.

S. LIN (1965). Computer solutions to the traveling salesman problem. *Bell System Tech. J. 44*, 2245-2269.

S. LIN, B.W. KERNIGHAN (1973). An effective heuristic algorithm for the traveling salesman problem. *Oper. Res. 21*, 498-516.

C. MILLER, A. TUCKER, R. ZEMLIN (1960). Integer programming formulation of travelling salesman problems. *J. Assoc. Comput. Mach. 7*, 326-329.

I. OR (1976). *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking*, Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

C.S. ORLOFF (1976). Route constrained fleet scheduling. *Transportation Sci. 10*, 149-168.

H. PSARAFTIS (1983a). An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Sci. 17*, 351-360.

H. PSARAFTIS (1983b). *k*-Interchange procedures for local search in a precedence-constrained routing problem. *European J. Oper. Res. 13*, 391-402.

R.A. RUSSELL (1977). An effective heuristic for the *m*-tour traveling salesman problem with some side constraints. *Oper. Res. 25*, 517-524.

M.W.P. SAVELSBERGH (1985). Local search for routing problems with time windows. *Ann. Oper. Res. 4*, 285-305.

M.W.P. SAVELSBERGH (1987). *Local Search for Constrained Routing Problems*, Report OS-R8711, Centre for Mathematics and Computer Science, Amsterdam.

M.W.P. SAVELSBERGH (1988). *Computer Aided Routing*, Ph.D. thesis, Centre for Mathematics and Computer Science, Amsterdam.

T.R. SEXTON, L.D. BODIN (1985a). Optimizing single vehicle many-to-many operations with desired delivery times: I. Scheduling. *Transportation Sci. 19*, 378-410.

T.R. SEXTON, L.D. BODIN (1985b). Optimizing single vehicle many-to-many operations with desired delivery times: II. Routing. *Transportation Sci. 19*, 411-435.

M.M. SOLOMON (1983). *Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms*, Report 83-02-01, Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia.

M.M. SOLOMON (1986). On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks 16*, 161-174.

M.M. SOLOMON, E.K. BAKER, J.R. SCHAFFER (1988). Vehicle routing and scheduling problems with time window constraints: implementations of solution improvement procedures. This volume.

B. SÖRENSEN (1986). *Interactive Distribution Planning*, Ph.D. thesis, Technical University of Denmark, Lyngby.

F. SOUMIS, J. DESROSIERS, M. DESROCHERS (1985). Optimal urban bus routing with scheduling flexibilities. *Lecture Notes in Control and Information Sciences 59*, Springer, Berlin, 155-165.

A.J. SWERSEY, W. BALLARD (1984). Scheduling school buses. *Management Sci. 30*, 844-853.