Time Constrained Routing and Scheduling

Chapter · January 1995

CITATIONS
680

READS
3,715

4 authors, including:

Jacques Desrosiers
HEC Montréal - École des Hautes Études commerciales
129 PUBLICATIONS 10,234 CITATIONS
SEE PROFILE

François Soumis
Polytechnique Montréal
165 PUBLICATIONS 7,665 CITATIONS
SEE PROFILE

Time Constrained Routing and Scheduling

Jacques Desrosiers
GERAD and École des Hautes Études Commerciales
Montréal, Canada, H3T 1V6

Yvan Dumas
GERAD and École des Hautes Études Commerciales
Montréal, Canada, H3T 1V6

Marius M. Solomon
Northeastern University and GERAD
Boston, Massachusetts, 02115, U.S.A.

François Soumis
GERAD and École Polytechnique de Montréal
Montréal, Canada, H3C 3A7

April, 1992

Revised: May, 1993
September, 1993
August, 1994

Forthcoming in Handbooks in Operations Research and Management Science
Volume on Networks, North-Holland, Amsterdam.

À la mémoire de Martin Desrochers, décédé prématurément le 3 juin 1991 à l'âge de 33 ans, pour sa contribution exceptionnelle au succès de notre équipe.

Abstract

This survey paper describes the significant advances made in time constrained vehicle routing and crew scheduling problems. In terms of solution methodology capable of solving realistic size problems, this field has seen a natural progression from ad-hoc methods to simple heuristics, to optimization-based heuristics and recently optimal algorithms. The paper provides an extensive overview of the algorithms developed and focuses on optimization methods.

It first addresses fixed schedule problems and develops in detail the Dantzig-Wolfe decomposition/column generation approach which can be applied to many of the other problem types. The paper treats next the traveling salesman problem with time windows and a variety of shortest path problems with time windows or resource intervals. These constitute important modules of more complex optimization models used for time constrained vehicle routing and crew scheduling problems. It then examines the vehicle routing problem with time windows and a number of its variants and generalizations including the pick-up and delivery problem. The paper finally presents a unified solution framework which is being successfully implemented in school-busing and urban transit areas, as well as in airline and railway industries.

Sommaire

Ce texte de synthèse décrit les récents résultats méthodologiques et pratiques obtenus dans le domaine de la confection d'itinéraires de véhicule et d'horaires d'équipage régis par des contraintes temporelles. Au niveau des algorithmes capables de résoudre des problèmes pratiques, ce domaine d'application a suivi la progression naturelle des implantations ad-hoc aux heuristiques simples, puis des heuristiques intégrants des concepts mathématiques et tout récemment, des algorithmes optimaux issus de la programmation mathématique. Ce texte donne une vue d'ensemble des algorithmes développés mais insiste plus particulièrement sur les plus récentes méthodes optimales.

Nous traitons en premier lieu de problèmes à horaire fixe où est présentée en détail la méthode de décomposition de Dantzig-Wolfe, aussi appelée méthode par génération de colonnes. Celle-ci s'applique à une vaste classe de problèmes traités par la suite. Nous présentons également divers algorithmes pour résoudre le problème du voyageur de commerce sous des contraintes d'intervalles de temps, les problèmes de plus court chemin avec le même type de restriction temporelle, ainsi que les extensions de ce dernier type de problème à des contraintes portant sur des ressources. Ces problèmes sont souvent utilisés comme modules spécialisés dans plusieurs méthodes optimales de résolution des problèmes de confection de tournées de véhicule et d'horaires d'équipage. Nous présentons en dernier lieu une approche unifiée de résolution de ces problèmes. Le modèle mathématique et la méthodologie utilisée ont été implantés avec succès dans plusieurs secteurs d'activités, notamment en transport scolaire, public, aérien et ferroviaire.

Acknowledgements

During the writing of this survey we have benefitted from numerous discussions with our friends and colleagues including Pierre Hansen, Brigitte Jaumard, André Langevin, Gilbert Laporte, Celso Ribeiro, Brunilde Sansó and Ben Smith. We are particularly grateful to Ed Baker, Oli Madsen and Alexander Rinnooy Kan who have taken the time to read an earlier version of this manuscript and have provided us with very helpful suggestions. We wish to especially thank Paolo Toth for his extensive and insightful comments. Mike Ball and an anonymous referee have made additional comments that have improved the manuscript. Special thanks are due to Irina Ioachim and Sylvie Gélinas who have also been instrumental in going over the paper with a fine comb. We are very grateful to Francine Benoît and Nicole Paradis at GERAD who have spent inumerable hours typing and editing the many versions of the manuscript.

This text has been used at the NorFA Research Course in Narvik, Norway, June 21–28, 1993. We would like to thank the organizers, Min Hwei Chuang, Per Olov Lindberg and Oli Madsen for providing us with a vehicle for testing this manuscript in the classroom. We have been involved in fruitful discussions with Susan Powell, Erik Anderson, Marielle Christiansen, Eric Gélinas, Niklas Kohl, Michael Lind, Andreas Nöu, Dag Wedelin and others during this Summer Course.

This work could have never been accomplished without the support and understanding of our families and special friends. We are most grateful to them.

Jacques Desrosiers, Yvan Dumas and François Soumis were partially supported by the Natural Sciences and Engineering Research Council of Canada grants. Marius Solomon was partially supported by the Patrick F. and Helen C. Walsh Research Professorship.

Contents

List of Figures

1 Introduction

Over the last two decades, economic phenomena such as the oil crisis of the early '70s, the deregulation of the U.S. airline and trucking industries in the '80s, and Europe '92 have highlighted transportation as an area where large productivity improvements could be achieved. This interest in transportation has also been fueled by other important developments such as the continuing reduction in the labor cost of products time-based competition, and the constant breakthroughs in computer technology. In turn, operation researchers have found a renewed interest in routing and scheduling problems. These problems generally involve the assignment of vehicles to trips such that the assignment cost and the corresponding routing cost is minimized.

During this period, the temporal aspect of these intrinsically spatial problems has also become increasingly important as manufacturing, service and transportation companies have tried to not only cut their logistics costs, but also compete on service differentiation. Not surprisingly, we have then witnessed the development of a fast growing body of research focused on time constrained routing and scheduling. The time dimension has been incorporated in these problems in the form of customer-imposed time window constraints. These constraints restrict the start of service at a customer to begin later than or at a prespecified earliest time and earlier than or at a prespecified deadline. When each time window consists of a single value, a fixed schedule problem is obtained. When the time windows are treated as hard constraints, a vehicle is not permitted to arrive at a node after the latest time to begin service. However, if a vehicle arrives too early at a customer, it is permitted to wait until the customer is ready for the beginning of service. A much less studied, but nevertheless important, variant of the problem incorporates soft time windows which can be violated at a cost.

In general, researchers have considered a homogeneous vehicle fleet mix. Yet, as the field has matured, the increased level of sophistication of the methods developed has permitted the treatment of heterogeneous fleet mixes. The fleet size has been assumed either fixed a priori or free, i.e., to be determined simultaneously with the best set of routes and schedules. However, many methods can actually optimize the fleet size in the process.

The costs involved in time-constrained routing and scheduling consist of fixed vehicle utilization costs and variable routing and scheduling costs. These latter costs include distance and travel time costs, waiting time costs, and loading/unloading time costs. In addition, inconvenience costs are associated with the transport of individuals. For problem variants involving the assignment of personnel to tasks, corresponding costs are defined.

The first articles to mention problems with temporal constraints date back to Dantzig and Fulkerson (1954), Ford and Fulkerson (1962), Appelgren (1969, 1971), Levin (1971), Madsen (1976), and Orloff (1976). Since then, a flurry of activity has been directed at the classical routing model, many of its realistic variants, important generalizations including the temporal aspect, and a variety of practical applications. This research has been reviewed in several insightful surveys written by Magnanti (1981), Bodin, Golden, Assad and Ball (1983), Desrochers, Lenstra, Savelsbergh and Soumis (1988), Solomon and Desrosiers (1988), and Halse (1992).

Time constrained routing and scheduling problems are encountered in a variety of industrial and service sector applications, ranging from logistics and transportation systems to material handling systems in manufacturing. Fixed schedule problems appear in several domains such as airline and rail transportation. The shortest path problem with time windows has been successfully utilized as a subproblem for the multiple traveling salesman problem used to construct school-bus routes. The traveling salesman problem with time windows has applications in single and multiple vehicle problems. The vehicle routing problem with time windows has many industrial applications including those where dock availability is a bottleneck such as for distribution centers. Pick-up and delivery problems with time windows have been applied to the transportation of the handicapped and elderly persons. Fixed schedule problems with resource constraints have found a fertile area in urban-bus driver scheduling and airline crew scheduling environments. In addition to the hard time window applications just described, soft time windows constraints have been applied to the transport of individuals in dial-a-ride problems.

The goal of this paper is to describe the significant advances made in time-constrained routing and scheduling. In terms of solution methodology capable of solving realistic size problems, this field has seen a natural progression from ad-hoc methods to simple heuristics, to optimization-based heuristics and recently optimal algorithms. The paper provides an extensive overview of the algorithms developed and focuses on optimization methods. Helped by continuously better insights into problem structures with time windows and rapid advances in computer technology, these methods are becoming a viable tool for solving practical size problems. While we have tried to be as comprehensive as possible, we have undoubtly omitted a few papers. We apologize in advance for any such occurrence.

The paper is organized as follows. Section 2 addresses fixed schedule problems and develops in detail the Dantzig-Wolfe decomposition/column generation approach which will then be applied to many of the other problem types. Next, Section 3 treats the traveling salesman problem with time windows. In Section 4, we examine time and resource constrained shortest path problems which constitute important modules of many of the algorithms to be described later. Section 5 explores the vehicle routing problem with time windows and several important problem variants including the multiple traveling salesman problem. Then, Section 6 treats the pick-up and delivery problem with time windows. In particular, the dial-a-ride problem with time windows is treated in this section. Section 7 examines an unified framework for fleet and crew scheduling problems. Finally, our conclusions and perspectives on future research are presented in Section 8.

2 Fixed Schedule Problems

The problem treated in this section considers a set of tasks (or trips) characterized by an origin, a destination, a duration and a fixed starting time. The tasks are given a priori and are composed of one or several activities. Between two successive tasks there is an inter-task transit time period of a certain duration and cost. The problem to be solved consists of forming a schedule, i.e., sequences of tasks where each task is performed exactly once. These sequences are sometimes called routes.

This type of problem is encountered in several fields, such as airline, rail, school-bus and urban transportation. In airline transportation, the trips are the flight legs: they are described by a city pair, an aircraft type and a schedule. Connections are possible only between two flight segments where the first ends and the second begins at the same airport, respectively. In rail transportation, this problem is found in the assignment of engines to train cars. A trip is determined by a train to haul at a given time. Connections are possible between the trains arriving at a station and those leaving the same station or other stations nearby. Several locomotives can be assigned to the same train. In school-bus transportation (the morning problem), a trip consists of visiting a sequence of stops, where school children are picked-up, and one or several schools, where these children are delivered. Connections are possible between the trips belonging to the same geographical region. In urban transportation, buses must be assigned to trips and connections are possible on the city network, in conformance to certain rules.

Each application differentiates itself by the structure of the network, its size and its periodicity (e.g., monthly, weekly or daily). Dantzig and Fulkerson (1954) were the first to formulate a fixed schedule problem as a minimum cost flow circulation problem. This section presents several refinements brought since then to the formulation of the problem as well as to the underlying network. We will be particularly interested in the construction of routes for vehicles to be assigned to fixed-schedule trips. The application to the assignment of personnel to tasks is immediate. In what follows we assume that all the vehicles are identical and that they all originate from a unique depot. It is therefore possible to solve the problems addressed in polynomial time (Lenstra and Rinnooy Kan 1981). The problem involving several depots is treated at the end of this section. More complex models involving flexible schedules, different vehicle types, and other generalizations, are treated in the subsequent sections.

2.1 The Single Depot Vehicle Scheduling Problem

Notation: To describe the Single Depot Vehicle Scheduling Problem (SDVSP), consider a set of n trips $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$, where trip $\mathcal{T}_i$ has a given duration and starts at time a_i, $i = 1, 2, \ldots, n$. Consider also a single depot where v vehicles are stationed. Let the set of nodes, $N = \{1, 2, \ldots, n\}$ represent the set of trips, and node $\{n+1\}$ represent the single depot. Let also t_{ij} denote the travel time between the ending point of trip $\mathcal{T}_i$ and the starting point of trip $\mathcal{T}_j$. For simplicity we will assume that t_{ij} includes the duration of the trip $\mathcal{T}_i$. An ordered pair of trips $(\mathcal{T}_i, \mathcal{T}_j)$ is said to be compatible if it satisfies the relation $a_i + t_{ij} \leq a_j$, for all $i, j \in N$. If two trips are to be covered by the same vehicle, additional restrictions may be imposed for compatibility. Denote the set of compatible trips by I, a subset of $N \times N$. Then define $A = I \cup (\{n+1\} \times N) \cup (N \times \{n+1\})$ to be the set of arcs. Let c_{ij}, $(i, j) \in I$ be the cost incurred if a vehicle performs trip $\mathcal{T}_j$ immediately after trip $\mathcal{T}_i$. This cost is usually a function of the distance between the ending point of trip $\mathcal{T}_i$ and the starting point of trip $\mathcal{T}_j$. It can also include the fixed cost incurred to cover one of the trips, say trip $\mathcal{T}_i$. Let next $c_{n+1,j}$ be the cost incurred if a vehicle undertakes trip $\mathcal{T}_j$, $j = 1, 2, \ldots, n$ as the first trip after leaving the depot. Similarly, the cost $c_{j,n+1}$ is incurred when trip $\mathcal{T}_j$ is the last trip before the vehicle returns to the depot. This cost can also include the fixed cost incurred to cover trip $\mathcal{T}_j$. Finally, let c be the fixed cost associated with the use of a vehicle. In general, the above costs are non-negative. The problem consists of finding an assignment of vehicles to trips in such a way that:

- each trip is covered exactly once by a vehicle;
- each vehicle used in the solution leaves from the depot, covers a sequence of pairwise compatible trips and returns to the depot at the end of the route;
- the number of vehicles leaving from the depot does not exceed v (this number may be fixed a priori at v, minimized during the solution process, or free);
- the sum of the costs of the routes traveled by the vehicles used in the solution is minimized.

To formally describe the SDVSP, define the graph $G = (V, A)$, where $V = N \cup \{n+1\}$ is the set of nodes and A is the set of arcs. This directed graph is acyclic on a cylinder. Note first that graph (N, I) is already acyclic. By making duplicates of the depot node, o and d for the origin and the destination of any feasible route, respectively, graph G can be unfolded to become acyclic in the plane (Figure 2.1). If however, depot nodes o and d are merged, graph G is no longer acyclic, except on a cylinder. Let the variables X_{ij} be the binary flow on arc $(i, j) \in A$: $X_{ij} = 1$, if a vehicle covers trips $\mathcal{T}_j$ after trip $\mathcal{T}_i$, and $X_{ij} = 0$, otherwise.

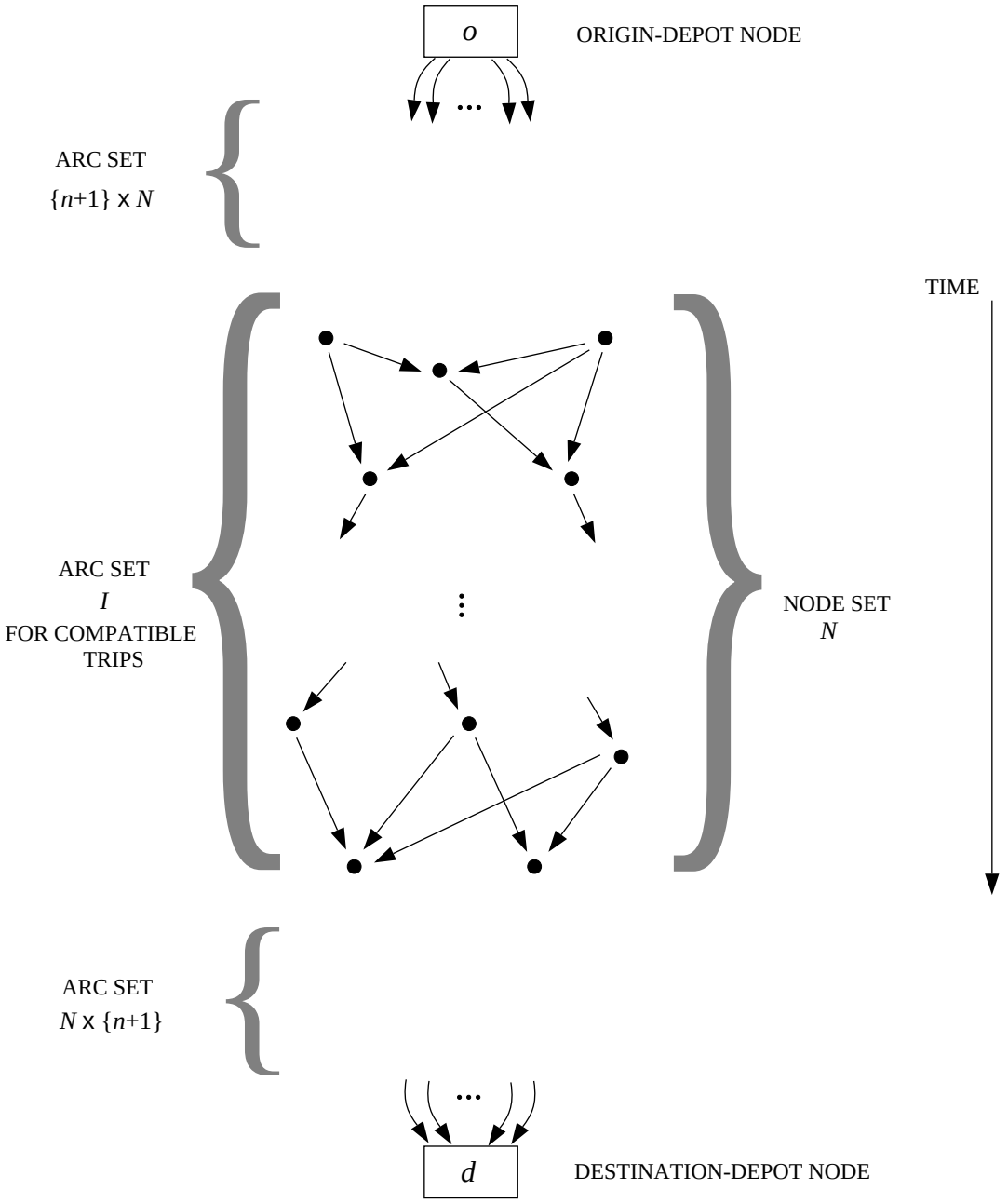


Figure 2.1: The unfolded acyclic graph

Formulation: The SDVSP can be formulated as a pure network flow problem:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}X_{ij} + \sum_{j\in N} cX_{n+1,j} \tag{2.1}$$

subject to:

$$\sum_{j\in V} X_{ij} = 1, \qquad \forall\, i \in N \tag{2.2}$$

$$\sum_{j\in N} X_{n+1,j} \leq v, \tag{2.3}$$

$$\sum_{i\in V} X_{ij} - \sum_{i\in V} X_{ji} = 0, \qquad \forall\, j \in V \tag{2.4}$$

$$X_{ij} \geq 0, \qquad \forall\, (i,j) \in A. \tag{2.5}$$

The objective function seeks to minimize the sum of travel and fixed vehicle utilization costs. Constraints (2.2) ensure that each trip is performed exactly once, while constraints (2.3) and (2.4) are fleet size and flow conservation constraints, respectively. The summations involved in the objective function as well as in the constraint sets are restricted to the defined variables. This convention is utilized throughout the text. If the number of vehicles is fixed at v, inequality (2.3) is replaced by an equality constraint. To minimize the number of vehicles, c is assigned a very large value. Finally, if this number is free, simply set $c = 0$. For simplicity, the value of c is generally incorporated in that of $c_{n+1,j},\ \forall\, j \in N$.

The above formulation can be solved either as a Minimum Cost Network Flow Problem or a Minimum Cost Circulation Problem. However, it must be noted that the networks on which algorithms are applied are different from that defined by the graph G. On these networks, each node representing a trip is split into two nodes, a trip-origin and a trip-destination node, respectively. These two nodes are then linked by an arc. A lower and upper bound equal to 1 is imposed on these n trip arcs. These transformations ensure that each trip is covered exactly once. The cost of such an arc can be set to zero or to the fixed cost incurred to cover that trip. Since each trip has to be covered exactly once, the sum of the costs of all the trips is a constant. Note that inter-trip costs must be defined accordingly. Similarly the depot node is split into an origin-depot node and a destination-depot node. These two nodes are then linked by a directed arc with zero cost. If this arc is directed from the origin to the destination depot it represents the slack variable in constraints (2.3). Given a supply of v at the origin-depot and an equal demand at the destination-depot, the capacity of this arc is $v - 1$, since it needs at least one vehicle to cover the trips. From the above description it follows that formulation (2.1)–(2.5) can be solved as a Minimum Cost Network Flow Problem. If however the arc has the reverse orientation, i.e., from the destination-depot node to the origin-depot node, then the flow on this arc is restricted to be

less than or equal to v, and a Minimum Cost Circulation Problem is derived. Each network is comprised of $2n+2$ nodes and $|A|+n+1$ arcs. The time complexity of the corresponding algorithms is $O((n+1)^3)$ (see Ahuja, Magnanti and Orlin 1989). In what follows, we examine several specialized cases of network representation, corresponding algorithms and cost structures.

Transportation/Assignment Network: Further analysis of the formulation (2.1)–(2.5) also reveals the structure of a Transportation Problem on a bipartite graph with $n+1$ nodes on each side (Figure 2.2). The $n+1$ nodes on the left are the origin-depot node o and the n trip-destination nodes. The $n+1$ nodes on the right are the destination-depot d and the n trip-origin nodes. The supply and the demand equal v for the depot node, and equal 1 for all the trips of N. The arcs are those of A together with the additional zero cost arc (o,d) symbolizing the unused vehicles or the slack variable in (2.3). The computational complexity of this Transportation Problem algorithm is also $O((n+1)^3)$. Note however that this Transportation Problem has a structure which very closely resembles an Assignment Problem and can be solved with a specialized $O(n^3)$ algorithm developed by Paixão and Branco (1987). Model (2.1)–(2.5) can be directly transformed into an Assignment Problem with $n+v$ rows and $n+v$ columns by replacing the depot node $(n+1)$ with v identical nodes $(n+1),\ldots,(n+v)$, and by setting the cost of the arcs connecting these nodes equal to zero (if the number of vehicles is to be minimized or if it is free) or equal to ∞ (if exactly v vehicles must be used), and by setting $c_{n+k,j} = c + c_{n+1,j}$, for $k = 1,\ldots,v$ and $j \in N$. The time complexity of the corresponding algorithm is $O((n+v)^3)$.

