

# Optimization of Multi-Objective Routing Models for Multiple Level and Multi-Tiered Networks

RAVIENDER SINGH and ARIF NADEEM\*

Department of Mathematics Bareilly College Bareilly  
(M J P R University Bareilly) (INDIA)

(Acceptance Date 26th May, 2012)

## Abstract

In the present work, The authors describe two types of problem. One is the *vehicle routing problem with time windows* and the second problem is the *pickup and delivery problem with time windows*. for Optimization we use dynamic programming and branch and bound method. They will not be able to solve large-scale problems. So, 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.

*Key words:* Vehicle Routing, Time Windows, Raw Data.

## 1 Introduction

Over the past twenty 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

in the specification of time windows at customers, *i.e.*, time interval 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 the present<sup>1</sup> study we consider two types of models. One is the *vehicle routing problem with time windows* (VRPTW), which is defined as follows. A number of vehicles in located at a single depot and must serve a number of geographically dispersed customers. Each vehicle has a given capacity. Each

---

*Address of Communication:* Dr Arif Nadeem 94, North City, Pilibhit Road, Bareilly 243002 U.P.  
Email: [drarifnadeem@gmail.com](mailto:drarifnadeem@gmail.com), Phone: 09897608566

customer has a given demand and must be served within a specified time window. The object is to minimize the total cost of travel. The special case in which the vehicle capacities are infinite is called the *multiple traveling salesman problems 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 vehicle moving between trips are empty.

The second model 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 address  $i$  (whether it be a customer in the *vehicle routing problem with time windows* (VRPTW) or an origin or destination in the

*pickup and delivery problem with time windows* (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

$$e_i \leq D_i \leq l_i \text{ for all } i.$$

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<sup>5,6</sup> 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 relevant:

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

$T_1$  is the ratio between the average window width and the average travel time. If  $T_1$  is its minimum value 0, we have a pure scheduling problem. If  $T_1$  is in between 0 and 2, we can expect that there are not many time-feasible cycles, and the temporal aspect 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 indicators.

$T_2$  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, *vehicle routing problem* (VRP) denotes the *vehicle routing problem with the windows* (VRPTW) without time windows. *Traveling salesman problem with time windows* (TSPTW) is the *multiple traveling salesman problem with time windows* (m-TSPTW) with a single salesman, and *traveling salesmen problem* (TSP) is the *traveling salesman problem with time window* (TSPTW) without time windows. Since the *traveling salesman problem* (TSP) is already NP-hard, one has to obtain solution to the *vehicle routing problem with time window* (VRPTW) and *pickup and delivery problem with time windows* (PDPTW) by fast approximation or enumerative optimization. In section 2 we present mathematical programming formulations for these problem 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.

2. Formulation :

In this section, the *vehicle routing problem with time windows* (VRPTW) and the *pickup and delivery problem with time windows* (PDPTW) are defined and formulated as mathematical programs. We concentrate on the basic problem, 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 indicate the depot and  $N = \{1, \dots, n\}$  in the set of customers, and

$$A = (\{0\} \times N) \cup \bigcup (N \times \{0\}),$$

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 customers  $i \in N$ , 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 other considerations. The objective is to minimize total travel cost.

The mathematical programming formulation has three types of variables.

$$X_{ij} \ (c_{i,j} \in A), \text{ equal to } 1.$$

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

$$\sum_{(ij) \in A} C_{ij} X_{ij} \tag{1}$$

Subject to

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

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

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

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

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

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

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

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  $C_{ij}$ . 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 *vehicle routing problem with time windows* (VRPTW) formulation is more compact than the *vehicle routing problem* (VRP) formulation due to Bodin and Golden<sup>3</sup>.