The elimination of all the depot nodes is equally possible if the minimization of the number of vehicles (large c) is sought, or if this number is free ($c = 0$). This leads to an Assignment Problem formulation (Orloff 1976). The arcs $(i,j) \in I$ are assigned the cost c_{ij}. Otherwise, if $(i,j) \in N \times N - I$, such an arc is defined nevertheless and it is assigned a cost of $c_{i,n+1} + c + c_{n+1,j}$. This cost is the sum of the cost $c_{i,n+1}$ of returning from the end of trip i to the depot, the cost c of utilizing a new vehicle, and the cost $c_{n+1,j}$ of going from the depot to the beginning of trip j. The complexity of the corresponding algorithm reduces to $O(n^3)$. This formulation is however not suitable if the number of vehicles is fixed a priori since the additional constraint needed to fix this number destroys the assignment structure.

Minimal Fleet Size: Given that the fixed cost associated with a vehicle is often very high, an important objective is to determine the minimum fleet size to meet the schedule of trips. In this case, the solution could be developed on the acyclic graph (N,I). A chain in this graph is a set of trips of N and inter-trips of I where a single trip is considered as

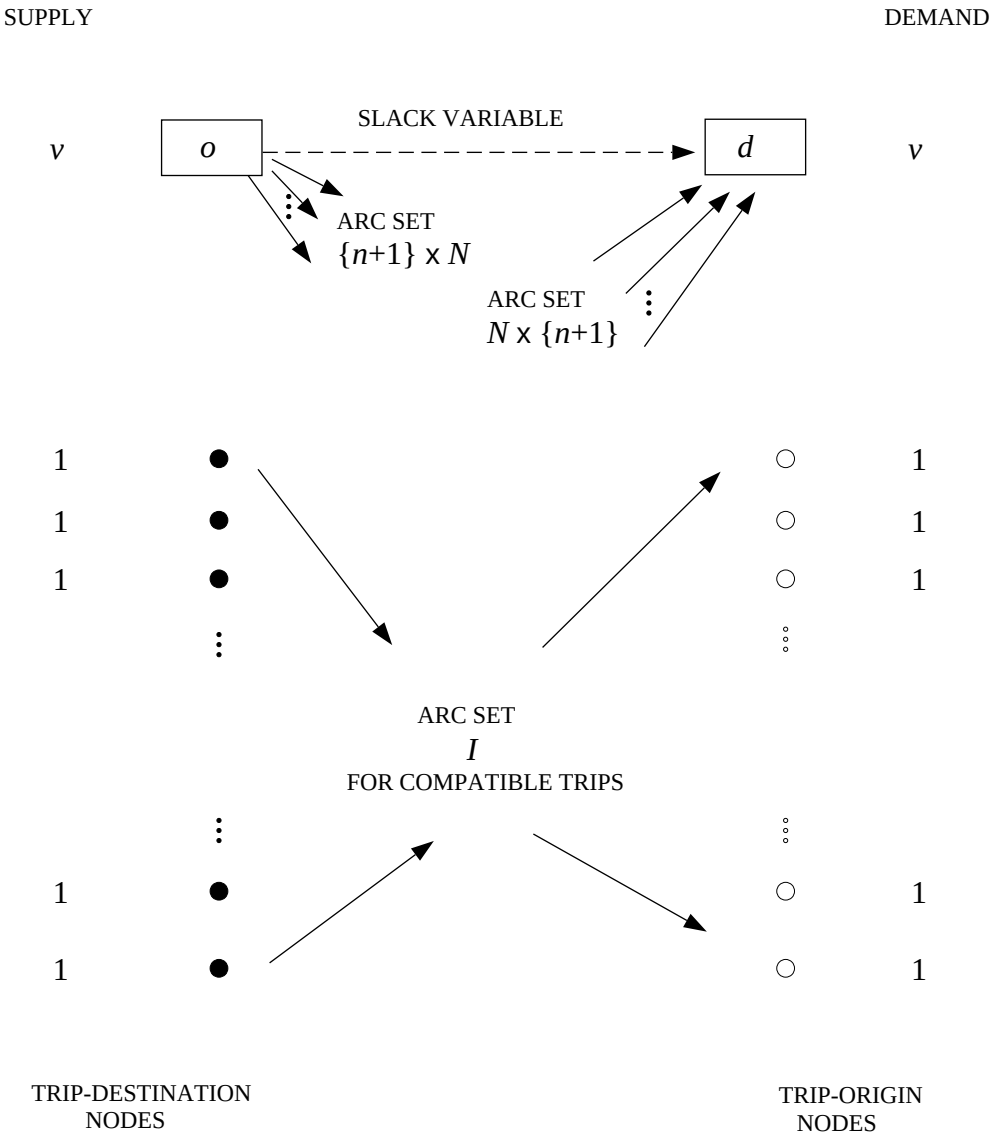


Figure 2.2: A transportation problem structure

an elementary chain. Determining the minimum number of vehicles needed to cover the set of trips is equivalent to determining the minimum number of chains which cover the nodes of (N, I). If the relation $(i, j) \in I$ defines a partial order over the set N, then the Dilworth Theorem (1950) on partially ordered sets asserts that this number is equal to the maximum number of mutually unrelated elements of N. For the construction of routes, under the relation $a_i + t_{ij} \leq a_j, i \in N, j \in N$, the elements of N form a partially ordered set. Therefore, two trips are unrelated if they cannot be covered by the same vehicle.

In any acyclic graph, the numbering of the nodes can be performed in such a way that if the arc (i, j) exists, then $i < j$. This numbering can be performed in $O(n^2)$ operations. Ford and Fulkerson (1962, pp. 64–65) have shown that the following algorithm generates the minimum number of chains: "*Assuming that the set has been numbered as stipulated above, select an undominated element (e.g., element 1), then proceed to its first (in terms of numbering) successor j, then to the first successor k of j, and so on until an element having no successor is reached. This traces out one chain of a minimal decomposition. Delete the elements of this chain and repeat the process.*" The computational complexity of this algorithm is $O(n)$.

Under the relation $(i, j) \in I$, the elements of N may not form a partially ordered set. For example, if long duration inter-trip connections are forbidden, the transitivity relations are not necessarily satisfied any more. Nevertheless, a minimal decomposition can be found by using the maximal flow algorithm (Ford and Fulkerson 1962, and Ahuja, Magnanti and Orlin 1989) which remains valid for acyclic graphs. The minimal fleet size covering all the trips is then given by $n - \sum_{(i,j)\in I} X_{ij}$. That is, the utilization of a number of inter-trips reduces the number of vehicles by the same amount. In the special case where the travel times t_{ij} are independent of j, they can be added to the service time for each trip $\mathcal{T}_i$ and the problem of finding the minimum number of vehicles reduces to the Fixed Job Schedule Problem (see Gertsbakh and Stern 1978, and also Fischetti, Martello and Toth 1987, 1989).

Network Flow Formulation (Revisited): The graphs used in the preceding sections enable the utilization of various specialized algorithms. Depending on the case, examples include the minimum cost flow, the minimum cost circulation, the transportation and the assignment algorithms, respectively. In addition, each application can also lead to the construction of a very specific and generally much smaller network. The time-space network of a fixed schedule problem involving only one depot and a single type of vehicle or individual is characterized by arcs corresponding to activities, specific to the context of each application. Furthermore, the quantity flowing on each arc can be constrained depending on the nature of the arc. Examples of such arcs include the case of a trip to be covered (in this

instance both the lower and the upper bounds are equal to one), of potential trips (here only the upper bound is equal to one), of diverse activities, such as periods of waiting, idle time, pauses for meals, and of travel.

The network type formulation which at the same time is the most general and the most simple is that where all the activities are on the arcs. In this case, each activity has an origin node and a destination node. For a graph defined by a set of nodes V and a set of arcs A, the structure of the costs $c_{ij}, (i,j) \in A$, and of the lower and upper bounds, ℓ_{ij} and u_{ij}, respectively, imposed on the variables X_{ij}, $(i,j) \in A$, completely characterizes a given application. Several examples are presented in the following section. The mathematical formulation of this network flow problem is then the following:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{2.6}$$

subject to:

$$\sum_{i\in V} X_{ij} - \sum_{i\in V} X_{ji} = 0, \qquad \forall\, j \in V \tag{2.7}$$

$$\ell_{ij} \le X_{ij} \le u_{ij}, \qquad \forall\, (i,j) \in A. \tag{2.8}$$

Relations (2.7) are the flow conservation equations at each node, while constraints (2.8) give the bounds on the X_{ij} variables. This is in fact a minimum cost circulation problem. The oriented graph (V, A) is no longer acyclic. We can however still consider that it is acyclic on a cylinder. This type of formulation allows the consideration of problems where a repetitive cyclic schedule is desired, that is where the initial and final conditions are identical but yet to be determined. The above formulation generalizes formulation (2.1)–(2.5). Any instance of (2.1)–(2.5) can be reformulated as (2.6)–(2.8) by appropriately defining the costs, setting the bounds ℓ_{ij} and u_{ij} to the value one on the trip-arcs, as well as imposing an upper bound for v on the returning arc assuring the circulation in the network and limiting the number of vehicles available. The advantages of this formulation will be clearly seen in the following section.

2.2 Time-Space Networks in Applications

The present section illustrates on several examples drawn from Desrosiers, Soumis and Desrochers (1982) how each application can give rise to the construction of a very specific and generally much smaller network. These networks are especially important in the more complex models: called several hundred times in relaxation or decomposition schemes, they permit a considerable acceleration of the solution time.

Aircraft Fleet Assignment: Consider a set of n flight legs. For each airport, a node is defined for each point in time where a flight leg starts or terminates. Next, for each flight leg define an arc connecting its departure node at one airport to its arrival node at a different airport. Finally, at each airport, ground arcs describing waiting are added to connect the nodes which are successive in time. The flight connection arcs exist only between the nodes at the same airport: such arcs describe a waiting time period and not a physical movement as is the case in road transportation.

When the initial conditions on the number of aircraft stationed at each airport are known, the network includes, for each airport, an origin-depot node which is connected to the node indicating the first departure. In the same manner, the node indicating the last arrival at each airport is connected with the destination-depot node. The final conditions are obtained by fixing the lower and upper bounds at the required value. If the initial and final conditions are imposed, in particular if they are the same as in a periodic schedule, ferries (empty flight legs) might be used to ensure feasibility of the problem.

Let us emphasize that the network constructed does not correspond to a multi-depot problem. In fact, even if we require that the number of aircraft available at each airport at the beginning of the planning period should equal that at the end of this period, it is however not required that the aircraft be the same. The multi-depot problems are formulated in the same manner as the problems with several types of vehicles by means of multi-commodity flow problems introduced in Section 2.3.

Consider next the case where the number of aircraft available at each airport at the beginning of the planning period is unknown and it is necessary that this number should be identical to that at the end of this period. To obtain a periodic solution, the depot nodes are eliminated and, at each airport, one arc, called a periodic arc, is directed from the last arrival node of the period considered to the first departure node of that period (Figure 2.3). The derived graph is acyclic on a cylinder and the problem to be solved becomes a minimum cost circulation problem. In this network, a lower and upper bound of value one is imposed on the flow of arcs corresponding to the flight legs that necessarily must be covered. In the case of potential flight legs, the lower bound is zero. On the waiting time arcs, it can frequently happen that the flows are larger than one, that is, several aircraft waiting at the same time at an airport. The cost structure is defined in terms of gains (or losses) associated with a necessary or potential flight leg. The number of aircraft is obtained by summing the total flow on all the arcs of the network at a given point in time. The size of this network is $2n$ nodes and $3n$ arcs, the degree of each node being 3 (an arc depicting a flight leg and two arcs depicting waiting time). Note that the size of the arc set is only $O(n)$ as compared to the potential $O(n^2)$ size.

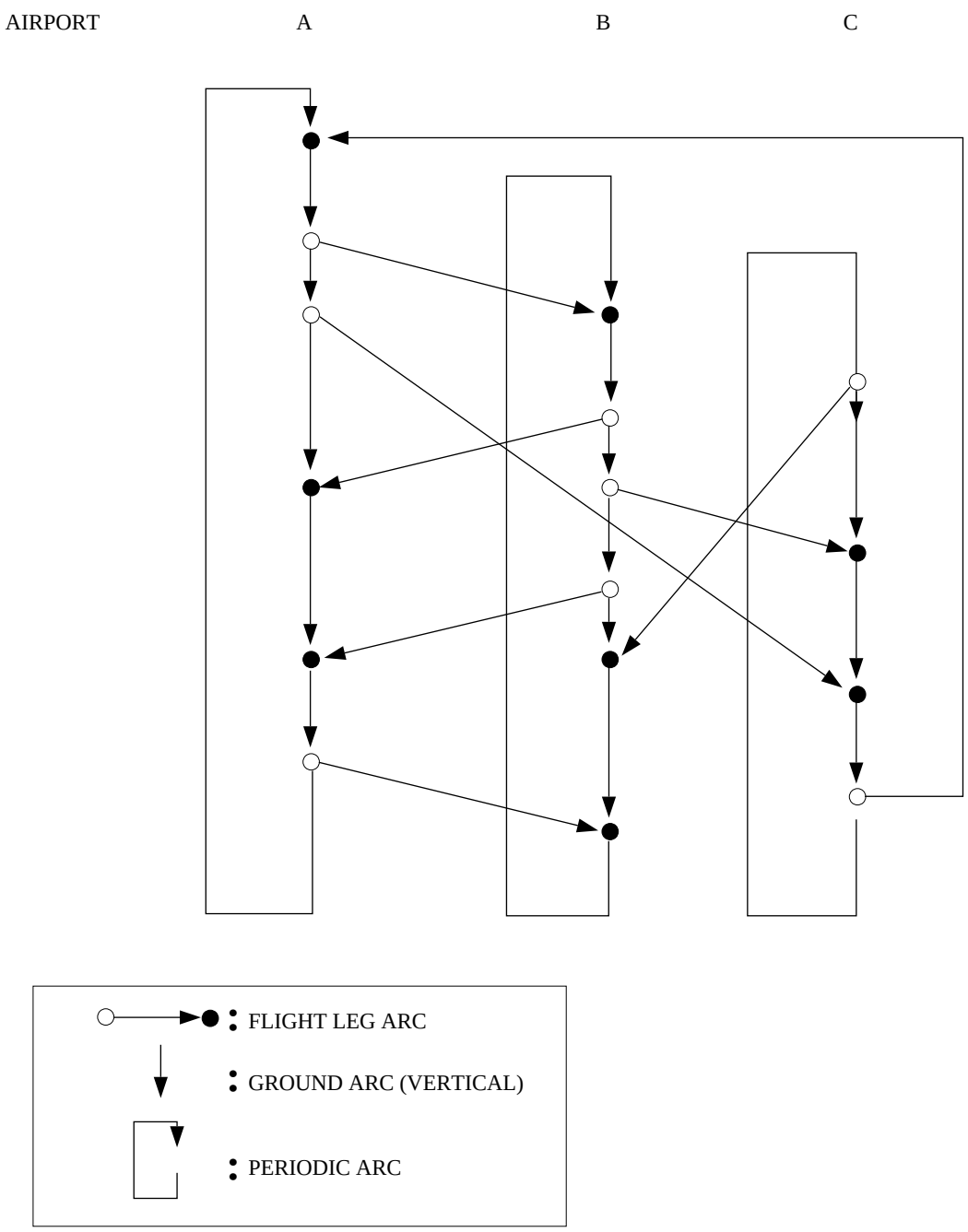


Figure 2.3: Time-space network for the Periodic Aircraft Assignment Problem

The size of this network can be reduced by grouping together some nodes corresponding to the same airport (Figure 2.4). At each airport, the nodes are examined in chronological order and a new group is started each time a flight arrival node is encountered, provided that in the current group there already exists a flight departure node. This aggregation rule ensures that a flight leaving an airport utilizes an aircraft already arrived at this airport.

To appreciate the size of the networks involved, consider the case of Air Canada described in Soumis, Ferland and Rousseau (1980). It consists of 325 flight legs and 300 potential flight legs. These give rise to an original time-space network which comprises 1250 nodes and 1875 arcs. The aggregation scheme groups together the nodes, on the average by sets of three, hence reducing the size of the network to 420 nodes and 1050 arcs.

Locomotive Assignment: Similar network transformations as above arise in rail transportation where engines are waiting at the same station. An additional distinctive aspect is to be noted: depending on the engine type, an origin-destination arc defining the trip of a train can have a lower and an upper bound different from the value one. On one hand, train cars may need to be hauled by several engines, while on the other hand it is possible to haul supplementary locomotives on that train to satisfy the demand in a neighboring city.

School-Bus Assignment: Consider the problem of assigning buses to school trips for the case involving students returning from school to their homes. The problem is to determine routes that start at the depot and cover a set of trips, each of which starts at a fixed time. There are no capacity constraints since each trip satisfies these by definition and vehicles moving between trips are empty. The essential feature of this application resides with the fact that each school is the departure point of a set of trips. Hence, the network contains only the n_S school-nodes which represent the origins of the trips and the n destination-nodes of the trips. The inter-trip arcs are defined between the destinations of the trips and the schools. Therefore, this network representation reduces the number of nodes of the school-bus network from $2n$ to $n + n_S$, while the number of arcs is reduced by approximatively a factor n_S/n. For a school-bus network of 273 trips and 70 schools, the number of inter-trip arcs is cut from 17,000 to less than 3000.

Urban-Bus Assignment: For this problem, the tasks are also bus trips. The inter-trip arcs correspond to two types of bus movements: 1) a bus going directly from a trip-destination node to a trip-origin node and incurring a bus/driver cost of travel and waiting,

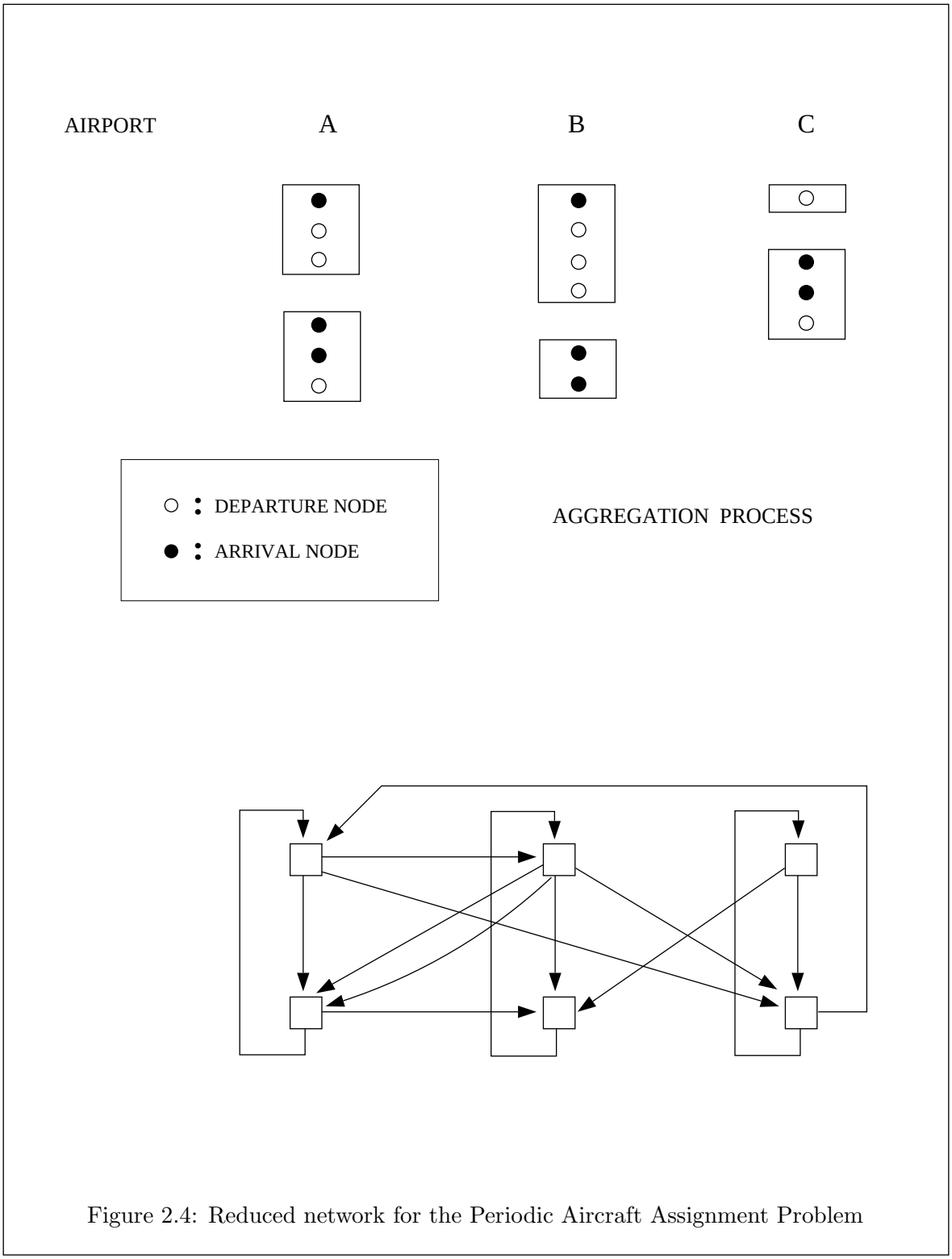


Figure 2.4: Reduced network for the Periodic Aircraft Assignment Problem

or 2) a bus returning to the depot for a driver rest period between two trips and incurring only a travel cost. The inter-trip arc chosen is the least costly of the two types. In the applications considered by Bökinge and Hasselström (1980) and by Soumis, Desrosiers and Desrochers (1985), the number of inter-trip arcs corresponding to buses returning to the depot is of the order of the square of the number of trips. In practice these arcs could correspond to returns to the depot after the morning peak hour and departures from the depot for the afternoon peak hour. A reduction in the number of arcs of this type is realized by creating new depot nodes, one for each potential bus departure and arrival time at the depot (Figure 2.5). Given n trips to be covered, $2n$ depot nodes are hence created, with only $2n$ waiting time arcs. In an example with $n = 128$ trips, the size of the original network comprises $2n+2 = 258$ nodes and $3n+|I| = 7085$ arcs, where the arcs belonging to I satisfy $a_i + t_{ij} \leq a_j,\ \ i, j \in N$. The transformed network comprises $4n = 512$ nodes and only 1360 arcs ($5n$ arcs + direct inter-trip arcs, i.e., $640 + 720$ arcs). An aggregation at the level of the $2n$ depot nodes identical to the one presented for aircraft fleet assignment can also be performed, reducing again the size of this network.

Note finally that the time-space network model for the single depot problem can be extended to a multi-commodity flow model for the multiple depot problem in a straightforward manner (Lamatsch 1992). An optimal solution approach to this problem is presented next.

2.3 The Multiple Depot Vehicle Scheduling Problem

Notation: The Multiple Depot Vehicle Scheduling Problem (MDVSP) is similar to the single depot version, except that now there is a set of depots K, the k^{th} depot housing v^k vehicles, $k \in K$. Furthermore, each vehicle used in the solution covers a sequence of pairwise compatible trips and must return to its depot at the end of the planning period. The vehicles have to be assigned to a set of fixed time-tabled trips so as to minimize the number of vehicles used and the overall operational cost. Let $N = \{1, 2, \ldots, n\}$ represent the set of trips. As previously defined for the SDVSP, let I denote the set of compatible trips. With each depot $k \in K$, we associate the graph $G^k = (V^k, A^k)$, where $n + k$ denotes the k^{th} depot, $V^k = N \cup \{n + k\}$, and $A^k = I \cup (\{n + k\} \times N) \cup (N \times \{n + k\})$. Let X^k_{ij} be the flow of type k through arc $(i, j) \in A^k$. The arc cost $c_{ij},\ \ (i, j) \in A^k$, is independent of k if $(i, j) \in I$, while $c_{n+k,j}$ for $j \in N$, and $c_{i,n+k}$ for $i \in N$, are usually dependent on k.

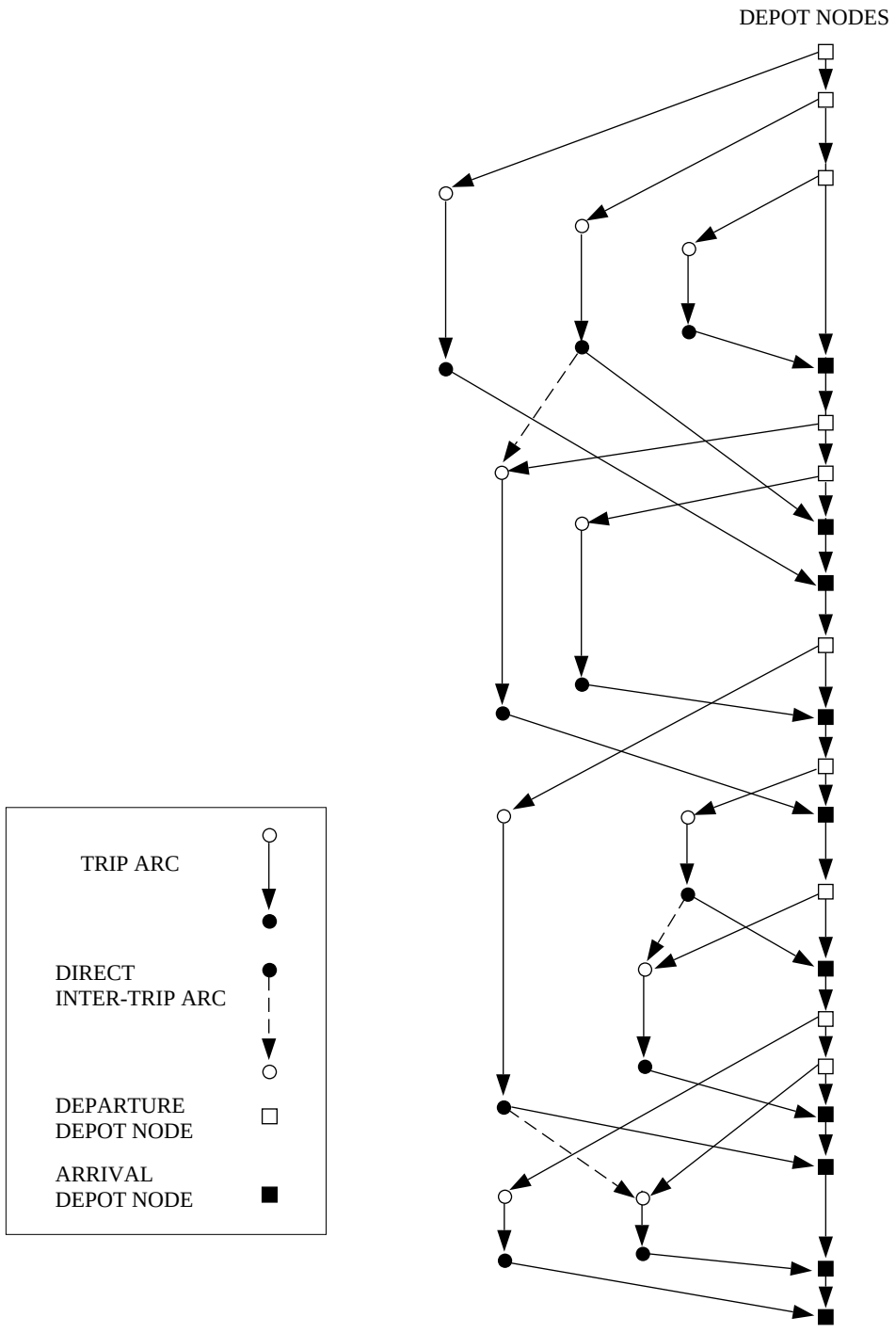


Figure 2.5: Time-space network for the Urban Bus Assignment Problem

Formulation: The MDVSP can be formulated as an integer multi-commodity flow problem (Ribeiro and Soumis 1994):

$$\text{Minimize} \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X_{ij}^k \tag{2.9}$$

subject to:

$$\sum_{k \in K} \sum_{j \in V^k} X_{ij}^k = 1, \qquad \forall\, i \in N \tag{2.10}$$

$$\sum_{j \in N} X_{n+k,j}^k \leq v^k, \qquad \forall\, k \in K \tag{2.11}$$

$$\sum_{i \in V^k} X_{ij}^k - \sum_{i \in V^k} X_{ji}^k = 0, \qquad \forall\, k \in K,\ \forall\, j \in V^k \tag{2.12}$$

$$X_{ij}^k \geq 0, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{2.13}$$

$$X_{ij}^k \quad \text{binary}, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k. \tag{2.14}$$

The objective function (2.9) seeks to minimize the total cost. Constraints (2.10) ensure that each trip is performed exactly once, while constraints (2.11) and (2.12) are fleet size and flow conservation constraints, respectively. The MDVSP has been shown to be NP-hard when $|K| \geq 2$ (Bertossi, Carraresi and Gallo 1987). As discussed in the previous section, the case $|K| = 1$ is solvable in polynomial time as a minimum cost network flow problem. When the costs $c_{n+k,j}$ and $c_{j,n+k}$ are independent of the depot k, $k \in K$, the problem reduces to the single depot case. In particular, the multi-depot minimum fleet size problem reduces to the single depot case. In fact, one has to only define the cost structure as follows: $c_{ij} = 0$ for all $(i,j) \in I$, $c_{n+k,j} = 1$ and $c_{j,n+k} = 0$, for all trips $j \in N$ and all depot $k \in K$. In general, solving the minimum fleet size problem is also a way to provide a feasible solution (if one exists) to the MDVSP in polynomial time.

Literature: Heuristic algorithms have been proposed by Bodin, Rosenfield and Kydes (1978), Ceder and Stern (1981), Smith and Wren (1981), El-Azm (1985), and Bertossi, Carraresi and Gallo (1987). Surveys on the subject can be found in Bodin and Golden (1981), Wren (1981), Bodin, Golden, Assad and Ball (1983), and Carraresi and Gallo (1984). Two new heuristics have also been published very recently by Lamatsch (1992) and Mesquita and Paixão (1992). Both are based on Lagrangean Relaxations of integer programming formulations.

Lamatsch (1992) has proposed a multi-commodity flow model equivalent to the formulation (2.9)–(2.14). The solution approach he used requires the sum over k of constraint set (2.12), yielding an additional set of constraints

$$\sum_{k\in K}\sum_{i\in V^k} X_{ij}^k - \sum_{k\in K}\sum_{i\in V^k} X_{ji}^k = 0, \quad \forall\, j \in N \cup \bigcup_{k\in K}\{n+k\}. \tag{2.12a}$$

The problem obtained from the relaxation of the multi-commodity flow conservation constraints (2.12) in the objective function may then be regarded as a single-commodity flow problem with new flow variables X_{ij} defined as $\sum_{k\in K} X_{ij}^k$. The resulting subproblem possesses the Integrality Property (see Section 5.5), so that the best bound obtained using this Lagrangean Relaxation is also equal to the linear relaxation value, i.e., as good as the bound provided by the Dantzig-Wolfe decomposition described below. For large problems however, the computation time reported is extremely high, as convergence is very slow. For this reason, a strategy is developed to modify the Lagrangean Relaxation solution in order to obtain a good heuristic solution for the original multi-commodity flow problem without going in a branch-and-bound exploration tree. Computational results are reported for three small real-world problems of a company running up to 250 trips from two depots.

The most recent polynomial time heuristic algorithm, which also always guarantees the use of the minimum number of vehicles, is due to Dell'Amico, Fischetti and Toth (1990). These authors report extensive computational results on test problems involving up to 1000 trips and 10 depots, showing that it outperforms other heuristics from the literature. Two exact algorithms have also been proposed. The first (Carpaneto, Dell'Amico, Fischetti and Toth 1989) is a branch-and-bound algorithm based on the computation of lower bounds by an additive scheme (see Fischetti and Toth 1989, for the theoretical exposition). The average integrality gap is relatively small, on the order of 0.9% on a set of randomly generated test problems, involving up to 70 trips and 3 depots. The second exact method (Ribeiro and Soumis 1994) is also a branch-and-bound algorithm. It is based on the solution of the linear relaxation of formulation (2.9)–(2.14) by a decomposition scheme (Dantzig and Wolfe 1960). Computational results are reported for problems generated similarly to the ones above but of size up to four to five times larger (up to $n = 100$ and $|K| = 10$, and $n = 250$ and $|K| = 6$). On these problems the average gap is only 0.0008%.

To explain the gap difference between the two methods, let Z_{IP} denote the optimal value for an integer programming problem. Denote by Z_{LP} the value of its linear relaxation. Briefly, the additive bounding procedure applied to the multi-depot vehicle scheduling problem uses a combination of an assignment bound and shortest path bounds. Let also Z_{ADD}

be the value of the lower bound provided by this additive bounding procedure. Finally, let Z_{DW} be the value of the lower bound obtained by the Dantzig-Wolfe decomposition presented next. Then the following result is obtained by Ribeiro and Soumis (1994):

$$Z_{ADD} \leq Z_{LP} = Z_{DW} \leq Z_{IP}. \tag{2.15}$$

This relation states that the value provided by the additive bounding procedure can be below that of the linear relaxation of the problem while the decomposition scheme used provides a bound which is equal to the LP bound. In practice, the value Z_{ADD} is very good. In theory the worst case ratio can be arbitrarily bad. Even under the symmetric cost assumption for the underlying travel network, the worst case ratio is greater or equal to 50% and this bound is tight.

Theoretical Aspects: The remaining part of this section exposes the decomposition principle applied to the MDVSP. The theory is based on Minkowski's theorem (Lasdon 1970, and Nemhauser and Wolsey 1988). Let $\mathcal{X} = \{X \in \mathbb{R}^n_+ | AX \leq b\}$ be a nonempty polyhedron defined by a finite set of constraints and lying within the non-negative orthant of real numbers. A point $x_p \in \mathcal{X}$ is defined to be an extreme point of $\mathcal{X}$ if there do not exist $X^1, X^2 \in \mathcal{X}$, $X^1 \neq X^2$, such that $x_p = 1/2X^1 + 1/2X^2$. If $\mathcal{X}(0) = \{X \in \mathbb{R}^n_+ | AX \leq 0\} \neq \{0\}$, then $X \in \mathcal{X}(0) \setminus \{0\}$ is called a ray of $\mathcal{X}$. A point $x_p \in \mathcal{X}$ is defined to be an extreme ray of $\mathcal{X}$ if there do not exist rays $X^1, X^2 \in \mathcal{X}(0) \setminus \{0\}$, $X^1 \neq \lambda X^2$, for any constant $\lambda > 0$, such that $x_p = 1/2X^1 + 1/2X^2$.

Then, a polyhedron $\mathcal{X}$ has a finite number of extreme points and extreme rays, and any point $X \in \mathcal{X}$ can be written as a convex combination of extreme points of $\mathcal{X}$ plus a non-negative linear combination of extreme rays of $\mathcal{X}$, i.e.,

$$\begin{aligned} & X = \sum_p x_p \theta_p, \\ \text{where} \quad & \sum_p \theta_p \delta_p = 1, \quad \theta_p \geq 0, \\ \text{and} \quad & \delta_p = \begin{cases} 1, & \text{if } x_p \text{ is an extreme point of } \mathcal{X} \\ 0, & \text{if } x_p \text{ is an extreme ray of } \mathcal{X}. \end{cases} \end{aligned}$$

A Dantzig-Wolfe Decomposition: The decomposition scheme is applied to the linear relaxation of the MDVSP obtained by dropping the integrality requirements (2.14). The master problem is given by (2.9)-(2.11). The subproblem involves a modified objective

function $\sum_{k\in K}\sum_{(i,j)\in A^k}\overline{c}_{ij}X_{ij}$, where the coefficients $\overline{c}_{ij}$ will be defined later. The subproblem set of constraints is defined by the flow conservation equations (2.12) over the non-negative X_{ij}^k variables (2.13).

The Subproblem: The subproblem constraints define a homogeneous system. Therefore the only extreme point of the subproblem is the null solution and any positive solution can be expressed as a non-negative linear combination of the extreme rays. Hence, the convexity constraint will not appear in the decomposition scheme. Let $(x_{ijp}^k, k\in K, (i,j)\in A^k), p\in\Omega$, be the set of extreme rays of the subproblem set of constraints. It will be shown later that these extreme rays may be defined by paths starting from some depot, covering a number of trips and returning to the same depot. To characterize an extreme ray it is then sufficient to send only one unit of flow on the corresponding path. Hence, the constant x_{ijp}^k takes only binary values.

Express next any solution X_{ij}^k to the master problem as a non-negative linear combination of the extreme rays, i.e.,

$$X_{ij}^k = \sum_{p\in\Omega} x_{ijp}^k\theta_p, \qquad \forall\, k\in K,\ \forall\,(i,j)\in A^k$$
$$\theta_p \geq 0, \qquad \forall\, p\in\Omega.$$

The Master Problem: Making the substitution into (2.9)–(2.11) and rearranging the summation order gives the revised formulation:

$$\text{Minimize}\sum_{p\in\Omega}\Big(\sum_{k\in K}\sum_{(i,j)\in A^k} c_{ij}x_{ijp}^k\Big)\theta_p \tag{2.16}$$

subject to:

$$\sum_{p\in\Omega}\Big(\sum_{k\in K}\sum_{j\in V^k} x_{ijp}^k\Big)\theta_p = 1, \qquad \forall\, i\in N \tag{2.17}$$

$$\sum_{p\in\Omega}\Big(\sum_{j\in N} x_{n+k,jp}^k\Big)\theta_p \leq v^k, \qquad \forall\, k\in K \tag{2.18}$$

$$\theta_p \geq 0, \qquad \forall\, p\in\Omega. \tag{2.19}$$

Let the cost coefficients of the objective function be denoted by $c_p, p\in\Omega$. The costs c_p are the sum of the costs of the arcs used on the path. Therefore,

$$c_p = \sum_{k\in K}\sum_{(i,j)\in A^k} c_{ij}x_{ijp}^k. \tag{2.20}$$

Since each node is visited at most once by a path, define

$$a_{ip} = \sum_{k \in K} \sum_{j \in V^k} x^k_{ijp}, \tag{2.21}$$

a binary constant taking value 1 if path p visits node i. Finally define

$$b^k_p = \sum_{j \in N} x^k_{n+k,jp}, \tag{2.22}$$

another binary constant taking the value 1 only if path p starts from depot $k \in K$. The resulting master problem is then given by:

$$\text{Minimize} \sum_{p \in \Omega} c_p \theta_p \tag{2.23}$$

subject to:

$$\sum_{p \in \Omega} a_{ip} \theta_p = 1, \qquad \forall\, i \in N \tag{2.24}$$

$$\sum_{p \in \Omega} b^k_p \theta_p \leq v_k, \qquad \forall\, k \in K \tag{2.25}$$

$$\theta_p \geq 0, \qquad \forall\, p \in \Omega. \tag{2.26}$$

Linear Relaxation Solution: The above formulation defines the linear relaxation of a set partitioning problem with availability constraints for each depot. Given an optimal solution to the restricted master problem on a subset of variables $\Omega' \subset \Omega$, let $\alpha_i, i \in N$ and $\beta_k, k \in K$, be the dual variables associated with the covering of the trips $i \in N$ and the use of a vehicle from a depot $k \in K$, respectively. A new minimum marginal cost column is generated by solving:

$$\underset{p \in \Omega - \Omega'}{\text{Minimize}} \quad c_p - \sum_{i \in N} \alpha_i a_{ip} - \sum_{k \in K} \beta_k b^k_p. \tag{2.27}$$

The constants c_p, a_{ip} and b^k_p are unknown on $\Omega - \Omega'$. If a negative marginal cost column exists, it is added to the current restricted master problem. Since there is no negative marginal cost columns on the subset Ω' (the current solution is optimal for the restricted master), optimization (2.27) can be performed on Ω. Constants c_p, a_{ip} and b^k_p can be represented by equations (2.20)–(2.22) which define them, expressed in terms of the original decision variables X^k_{ij}. These variables must satisfy the subproblem constraints (2.12) and (2.13). Hence, minimization (2.27) can be reformulated as:

$$\text{Minimize} \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X^k_{ij} - \sum_{i \in N} \alpha_i \Big(\sum_{k \in K} \sum_{j \in V^k} X^k_{ij} \Big) - \sum_{k \in K} \beta_k \Big(\sum_{j \in N} X^k_{n+k,j} \Big) \tag{2.28}$$

subject to: (2.12)–(2.13).

The set of arc A^k, $k \in K$, can be partitioned the following way: (i,j), with $i \in N$ and $j \in V^k$, or $(n+k,j)$, with $k \in K$ and $j \in N$. Hence the objective function can be rewritten as

$$\sum_{k\in K}\sum_{i\in N}\sum_{j\in V^k}(c_{ij}-\alpha_i)X_{ij}^k - \sum_{k\in K}\sum_{j\in N}(c_{n+k,j}-\beta_k)X_{n+k,j}^k. \tag{2.29}$$

It follows that the marginal cost of an arc $(i,j) \in A^k$, $k \in K$ is given by:

$$\overline{c}_{ij} = \begin{cases} c_{ij}-\alpha_i, & \text{if } i \in N \\ c_{ij}-\beta_k, & \text{if } i = n+k, \quad k \in K\ . \end{cases} \tag{2.30}$$

Then, minimization (2.27) simply becomes:

$$\begin{aligned} \text{Minimize} \quad & \sum_{k\in K}\sum_{(i,j)\in A^k}\overline{c}_{ij}X_{ij}^k \\ \text{subject to:} \quad & (2.12)-(2.13). \end{aligned} \tag{2.31}$$

If a negative marginal cost solution exists, then, by the flow conservation equations, it is defined by a number of negative marginal cost paths. We can choose one path or any number of paths from any number of different depots. The way to characterize an extreme ray is to choose only one path. Then, one unit of flow is sent on it to define the extreme ray. Note that the subproblem is separable by depot, and that for each depot, the problem solution amounts to a shortest path, on an acyclic graph, which admits an easy solution. The algorithmic complexity is only $O(n^2)$ for a n-node problem (Fulkerson 1972).

The solution process entails alternating between the restricted master problem and the subproblem. The procedure is started with an artificial set of columns for the restricted master problem which forms the initial identity basis. Each time the restricted master problem is solved on a subset of columns, the dual variables are updated and used for the subproblem solution. Then, to accelerate the process, each time the generating phase is called, every depot may provide several path columns. The column generation procedure terminates when no more negative marginal cost columns exist for any depot. The process then gives the value Z_{LP}, i.e., the solution value of problem (2.9)–(2.13), which is the linear relaxation of the MDVSP.

Optimal Integer Solution: To obtain an optimal integer solution to the original problem (2.9)–(2.14), cuts and branch-and-bound strategies are used, if necessary. An integer solution is found if constraints (2.14), i.e., X_{ij}^k binary, $\forall k \in K$, $\forall (i,j) \in A^k$, are all satisfied. Initially, one needs to check if all θ_p variables take binary values. This is a sufficient condition since, in this case, the solution decomposes into a set of disjoint paths. Otherwise, one has to evaluate if

$$X_{ij}^k = \sum_{p \in \Omega} x_{ijp}^k \theta_p \text{ binary}, \quad \forall\, k \in K,\, \forall\, (i,j) \in A^k \tag{2.32}$$

are all satisfied. This is accomplished by using only the fractional θ_p variables in the optimal basis, i.e., $p \in \Omega$ such that $0 < \theta_p < 1$.

If the solution is fractional, the value Z_{LP} can be rounded up to the next integer, if all coefficients c_{ij} are integer. In other words, the following constraint is added to the linear relaxation (2.9)–(2.13):

$$\sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij} X_{ij}^k \geq \lceil Z_{LP} \rceil. \tag{2.33}$$

Branching can be performed on the number of vehicles used at a depot, i.e., on $\sum_{j \in N} X_{n+k,j}^k$, for a given k. It may also be done on a specific binary flow variables X_{ij}^k, or on a subset of flow variables like $\sum_{k \in K} X_{ij}^k$. Branching decisions to set these variables at zero or at one define new arc sets for each subproblem. If new constraints are added to the linear relaxation (2.9)–(2.13), the decomposition process can be applied as described before so that new constraints appear in the master problem definition. Otherwise, branching decisions are easily transferred to the subproblem structure (elimination of arcs, dual variables) so that only feasible negative marginal cost columns are generated. The column generation scheme is then applied at each node of the branch-and-bound tree to obtain the overall optimal solution. Note that while solving (2.23)–(2.26) by a simplex algorithm, if a feasible integer solution is found, it is recorded and whenever a better solution is encountered, the previous best solution is updated. The experimental behavior of the algorithm is described in more details in Ribeiro and Soumis (1994).

Additional details on the solution of integer programs using Dantzig-Wolfe decomposition are provided in Sections 5, 6 and 7. We can nevertheless offer here some insights into branching schemes for integer programs solved by decomposition methods, including column generation. If the formulation (2.9)–(2.14) is solved directly, branching decisions will naturally be taken on the original variables (see for example Forbes, Holt and Watts 1991). This is also true should the problem be solved by Lagrangean Relaxation where no new variables are defined. Given that Lagrangean Relaxation and Dantzig-Wolfe decomposition are equivalent dual and primal methods, respectively (see Magnanti, Shapiro and Wagner 1976), this suggests that in a decomposition approach, branching should be performed on the original variables, even if new variables have been introduced for extreme points or rays.

3 The Traveling Salesman Problem with Time Windows

The traveling salesman problem (TSP) with time windows (TSPTW) consists of finding the minimum cost path to be traveled by a vehicle, starting and returning at the same depot, which must visit a set of nodes exactly once. The service at a node must begin within the time window defined by the earliest time and the latest time when the start of service is allowed at that node. A vehicle is not permitted to arrive at a node after the latest time to begin service. However, if a vehicle arrives too early at a node, it is permitted to wait until the node is ready for the beginning of service. The TSPTW is a time-constrained vehicle routing problem involving only one uncapacitated vehicle.

This type of problem is encountered in a variety of industrial and service sector applications. Examples include bank deliveries, postal deliveries and school-bus routing and scheduling. The TSPTW constitutes an important element of more complex vehicle routing problems. In particular, for *cluster-first, route second* approaches to such problems, if the clustering phase is performed heuristically, the a posteriori optimization of the TSPTW in each cluster is necessary. This type of problem has also notable applications in manufacturing. In automated manufacturing systems, for example, an automated guided vehicle must visit a set of flexible machines. To achieve full machine utilization, the automated guided vehicle must visit each machine within the time window determined by the processing time of the job currently being machined.