The latter formulation has  $O(n^3)$  variable 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 constraint:

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

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

In their *traveling salesman problem* (TSP) formulation, Miller, Tucker and Zemlin<sup>29</sup> propose the following subtour elimination constraints:

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

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 on 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

$$x_{ij}^k \text{ ( } k \in M \text{), where } x_{ij}^k = 1.$$

If arc  $(i, j)$  is used by a vehicle by 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<sup>21-22</sup> 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 vehicle 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 the 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  $i \in N$  request 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

$$A = (\{0\} \times N^+) \cup I \cup (N^- \times \{0\}), \text{ where } I \subset ((N^+ \cup N^-) \times (N^+ \cup N^-))$$

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

$$[e_i^+, l_i^+] \text{ and } [e_i^-, l_i^-] \text{ for each arc } (i, j) \in A,$$

There is a cost  $C_{ij}$  and a travel time  $t_{ij}$ .

Finally, there is 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 *vehicle routing problem with time windows* (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

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

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

Subject to

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

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

$$\sum_{j \in V} x_{i^+j}^k - \sum_{j \in V} x_{j^-i^-}^k = 0 \text{ for } i \in N, k \in M, \quad (12)$$

$$D_{i^+} + t_{i^+i^-} \leq D_{i^-} \text{ for } i \in N, \quad (13)$$

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

$$e_i \leq D_i \leq l_i \text{ for } i \in N^+ \cup N^-, \quad (15)$$

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

$$0 \leq y_i \leq Q \text{ for } i \in N^+, \quad (17)$$

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

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 *vehicle routing problem with time windows* (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 point. Constraint (14) and (15) ensure feasibility of the schedule, and constraint (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 modal extensions presented for the *vehicle routing problem with time windows* (VRPTW) can be applied to the *pickup and delivery problem with time windows* (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 brands 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 problem. 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 method is reviewed below.

Baker<sup>2</sup> presents a branch and bound

method for the *traveling salesman problem with time windows* (TSPTW), in which bound 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<sup>31</sup> which is essentially an *multiple traveling salesman problem with time windows* ( $m$  - 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<sup>45</sup> 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, Sauve and Soumis<sup>17</sup> 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 *multiple-traveling salesman problem with time windows* ( $m$ -TSPTW) with the second objective function, Desrosiers, soumis, Desrochers and Sauve<sup>19</sup> 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<sup>44</sup> apply the first rule to solve problem 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. Sorensen<sup>43</sup> suggests the use of lagrangean decomposition<sup>23,25</sup> for the *vehicle routing problems with time windows* (VRPTW). The two resulting subproblems are the shortest path problems with time windows and the generalized assignment problem. No computational results have been reported.

3.1. *Dynamic programming :*

Dynamic<sup>26</sup> programming is traditional solution method for constrained shortest path problem. 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  $O \in V$ , 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 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 restricted to state space 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 *traveling salesman problem with time window* (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<sup>7</sup> propose the following dynamics programming algorithm. There are states of the form  $(S, j)$  with

$$S \subset N$$

$$\text{And } j \in S \text{ and, } d(S,j)$$

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.

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

$$d(S, j) = \min_{i \in S - \{j\}, \{i, j\} \in A} \{d(S - \{j\}, i) + t_{ij}\}$$

for  $j \in \cup\{n + 1\}$ ,

Where we redefine

$$d(S, j) = e_j \text{ in case } d(S, j) < e_j \text{ and } d(S, j) = \infty \text{ in case } d(S, j) > l_j.$$

Psaraftis<sup>32</sup> uses dynamic programming to solve the single vehicle *dial-a-ride problem* (DARP). The states are of the form  $(j, k_j, \dots, k_n)$  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 straight forward to define the feasible transitions between states. The Algorithm has  $2n$  stages, each of which extends the path 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<sup>14</sup> give a similar  $2n$ -stage algorithm for the capacitated single-vehicle *pickup and delivery problem with time windows* (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 problem with up to 40 customers<sup>41</sup>.