3.1 Problem Formulation

Notation: Consider a set of nodes $N = \{1, \ldots, n\}$ to be visited and the additional single depot. Next, duplicate the depot into an origin-depot o and a destination-depot d. Let $V = N \cup \{o, d\}$. Next, associate with each node $i \in V$ a time window $[a_i, b_i]$. The constant a_o is the earliest starting time of the vehicle from the depot, while b_d is the latest arrival time of the vehicle to the depot. Denote by A the set of arcs and associate with each arc a non-negative duration t_{ij} and a cost c_{ij}. We assume next that the service time at node i is included in the time value t_{ij}, for all $i \in N \cup \{o\}$. Finally, an arc (i, j) is defined in the set A if the following condition holds:

$$a_i + t_{ij} \leq b_j, \qquad i \in N \cup \{o\}, \quad j \in N \cup \{d\}, \quad i \neq j. \tag{3.1}$$

Formulation: The mathematical programming formulation given below involves two types of variables: the set of binary flow variables $X = (X_{ij},\ (i,j) \in A)$ and the set of time variables $T = (T_i,\ i \in V)$. The variable X_{ij} is equal to 1, if arc (i,j) is used in the optimal tour, and is set at 0 otherwise. Given that waiting at a node before the specified time window is permitted, the variable T_i specifies the start of service at node i, $i \in N \cup \{o\}$, while the variable T_d gives the arrival time at the destination-depot node.

The minimal cost tour starting within the time interval $[a_o, b_o]$ at the origin-depot node, visiting exactly once all the nodes of N within their time windows, and finishing at the destination-depot node before the time limit b_d, can be formulated as follows:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{3.2}$$

subject to:

$$\sum_{j \in N \cup \{d\}} X_{ij} = 1, \qquad \forall\, i \in N \tag{3.3}$$

$$\sum_{j \in N} X_{o,j} = 1, \tag{3.4}$$

$$\sum_{i \in N \cup \{o\}} X_{ij} - \sum_{i \in N \cup \{d\}} X_{ji} = 0, \qquad \forall\, j \in N \tag{3.5}$$

$$\sum_{i \in N} X_{i,d} = 1, \tag{3.6}$$

$$X_{ij}(T_i + t_{ij} - T_j) \le 0, \qquad \forall\, (i,j) \in A \tag{3.7}$$

$$a_i \le T_i \le b_i, \qquad \forall\, i \in V \tag{3.8}$$

$$X_{ij} \ge 0, \qquad \forall\, (i,j) \in A \tag{3.9}$$

$$X_{ij} \text{ binary}, \qquad \forall\, (i,j) \in A. \tag{3.10}$$

This formulation closely replicates the fixed schedule formulation introduced in Section 2. It is a nonlinear program with $O((n+2)^2)$ variables and $O((n+2)^2)$ constraints, where the objective function (3.2) represents the total cost. The subproblem defined by (3.2)–(3.5) and (3.9) is a minimum cost flow problem (in fact an Assignment Problem with $(n+2)$ rows and $(n+2)$ columns). Constraints (3.7)–(3.8) ensure feasibility of the time schedule.

Subtours are eliminated by constraints (3.7). Using a big M constant, these constraints can be linearized and rewritten as follows (Desrosiers, Pelletier and Soumis 1983, and Solomon 1983):

$$T_i + t_{ij} - T_j \le M(1 - X_{ij}), \qquad \forall\, (i,j) \in A. \tag{3.7a}$$

They represent a generalization of the classical TSP subtour elimination constraints proposed by Miller, Tucker and Zemlin (1960). Note that the large constant M can be replaced by $M_{ij} = \max\{b_i + t_{ij} - a_j, 0\}$, $(i,j) \in A$. When $b_i + t_{ij} \leq a_j$, these constraints are always satisfied for all values of T_i, T_j and X_{ij}. Then, the formulation requires the presence of constraints (3.7) or (3.7a) only for arcs $(i,j) \in A$, such that $M_{ij} > 0$.

Nonlinear Integrality Property: The nonlinear formulation (3.2)–(3.10) has also an interesting property. If the problem is feasible, integrality requirements (3.10) can be eliminated from the above formulation (Desrosiers, Soumis and Desrochers 1984). To show that this can be done, let (X^*, T^*) be an optimal solution. If X^* is not integer, define $A^* = \{(i,j) \in A | X^*_{ij} > 0\}$. Now fix T^*, and consider the minimum cost Assignment Problem on the subnetwork obtained by retaining only the arcs of A^* for which there exists an integer optimal solution $\overline{X}$. It is then easy to verify that $(\overline{X}, T^*)$ is an optimal integer solution to (3.2)–(3.9). Note however that the linearized formulation containing (3.7a) does not necessarily provide an integer solution. Hence, constraints (3.10) must be retained in the linearized formulation.

Literature: Research on the TSPTW has been scant. Savelsbergh (1985) has proved that even finding a feasible solution to the TSPTW is an NP-complete problem. Therefore, the author has proposed heuristic algorithms based on the k-interchange concept (Lin 1965, and Lin and Kernighan 1973). This involves the replacement of k arcs currently on the TSP tour with k other arcs. For the TSP, the processing of a single k-interchange can be performed in constant time, for any k. This is because one has to only examine the savings derived from such an interchange. For the TSPTW, however, the treatment of a k-interchange could require $O(n)$ time. Such an interchange may shift the time to start the service at a node, which in turn could alter the start of service at all the other nodes on the tour. Therefore, in addition to its profitability one has to also test the feasibility of an interchange. Savelsbergh (1985) shows that for 2-interchanges and the restricted 3-interchanges proposed by Or (1976), the processing of a single interchange can still be executed in constant time for the objective of minimizing the total travel time. Savelsbergh (1988, 1989) has extended this result to the case of minimizing the total schedule time. Extensive research has been conducted on heuristics for more complex routing problems and this will be discussed in Sections 5 and 6.

Despite the difficulty of the TSPTW, several other authors have focused on exact algorithms to minimize the *total cost* or the *total schedule time*. Objective (3.2) describes the former case; minimizing the value of $T_d - T_o$ is appropriate for the latter case. For this objective, it is also easy to define waiting time variables $W_i \geq 0$, $i \in N \cup \{d\}$ at each node

to be serviced and the destination-depot node, introduce them in constraints (3.7) or (3.7a) and replace the inequality sign by an equality sign, and rewrite the objective function as:

$$\text{Minimize } T_d - T_o = \sum_{(i,j)\in A} t_{ij} X_{ij} + \sum_{i \in N \cup \{d\}} W_i \qquad (3.2a)$$

Under special conditions, Baker (1983) introduces another model for the minimization of the total schedule time. His model as well as the solution process are discussed in Section 3.5.

The formulation (3.2)–(3.10) or the model with the objective function (3.2) replaced by (3.2a) can be used in a branch-and-bound scheme to determine an optimal time window constrained tour. For example, one can relax constraints (3.7) and (3.8) and use the lower bound provided by the Assignment Problem solution. Branching on the X_{ij} variables as do Carpaneto and Toth (1980) for the asymmetric TSP may provide a very good algorithm. Considering the linearized version, another lower bound is given by the solution of the linear relaxation. This bound is at least as good as the assignment bound, yet much more time-consuming to calculate. If the graph is dense, it is expected to be very close to the assignment bound (see Section 5.2).

Many other relaxations or decompositions can be designed using the previous formulations. In any search tree, branching on the flow variables X_{ij} $(i,j) \in A$ is easily implemented. However when the most important component of a specific TSPTW application involves the time window restrictions, branching on the time variables T_i is definitely much more appropriate. This is also true for any time-constrained vehicle routing problem solved by branch-and-bound.

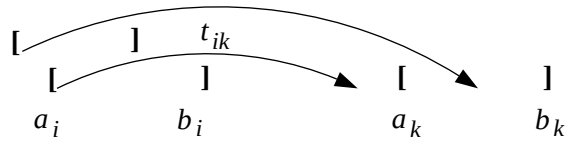
3.2 Time Window Reduction and Arc Elimination

The complexity of the algorithms proposed for time window constrained routing and scheduling problems is a function of the width of the time windows. To improve the efficiency of these algorithms, the time windows' width can be reduced a priori by applying several rules described in Cyrus (1988) and in Desrochers, Desrosiers and Solomon (1992). These rules, which are valid for both objective functions (i.e., the minimization of the total cost or the total schedule time), are presented in Figure 3.1.

At each node, an attempt is made to reduce the time window width by testing conditions (3.11)–(3.14) in order. Arcs which become infeasible are removed from the arc set A. After examining all the nodes, a new iteration is started. In practice, after two or three iterations no further reductions are possible.

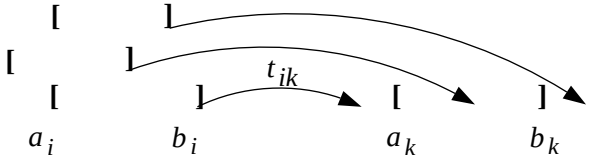
LOWER BOUND ADJUSTMENT DERIVED FROM THE EARLIEST ARRIVAL TIME AT NODE k FROM ITS PREDECESSORS, FOR $k \in N \cup \{d\}$:

$$a_k := \max\{a_k, \min_{(i,k)\in A}\{a_i + t_{ik}\}\} \tag{3.11}$$



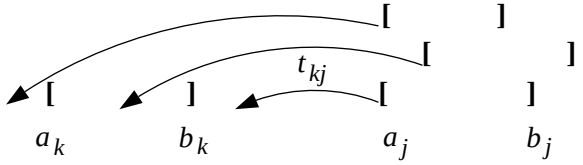
UPPER BOUND ADJUSTMENT DERIVED FROM THE LATEST ARRIVAL TIME AT NODE k FROM ITS PREDECESSORS, FOR $k \in N \cup \{d\}$:

$$b_k := \min\{b_k, \max\{a_k, \max_{(i,k)\in A}\{b_i + t_{ik}\}\}\}; \tag{3.12}$$



LOWER BOUND ADJUSTMENT DERIVED FROM THE EARLIEST DEPARTURE TIME FROM NODE k TO ITS SUCCESSORS, FOR $k \in N \cup \{o\}$:

$$a_k := \max\{a_k, \min\{b_k, \min_{(k,j)\in A}\{a_j - t_{kj}\}\}\}; \tag{3.13}$$



UPPER BOUND ADJUSTMENT DERIVED FROM THE LATEST DEPARTURE TIME FROM NODE k TO ITS SUCCESSORS, FOR $k \in N \cup \{o\}$:

$$b_k := \min\{b_k, \max_{(k,j)\in A}\{b_j - t_{kj}\}\}. \tag{3.14}$$

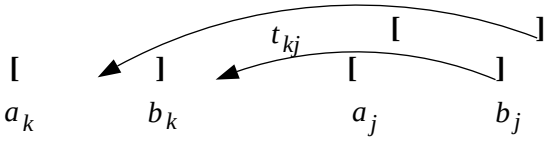


Figure 3.1: Time Window Reduction

Arc Elimination for the TSPTW: Additional computational efficiency can be gained by reducing the set of arcs for the TSPTW by applying the rules developed by Langevin, Desrochers, Desrosiers and Soumis (1993). To present these conditions, let $EAT(i,j)$ be the earliest arrival time at node j from node i. This can be computed by solving a shortest time path problem (including waiting times) starting at time a_i and satisfying the time window constraints from node i to node j. This can be calculated using dynamic programming (see Section 4).

For arc (i,j) to exist it should be possible to visit node k either before node i or after node j, for all $k \in N$. Therefore, arc (i,j) can be eliminated if $\exists k \in N$ such that

$$EAT(k,i) > b_i \quad \text{and} \quad EAT(j,k) > b_k. \tag{3.15}$$

Note that when the depot is node j (node i), only the former (latter) condition needs to be tested. Rule (3.15) can even be strengthened. Let $LTD(i,j)$ be the latest departure time from i such that the time to begin service at j is feasible. This can also be computed by solving a shortest time path problem (including waiting times) by using the reverse arc directions and starting at time b_j at node j. Let $k \rightsquigarrow i$ represent nodes k and i linked by a path, or the single arc (k,i) in the degenerate case. Then, arc (i,j) can be eliminated if there exists a node k such that the two sequences $k \rightsquigarrow i \rightsquigarrow j$ and $i \rightsquigarrow j \rightsquigarrow k$ are not feasible, i.e., if $\exists k \in N$ such that

$$EAT(k,i) > LDT(i,j) \quad \text{and} \quad EAT(i,j) > LDT(j,k). \tag{3.16}$$

Time Window Reduction for the TSPTW: Gélinas (1993) suggests specific time window reductions to be performed for the TSPTW. Let $BEFORE(k)$ be the set of all nodes that must necessarily be visited before k. Therefore, $BEFORE(k) = \{i \in N | EAT(k,i) > b_i\}$. Similarly define $AFTER(k) = \{j \in N | EAT(j,k) > b_k\}$, the set of all nodes that must necessarily be visited after k. Given these sets, the adjustment relations (3.11) and (3.14) can be strengthened. Another lower bound adjustment can then be derived from the earliest arrival time at node k from nodes only in $BEFORE(k)$, for $k \in N \cup \{d\}$:

$$a_k := \max\{a_k, \max_{i \in BEFORE(k)} \{EAT(i,k)\}\}. \tag{3.11a}$$

Symmetrically, another upper bound adjustment can be derived from the latest departure time from node k to nodes in $AFTER(k)$, for $k \in N \cup \{o\}$:

$$b_k := \min\{b_k, \min_{j \in AFTER(k)} \{LDT(k,j)\}\}. \tag{3.14a}$$

For the TSPTW, preprocessing involves time window reduction and arc elimination procedures until no further modifications are possible. Finally, note that the values $EAT(i,j)$ and $LDT(i,j)$ can be overestimated and underestimated, respectively, by using unconstrained shortest time paths. This is much faster and it gives satisfactory results in practice. For the rest of this section we assume that A represents the set of arcs after time window reduction and arc elimination.

The next sections focus on three optimal branch-and-bound approaches (Sections 3.3, 3.4 and 3.6) and on a fourth approach (Section 3.5) which solves the problem directly. The first two methods solve the problem of minimizing the total schedule time (Baker 1983, Christofides, Mingozzi and Toth 1981b). The last two approaches are suitable for both types of objective function. The third algorithm is a very powerful dynamic programming algorithm (Dumas, Desrosiers, Gélinas and Solomon 1991). The fourth method is presented not for its numerical performance, but rather for its original formulation involving only $O(n^2)$ variables and $O(n)$ constraints in the linear relaxation (Langevin, Desrochers, Desrosiers and Soumis 1993).

3.3 A Time Constrained Critical Path Approach

In this subsection, it is assumed that the time matrix t_{ij}, $i,j \in V$ is symmetric and that the triangle inequality holds. Under these conditions, Baker (1983) presents a branch-and-bound algorithm for the following time oriented formulation of the TSPTW:

$$\text{Minimize} \quad T_d - T_o \tag{3.17}$$

subject to:

$$T_i \geq T_o + t_{o,i}, \qquad \forall\, i \in N \tag{3.18}$$

$$|T_i - T_j| \geq t_{ij}, \qquad 1 \leq i < j \leq n \tag{3.19}$$

$$T_d - T_i \geq t_{i,d}, \qquad \forall\, i \in N \tag{3.20}$$

$$a_i \leq T_i \leq b_i, \qquad \forall\, i \in V. \tag{3.21}$$

The solution process also imposes a third condition which specifies that the tour starts at a fixed time value, i.e., $a_o = b_o = T_o$. As discussed later, allowing $b_o - a_o > 0$ implies an additional degree of complexity.

The approach involves constructing a related acyclic network. This network has the same set of nodes V. The arcs in the network are constructed as follows. Given any two

nodes i and j, if the time window constraints impose a precedence relationship between these two nodes, say $T_i < T_j$ if $a_j + t_{ji} > b_i$, then this relationship is represented by an arc. The constraints (3.19) which do not create precedence relationships are relaxed.

In order to minimize the total schedule time $T_d - a_o$, the critical path, starting at time a_o from the origin-depot node and returning to the destination-depot node at the end of the tour, is computed on the related acyclic network as a longest path with time window constraints. Given that the starting time of the critical path is fixed, the computational complexity of such an algorithm is $O(n^2)$ for a n-node network (see Section 4.5). This critical time path provides a lower bound in the search tree. Branching creates two subproblems corresponding to the case where arrival at node i precedes the arrival at node j, and vice versa. Therefore the branching decisions are taken on time variables. The best solution gives the minimal value of $T_d - a_o$. The optimal tour can be found on the corresponding acyclic network of the search tree. In this case, a maximum cardinality path labeling convention must be adopted for the longest path algorithm to resolve ties in the triangle inequality. Otherwise, paths of equal schedule time to the critical path (the optimal tour) but containing less than $n+1$ arcs would be allowed. Baker (1983) reports good performance for the algorithm on 50-node problems where only a small percentage of time windows overlap.

Note: Baker (1983) presented the above algorithm to find the minimum schedule time for the TSPTW under several conditions including the fixed starting time of the tour. If waiting at the origin-depot node is feasible within the time window $[a_o, b_o]$ for $b_o > a_o$, then a specific critical path length must be calculated for each possible starting time value, i.e., for $D_o = b_o - a_o + 1$ values in the case of integer values, for all the time parameters. The optimal critical path length is then given by the smallest value. For a n-node problem, the computational complexity of the algorithm becomes $O(n^2 D_o)$.

An alternative way to solve the critical path problem with time windows when the starting time is flexible at the origin-depot node is to use a two-dimensional labeling procedure. Such labeling procedures for the shortest path problem with time windows and dominance criteria which permit label elimination at a node are discussed in the next section. The first dimension of this two-dimensional labeling represents the start of service at a node within the time interval. The second dimension represents the negative of the elapsed schedule time (the duration). Additional comments on the critical path problem with time windows are given in Section 4.5.

3.4 State-Space Relaxation

State-space relaxation is a general relaxation procedure proposed by Christofides, Mingozzi and Toth (1981b) for a number of routing problems. The motivation for this methodology stems from the fact that very few combinatorial optimization problems can be solved by dynamic programming alone due to the dimensionality of their state-space. To overcome this difficulty, the number of states is reduced by mapping the state-space associated with a given dynamic programming recursion to a smaller cardinality space. This mapping, denoted by g, must associate to every transition from a state S_1 to a state S_2 in the original state space, a transition $g(S_1)$ to $g(S_2)$ in the new state-space. To be effective, the function g must give rise to a transformed recursion over the relaxed state-space which can be computed in polynomial time. Furthermore, this relaxation must generate a good lower bound for the original problem. This lower bound could then be used in a branch-and-bound algorithm for the solution of the original problem.

We present this approach in the context of the minimization of the total schedule time for the TSPTW. Assume that the time-constrained path starts at the fixed time value a_o. Define $F(S,i)$ as the shortest time it takes for a feasible path starting at node o, passing through every node of $S \subseteq N$ exactly once, to end at node $i \in S$. Note that the optimization of the total arc cost would involve an additional dimension to account for the arrival time at a node (see Section 3.5). The function $F(S,i)$ can be computed by solving the following recurrence equations:

$$F(S,j) = \min_{(i,j)\in A} \{F(S-\{j\},i) + t_{ij} | i \in S-\{j\}\} \quad \forall\, S \subseteq N, j \in S. \tag{3.22}$$

The recursion formula is initialized by

$$F(\{j\},j) = \begin{cases} \max\{a_j, a_o + t_{o,j}\} & \text{if } (o,j) \in A, \\ \infty, & \text{otherwise.} \end{cases} \tag{3.23}$$

The optimal solution to the TSPTW is given by:

$$\min_{j\in N} \{F(N,j) + t_{j,d}\}. \tag{3.24}$$

Note that equation (3.22) is valid if $a_j \leq F(S,j) \leq b_j$. If however $F(S,j) < a_j$, then $F(S,j) = a_j$; if $F(S,j) > b_j, F(S,j) = \infty$. Formulas (3.22)–(3.24) define a shortest path algorithm on a state graph whose nodes are the states (S,i) and whose arcs represent transitions from one state to another. This algorithm is a forward dynamic programming algorithm where at step s, $s = 1,\ldots,n+1$, a path of length s is generated. The states (S,i) of cost $F(S,i)$ are defined as follows: S is an unordered set of visited nodes and i is the last visited node, $i \in S$.

Christofides, Mingozzi and Toth (1981b) suggest several alternatives for the mapping g. We present here the shortest q-path relaxation, i.e., $g(S) = q = \sum_{i \in S} q_i$, where $q_i \geq 1$ is an integer associated with node $i \in N$; then $g(S - \{j\}) = g(S) - q_j$. Define $Q = \sum_{i \in N} q_i$. Hence the transformed recursion equations are:

$$F(q,j) = \min_{(i,j) \in A} \{F(q - q_j, i) + t_{ij} | q - q_j \geq q_i\}, \qquad q \in \{1, \ldots, Q\}, j \in N. \tag{3.25}$$

Recursions (3.25) hold if $a_j \leq F(q,j) \leq b_j$. Otherwise, if $F(q,j) < a_j$, then $F(q,j) = a_j$ and if $F(q,j) > b_j$, then $F(q,j) = \infty$. The recursion formula is initialized by

$$F(q,j) = \begin{cases} \max\{a_j, a_o + t_{o,j}\}, & \text{if } (o,j) \in A \text{ and } q = q_j, \\ \infty, & \text{otherwise, for } q \in \{1, \ldots, Q\}, j \in N. \end{cases} \tag{3.26}$$

The lower bound is given by:

$$\min_{j \in N} \{F(Q,j) + t_{j,d}\}. \tag{3.27}$$

The complexity of the bounding procedure for a n-node problem is $O(n^2 Q)$.

This lower bound can be increased by applying node penalties. This method consists of associating Lagrangean multipliers with the travel times of arcs incident to nodes that are not visited, or visited more than once. These multipliers decrease the former travel times and increase the latter. They are adjusted through subgradient optimization. The lower bound can also be increased by modifying the state-space through the application of subgradient optimization to the weights q_i. Branching on flow variables, Christofides, Mingozzi and Toth (1981b) report solving 50-node problems with moderately tight time windows. The reader may also refer to Friis (1985) for a Pascal code of this algorithm.

The same methodology has been applied to more complex routing problems and it will be discussed in Section 5. State-space relaxation parallels constraint aggregation and Lagrangean relaxation for integer programming. Constraints in integer programming formulations appear as state variables in dynamic programming recursions and hence constraint aggregation and relaxation correspond to state-space relaxation.

3.5 A Dynamic Programming Method

In the previous section, we have illustrated the use of transformed recursions derived from relaxing the state-space to obtain lower bounds which are then embedded in a tree search approach. However, when enough constraints are present, dynamic programming can be used directly to solve realistic size problems. The method presented in this section is based

on the paper by Dumas, Desrosiers, Gélinas and Solomon (1991) which takes advantage of the time window constraints. These constraints permit the reduction of the state-space dimension and the number of state transitions which is achieved in a preprocessing stage as well as while the algorithm is being run. The dynamic programming method presented here aims at minimizing the total arc cost. This is a more difficult optimization than when the minimum schedule time is sought. As in the previous methods, it is also assumed that the tour starts at time a_o. If the starting time at node o is within a time window $[a_o, b_o]$, the optimal tour can still be found by starting at time a_o, since waiting time does not increase the objective function.