Two types of constrained shortest path

problems have been considered: the *shortest path problem problems with time windows* (SPPTW) and the capacitated *shortest path problem with pickups, deliveries and time windows* (SPPPDWTW). The main difference between these problems and the single-vehicle *dial-a-ride problem* (DARP) is that the path is no longer required to visit all customers. For the *shortest path problem problems with time windows* (SPPTW) which is defined by (1), (3) – (5) and (8) Desrosiers, Palletier and Soumis<sup>16</sup> propose a label correcting method. Desrochers and Soumis<sup>11,12</sup> give two pseudo-polynomial algorithms. One is a label setting method, the other a primal-dual method. Desrochers<sup>10</sup> generalizes the latter algorithm to the case of multidimensional time windows. For the *shortest path problem with pickup, deliveries and time windows* (SPPPDWTW) [(9) and (11) – (18)], Dumas<sup>20</sup> and Dumas and Desrosiers<sup>20</sup> present an Algorithm which is similar to the one for the capacitated single vehicle *pickup and delivery problem with time windows* (PDPTW).

As we have mentioned before<sup>3</sup>, dynamic programming algorithm are mostly used as subroutines in other solution method. This is because the problems considered in this section occur as subproblem in multi-vehicle problems. The *traveling salesman problem with time windows* (TSPTW) and the single vehicle *dial-a-ride problem* (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 *shortest path problems with time windows* (SPPTW) occurs as a subproblem in the set partitioning algorithm for the *multiple traveling salesman*

*problem with time windows*(m-TSPTW) due to Desrosiers, Soumis and Desrochers<sup>18</sup>, in the Lagrangean relaxation algorithm for the fleet size problem due to Desrosiers, Sauve and Soumis<sup>17</sup>: in the Lagrangean decomposition algorithm for the *vehicle routing problem with time windows* (VRPTW) due to Sorensen<sup>43</sup>. The *shortest path problem with pickup, deliveries and time windows* (SPPDPTW) is a subproblem in the set partitioning algorithm for the *pickup delivery problem with time windows* (PDPTW) due to Desrosiers, Dumas and Soumis<sup>15</sup>.

### 3.1.2 State space relaxation :

For a number of problems, christofides, Mingozi and Toth<sup>5,6,7</sup> 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 starting point and replace it 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 *Traveling salesman problem with time windows* (TSPTW) [christofides, Migozzi and Toth (7)].

With each vertex  $i$ , an arbitrary

integer  $\beta_i$  is associated, with  $\beta_0 = \beta_{(n+1)} = 0$ . The mapping is defined by

$$g(S,j) = (k, \beta, j),$$

Where

$$k = |S| \text{ and } \beta = \sum_{i \in S} \beta_i$$

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 time of arcs incident to undercovered vertices and to increase the travel time of arcs incident to overcovered vertices; these penalties are adjusted by subgradient optimization. Similarly, the weights  $\beta_i$  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<sup>26</sup> extend this approach to the *vehicle routing problem with time windows* (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

*traveling salesman problem with time windows* (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 cost of  $m$  routes with total load

$$\sum_{i \in N} q_i.$$

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 vortex 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 problem and in particular the *vehicle routing problem with time windows* (VRPTW) and the *pickup and delivery problem with find windows* (PDPTW) can be reformulated as set partitioning problem, 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  $\gamma_r$  as the sum of the costs of its arcs and

$$\delta_{ri} \quad (i \in N)$$

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.

$$\sum_{r \in R} \gamma_r x_r \quad (19)$$

Subject to

$$\sum_{r \in R} \delta_{ri} x_r = 1 \quad \text{for } i \in N \quad (20)$$

$$x_r \in \{0, 1\} \quad \text{for } r \in R \quad (21)$$

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 feasible solution to the relaxation of (9) – (18), but not vice versa. We can therefore expect to obtain better lower bound 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 problem, we first describe some general aspects of this approach.

#### 3.2.1 The submodel :

The objective function of the subproblem has coefficients that depend on the values of the dual variables  $\pi_i$  ( $i \in N$ ) 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 *vehicle*

*routing problem with time windows (VRPTW), (3) – (5) and (8) for the multiple traveling salesman problem with time windows (m-TSPTW), and (11) – (18) for the pickup and delivery problem with time windows (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 optimization algorithm :