To present this method, let $F(S, i, t)$ be the least cost of a path originating at node o, visiting every node of $S \subseteq N$ exactly once, ending at node $i \in S$, and ready to service node i at time t or later. Note the difference between the cost $F(S, i, t)$ of a path, and the time t when service can start at node i. Given $S \subseteq N$ and $i \in S$, this creates two-dimensional labels $(t, F(S, i, t)), a_i \leq t \leq b_i$, of the time and cost components since there is no relation of total order in $\mathbb{R}^2$. The function $F(S, i, t)$ can be obtained by solving the recursions:

$$\begin{aligned} F(S, j, t) \quad = \quad & \min_{(i,j)\in A} \{F(S - \{j\}, i, t') + c_{ij} | i \in S - \{j\}, t' \leq t - t_{ij}, a_i \leq t' \leq b_i\} \\ & \text{for all } \; S \subseteq N, j \in S \; \text{ and } \; a_j \leq t \leq b_j. \end{aligned} \tag{3.28}$$

Equation (3.28) holds true for $a_j \leq t \leq b_j$. If instead $t < a_j$, then $F(S, j, t) = F(S, j, a_j)$, while if $t > b_j$, then $F(S, j, t) = \infty$. Equation (3.28) is initialized by

$$F(\{j\}, j, \max\{a_j, a_o + t_{o,j}\}) = \begin{cases} c_{o,j}, & \text{if } (o, j) \in A, \\ \infty, & \text{otherwise.} \end{cases} \tag{3.29}$$

The optimal solution to the TSPTW is given by:

$$\min_{(i,d)\in A} \min_{a_i \leq t \leq b_i} \{F(N, i, t) + c_{i,d} | i \in N, t \leq b_d - t_{i,d}\}. \tag{3.30}$$

We present now the tests used to reduce the state-space dimension and the number of transitions. The first test deals with the situation where several paths that visit set S and end at i have different times to begin service and costs.

TEST 1. For two different states, (S, i, t^1) and (S, i, t^2), if $t^1 \leq t^2$ and $F(S, i, t^1) \leq F(S, i, t^2)$, then state (S, i, t^1) dominates state (S, i, t^2).

After the application of this test only the Pareto optimal states are further considered. A number of applications of dominance between states as a function of time and cost values can be found in the literature (see for example Desrochers and Soumis 1988ab, and Kolen,

Rinnooy Kan and Trienekens 1987). Incidentally, this test has been successfully applied in algorithms for the shortest path problem with time windows to be presented in Section 4. If integer data are used to define the time windows and the traveling times, the number of states (S, i, t), $a_i \leq t \leq b_i$ is countable in t. Then, $F(S, i, t)$ is stepwise decreasing as a function of t over the interval $[a_i, b_i]$. Therefore, after applying TEST 1, the two-dimensional labels $(t, F(S, i, t))$ of (S, i) can be ordered in a list by increasing time and decreasing cost value. Let $FIRST(S, i)$ be the time value of the first label of that list.

An optimal solution for the TSPTW can be thought of as a set of n nodes, ordered according to the time to begin service at each node. Partial orderings of the nodes are also dictated when precedence relationships are present. The state-space dimension and the number of transitions can be greatly reduced by examining whether partial paths satisfy these orderings. The tests developed for this purpose detect when a current partial path cannot be farther extended, thereby eliminating the corresponding states and transitions.

To present these tests, recall that $LDT(i, j)$ is the latest departure time from i such that the time to begin service at j is feasible. Recall also that $BEFORE(i)$ is the set of nodes k that must necessarily be visited before i. Finally, a state (S, i, t) admits a feasible extension towards j, i.e., the state $(S \cup \{j\}, j, \max\{a_j, t + t_{ij}\})$ can be created, if $t + t_{ij} \leq b_j$. We now present several post-feasibility tests. The first test was introduced by Desrosiers, Dumas and Soumis (1986) for the single vehicle pick-up and delivery problem.

TEST 2. Given the states (S, i, t), for all $a_i \leq t \leq b_i$, if the smallest time value to begin service at node i is greater than the latest feasible departure time towards a certain node j, $j \notin S$, i.e., $FIRST(S, i) > \min_{j \notin S} LDT(i, j)$, then the states (S, i, t), for all $a_i \leq t \leq b_i$, do not admit feasible extensions towards any node.

This global test eliminates the states (S, i, t) for all t by only examining the earliest time to begin service at i. If there exists a node which cannot feasibly be reached from node i even when service at node i begins as early as possible, then there is no reason to consider the states (S, i, t) any further. The next test asserts that all the nodes in the set $BEFORE(j)$ must necessarily be visited before a given node j.

TEST 3. Given the states (S, i, t), for $a_i \leq t \leq b_i$, and given node j, $j \notin S$, $(i, j) \in A$, if $BEFORE(j) \not\subset S$, then no feasible extensions exist towards j.

TEST 4. Given the state (S, i, t), for a fixed t, $a_i \leq t \leq b_i$, and a node j, $j \notin S$, $(i, j) \in A$, assume that node j is such that it admits a feasible extension of (S, i, t) towards j, i.e., $t \leq LDT(i, j)$. If there exists a node $k, k \notin S \cup \{j\}$ such that $t + t_{ij} > LDT(j, k)$, i.e., the additional extension from j to k does not satisfy the time window constraint at node k, then j cannot succeed i for (S, i, t). In this case the states (S, i, t'), for all $t' \geq t$, are not extended towards j.

This fourth test is implemented by choosing node k only among the immediate successors of i.

The time complexity of this algorithm is clearly exponential. Nevertheless Dumas, Desrosiers, Gélinas and Solomon (1991) report having optimally solved large scale problems with fairly wide, overlapping time windows using integer data for all parameters. When the density of the points inside the chosen geographical region increased with problem size, problems with up to 200 nodes were solved. As expected, the CPU time increases with the width of the time windows. However, for small time window widths, the behavior of the algorithm is less than exponential. When the density of the nodes in the geographical region was kept constant as the problem size was increased, the algorithm was capable of solving problems with up to 800 nodes. For these problems, the CPU time increased linearly with problem size.

Several extensions of the approach presented here can be readily seen. First, as can be observed from TEST 3, the algorithm is already adapted to account for additional precedence constraints. Second, as forward dynamic programming is used to solve the TSPTW, much more general cost functions could be utilized. These include step functions, nonlinear functions, and functions capturing the additional cost for waiting before service can begin at a node. In fact, as long as the cost function is a non-decreasing function of time, the algorithm and the dominance process can be adapted. Finally, when there are several nodes at the same physical location but with different time windows, the algorithm can be accelerated by removing a priori a set of arcs between these nodes. This is the *same location criterion* used in Dumas, Desrosiers and Soumis (1991).

3.6 A Two-Commodity Flow Formulation

Finke, Claus and Gunn (1984) have introduced a two-commodity network flow formulation for the classical TSP. In their formulation, one unit of the first commodity, flow Y, is delivered at each node while one unit of the second commodity, flow Z, is picked-up at each node. Hence the total combined amount of the two flows at each node is always the same. Lucena (1986) has considered a similar formulation with various demands d_i at each node

$i \in N$. Langevin, Desrochers, Desrosiers and Soumis (1993) have extended the approach to the TSPTW. Figure 3.2 depicts several two-commodity flow representations for Traveling Salesman tours, each one being described by the sequence $o \rightarrow i_1 \rightarrow i_2 \ldots i_n \rightarrow d$.

In time constrained Traveling Salesman Problems, time represents each of the two commodities. Along any feasible traveling salesman path, flow $Y_{ij}, (i,j) \in A$ is a decreasing flow at each node, while flow $Z_{ij}, (i,j) \in A$ increases at each node. The total combined amount of flow is always the same at each node and equals the upper bound value b_d at the destination-depot node. Recall that $W_i, i \in N \cup \{d\}$ is the waiting time before the start of service of node i. The two-commodity formulation of the TSPTW is given by:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij}(Y_{ij} + Z_{ij})/b_d + \sum_{i \in N \cup \{d\}} W_i \tag{3.31}$$

subject to:

$$\sum_{j \in N \cup \{d\}} (Y_{ij} + Z_{ij}) = b_d, \qquad \forall\, i \in N \tag{3.32}$$

$$\sum_{i \in N \cup \{o\}} (Y_{ij} + Z_{ij}) = b_d, \qquad \forall\, j \in N \tag{3.33}$$

$$\sum_{j \in N} (Y_{o,j} + Z_{o,j}) = b_d, \tag{3.34}$$

$$\sum_{i \in N} (Y_{i,d} + Z_{i,d}) = b_d, \tag{3.35}$$

$$T_o = \sum_{j \in N} Z_{o,j}, \tag{3.36}$$

$$T_i = \sum_{j \in N \cup \{d\}} Z_{ij}, \qquad \forall\, i \in N \tag{3.37}$$

$$T_i = \sum_{j \in N \cup \{o\}} (Z_{ji} + t_{ji}(Y_{ji} + Z_{ji})/b_d) + W_i, \qquad \forall\, i \in N \tag{3.38}$$

$$T_d = \sum_{i \in N} (Z_{i,d} + t_{i,d}(Y_{i,d} + Z_{i,d})/b_d) + W_d, \tag{3.39}$$

$$a_i \leq T_i \leq b_i, \qquad \forall\, i \in V \tag{3.40}$$

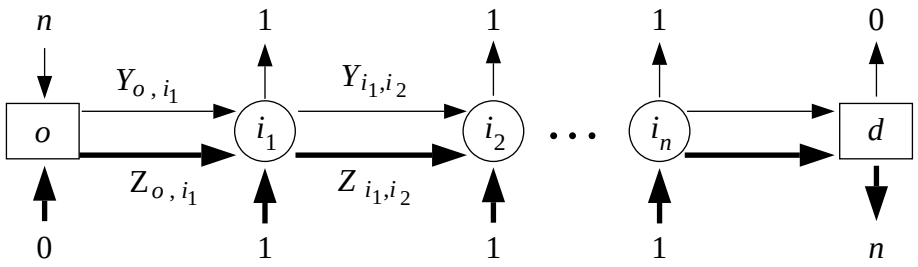
$$Y_{ij} \geq 0, \quad Z_{ij} \geq 0, \qquad \forall\, (i,j) \in A \tag{3.41}$$

$$W_i \geq 0, \qquad \forall\, i \in N \cup \{d\} \tag{3.42}$$

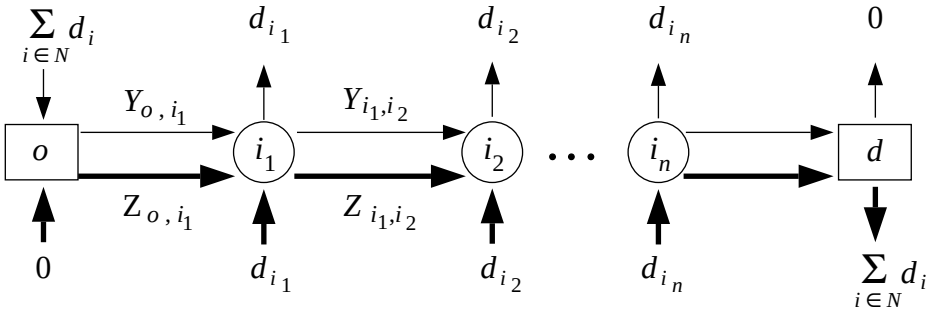
$$(Y_{ij} + Z_{ij})/b_d \quad \text{binary}, \qquad \forall\, (i,j) \in A. \tag{3.43}$$

The objective function (3.31) minimizes the sum of the costs over the arcs and the waiting time costs. If the waiting time costs do not appear in the objective function, then variables $W_i, i \in N \cup \{d\}$ can be considered simply as surplus variables and thus removed.

Finke, Claus and Gunn (1984)



Lucena (1986)



Langevin, Desrochers, Desrosiers and Soumis (1993)

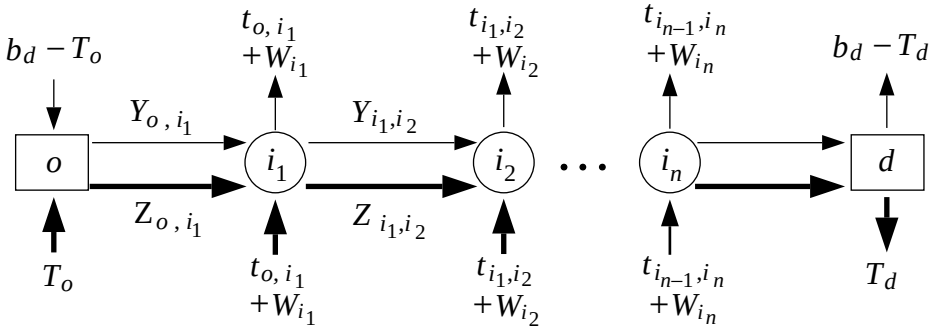


Figure 3.2: Two-commodity flow illustrations

At each node, the sum of the Y and Z flows in constraints (3.32)-(3.35) is always equal to b_d. Relation (3.36) describes the starting time variable at the origin-depot node, while the times to begin service at the other nodes are given in (3.37)–(3.39). For example, relation (3.38) states that at node $i, i \in N$, the total amount of flow on arcs entering node i is increased by the sum of the amount of time needed to travel on those arcs and the waiting time incurred before node i is ready for service. Since $(Y_{ij} + Z_{ij})/b_d$ are restricted to binary values by (3.43), relation (3.38) states that in any feasible integer solution, the time to begin service at node i (evaluated by flow Z) is given by the sum of the time at the previous node, say j, the travel time t_{ji} to reach node i, and the waiting time W_i incurred before the lower bound a_i. Time window constraints appear in (3.40), while non-negativity requirements for all variables are given in relations (3.41) and (3.42). The binary variables $(Y_{ij} + Z_{ij})/b_d$, $(i, j) \in A$ defined by (3.43) are the flow variables X_{ij} of the previous formulations while time variables T_i, $i \in N \cup \{o, d\}$ have been defined in (3.36)–(3.39).

This model is very general and as can be seen, it already accounts for a flexible starting time at the origin-depot node. If on each arc, the cost is identical to the time needed to travel on the arc, i.e., $c_{ij} = t_{ij}$, $(i, j) \in A$, and if $c_i = 1, i \in N \cup \{d\}$, then the objective function minimizes the total schedule time of the optimal tour while respecting the time windows. The formulation (3.31)–(3.43) for the TSPTW is valid only if the fixed parameter b_d is sufficiently large not to eliminate integer solutions. If there exists an integer solution, the upper bound value may be taken as $b_d = \max_{(i,d)\in A} \{b_i + t_{i,d}\}$. Relaxing the integrality conditions (3.43), the linear relaxation formulation of the TSPTW involves $O(n^2)$ variables and only $O(n)$ constraints. The lower bound obtained from the linear relaxation is at least as good as the assignment bound (given by the solution of (3.31)–(3.35) and (3.41)–(3.42)). It can be demonstrated that it improves as the value of b_d decreases, as long as the TSPTW instance stays feasible.

It should be noted that even though the subtour elimination constraints are verified for any feasible integer solution for the TSPTW, they are not necessarily verified by the linear relaxation of the previous formulation. It is then possible to improve the lower bound obtained using the LP relaxation of the two-commodity flow formulation by detecting violated subtour elimination constraints and by adding them to the LP relaxation. The classical subtour elimination constraints for the TSP (Dantzig, Fulkerson and Johnson 1954) can be adapted to the various two-commodity flow formulations and in our case they become

$$\sum_{i,j\in S} (Y_{ij} + Z_{ij}) \leq b_d(|S| - 1), \qquad \forall\, S \subset V,\ |S| \geq 2. \tag{3.44}$$

Computational experiments with this new time-constrained formulation show that it can solve problems of up to 40 nodes. Tests have been conducted using a fixed starting time value, i.e., $T_o = a_o = b_o$. Further research might possibly introduce improvements and extensions of the Miller-Tucker-Zemlin subtour elimination constraints (Desrochers and Laporte 1990). Despite their relative weakness, it is straightforward to identify which of these constraints are violated. Furthermore they are valid for any problem of the branch-and-bound tree. Computational tests should show the viability of this approach.

3.7 Special Routing Structures and Alternative Objective Functions

Related research has examined the TSP with precedence constraints, special routing structures and alternative objective functions. The TSP with precedence constraints will be addressed in Section 6. We next examine the other two areas.

The Shoreline Network: A routing structure which has recently received attention is the shoreline network. Following the work of Psaraftis, Solomon, Magnanti and Kim (1990) we define an ordered set of points $N = \{1, 2, \ldots, n\}$ to be located on the shoreline if the interpoint travel distances $c_{ij}, i \in N, j \in N$ satisfy the following conditions: For all $1 \leq i \leq k \leq j \leq n$,

$$c_{ii} = 0, \tag{3.45}$$

$$c_{ij} = c_{ji}, \tag{3.46}$$

$$c_{ij} \geq c_{ik}, \tag{3.47}$$

$$c_{ij} \geq c_{kj}, \tag{3.48}$$

$$c_{ij} \leq c_{ik} + c_{kj}. \tag{3.49}$$

Note that a shoreline network is the triangle-inequality metric with two additional restrictions, (3.47) and (3.48). Note further that it is not necessary for a path along the shoreline to be convex, i.e., lie on the boundary of a convex region. A further restriction of the shoreline problem is the straight-line case where condition (3.49) is strengthened to $c_{ij} = c_{ik} + c_{kj}$. Shoreline problems arise in transportation environments such as the routing and scheduling of cargo ships. In addition, the straight-line case occurs in a wide range of applications including rail, rivers, and highways.

In the path version of the shoreline TSPTW, a vehicle starting at time zero from point 1 and traveling at unit speed must visit once all the points within their respective time windows. In the tour version, the vehicle must return to point 1 after visiting all the other points exactly once.

The computational complexity of the shoreline TSPTW remains open at this time. It should however be noted that the shoreline TSP is polynomially solvable, while the Euclidean TSP is NP–complete (Papadimitriou 1977). In this light, Psaraftis, Solomon, Magnanti and Kim (1990) propose heuristic algorithms for its solution and derive data-dependent worst-case performance ratios for these heuristics. The authors also illustrate the performance of these algorithms on a real-world shoreline. Kim (1985) has performed a cursory analysis of other problem variants involving alternative objective functions, different types of time window constraints and multi-vehicles.

Research has also been carried out for the straight-line case. We discuss here only the path variant of the problem which has proved to be more challenging than its tour counterpart. Psaraftis, Solomon, Magnanti and Kim (1990) have examined the version of the problem where only earliest pick-up time-constraints are present. The authors develop an $O(n^2)$ optimal dynamic programming algorithm which minimizes the total schedule time, or equivalently, the maximum completion time. When service times are present, the problem was later shown to be strongly NP-complete by Tsitsiklis (1992). This author has also constructed an $O(n^2)$ dynamic programming algorithm for the version of the problem involving solely deadlines. Furthermore, he has shown that when general time windows are present the problem is strongly NP-complete.

Alternative Objective Functions: A related problem is the traveling repairman problem (TRP) with time windows (TRPTW). The objective is to minimize the sum of completion times, i.e., $\sum T_i$, or equivalently, the sum of flowtimes, i.e., $\sum (T_i - a_i)$. Afrati, Cosmadakis, Papadimitriou, Papageorgiou and Papakonstantinou (1986) have shown that the TRP can be solved in $O(n^2)$ time. They have further shown that when deadlines are imposed, the problem becomes NP-complete but can be solved by a pseudo-polynomial algorithm. Recently, Tsitsiklis (1992) proved that the TRPTW is strongly NP–complete. The author also examines the TSPTW and the TRPTW when the number of nodes is bounded by a constant. We refer the reader to the original paper for details on the different complexity results and open problems. If the order in which the nodes are visited is fixed in advance, the TRPTW reduces to optimizing the schedule for a fixed route. We address this problem in Section 6.3.

Research has also been recently undertaken on the objective of minimizing the number of late deliveries. This is an important objective encountered in several practical applications. One example is encountered in technologically advanced manufacturing systems where an automated guided vehicle must deliver parts to a set of machines such that the number of idle machines is minimized. Another example involves express pizza delivery systems. Gendreau, Laporte and Solomon (1992) develop a specialized enumerative algorithm capable of optimally solving problems involving up to 100 customers.

Finally, the time dependent TSP, i.e., the TSP where the travel time between every pair of nodes depends not only on the distance between the nodes but also on the time of day, has been studied by Malandraki and Daskin (1989). The authors propose a cutting plane heuristic.

4 Constrained Shortest Path Problems

The shortest path problem with time windows consists of finding the least cost route between two specified nodes in a graph whose nodes can only be visited within a specified time interval. This problem has found its first application as a subproblem in the construction of school-bus routes in the early '80s. Since then it has appeared as a subproblem in many time constrained routing and scheduling problems to be described in the following sections. The shortest path problem with time windows is a relaxation of the Traveling Salesman Problem with Time Windows obtained by relaxing the requirement that each node must be visited.

4.1 The Shortest Path Problem with Time Windows

Notation: Let $G = (V, A)$ be a graph where A is the set of arcs and V is the set of nodes $N \cup \{o, d\}$, where N consists of nodes that can be visited from an origin o to a destination d. A time window $[a_i, b_i]$, $i \in V$ is associated with each node. A path in the graph G is defined by a sequence of nodes $i_0, i_1, \ldots, i_H$, such that each arc (i_{k-1}, i_k) belongs to A. All paths start at time a_o from node $i_0 = o$ and finish at node $i_H = d$ no later than b_d. A path is elementary if it contains no cycles. Each arc $(i, j) \in A$ has a positive or negative cost c_{ij} and a positive duration t_{ij}. We assume next that the service time at node i is included in the time value t_{ij}, for all $i \in N$. An arc (i, j) is defined in the set A only if it is feasible, i.e., if it respects the condition: $a_i + t_{ij} \leq b_j$. This problem was first introduced in Desrosiers, Pelletier and Soumis (1983) and Desrosiers, Soumis and Desrochers (1984) as a subproblem for the Multiple Traveling Salesman Problem with Time Windows.

Formulation: The mathematical programming formulation involves two types of variables: flow variables X_{ij}, $(i, j) \in A$, and time variables T_i, $i \in V$. Define variable X_{ij} to be the flow on arc $(i, j) \in A$, and variable T_i to be the start of service at node $i, i \in V$. Waiting time is allowed before the start of service at each node. Using this notation, the Elementary (or acyclic) Shortest Path Problem with Time Windows (ESPPTW) may be formulated as follows:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{4.1}$$

subject to:

$$\sum_{j\in V} X_{ij} - \sum_{j\in V} X_{ji} = \begin{cases} +1, & i = o \\ 0, & \forall\, i \in N \\ -1, & i = d \end{cases} \tag{4.2}$$

$$X_{ij} \geq 0, \qquad \forall\, (i,j) \in A \tag{4.3}$$

$$X_{ij}(T_i + t_{ij} - T_j) \leq 0, \qquad \forall\, (i,j) \in A \tag{4.4}$$

$$a_i \leq T_i \leq b_i, \qquad \forall\, i \in V. \tag{4.5}$$

The objective function (4.1) seeks to minimize the total travel cost. Constraints (4.2)–(4.3) define flow conditions on the graph G, time windows appear in (4.5), and compatibility requirements between flow and time variables are given in (4.4).