The continuous relaxation of the set partitioning problem is solved by the simplex Algorithm. This method produces the dual values  $\pi_i$  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, 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

dual value  $\pi$ , 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 where 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 set  $x_r = 0$ : 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 vehicle 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 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 subproblem 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 which 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<sup>18</sup> propose a set partitioning approach to the *multiple traveling salesman problem with time windows* (m-TSPTW). The column generation problem is the *shortest path problem with time windows* (SPPTW). 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 problem 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 algorithms 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<sup>13</sup> 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 vehicle of a given type; when this number is integral for each types, the usual branching on flow variables is applied. No set partitioning algorithm for the vehicle routing problem with time windows

(PDPTW) has been proposed so far. However, Desrochers<sup>10</sup> presents a dynamic programming algorithm for the shortest teat 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<sup>20</sup> develops a set partitioning approach for the *pickup and delivery problem with time windows* (PDPTW). He solves problems with 30 customers (60 vertices) within 100 second on a CDC Cyber 173. These problems have tight capacity constraints ( $q_i \geq Q/3$ ) and loose time Window constraints. Narrowing the time windows significantly decreases the cardinality of the state space and there by the computation time.

The subproblem in this case is the *shortest path problem with pickups, deliveries and time window* (SPPDPTW), which was reviewed in section 3.1.1 The algorithm of Dumas<sup>20</sup> first branches an the number of vehicles per type and then on flow variables. Desrosiers, Dumas and Soumis<sup>15</sup> 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 O) 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 i<sup>th</sup> custom visited by the vehicle. There are two quantities associated with a subpath (h, ...,k) that play a dominant role in the algorithm 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):

$$S_{hk}^+ = \min_{h \leq j \leq k} \left\{ \ell_j - (D_h + \sum_{h \leq i < j} t_{i,i+1}) \right\}$$

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):

$$S_{hk}^- = \min_{h \leq j \leq k} \prod D_j - e_j \int$$

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

$$S_{hk}^+ \text{ and } S_{hk}^-$$

For  $j = h, \dots, k$  can be computed in  $O(n)$  time. [Savelsborgh (35)].

#### 4.1 Construction :

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

1. Solution 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 model with time windows :

Solomon<sup>40</sup> was one of the first who attempted to adopt the existing approximation algorithms for the *vehicle routing problem* (VRP) to the *vehicle routing problem with time windows* (VRPTW). Much of the material in this section is based on his work.

*Savings.* The savings method of Clarke and Wright<sup>8</sup> is probably the first and certainly the best known heuristic proposed to solve the *vehicle routing problem* (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 saving and subject to vehicle capacity constraints. Note that in his 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 *vehicle routing problem with time windows* (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 time 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  $(1, \dots, n+1)$ , 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 the both spatial and temporal closeness of customers, e.g., by adding a waiting time penalty to the cost saving.

*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 now end point. The selection is restricted to those customers whose addition is feasible with respect to capacity and time window 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<sup>40</sup> proposes following:

$$\alpha_1 t_{ij} + \alpha_2 (\max\{e_j, D_i + t_{ij} - D_i\} + \alpha_3 |e_j - (D_i + t_{ij})|)$$

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 separately. 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  $\mu \in U$ , We first determine the best feasible point  $i_\mu$  after which it could be inserted into the emerging route:

$$i(\mu, i_\mu) = \min_{0 \leq i \leq n} \{i(\mu, i)\} \quad \text{for } \mu \in U$$

We next select the customer  $\mu^*$  to be inserted into the route:

$$\sigma(\mu^*, i_{\mu^*}) = \min_{\mu \in U} \{ \sigma(\mu, i_\mu) \}$$

The insertion criterion  $i$  and selection criterion  $\sigma$  are still to be specified; we refer to Solomon<sup>40</sup>

and Savelsbergh<sup>35</sup> for a number of possible definition which takes both spatial and temporal aspects into account. When no more customers can be inserted, a new route is started, unless all customers have been route.

Insertion of  $\mu$  between  $i$  and  $i+1$  could changes all departure time on the path  $(i+1, \dots, n+1)$ . 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<sup>40</sup> concludes on the basis of extensive computational experiments that insertion methods outperform other type of construction methods.

4.1.2 The pickup and delivery model with time windows :

Jaw, Odoni, Psaraftis and Wilson<sup>24</sup> consider a variant of the dial-a-ride problem (DARP). Their approach seems to be applicable to the proper dial-a-ride problem (DARP) as well.

The customers that are to be pickup and delivered have the following types of service constraints. Each customer  $i$  specifies either a desired pickup time

$$\bar{D}_{i+}$$

Or a desired delivery time

$$\bar{A}_{i-},$$

And a maximum travel time

$$\bar{T}_i;$$

In addition, there is a tolerance

$$\bar{U}.$$

If customer  $i$  has specified a desired pickup

time, the actual pickup time  $D_{i+}$  should fall within the time window

$$[\bar{D}_{i+}, \bar{D}_{i+} + \bar{U}],$$

If he has specified a desired delivery time, the actual delivery time  $A_{i-}$  should fall within the window

$$[\bar{A}_{i-} - \bar{U}, \bar{A}_{i-}].$$

Moreover, his actual travel time should not exceed his maximum travel time:

$$A_{i-} - D_{i+} \leq \bar{T}_i.$$

Note that this information suffices to determine two time windows

$$[e_{i+}, l_{i+}] \text{ and } [e_{i-}, l_{i-}]$$

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 period of time a vehicle is active between two successive period of slack time. For convenience, we drop the superscript indicating pick up or delivery. For each visit to an address  $i$  during an active period, we define the following variants of possible backward and forward shifts.

$$\sum_i^- = \min \{ \min_{j \leq i} |A_i - e_j|, \Lambda S \}$$

$$\sum_i^+ = \min_{j \leq i} |l_i - A_j|$$

$$S_i^- = \min_{j \geq i} |A_i - e_j|$$

$$S_i^+ = \min \{ \min_{j \geq i} |l_i - A_j|, LS \}$$

Where  $\Delta$  and  $L$  are the duration of the slack periods immediately preceding and following the active period in question

$$\Sigma_i^-(\Sigma_i^+)$$

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 segment of an active period can be displaced to accommodate an additional customer. Once it is established that same way of inserting the pickup and delivery of customer  $i$  satisfies the time window constraints, it must be ascertained that it satisfied 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<sup>38,39</sup> consider a variant of the single-vehicle *dial-a-ride problem* (DARP) in which only deadlines for the deliveries are specified. Their solution approach is to apply Bender decomposition to a mixed 0-1 nonlinear programming formulation, which separates the routing and scheduling component.

#### 4.2 Iterative improvement :

Cross<sup>9</sup> and Lin<sup>27</sup> introduced the notion of k-exchanges to improve solutions to the *traveling salesman problem* (TSP) Lin and Kernighan<sup>28</sup> generalized the approach and

many author reported on its applications to related problem. In the context of vehicle routing, christofides and Eilon<sup>4</sup> and Russell<sup>34</sup> adapted the approach to the basic *vehicle routing problem* (VRP) and Psaraftis<sup>34</sup> used it for the *dial-a-ride problem* (DARP). In this section, we will show how time windows can be handled in k-exchange procedures for *traveling salesman problem with time windows* (TSPTW) [Savelsbergh<sup>35</sup>]. It is not hard, however, to extend the techniques to multi-vehicle problem with various types of side constraints [Savelsbergh<sup>36,37</sup>]. The issue is also addressed in another contribution to this volume [Solomon, Baker and Schaffer<sup>42</sup>]. A k-exchange is a substitution of  $k$  arcs of a route with  $k$  other arcs. In the *traveling salesman problem* (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 *traveling salesman problem with time window* (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 model with time windows :

The number of possible k-exchange in a given routes is  $O(n^k)$ . The computational requirement of k-exchange procedures thus increase rapidly with  $k$ , and one usually only considers the case  $k=2$  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 revering the path  $(i+1, \dots, j)$ . In a-3 exchange, three arcs are deleted and there are seven possibilities to construct a new route from the remaining segment. Or<sup>30</sup> proposes to restrict attention to those 3-exchange 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 exchange in which one customers in relocated. The basic idea of the approach is the use of 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 case 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  $i-1, i-2, \dots, 0$ ; note that in each exchange the path  $(j+1, \dots, i-1)$  of the previous exchange is expanded with the arc  $(j, j+1)$ . In the latter case,  $j$  assume the values  $i+1, i+2, \dots, n$  in that order; in each exchange the path  $(i+1, \dots, j-1)$  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 through  $i$  to  $j+1$