Nonlinear Integrality Property: This nonlinear formulation has an appealing characteristic: as in the case of the TSPTW, it can be shown that if the problem is feasible, then there exists an optimal integer solution. To prove this, note that constraints (4.4) indicate that if $X_{ij} > 0$, then $T_i + t_{ij} \leq T_j$. If the flow values are fractional, the optimal solution of value Z^* is composed of several paths of cost c_p, each of which has a positive flow θ_p, i.e., $Z^* = \sum\limits_p c_p\theta_p$, where $\sum\limits_p \theta_p = 1$. Assign a unit flow to the arcs of the minimum cost path $c_{\min}$: then, this path satisfies the time constraints and also constitutes an optimal integer solution since:

$$Z^* \leq c_{\min} = \sum_p c_{\min}\theta_p \leq \sum_p c_p\theta_p = Z^*.$$

Literature: The shortest path with additional linear constraints is a related problem which has been discussed by a number of authors, e.g., Joksch (1966), Minoux (1975), Hansen (1980), Aneja, Aggarwal and Nair (1983), Jaffe (1984) and Martins (1984). The proposed algorithms use Lagrangean relaxation, branch-and-bound and dynamic programming. These algorithms are designed, however, for problems with only a few additional linear constraints, while the ESPPTW involves many such constraints.

Dynamic Programming Algorithms: The ESPPTW can be solved by dynamic programming. To introduce this approach, define $F(S,i,t)$ as the minimum cost of the path going from node o to node i, $i \in N \cup \{d\}$, visiting all nodes in set $S \subseteq N \cup \{d\}$ only once, and servicing node i at time t or later. The cost $F(S,i,t)$ can be computed by solving the following recurrence equations:

$$F(\phi, o, a_o) = 0 \tag{4.6}$$

$$F(S, j, t) = \min_{(i,j)\in A} \left\{ F(S - \{j\}, i, t') + c_{ij} | i \in S - \{j\}, t' \leq t - t_{ij}, a_i \leq t' \leq b_i \right\},$$

$$\text{for all} \quad S \subseteq N \cup \{d\}, \ j \in S \quad \text{and} \quad a_j \leq t \leq b_j. \tag{4.7}$$

The optimal solution is given by

$$\min_{S\subseteq N\cup\{d\}} \ \min_{a_d \leq t \leq b_d} F(S, d, t). \tag{4.8}$$

Note that equation (4.7) is valid only if $a_j \leq t \leq b_j$. If however $t < a_j$, then $F(S, j, t) = F(S, j, a_j)$, and if $t > b_j$, then $F(S, j, t) = \infty$. The ESPPTW is NP-hard in the strong sense as can be shown by reduction from the problem denoted *Sequencing within Intervals* (Dror 1994; see also Garey and Johnson 1979). Thus, the proposed dynamic programming algorithm has an exponential complexity and no pseudo-polynomial algorithm is known for this problem.

An easier problem to solve is obtained from the ESPPTW by relaxing the elementary path requirement. This relaxed problem called the Shortest Path Problem with Time Windows (SPPTW) admits paths with cycles when some c_{ij} are negative. However, the time window constraints and the positive arc durations t_{ij} guarantee the finiteness of all paths. The SPPTW can also be solved by dynamic programming. To present this methodology, define $F(i, t)$ as the minimum cost of the path going from node o to node i, $i \in N \cup \{d\}$, and servicing node i at the latest at time t. The shortest path from o to d can be computed by solving the recursions:

$$F(o, a_o) = 0 \tag{4.9}$$

$$F(j, t) = \min_{(i,j)\in A} \{F(i, t') + c_{ij} | i \in N \cup \{d\} - \{j\}, \ t' \leq t - t_{ij}, a_i \leq t' \leq b_i\},$$

$$\text{for all} \quad j \in N \cup \{d\} \quad \text{and} \quad a_j \leq t \leq b_j. \tag{4.10}$$

The optimal solution is given by

$$\min_{a_d \leq t \leq b_d} F(d, t). \tag{4.11}$$

Again, equation (4.10) is valid only if $a_j \leq t \leq b_j$. If however $t < a_j$, then $F(j, t) = F(j, a_j)$ and if $t > b_j$, then $F(j, t) = \infty$. The SPPTW includes the multiple knapsack problem as a special case and therefore is also NP-hard. However, there are pseudo-polynomial algorithms to solve it. These algorithms will be discussed in the next subsection.

4.2 Dynamic Programming Algorithms for the SPPTW

Solving the SPPTW by dynamic programming is equivalent to solving an unconstrained shortest path problem on an acyclic graph. Given that the time window parameters are integer values, the maximum number of nodes in the acyclic graph is equal to $D = \sum_{i \in V} (b_i - a_i + 1)$. This requires large amounts of computer time and memory when the set of nodes is large and especially when the time windows are wide. Nevertheless, we present here two efficient algorithms to solve the SPPTW.

First, introduce the concept of a *label* which permits to solve the problem on the graph $G = (V, A)$, rather than on the acyclic graph with D nodes. With each path $\mathcal{P}_i$ from the origin o to the node i satisfying the time windows is associated a two-dimensional (time, cost) label corresponding to the start of service at node i and the cost of the path $\mathcal{P}_i$, respectively. At node i, these labels will be denoted by

$$(T_i^k, C_i^k), \qquad i \in V,\ k \geq 1$$

to indicate the characteristics of the k^{th} path from o to i. The indices k and i may be dropped when the context is unambiguous. These labels are calculated iteratively along the path $\mathcal{P}_i = (i_0, i_1, i_2, \ldots, i_H)$ as:

$$\begin{aligned} (T_{i_0}, C_{i_0}) &= (a_o, 0) \\ (T_{i_h}, C_{i_h}) &= (\max\{a_{i_h}, T_{i_{h-1}} + t_{i_{h-1},i_h}\}, C_{i_{h-1}} + c_{i_{h-1},i_h}), \quad h = 1, \ldots, H. \end{aligned}$$

where $i_0 = o$ and $i_H = i$

Definition: Let (T_i^1, C_i^1) and (T_i^2, C_i^2) be two different labels for two paths from o to i. The first label dominates the second, i.e., $(T_i^1, C_i^1) \prec (T_i^2, C_i^2)$ if and only if $(T_i^2, C_i^2) - (T_i^1, C_i^1) \geq (0, 0)$.

Definition: A label (T_i, C_i) at a given node i is said to be efficient if no other labels at i dominate it. A path from o to i is said to be efficient if the corresponding label is efficient.

This dominance relation is not a total ordering and therefore does not allow all paths to be ordered. However, this relation defines a partial order on the labels and therefore it does allow us to conclude that the efficient path $\mathcal{P}_i$ is the shortest path from node o to node i such that service at node i starts at the latest at time T_i. This implies the possibility of

several efficient paths for each node. The importance of the dominance relation stems from allowing the cost of the feasible path to be defined as a decreasing step function of the time at which the service begins.

Define now Q_i, $i \in V$, to be the set of labels of node i and let $EFF(Q_i)$ denote the set of efficient labels among the set of labels Q_i of node i. The set of efficient labels at each node can be calculated by dynamic programming. The shortest path from o to d satisfying the time window constraints is obtained directly from the set $EFF(Q_d)$: it is represented by the least cost label.

Since arc costs c_{ij} might be negative, we first present a *label correcting* algorithm which is an adaptation of the Ford-Bellman-Moore algorithm (1956, 1958 and 1959, respectively) for the classical shortest path problem. We then present a *permanent labeling* algorithm, which is a generalization of the Dijkstra's algorithm (1959) on graphs with non-negative arc costs.

A Label Correcting Algorithm (Desrosiers, Pelletier and Soumis 1983): Let $\Gamma(i) = \{j|(i,j) \in A\}$ be the set of successors of node i, and $Q_i = \bigcup_k \{(T_i^k, C_i^k)\}$ be the set of labels of node $i \in V$. A basic operation in shortest path algorithms is the *treatment of a label* (T_i^k, C_i^k). It consists of creating new labels at nodes $j \in \Gamma(i)$ by adding arcs (i,j) to the path from o to i associated with label (T_i^k, C_i^k). The new label for a given $j \in \Gamma(i)$ is:

$$f_{ij}(T_i^k, C_i^k) = \begin{cases} (\max\{a_j, T_i^k + t_{ij}\}, C_i^k + c_{ij}), & \text{if} \quad T_i^k + t_{ij} \leq b_j \\ \phi, & \text{otherwise.} \end{cases} \tag{4.12}$$

The *treatment of node* i is the treatment of all labels in Q_i. The set of new labels at each node $j \in \Gamma(i)$ is $\bigcup_k f_{ij}(T_i^k, C_i^k)$. Since the definition of f_{ij} allows waiting at node j, the new labels will not all be efficient, as the time values T_j^k may pile up at the value a_j. In fact $T_j^k = a_j$, for all k such that $T_i^k + t_{ij} \leq a_j$. Furthermore, some new labels may dominate or may be dominated by some labels already in Q_j. Hence, the new set of efficient labels at node j is given by $EFF(\bigcup_k f_{ij}(T_i^k, C_i^k) \cup Q_j)$. The algorithm to be presented next uses a list $\mathcal{L}$ of nodes which consists of the nodes which have to be treated after improvement or modification of their labels. This algorithm can be described as follows:

A Label Correcting Algorithm for SPPTW

Step 0:

Initialization

$Q_o = \{(T_o^1 = a_o, C_o^1 = 0)\}$;

$Q_i = \{(T_i^1 = a_i, C_i^1 = \infty)\} \quad \forall\, i \in V \setminus \{o\};\ \mathcal{L} = \{o\}.$

Step 1.

Treatment of node i

Choose a node $i \in \mathcal{L}$;

For all $j \in \Gamma(i)$ do:

$Q'_j = EFF(\bigcup_k f_{ij}(T_i^k, C_i^k) \cup Q_j)$,

If $Q'_j \neq Q_j$ then $Q_j = Q'_j$ and $\mathcal{L} := \mathcal{L} \cup \{j\}$.

Step 2.

Reduction of $\mathcal{L}$

$\mathcal{L} := \mathcal{L} \setminus \{i\}$;

If $\mathcal{L} = \phi$ then STOP, otherwise return to **Step 1**.

The labels of set Q_i of a node i are placed in a list in increasing time order. Next, let Δ_i, a subset of Q_i, be the set which only contains the labels which have been modified since i appeared in $\mathcal{L}$. Labels in Δ_i are the only labels to generate modifications at node j when coming from node i. The set Q_j is then obtained by a sequential update of set Q_j based only on Δ_i.

The complexity of this algorithm depends on the rule for selecting the node to be treated next. FIFO or LIFO strategies can produce an exponential worst-case complexity. A polynomial complexity of order $O(D^3)$ can however be obtained by utilizing the $\mathcal{L} - 2$ queue method (Pape 1980, Pallottino 1984, and Desrochers and Soumis 1988a). In short, this method uses a double queue to represent the list $\mathcal{L}$. The first time a node is inserted in $\mathcal{L}$, it is added at the tail of the second queue. Subsequent insertions of this node occur at the tail of the first queue. The nodes to be treated are withdrawn from the head of the first queue. If the first queue is empty, the second queue becomes the first, and the second queue becomes empty.

In an acyclic graph the nodes can be numbered in such a way that if $(i, j) \in A$, then $i < j$. This numbering of the nodes ensures that each node is inserted in list $\mathcal{L}$ once and only once. In a graph with time windows where cycles exist, the nodes can be numbered in chronological order, for example, by start time. If the time windows are sufficiently narrow, this numbering ensures that the relation $(i, j) \in A \Rightarrow i < j$ is satisfied by the majority of the arcs. Therefore, the treatment of the nodes from the first queue in this order significantly reduces reinsertions and computation time (see the computational experiments in Desrosiers, Pelletier and Soumis 1983).

The label correcting algorithm solves problems with 500 nodes, 50,000 arcs and 100 discrete time units per window in a matter of a few seconds. This corresponds to solving a classical shortest path problem in an expanded network with 50,500 nodes and 5,050,000 arcs.

A Permanent Labeling Algorithm (Desrochers and Soumis 1988a): This method is a generalization of Dijkstra's algorithm to the SPPTW. This permanent labeling algorithm can be used even if the SPPTW contains negative costs and negative cycles. The procedure is based on a new treatment order of labels.

In classical shortest path problems, the notions of *node* and *label* are closely linked: one label is associated to each node. The order of treatment is thus defined simultaneously for nodes and labels. In the case of time constrained shortest path problems with an objective function which minimizes the travel cost, a set of labels is associated with each node, thus the two notions are distinct. The label correcting algorithm previously presented did not fully exploit this distinction. The treatment order was defined on the *nodes* and all the labels associated with a node were treated. The computational aspect was improved by the use of subsets Δ_i, $i \in V$, and by the chronological numbering of the nodes by start time.

In contrast, the permanent labeling algorithm treats sequentially the *labels* in increasing order of time. The positive duration of the arcs insures that once a label has been treated it is impossible to improve it any further. The algorithm does not necessitate the use of list $\mathcal{L}$. Instead, the method uses sets P_i of permanent labels at node i. Each set P_i contains the labels of node i which has been previously treated. The algorithm consists of the following steps:

A Permanent Labeling Algorithm for SPPTW

Step 0:
Initialization
$Q_o = \{(T_o^1 = a_o, C_o^1 = 0)\}$; $Q_i = \emptyset, \quad \forall\, i \in N \cup \{d\}$;
$P_i = \emptyset, \quad \forall\, i \in V$.

Step 1.
Selection of the next label to be treated
Choose a label (T_i^k, C_i^k) with minimal T_i^k from $\underset{i \in V}{\cup}(Q_i \setminus P_i)$;
If $\underset{i \in V}{\cup}(Q_i \setminus P_i) = \emptyset$ then STOP.

Step 2.
Treatment of label (T_i^k, C_i^k)
For all $j \in \Gamma(i), \quad Q_j := EFF(f_{ij}(T_i^k, C_i^k) \cup Q_j)$;
$P_i := P_i \cup \{(T_i^k, C_i^k)\}$;
Return to **Step 1**.

The complexity of Step 1 can be reduced by using the concept of a *bucket* introduced by Denardo and Fox (1979). For classical shortest path problems, a bucket is a set of nodes whose label costs lie within a specified interval. For time constrained shortest path, a bucket contains those labels whose time lies within a specified interval. The k^{th} bucket consists of the set of labels with a time label in the interval $[kw, (k+1)w)$, where the width of the bucket, w, is fixed at $w = \min_{(i,j) \in A} t_{ij}$. The search for the earliest untreated time label is thus replaced by the search for an element of the earliest untreated bucket. The optimality of the algorithm with buckets can be easily proved. Its worst-case complexity is of order of $O(D^2)$. The algorithm solves problems with up to 2,500 nodes and 250,000 arcs in only a few seconds.

4.3 The Shortest Path Problem with Resource Constraints

The Shortest Path Problem with Resource Constraints (SPPRC) was introduced by Desrochers (1988) as a multi-dimensional generalization of the SPPTW. The SPPRC appears as a subproblem in the Dantzig-Wolfe decomposition/column generation and in the

Lagrangean Relaxation approaches used to solve vehicle routing problems (Section 5), vehicle fleet planning problems, bus driver scheduling problems, airline crew scheduling problems and many other time constrained routing and scheduling problems (Section 7).

Notation: The problem is defined on a graph $G = (V, A)$ where A is the set of arcs and V is the set of nodes $N \cup \{o, d\}$, where N consists of the nodes that can be visited from an origin o to a destination d. A path in the graph G is defined as a sequence of nodes $i_0, i_1, \ldots, i_H$, such that each arc (i_{h-1}, i_h) belongs to A. All paths start at the origin ($i_0 = o$) and end at the destination ($i_H = d$). A cost c_{ij} is associated with each arc $(i, j) \in A$. The cost of a path is defined as the sum of the costs of the arcs of the path. The paths need not be elementary, i.e., the same node is allowed to be visited more than once (i_h can be equal to $i_{h'}$, $h \neq h'$).

Let R denote the set of resources used to model a complex situation. The travel time t_{ij} on each arc (i, j) is replaced by the consumption of t^r_{ij} units of resource r, for all $r \in R$. The time interval constraint $[a_i, b_i]$ on node i is replaced by $|R|$ interval constraints of the form $[a^r_i, b^r_i]$, $r \in R$, restricting the amounts of resources used by the path to reach node i. The resource consumptions accumulate along a path. Define then T^r_i as the amount of the r^{th} resource used to reach node i using a path from o to i. A path reaching node i and using less than a^r_i units of the r^{th} resource becomes feasible by wasting some units of the r^{th} resource. However, a path reaching node i and using more than b^r_i units of the r^{th} resource is infeasible.

An arc (i, j) is infeasible if it is not possible to visit node j after node i while respecting the feasibility windows for both nodes i and j. All infeasible arcs are then excluded from the set A. Conversely, all arcs $(i, j) \in A$ respect the following conditions:

$$a^r_i + t^r_{ij} \leq b^r_j, \quad \forall\, r \in R.$$

The SPPRC is then defined as follows: find a minimal cost feasible path from the origin o to the destination d, i.e., a sequence of nodes $(i_0, i_1, \ldots, i_H)$ and an associated resource consumption, $T^r_{i_h}$, $0 \leq h \leq H$, for each resource such that:

1) $(i_{h-1}, i_h) \in A$, for $1 \leq h \leq H$, with $i_0 = o$ and $i_H = d$;

2) $T^r_{i_{h-1}} + t^r_{i_{h-1}, i_h} \leq T^r_{i_h}$, for $1 \leq h \leq H$ and $r \in R$;

3) the resource constraints are respected at all the visited nodes:
$a^r_{i_h} \leq T^r_{i_h} \leq b^r_{i_h}$, for $0 \leq h \leq H$ and $r \in R$;

4) the total cost ($\sum_{h=1}^{H} c_{i_{h-1},i_h}$) of this path is the minimum among all feasible paths.

Formulation: The mathematical formulation (4.1)–(4.5), introduced for the elementary shortest path problem with time windows can easily be extended to the SPPRC. Let X_{ij}, $(i,j) \in A$, be the flow variables, and T_i^r, $i \in V$, $r \in R$, be the resource variables. Then, the formulation is:

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{4.13}$$

subject to:

$$\sum_{j\in V} X_{ij} - \sum_{j\in V} X_{ji} = \begin{cases} +1, & i = o \\ 0, & \forall\, i \in N \\ -1, & i = d \end{cases} \tag{4.14}$$

$$X_{ij} \geq 0, \qquad \forall\, (i,j) \in A \tag{4.15}$$

$$X_{ij}(T_i^r + t_{ij}^r - T_j^r) \leq 0, \qquad \forall\, (i,j) \in A,\ \forall\, r \in R \tag{4.16}$$

$$a_i^r \leq T_i^r \leq b_i^r, \qquad \forall\, i \in V,\ \forall\, r \in R. \tag{4.17}$$

The above formulation contains only elementary shortest paths in the solution space. If the graph G is acyclic, like in many practical fleet assignment and crew scheduling applications, the dynamic programming described next is optimal. Otherwise, the solution space may include paths with finite cycles.

4.4 A Dynamic Programming Algorithm for the SPPRC

Solving the SPPRC by dynamic programming may be done by solving an unconstrained shortest path on an acyclic graph. Given that resource parameters are integer values, the maximum number of nodes of the resulting acyclic graph is equal to $\sum_{i\in V} \prod_{r\in R} (b_i^r - a_i^r + 1)$. This may require a large amount of computer time and memory. Note that an interval reduction process can be performed a priori, similar to the one described for the TSPTW in Section 3.1. Next, we use again the concept of labels which permits to solve the problem on the original graph $G = (V, A)$.

With each path $\mathcal{P}_i$ from the origin o to node i is associated a state denoted T_i^R, corresponding to the quantity of each resource $(T_i^1, T_i^2, \ldots, T_i^{|R|})$ used by the path, and a cost value C_i, which is a function of the state. For a given node i, the set of feasible states is $\{(T_i^1, T_i^2, \ldots, T_i^{|R|}) | a_i^r \leq T_i^r \leq b_i^r, \text{ for all } r \in R\}$. To use a simpler notation, for each path

we define a label (T_i^R, C_i). The label represents both the state value and the cost of a path $\mathcal{P}_i$. The definitions of dominance, efficient label and efficient path can be generalized to this multi-dimension problem.

Definition: Let (T_i^{1R}, C_i^1) and (T_i^{2R}, C_i^2) be two distinct labels. Then, the first label dominates the second, i.e., $(T_i^{1R}, C_i^1) \prec (T_i^{2R}, C_i^2)$ if and only if $C_i^1 \leq C_i^2$, $T_i^{1r} \leq T_i^{2r}$ for all $r \in R$.

Definition: A label (T_i^R, C_i) at a given node i is said to be efficient if no other label at i dominates it. A path $\mathcal{P}_i$ from o to i is said to be efficient if its label is efficient.

A Pulling Dynamic Programming Algorithm: Similarly to the SPPTW, the SPPRC can be solved by a dynamic programming algorithm which preserves only non-dominated labels. For the SPPTW, the basic step of the Label Correcting Algorithm and the Permanent Labeling Algorithm is the *reaching* process. This consists of starting with a label associated with a path $\mathcal{P}_i$, and for all $j \in \Gamma(i)$, determining the labels corresponding to feasible paths $\mathcal{P}_j$ which extend the path $\mathcal{P}_i$. For each node i, these two algorithms create new labels for nodes $j \in \Gamma(i)$; in other words, labels can be created for node j every time one of its predecessors is treated. The labels at each node j must therefore be updated several times. This is costly, because the process is more complex than a sequential update, especially if there are a significant number of resource constraints.

The efficiency of the dynamic programming algorithm can be increased by using a *pulling* process. This process was introduced by Desrochers and Soumis (1988b) in a reoptimization algorithm for the SPPTW and in Desrochers (1988) for the SPPRC. For a given node j, the *pulling* process consists of creating labels at this node for all feasible labels (T_i^R, C_i) associated with paths $\mathcal{P}_i$, obtained by extending all these paths for which the addition of arc $(i, j) \in A$ allows arrival at node j in a feasible state T_j^R. A new label $(T_j^R, C_i + c_{ij})$, with T_j^R given by

$$(\max\{a_j^r, T_i^r + t_{ij}^r\}, \forall\, r \in R) \tag{4.18}$$

is created at node j if $T_i^r + t_{ij}^r \leq b_i^r$ for all $r \in R$. As the new labels are created at a single node, this algorithm requires the updating of labels at only one node, in contrast with the reaching approach requiring updating at several nodes.