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

5. The waiting time  $W$  on the path  $(j+1, \dots, i+1)$

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

During the backward search, an exchange is feasible if

$$D_k^{new} \leq l_k$$

For  $k = j+1, \dots, i-1$ , and potentially profitable if

$$D_{i+1}^{new} < D_{i+1}.$$

The superscript ‘new’ indicate 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 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

$$D_k^{new} \leq l_i$$

And potentially, profitable if

$$D_{i+1}^{new} < D_{j+1}.$$

This equivalent to

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

The only update is  
 $S^- := \min \{ D_j - e_j, S^- \}$ .

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

#### 4.3 In complete optimization :

Fast approximation algorithms can also be derived from the optimization algorithms prevented 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 problem subject to real life

constraints. A striking example is the set partitioning approach, which appears to be particularly efficient for strongly constrained problem. The continuous relaxation of the set partitioning formulation can be solved 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 problem of a realistic size. Optimization algorithms of this type are being used for school bus scheduling.

### References

1. Anthonisse J.M., Lenstra J.K., Savelsbergh M.W.P., *Function description of CAR, an interactive system for computer aided routing*, Report OR-R86716, Centre for Mathematic and Computer Science Amsterdam (1987).
2. Baker E. K., An exact algorithm for the time – constrained traveling salesman problem. *Oper. Res.* 31, 65-73 (1983).
3. Bodin L., Golden B., Classification in vehicle routing and scheduling *network* 11, 97-108 (1981).
4. Christofides N., Eilon S., An algorithm for the vehicle dispatching problem. *Oper. Res. Quart.* 20, 309-318 (1969).
5. Christofides N., Mingozzi A., Toth P., Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations math. *Programming* 20, 255-282 (1981a).
6. Christofides N., Mingozzi A., Toth P., State space relaxation procedures for computation of bounds to routing problem. *Network*, 11, 145-164 (1981b).
7. Christofides N., Mingozzi A., Toth P., *Exact*

- algorithms for the traveling salesman problem with time constraints, based on state-space relaxation.* Unpublished manuscript.
8. Clarke G., Wright J.W., Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* 12, 568-581 (1964).
  9. Croes A., A method for solving traveling salesman problem. *Oper. Res.*, 5, 791-812 (1958).
  10. Desrochers M., *An algorithm for the shortest path problem with resource constraints.* Publication 421A. centre de recherché sur les transports, Université de Montreal.
  11. Desrochers M., Soumis F., *A generalized permanent Labelling algorithm for the shortest path with time windows.* Publication 394A, centre de recherché sur les transports, Université de Montreal.
  12. Desrochers M., Soumis F., *A reoptimization algorithm for the shortest path problem with time windows.* Publication (397A). centre de recherché sur les transports, Université de Montreal (1985b).
  13. Desrosiers J., Dumas Y., Soumis F., *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 Montreal (1986a).
  14. Desrosiers J., Dumas Y., Soumis F., 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 (1986b).
  15. Desrosiers J., Dumas Y., Saumis F., *Vehicle routing problem with pickup delivery and time windows,* cahiers du GERAD, Ecole des Hautes Etudes Commerciales de Montreal (1987).
  16. Desrosiers J., Pelletier P., Soumis F., Plus court chemin avec contraintes d'horaires *Rairo Reach. Oper.* 17(4), 357-377 (1984).
  17. Desrosiers J., Sauve M., Soumis F., *Lagrangian relaxation methods for solving the minimum fleet size multiple traveling salesman problem With time windows,* Publication 396, centre de recherché sur les transports, Université de Montreal (1885).
  18. Desrosiers J., Soumis F., Desrochers M., Routing with time windows by column generation. *Network* 14, 545-565 (1984).
  19. Desresters J., Soumis F., Desrochers M., Sauve, Routing and scheduling with time windows solved by network relaxation and branch and bound on time variables. J.M. Rousseau (ed.) *computer scheduling of pbclic transport* 2, North-Holland Amsterdam 451-469 (1985).
  20. Dumas Y., *Confection d'itineraire de vehicules du transport de plusieurs origins a plusieurs destinations.* Publication 434. Université De Montreal (1985).
  21. Dumas Y., Desrosiers J., *A shortest path problem for vehicle routing with pickup, delivery and time windows.* Cahiers du GERAD G-86-09, Ecole des Hautes Etudes Commerciales de Montreal (1986).
  22. Golden B.L., Assad A.A., *Vehicle routing with time window constraints. Algorithm solution.* *Amer.j. math. Management Sci.*, 6, 251-428 (1986).
  23. Guignard M., *Lagrangen Decompoistion: an improvement over Lagrangean and Surrogate duals,* report 62. Department of statistics, University of Pennsylvania Philadelphia.
  24. Jaw J.J., Odoni A.R., Psaraftis H.N., Wilson N.H.M., A heuristic algorithm for the

- multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Res. Part B* 20B, 243-257 (1986).
25. Jornsten K.O., Nasbers M., Smeds P.A., *Variable splitting- a new lagrangean relaxation approach to some mathematical programming model* report LITH-MAT-R- 85-04. Department of mathematics, Linkoping Institute of Technology (1985).
  26. Kolen A.W.J., Rinnooy A.H.G, Trienekens H.W.J.M., Vehicle routing with time windows. *Oper. Res.* To Appear (1987).
  27. Lin S., Computer solutions to the traveling salesman problem. *Bell System Tech. J.* 44, 2245-2269 (1965).
  28. Lin S., Kernighan B.W., An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* 21, 498-516 (1973).
  29. Miller C., Tucker A., Zemlin R., Integer programming formulation of traveling salesman problems. *J. Assoc. Comput. Mach.* 7, 326-329 (1960).
  30. Or I., *Traveling salesman type combinatorial problem and their relation of the logistics of blood banking*, Ph.D. thesis, Northwestern University, Evanston, IL (1976).
  31. Orloff C.S., Route constrained fleet scheduling. *Transportation sci.* 10, 149-168 (1976).
  32. Psaraftis H., An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Sci.* 17, 351-360 (1983a).
  33. Psaraftis H., *k-interchange procedures for local search in a precedence-constrained routing problem.* *European J. Oper. Res.* 13, 391-402 (1983b).
  34. Russell, R. A., *Oper. Res.* 25, 517-524 (1977).
  35. Savelsbergh M.W.P., *Open. Res.* 4, 285-305.
  36. Savelsbergh, M.W.P., *Local search for constrained routing problem*, report OS-R8711 Centre for Mathematics and Computer Science, Amsterdam (1987).
  37. Savelsbergh, M.W.P., *Computer aided routing*, Ph.D. thesis, Centre for Mathematics and Computer Science, Amsterdam (1988).
  38. Sexton T.R., Bodin L.D., Optimizing single vehicle many to many operations with desired delivery times II. *Putting transportation. Sci.* 19, 378-410 (1985a).
  39. Sexton T.R., Bodin L.D., II Routing *transportation. Sci.* 19, 411-435 (1985b).
  40. Solomon M.M., *Vehicle routing and scheduling with time windows constraints. Models and algorithms* report 83-02-01. The Wharton school, University of Pennsylvania Philadelphia (1983).
  41. Solomon M.M., On the worst case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Network* 16, 161-174 (1986).
  42. Solomon M.M., Baker E.K., Schaffer J.R., Vehicle routing and scheduling problem with time window constraints: implementation of solution improvement procedures. This volume (1988).
  43. Sorensen B., *Interactive distribution planning*, Ph.D. thesis, Technical University of Denmark, Lyngby (1986).
  44. Soumis F., Desrosiers J., Desrochers M., Optimal urban bus routing with scheduling flexibilities. *Lecture notes in control and information. Science* 59, Springer, Berlin, 155-165 (1985).
  45. Swersey A. J., Ballard W., Scheduling school buses. *Management Sci.* 30, 844-853 (1984).