A permanent labeling algorithm can be developed if, for each arc (i, j), at least one resource consumption or cost takes positive values, i.e., $(t_{ij}^1, t_{ij}^2, \ldots, t_{ij}^{|R|}, c_{ij}) \not\leq 0$, for all $(i, j) \in A$. To simplify the notation, we will use a slightly stronger hypothesis:

$$\text{for all } (i,j) \in A, \quad t_{ij}^1 \geq w > 0. \tag{4.19}$$

A simple algorithm is then obtained. To present it, the following notation is needed. For each node $i \in V$, let Q_i be the set of labels and P_i the subset of permanent labels. The set P_i is characterized by using a variable bound π_i on the consumption of the first resource: P_i is the subset of labels such that $a_i^1 \leq T_i^1 < \pi_i \leq b_i^1$. The algorithm consists of the following steps:

A Permanent Labeling Algorithm for SPPRC

Step 0.
Initialization
$Q_o = \{(a_o^1, \ldots, a_o^{|R|}, 0)\}; \qquad \pi_o = b_o^1;$
$Q_i = \phi,\ \pi_i = a_i^1, \quad \forall\, i \in N \cup \{d\}.$

Step 1.
Selection of node j
If $\pi_i = b_i^1 \quad \forall\, i \in N \cup \{d\}$, then STOP.
Otherwise select $j = \arg \min_{i \in N \cup \{d\}} \{\pi_i | \pi_i < b_i^1\}$;

Step 2.
Pulling at node j
Q'_j is the set of labels created such that $\pi_j \leq T_j^1 \leq \min\{b_j, \pi_j + w\}$.
$P_j := EFF(Q'_j \cup P_j)$;
$\pi_j = \min\{b_j, \pi_j + w\}$;
Return to **Step 1**.

The new labels created in set P_j at Step 2 are permanent labels. The multi-dimensional procedure $EFF(\cdot)$ used to select the subset of efficient labels is complicated and it needs specialized data structures to obtain a pseudo-polynomial algorithm of low complexity. An efficient algorithm using the pulling process was designed by Desrochers (1988). The author describes a primal-dual approach which permits to identify permanent and non-permanent labels using the bucket concept. Problems with 1,000 nodes, 50,000 arcs, and 5 resource constraints at each node were solved in less than 1 minute. Such problems necessitate the creation of about 6,000,000 states.

4.5 Extensions

The first special case of the SPPTW is the problem where the cost of a path is the path duration, including the waiting time. In this case, for any label (T_i, C_i) at node i, $C_i = T_i - a_o$. The two dimensional labels can be replaced by one dimensional labels. Since a relationship of total order exists between the one dimensional labels, it is sufficient to keep only one non-dominated label at each node. Hence, the classical shortest path algorithms can be adapted to the context of time windows: if $T_i > b_i$, the label is eliminated; if $T_i < a_i$, the label is set to the value a_i. If the graph is acyclic or if arc durations are non-negative, an $O(n^2)$ permanent labeling algorithm can be used. Otherwise, an $O(n^3)$ label correcting algorithm must be utilized.

A second special case of the SPPTW is the longest time path problem (the critical path problem) with time windows. This is obtained by minimizing the negative value of the duration. In this case, for any label (T_i, C_i) at node i, $C_i = -(T_i - a_o)$. Hence, there is no dominance relation between the labels of a node. Note however that the pseudo-polynomial algorithms can still be used. If the graph is acyclic, the implementation is facilitated by treating nodes in topological order. In the special case where time windows impose a precedence relationship between the nodes, an $O(n^2)$ can be used. This fact was exploited by Baker (1983) in his approach for the TSPTW described in Section 3.

The algorithms presented can also be adapted to solve the 2–cycle free SPPTW, a relaxation of ESPPTW, yet stronger than SPPTW. A 2–cycle free path is a path without cycles of the form $(j \rightarrow i \rightarrow j)$. For the classical shortest path problem, the dynamic programming algorithm can be adapted for 2–cycle elimination by keeping the best and the second best labels at each node (Houck, Picard, Queyranne and Vemuganti 1980, and Christofides, Mingozzi and Toth 1981a). The modification appears in the treatment of a node i for which $j \in \Gamma(i)$, and the best label is obtained by using the arc (j, i). The creation of a path with the cycle $(j \rightarrow i \rightarrow j)$ is avoided by adding the arc (i, j) to the second best label. It creates the best 2–cycle free path arriving at node j via node i. Algorithms for 2-cycle free SPPTW are easily obtained by doubling the number of labels in the SPPTW algorithms (Kolen, Rinnooy Kan and Trienekens 1987, and Desrochers, Desrosiers and Solomon 1992).

Some generalizations of the SPPTW involving linear costs on the waiting time at each node can also be solved. Consider first a linear penalty function $c_j(T)$, with a non-negative slope within the time window $a_j \leq T \leq b_j$, for each node j. In this case, the least cost value of a given partial path at node j is the one with the earliest arrival time. Provided that the global cost function is updated correctly, i.e., c_{ij} is replaced by $c_{ij} + c_j(T_j)$ in the computa-

tions, no other modifications to the algorithm are required. If all the penalty functions are defined using non-positive slopes, the shortest path problem should be solved on the mirror network, i.e., the direction of all arcs should be reversed and the time windows should be updated according to a certain constant parameter. Such an application is presented in Dumas, Salomon, Solomon and Van Wassenhove (1993) in the manufacturing context. When positive and negative slope penalty functions appear simultaneously, the algorithmic difficulties increase so much that this extension will not be discussed here (Ioachim, Gélinas, Desrosiers and Soumis 1993).

Additional extensions of the SPPRC concern the cost function and the resource functions. Since the shortest path problem with resource constraints is solved using forward dynamic programming, nonlinear cost functions can easily be used. Desrochers, Gilbert, Sauvé and Soumis (1992) provide examples of piecewise linear and step functions utilized for the Urban Crew Scheduling Problem (see also Sections 6 and 7). The essential condition on a valid cost function is that it must be non-decreasing (see Desrosiers, Dumas, Solomon and Soumis (1993) for a proof). Finally, as the cost can be considered as a resource, nonlinear non-decreasing resource functions can also be used to model complex situations.

5 The Vehicle Routing Problem with Time Windows

The vehicle routing problem with time windows (VRPTW) consists of designing a set of minimum cost routes, originating and terminating at a central depot, for a fleet of vehicles which services a set of customers with known demands. The customers must be assigned exactly once to vehicles such that the vehicle capacities are not exceeded. The service at a customer must begin within the time window defined by the earliest time and the latest time when the customer permits the start of service.

Time windows can be hard or soft. In the hard time window case, if a vehicle arrives too early at a customer, it is permitted to wait until the customer is ready to begin service. However, a vehicle is not permitted to arrive at a node after the latest time to begin service. In contrast, in the soft time window case, the time windows can be violated at a cost. We are going to focus on the hard time window variant where most of the research effort has been directed.

The VRPTW is a generalization of the vehicle routing problem (VRP) involving the added complexity of time windows. The VRP and a variety of its practical applications have been the subject of a wide body of research (see Magnanti 1981, Bodin, Golden, Assad and Ball 1983, Laporte and Nobert 1987, and Laporte 1992 for survey papers). A similar flurry of activity is currently being experienced in the VRPTW area. Figures 5.1 and 5.2 illustrate the temporal and spatial interplay of VRPTW routes which in many cases do not exhibit the geographical cohesiveness of VRP routes.

The special case in which the vehicle capacities are not binding is called the Multiple Traveling Salesman Problem With Time Windows (m–TSPTW). It is also a direct extension of the Single Depot Vehicle Scheduling Problem, a fixed schedule problem introduced in Section 2.1. The m–TSPTW involves the determination of routes that start at a single depot and cover a set of trips, each of which starts within a time window. Note that trips are considered as customers. In addition, there are no capacity constraints, since each trip satisfies these by definition, and vehicles traveling between trips are empty. An additional complexity encountered in some VRPTW consists of precedence relationships among certain customers. An example is the backhauling problem with time windows which involves both pick-ups and deliveries.

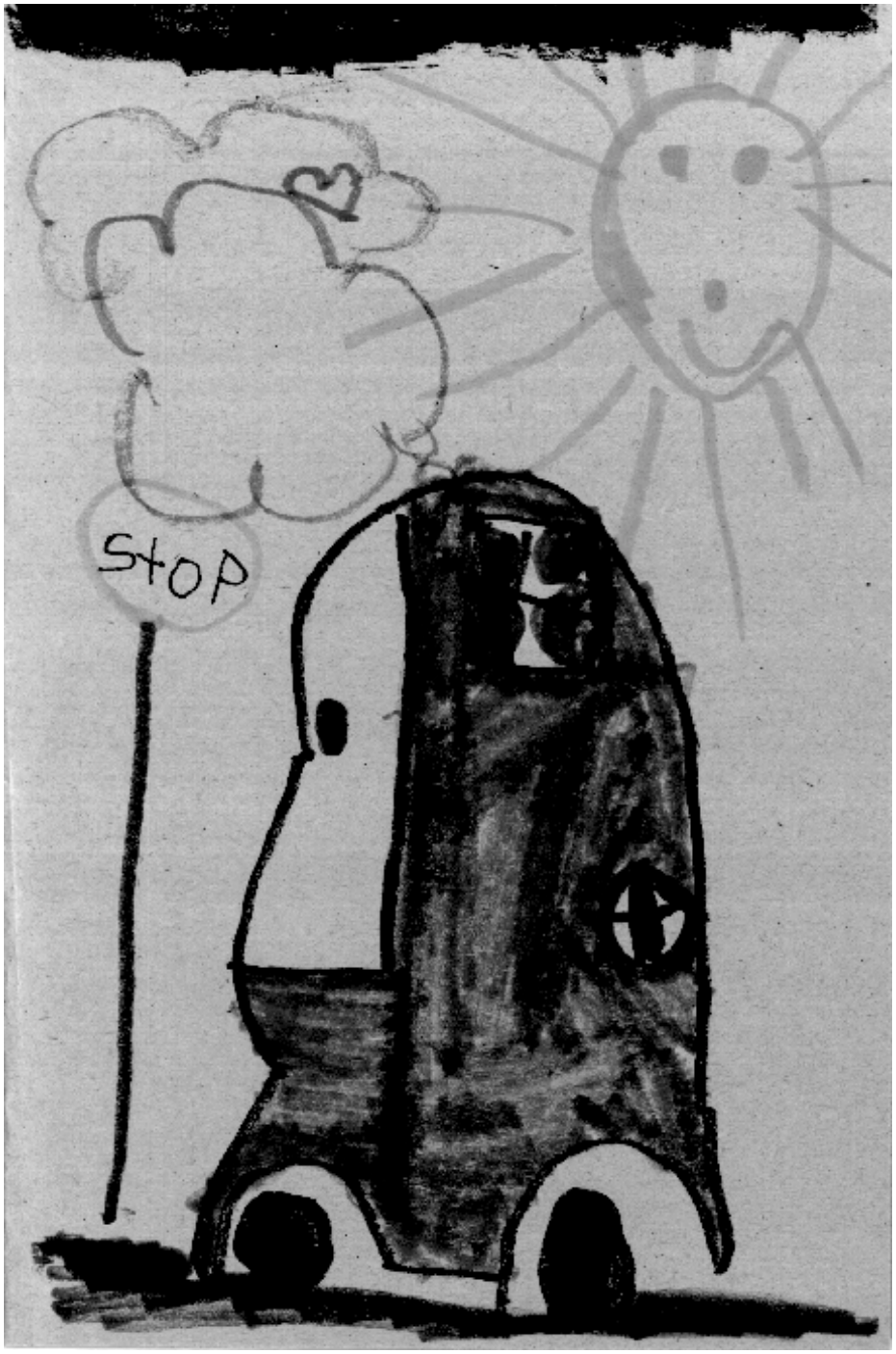


Figure 5.1: Léa's perception of the VRPTW

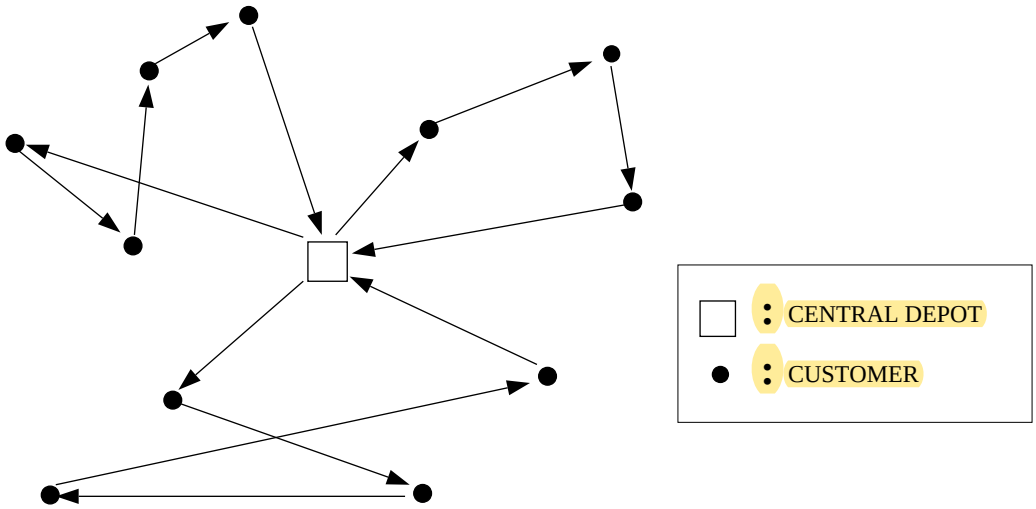


Figure 5.2: Temporal and spatial interplay of VRPTW routes

Time windows arise naturally in problems faced by business organizations which work on flexible time schedules. Specific examples of problems with hard time windows include bank deliveries, postal deliveries, industrial refuse collection and school-bus routing and scheduling. Among the soft time window problems, dial-a-ride problems constitute an important example.

Since the VRP is NP-hard, by restriction, the VRPTW is also NP-hard. In fact, even finding a feasible solution to the VRPTW when the number of vehicles is fixed a priori, is itself a NP-complete problem. This is a corollary of the result derived by Savelsbergh (1985) for the case of one uncapacitated vehicle, i.e., the TSPTW, mentioned in Section 3. Therefore, many algorithms have been developed under the assumption that the number of vehicles used is free. This implies the simultaneous determination of the vehicle fleet size and the optimal set of routes and schedules for this fleet. While most methods have assumed a single depot and a homogeneous fleet, they admit easy extensions. Such extensions with specific initial and final conditions for each vehicle of the fleet are explicitly given here in the formulation provided in Section 5.1. This generalized model and the solution approach are therefore well suited to be used in a planification phase as well as in an operational mode.

Literature: Given the inherent computational challenge of this problem class, the early work on the VRPTW has consisted of case studies (Pullen and Webb 1967, Knight and Hofer 1968, and Madsen 1976). One line of later research has been directed at the development

and analysis of heuristics capable of solving realistic size problems. A second line has been the design of effective optimal approaches for easier problem variants such as the m–TSPTW (Desrosiers, Soumis and Desrochers 1984). Recently, substantial progress has been made in removing computational barriers for optimal VRPTW algorithms (Kolen, Rinnooy Kan and Trienekens 1987, Desrochers Desrosiers and Solomon 1992, and Halse 1992). We present these approaches later in this section.

Heuristic Approaches: A number of route construction and route improvement procedures have been proposed. Route construction algorithms build a feasible solution by inserting at every iteration one unrouted customer into a current partial route. Sequential methods construct one route at a time, while parallel methods build several routes simultaneously. Insertion criteria based on maximum savings, minimum additional distance and time, and nearest neighbor concepts have been proposed. Iterative improvement methods start from a feasible solution and seek to improve it through a sequence of local modifications. They terminate when no more improvements are possible, generally at a local optimum. Branch exchange improvement methods implement local modifications by interchanging k arcs in the current solution with k arcs presently not used. A k-interchange is performed if and only if it is feasible and it generates an improved solution. Such exchanges can be performed within and/or between routes. These procedures terminate with a k-optimal solution, i.e., one that cannot be improved through any k-interchange.

Solomon (1987) was the first to generalize a number of VRP route construction heuristics to the VRPTW. His computational results show the effectiveness of a two-phase sequential algorithm based on insertion criteria. First, the best feasible insertion position for each unrouted customer is chosen with respect to the minimum additional distance and time required. Second, the customer to insert is selected using a measure of the maximum savings derived. Insertion based algorithms have also been successfully utilized in routing environments where additional precedence constraints exist, such as the Dial-A-Ride Problem which will be discussed in Section 6. Nevertheless, the worst-case ratio behavior of these approximation methods and of k-interchange heuristics on n-customer problems was proven to be at least of order n (Solomon 1986).

Route improvement procedures have been developed by Russell (1977), Cook and Russell (1978), and Baker and Schaffer (1986) for the VRPTW, and by Potvin, Lapalme and Rousseau (1989) specifically for the m-TSPTW. These methods are extensions to the VRPTW of the k-interchange heuristic of Lin and Kernighan (1973) and of the 2-opt and 3-opt branch exchange procedures of Lin (1965). While effective, these methods suffer from the major drawback of a very large processing time requirement. To alleviate this problem, Solomon, Baker and Schaffer (1988) have proposed a very efficient within route improve-

ment algorithm based on the OR-opt procedure (Or 1976). The algorithm restricts the branch exchanges examined to only those where one, two, or three adjacent customers are inserted in a later position on the route, between two currently consecutive customers. It also incorporates the lexicographic ordering for processing suggested by Savelsbergh (1985) and uses push-forward and backward shifts in customer arrival times for feasibility checks. The authors report that the algorithm obtained solutions which were within 1% of those derived by 3-opt and required only 26% of its processing time.

A competitive algorithm based on a parallel insertion heuristic and the OR-opt method has been proposed by Potvin and Rousseau (1993). Another competitive approach and a similar OR-opt implementation has been suggested by Thompson and Psaraftis (1989). Their method is based on the concept of cyclic k-transfers which involves transferring k demands from route I^j to route $I^{\delta(j)}$ for each j and a fixed integer k. The set of routes $\{I^r\}$, $r = 1, \ldots, m$ constitutes a feasible solution and δ is a cyclic permutation of a subset of $\{1, \ldots, m\}$. In particular, when δ has fixed cardinality $\mathcal{C}$, we obtain a $\mathcal{C}$-cyclic k-transfer. By allowing k dummy demands on each route, demand transfers can be performed among permutations rather than cyclic permutations of routes. Due to the complexity of the cyclic transfer neighborhood search, this is performed heuristically. In particular, similarly to OR-opt, only sets of demands belonging to adjacent customers are considered for transfer. Several variants of the generic algorithm are developed. One that proved effective consists of a mix of modules involving 2- and 3-cyclic 1-transfers, some of which consider dummy demands. The method has also been applied to problems to be discussed later in Section 6. Finally, Potvin, Kervahut and Rousseau (1992) have introduced a new Tabu search heuristic which appears to be quite good at improving Solomon's initial solutions.

Recently, Kontoravdis and Bard (1992) have developed a parallel heuristic and lower bounds for the fleet size. The approach is a greedy randomized adaptive search procedure (GRASP) which combines a greedy heuristic and randomization to construct a feasible solution. Local search is then used to improve upon this solution. During this phase, each route is considered for elimination, with routes having fewer customer examined first. The authors have also developed three different lower bounds. The first stems from the underlying Bin Packing structure created by the capacity constraints. The second is derived from the maximum clique of the associated incompatibility graph. An arc in this graph corresponds to a pair of customers which cannot be on the same route due to capacity or time window violations. The third also considers a Bin Packing problem generated by the time window constraints. Here, bin capacity is the length of the scheduling horizon, while the items are of size equal to either the time needed to go from a customer to its closest neighbor or the depot, or the time from the depot to the first customer on each route. The authors report computational results which indicate that on random and semi-clustered

problems, GRASP performed better than Solomon's heuristic. GRASP performed at least as well as Potvin and Rousseau's heuristic on all data sets and was superior for problems with a short scheduling horizon. The lower bounds have shown an encouraging performance as, for 14 problems from Solomon's data set, the gap between the best lower bound and the heuristic was zero.

Research on problems involving soft time windows has been scant. The objective function considered is to minimize a linear combination of total vehicle operating cost and total customer penalty due to missing any of the time windows. Sexton and Choi (1985) present a heuristic Benders decomposition algorithm for the single vehicle pick-up and delivery problem with soft time windows. It is a two-phase routing and scheduling procedure similar to the one developed in the dial-a-ride context. Koskosidis, Powell and Solomon (1992) have developed an iterative optimization-based heuristic which extends the generalized assignment heuristic of Fisher and Jaikumar (1981). The problem is decomposed into an assignment/clustering phase and a series of TSP with soft time windows components. At each iteration, the assignment of customers to vehicles is solved as a capacitated clustering problem. The routes and schedules are obtained by solving a TSP subproblem for each vehicle. These solutions also provide the improved approximate clustering costs to be used at the next iteration.

5.1 Problem Formulation

Notation: Let $N = \{1, \ldots, n\}$ be the set of customers and K, indexed by k, be the set of available vehicles to be routed and scheduled. Consider the graphs $G^k = (V^k, A^k)$, for all $k \in K$, each of them consisting of a set V^k of nodes and a set A^k of arcs. The set V^k consists of $N \cup \{o(k), d(k)\}$, where $o(k)$ and $d(k)$ represent respectively the origin-depot and the destination-depot of vehicle k, $k \in K$ (or any initial and final locations). The set A^k contains all feasible arcs, which is a subset of $V^k \times V^k$.

In the pick-up version, for each customer $i \in N$, there is a known demand p_i to be picked-up and a time window $[a_i, b_i]$ within which the customer permits the start of service. Define now a load parameter $\ell_i = p_i$, $i \in N$. For each vehicle $k \in K$, assume an initial load value $\ell_{o(k)}$, a time window $[a_{o(k)}, b_{o(k)}]$ at the starting location place as well as another time window at the destination location defined by $[a_{d(k)}, b_{d(k)}]$. Without loss of generality we assume that all the parameters are integer. Note that any VRPTW involving only deliveries can be reformulated as an equivalent pick-up problem by letting $\ell_i = d_i$, for all $i \in N$.

For each arc $(i, j) \in A^k, k \in K$, there is a cost c_{ij}^k and a travel time t_{ij}^k. As in previous sections, we assume that the service time at node i is included in the travel time t_{ij}^k, for all i. All the customers must be assigned to at most v vehicles, $v \leq |K|$, such that the

capacity Q^k of each vehicle used is not exceeded. Note that an arc $(i,j) \in A^k, k \in K$ can be eliminated by temporal constraints if $a_i + t_{ij}^k > b_j$, by capacity constraints if $\ell_i + \ell_j > Q^k$, or by other considerations.

The mathematical programming formulation to be introduced next involves three types of variables: flow variables X_{ij}^k, $(i,j) \in A^k, k \in K$, equal to 1 if arc (i,j) is used by vehicle k, and 0 otherwise; time variables T_i^k, $i \in V^k, k \in K$ specifying the start of service at node i; and load variables L_i^k, $i \in V^k, k \in K$, specifying the load of the vehicle k just after servicing node i.

Formulation: The problem of finding the minimal cost set of routes satisfying the VRPTW constraints can then be formulated as follows:

$$\text{Minimize} \sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k X_{ij}^k \tag{5.1}$$

subject to:

$$\sum_{k \in K} \sum_{j \in N \cup \{d(k)\}} X_{ij}^k = 1, \qquad \forall\, i \in N \tag{5.2}$$

$$\sum_{k \in K} \sum_{j \in N} X_{o(k),j}^k \leq v, \tag{5.3}$$

$$\sum_{j \in N \cup \{d(k)\}} X_{o(k),j}^k = 1, \qquad \forall\, k \in K \tag{5.4}$$

$$\sum_{i \in N \cup \{o(k)\}} X_{ij}^k - \sum_{i \in N \cup \{d(k)\}} X_{ji}^k = 0, \qquad \forall\, k \in K, \forall\, j \in V^k \setminus \{o(k), d(k)\} \tag{5.5}$$

$$\sum_{i \in N \cup \{o(k)\}} X_{i,d(k)}^k = 1, \qquad \forall\, k \in K \tag{5.6}$$

$$X_{ij}^k (T_i^k + t_{ij}^k - T_j^k) \leq 0, \qquad \forall\, k \in K, \forall\, (i,j) \in A^k \tag{5.7}$$

$$a_i \leq T_i^k \leq b_i, \qquad \forall\, k \in K, \forall\, i \in V^k \tag{5.8}$$

$$X_{ij}^k (L_i^k + \ell_j - L_j^k) \leq 0, \qquad \forall\, k \in K, \forall\, (i,j) \in A^k \tag{5.9}$$

$$\ell_i \leq L_i^k \leq Q^k, \qquad \forall\, k \in K, \forall\, i \in N \cup \{d(k)\} \tag{5.10}$$

$$L_{o(k)}^k = \ell_{o(k)}, \qquad \forall\, k \in K \tag{5.11}$$

$$X_{ij}^k \geq 0, \qquad \forall\, k \in K, \forall\, (i,j) \in A^k \tag{5.12}$$

$$X_{ij}^k \text{ binary}, \qquad \forall\, k \in K, \forall\, (i,j) \in A^k. \tag{5.13}$$

This is a nonlinear formulation where the objective function (5.1) represents the total cost. It is possible to include a fixed charge c of using a vehicle by adding it to all $c_{o(k),j}^k$, $j \in N$. To minimize the number of vehicles, c is assigned a very large value, while it

is set to zero if this number is free. To fix the number of vehicles used at exactly v, inequality (5.3) is replaced by an equality. Additional restrictions on the fleet composition can easily be formulated and incorporated in the above model. Constraints (5.2) impose that each customer be assigned exactly once to a vehicle route. Constraints (5.4)–(5.6) describe the flow on the path that vehicle k will use. Constraints (5.7) and (5.8) ensure feasibility of the time schedule while constraints (5.9)–(5.11) guarantee feasibility of the loads. Binary conditions on the flow variables are given in (5.13).

In the presence of the binary conditions on the flow variables, constraints (5.7) and (5.9) can be linearized and rewritten as:

$$T_i^k + t_{ij}^k - T_j^k \leq \Big(1 - X_{ij}^k\Big) M_{ij}^k, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{5.7a}$$

$$L_i^k + \ell_j - L_j^k \leq \Big(1 - X_{ij}^k\Big) Q^k, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{5.9a}$$

where M_{ij}^k are large constants. Note that one can replace the large constant M_{ij}^k in constraints (5.7a) by the value $\max\{b_i + t_{ij}^k - a_j, 0\}$, $(i,j) \in A^k$, and only require constraints (5.7) or (5.7a) for arcs $(i,j) \in A^k$, such that $M_{ij}^k > 0$; when $b_i + t_{ij}^k \leq a_j$, these constraints are satisfied for all values of T_i^k, T_j^k and X_{ij}^k.

This very general formulation can account for a homogeneous as well as a heterogeneous fleet of vehicles, the single and multiple-depot cases, and even for optimization situations requiring specific initial conditions for each vehicle. This is a practical model useful in an operational mode. The definition of a single time window per customer can also be relaxed to include multiple time windows. This may necessitate changing the objective function to account for preferred service times. Multiple time windows have primarily been examined in the context of the multi-period vehicle routing problem where the time windows are full days. Each customer must receive a specified number of visits within the planning horizon. This problem has been surveyed by Solomon and Desrosiers (1988). Incidentally, this generalization can be treated by Dantzig-Wolfe decomposition, State-Space and Lagrangean relaxation schemes which use time window constrained shortest paths as sub-structures.

We consider next two special cases of the VRPTW.

The Multiple Traveling Salesman Problem with Time Windows: Eliminating constraints (5.9)–(5.11) yields an m–TSPTW formulation. This is a VRPTW involving uncapacitated vehicles. It is also a natural extension of the fixed schedule problem which occurs when the time windows consist of a single value, i.e., the arrival time at each customer is prespecified. Aircraft, ship, engine, school bus and urban bus scheduling have

proven to be very fruitful grounds for dealing with time window constraints. To our knowledge, Appelgren (1969) was the first author to use a Dantzig-Wolfe decomposition/Column Generation algorithm to solve a ship scheduling problem with time window constraints. The resulting master problem is a set partitioning problem, while the subproblem consists of a shortest path problem over an expanded network arising from the discretization of the time windows. Optimal integer solutions are not guaranteed in this first paper, nor in the second one where the author wrote: *"There are fundamental difficulties in combining these integer programming methods with the Dantzig-Wolfe decomposition, since the constraints generated in the master program have to be taken into account in the solution of the subprograms."* (Appelgren 1971). As reported in Section 2, these difficulties are all removed if, first, master program solutions are transferred back to the original formulation on which the decomposition method is applied, and second, all branching decisions are taken on this original structure. The decomposition process can then be reapplied on the modified structure. Additional details are provided in following sections.

Another approach is that of Levin (1971) who presents and directly solves an integer programming formulation including discretized time windows for the minimization of aircraft fleet size. The same integer programming formulation has been later used by Swersey and Ballard (1983) to solve school-bus scheduling problems. The authors have been successful in manually adjusting the few fractional variables obtained, without increasing the number of bus fleet size. Other heuristics have been proposed for school-bus scheduling by Orloff (1976), Bodin and Berman (1979), Graham and Nuttle (1986), and Desrosiers, Ferland, Rousseau, Lapalme and Chapleau (1986), and for aircraft scheduling by Martin (1981). Exact algorithms for the m–TSPTW are presented in the following subsections.

The Simple Backhauling Problem with Time Windows: The vehicle routing problem with backhauls under time window constraints is an extension of the VRPTW. Linehaul customers are located at sites which are to receive a quantity of goods from the depot, while backhaul customers are located at sites which have to send a quantity of goods back to the depot. In the simple backhaul problem, all deliveries must be made before any pick-ups occur. Heuristics have been designed only for the version without time windows at customers, and most of them are adaptations of insertion methods created for the VRP (Deif and Bodin 1984, Golden and Assad 1984, Casco, Golden and Wasil 1988, and Goetschalckx and Jacobs-Blecha 1989). To our knowledge, only one mathematical optimization approach has been successfully implemented by Yano, Chan, Ritcher, Cutler, Murty and McGettigan (1987). It is based on a set covering formulation and a priori enumeration of a restricted set of feasible routes.

Let us now modify formulation (5.1)–(5.13) to account for the simple backhauling problem. For this, define $N = N^D \cup N^P$ to be the set of customers N, divided into a set of delivery locations N^D, and a set of pick-up locations N^P. The delivery demands are given by $\ell_i = d_i$, $i \in N^D$, while the pick-up demands are fixed to $\ell_i = p_i$, $i \in N^P$. The network structure is also partitioned in two main parts: the delivery level (which needs to be addressed first), and the pick-up level. The only difference with formulation (5.1)–(5.13) is the replacement of (5.10) by

$$\ell_i \leq L_i^k \leq Q^k, \quad \forall\, k \in K, \quad \forall\, i \in N^D \tag{5.10a}$$

$$Q^k + \ell_j \leq L_i^k \leq 2Q^k, \quad \forall\, k \in K, \quad \forall\, i \in N^P \cup \{d(k)\}. \tag{5.10b}$$

At the delivery level, constraints (5.10) and (5.10a) are identical. Due to the capacity window constraints, at any customer i of N^D, the total quantity L_i^k delivered since the start is at most equal to Q^k. At the end of the delivery portion, any vehicle k should be empty, or equivalently, it should allow Q^k new units of capacity. Crossing from the delivery level to the pick-up level on an arc $(i, j) \in A^k$, $i \in N^D$ and $j \in N^P$, the load L_j^k is set to the lower bound of the capacity window at customer j, i.e., at the value

$$L_j^k = \max\{L_i^k + \ell_j,\, Q^k + \ell_j\} = Q^k + \ell_j = Q^k + p_j, \quad i \in N^D,\, j \in N^P.$$

Constraints (5.9) and (5.10b) describe such conditions. As stated before, waiting until the permitted start of service is allowed, that is, if customer j follows customer i, and the arrival time $T_i^k + t_{ij}$ at customer j is less than a_j^k on vehicle k, the service only starts at time $T_j^k = a_j = \max\{T_i^k + t_{ij}, a_j\}$. The vehicle load variables act in a similar fashion.

Note finally that (5.9) need not be defined for arcs crossing from N^D to N^P since they are always satisfied. The exact algorithms designed for the VRPTW can then be utilized for the backhauling case where all the deliveries are performed before the start of pick-ups (Gélinas, Desrochers, Desrosiers and Solomon 1992). More complex algorithms are however necessary when the problem involves pick-up or delivery requests performed in any order as well as for the problem which involves simultaneous pick-up and delivery requests at the same customer. Halse (1992) describes some exact and approximate algorithms for these cases.

All the optimization algorithms for the VRPTW or its special cases, such as the m-TSPTW and the backhauling problem, employ branch-and-bound enumeration trees. In the next subsections, we will review several approaches used to compute lower bounds. To improve the efficiency of these approaches, the time windows' width can be reduced using the conditions presented in Section 3.2.

5.2 Network and Linear Programming Relaxations

The network relaxation was analyzed for the m–TSPTW with a single depot and a homogeneous fleet of vehicles. It is obtained by dropping constraints (5.9)–(5.11) and then relaxing the scheduling and time window constraints (5.7)–(5.8). If $a_i = b_i$, for all $i \in N$, then the problem reduces to the fixed schedule problem and the network relaxation produces an optimal solution. The quality of the bound deteriorates with an increasing width of the time windows.

The network relaxation bound is often of poor quality as there is usually a gap on the number of vehicles. Two branching rules have been proposed: branching on flow variables and branching by partitioning the time windows. Using the former branching rule in the case of very tight time windows, Soumis, Desrosiers and Desrochers (1985) have optimally solved urban-bus scheduling problems involving up to 150 trips. However, for school-bus scheduling problems where the time windows become wider, the branch-and-bound tree grows too rapidly in size for the approach to be practical (Desrosiers, Soumis, Desrochers and Sauvé 1986). Hence, the authors used the latter branching rule which proved successful in handling the wider time intervals. Nevertheless, it was concluded that the network relaxation approach was inferior to the Dantzig-Wolfe decomposition scheme which will be presented in the next section.

The linear relaxation bound is obtained by replacing constraints (5.7) and (5.9) by their linearized versions (5.7a) and (5.9a) and by discarding the binary requirements (5.13). The resulting linear program is the above network flow problem with the additional time and capacity constraints. Following the work of Desrosiers, Sauvé and Soumis (1988), when setting the time variables at the center of the time window, i.e., $T_i^k = (a_i + b_i)/2$, the load variables similarly at $L_i^k = (\ell_i + Q^k)/2$, $i \in N, k \in K$, constraints (5.7a) and (5.9a) are satisfied if $X_{ij}^k \leq 1/2$, for all $(i,j) \in A^k$. It is thus relatively easy to obtain a fractional optimal linear programming solution to problem (5.1)–(5.13) for which the time and capacity constraints are inactive. In most cases, this bound is thus no better than the network relaxation bound.

5.3 Dantzig-Wolfe Decomposition / Column Generation

The method to be presented next has sometimes been referred to as a Dantzig-Wolfe decomposition or a column generation technique. As reported previously, early work with this approach dates back to Appelgren (1969, 1971) for a vessel scheduling problem with time window constraints. A modified version of the subproblem structure as well as the first exact branch-and-bound scheme was originally applied to the m–TSPTW (Desrosiers,

Soumis and Desrochers 1984), and subsequently to many vehicle routing and crew scheduling problems (see Sections 2, 6 and 7), including the VRPTW (Desrochers, Desrosiers and Solomon 1992).

Theoretical aspects: In the presence of binary constraints on the flow variables, the nonlinear problem (5.1)–(5.13) can be linearized to provide an integer problem denoted by IP. The classical Dantzig-Wolfe decomposition for linear programming can thus be applied to the linear relaxation LP of that integer problem. The decomposition scheme involves a master problem MP and a subproblem SP. If the binary requirements are kept in the subproblem definition and if the subproblem possesses the Integrality Property, i.e., the optimal value of SP, when solved as a linear program, is not altered by dropping the binary constraints, then $Z_{LP} = Z_{MP}$. If however SP does not possess the Integrality Property, then SP can be solved as an integer problem. Theoretically, the optimization can be done by solving a linear program defined on the convex hull of SP, or it can be done directly, as an integer problem, by some other means. Solving such a SP as an integer problem allows for a possible partial reduction of the integrality gap between Z_{IP} and Z_{LP}, that is

$$Z_{LP} \leq Z_{MP} \leq Z_{IP}.$$

To obtain an optimal integer solution to the original problem IP, branching decisions can be taken on the original variables of IP, and the Dantzig-Wolfe decomposition process embedded into a branch-and-bound enumeration tree. These features are exploited in the following Dantzig-Wolfe decomposition applied to the VRPTW.

A Dantzig-Wolfe Decomposition: The decomposition presented here parallels the scheme introduced in Section 2.3 for the Multiple Depot Vehicle Scheduling Problem. The master problem is given by (5.1)–(5.3), i.e., the objective function (5.1), the covering of each customer $i \in N$ exactly once (5.2), and the constraint on the total number of vehicles (5.3). The subproblem involves a modified objective function $\sum_{k \in K} \sum_{(i,j) \in A^k} \overline{c}_{ij}^k X_{ij}^k$, where the coefficients $\overline{c}_{ij}^k$ will be defined later, and constraint set (5.4)–(5.13). That is, the path constraints (5.4)–(5.6), constraints (5.7) and the time windows (5.8), which together insure the feasibility of the time schedule, also constraints (5.9), the capacity intervals (5.10) along with the initial load conditions (5.11), which guarantee the feasibility of the load, and finally the binary requirements (5.13) on the flow variables $X_{ij}^k, k \in K$, $(i,j) \in A^k$.

The Subproblem: The first observation to make on the subproblem structure is that it decomposes into $|K|$ disjoint subproblems, one for each vehicle. The second observation

characterizes each subproblem as an elementary shortest path problem with time and capacity constraints, its solution being obtained on a bounded polyhedron. Consequently, the flow variables X_{ij}^k can be expressed as a non-negative convex combination of the paths generated from the corresponding subproblem. Hence, the usual convexity constraints will appear in the master problem definition. We will show later in this section how they can be removed in some special cases, one of them being the homogeneous fleet and single depot case.

In Section 2.3, the subproblem for the MDVSP can be viewed as a classical shortest path problem defined on an acyclic network. This problem possesses the integrality property, i.e., a linear programming algorithm applied on the problem gives rise to an integer solution. Therefore, solving the shortest path problem using a dynamic programming algorithm is equivalent to using a linear programming algorithm, and thus the bound obtained from the Dantzig-Wolfe decomposition is equal to the optimal value of the linear relaxation of the MDVSP. In the case of the VRPTW, the linearized subproblem together with the integrality requirements is an elementary shortest path problem with time windows and capacity constraints. This subproblem does not possess the integrality property. Consequently, solving it as a nonlinear integer program permits a reduction of the integrality gap between the optimal solution of the linearized version of formulation (5.1)–(5.12) and the optimal integer VRPTW solution to (5.1)–(5.13).

As mentioned in Section 4.1, the elementary shortest path problem with resource constraints is a NP-hard problem for which no polynomial or pseudo-polynomial algorithms are known. It was also shown in Section 4.2 that when non-elementary path solutions for the subproblem are allowed, i.e., solutions where a path may contain cycles of finite duration and/or load due to the presence of time windows and capacity intervals, pseudo-polynomial algorithms have been developed for its solution.

The introduction of these paths containing cycles augments the size of the set of admissible columns generated for the master problem. The lower bound on the VRPTW solution obtained by solving the coordinating master problem may decrease, yet it can be slightly improved if the constrained shortest path problem is solved using a 2–cycle elimination procedure within the solution process (Houck, Picard, Queyranne and Vemuganti 1980, Kolen, Rinnooy Kan and Trienekens 1987, and Desrochers, Desrosiers and Solomon 1992). However, the covering constraints (5.2) of the master problem ensure that each customer is visited exactly once. Hence, the supplementary columns containing cycles cannot appear in any integer solution of the master problem, whether optimal or not, and they are automatically constrained to take a zero value during the branch-and-bound process.

The Master Problem: Let Ω_B^k be the set of feasible paths of subproblem $k, k \in K$. Each $p \in \Omega_B^k$ corresponds to an elementary path which can be described using *binary* values $(x_{ijp}^k, (i,j) \in A^k)$. Any solution X_{ij}^k to the master problem can be expressed as a non-negative convex combination of a finite number of elementary paths, i.e.,

$$\begin{aligned} X_{ij}^k &= \sum_{p \in \Omega_B^k} x_{ijp}^k \, \theta_p^k, && \forall\, k \in K,\ \forall\, (i,j) \in A^k \\ \sum_{p \in \Omega_B^k} \theta_p^k &= 1, && \forall\, k \in K \\ \theta_p^k &\geq 0, && \forall\, k \in K,\ \forall\, p \in \Omega_B^k. \end{aligned}$$

Since the solution process of the constrained shortest path subproblem may generate paths containing finite cycles, such a path p can be described using *integer* flow values x_{ijp}^k, $(i,j) \in A^k$. Let Ω_I^k be the set of the additional paths and define $\Omega^k = \Omega_B^k \cup \Omega_I^k$. Any solution X_{ij}^k to the master problem can still be expressed as a non-negative convex combination of paths, i.e.,

$$\begin{aligned} X_{ij}^k &= \sum_{p \in \Omega^k} x_{ijp}^k \, \theta_p^k, && \forall\, k \in K,\ \forall\, (i,j) \in A^k \\ \sum_{p \in \Omega^k} \theta_p^k &= 1, && \forall\, k \in K \\ \theta_p^k &\geq 0, && \forall\, k \in K,\ \forall\, p \in \Omega^k. \end{aligned}$$

Next, define the parameters c_p^k, a_p^k and b_p^k the following way:

$$\begin{aligned} c_p^k &= \sum_{(i,j) \in A^k} c_{ij}^k x_{ijp}^k, && \forall\, k \in K,\ \forall\, p \in \Omega^k \\ a_{ip}^k &= \sum_{j \in N \cup \{d(k)\}} x_{ijp}^k, && \forall\, k \in K,\ \forall\, i \in N,\ \forall\, p \in \Omega^k \\ b_p^k &= \sum_{j \in N} x_{o(k),jp}^k, && \forall\, k \in K,\ \forall\, p \in \Omega^k. \end{aligned}$$

We make these substitutions into (5.1)–(5.3) and rearrange the summation order. These substitutions together with the convex combination constraints gives the revised formulation of the master problem:

$$\text{Minimize} \sum_{k \in K} \sum_{p \in \Omega^k} c_p^k \theta_p^k \tag{5.14}$$

subject to:

$$\sum_{k \in K} \sum_{p \in \Omega^k} a_{ip}^k \theta_p^k = 1, \qquad \forall\, i \in N \tag{5.15}$$

$$\sum_{k\in K}\sum_{p\in\Omega^k} b_p^k\theta_p^k \le v, \tag{5.16}$$

$$\sum_{p\in\Omega^k} \theta_p^k = 1, \qquad \forall\, k \in K \tag{5.17}$$

$$\theta_p^k \ge 0, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k. \tag{5.18}$$

The coefficient c_p^k, $k \in K$, $p \in \Omega^k$ is the cost of the path $p \in \Omega^k$. Coefficient a_{ip}^k, $k \in K$, $i \in N$, $p \in \Omega^k$ is a constant taking a non-negative integer value: it indicates the number of times customer i is visited by the vehicle k on path p. Coefficient b_p^k, $k \in K$, $p \in \Omega^k$, takes only binary values: value 1 if vehicle k visits at least one customer in path p, and 0 otherwise. In the latter case, path $p \in \Omega^k$ only uses the arc $(o(k), d(k))$. In (5.17), the coefficient of θ_p^k is equal to 1, for all $k \in K$ and $p \in \Omega^k$. In fact, this constraint corresponds to (5.4), or equivalently to (5.6), in the original formulation, i.e.,

$$\sum_{j\in N\cup\{d(k)\}} X_{o(k),j}^k = \sum_{i\in N\cup\{o(k)\}} X_{i,d(k)}^k = 1.$$

The mathematical formulation (5.14)–(5.18) is then the linear relaxation of a set partitioning type problem with an additional constraint on the total number of vehicles and a set of convex combination constraints.

In the case of a single depot and a homogeneous fleet of vehicles, together with the same initial conditions for all vehicles, the sets $\Omega^k, k \in K$ are all identical, i.e., $\Omega = \Omega^k$, for all $k \in K$. The subproblem solution involves only a single network $G = (V, A) = (V^k, A^k)$, for all $k \in K$. In this case, the convex combination constraints (5.17) can be aggregated. First, let $\theta_p = \sum_{k\in K} \theta_p^k$; then $\sum_{p\in\Omega} \theta_p = |K|$ and index k can then be removed from (5.14)–(5.16) and (5.18). Since the aggregated constraint counts the total number of used and unused vehicles, it is redundant when compared to constraint (5.16) limiting the number of vehicles used to $v, v \le |K|$. The resulting formulation given below becomes the classical linear relaxation of the set partitioning formulation with the additional restriction on the number of vehicles routed and scheduled:

$$\text{Minimize} \sum_{p\in\Omega} c_p\theta_p \tag{5.19}$$

subject to:

$$\sum_{p\in\Omega} a_{ip}\theta_p = 1, \qquad \forall\, i \in N \tag{5.20}$$

$$\sum_{p\in\Omega} \theta_p \le v, \tag{5.21}$$

$$\theta_p \ge 0, \qquad \forall\, p \in \Omega. \tag{5.22}$$

In the presence of multiple depots and a heterogeneous fleet size, similar aggregations can be performed as long as the conditions are identical for all vehicles in the same group. One constraint is retained for each group and it describes the number of available vehicles within that group. The route assignment to a specific vehicle number within a group can be done a posteriori, i.e., after the solution is obtained.

Linear Relaxation Solution: The solution process consists of two levels. At the first level, the coordinating master problem, a linear program, is optimized using the current columns. At the second level, the subproblems are solved to find minimum marginal cost columns.

If a path with negative marginal cost is found, the corresponding column is added to the known ones in the master problem and the solution process returns to the first level. Otherwise the current solution is optimal for the master problem and it provides a lower bound on the optimal integer solution of the constrained multi-commodity flow model.

Solving the master problem (5.14)–(5.18) over the current set of columns $\Omega' \subset \cup_{k\in K}\Omega^k$ by using the simplex method gives the dual variables α_i, $i \in N$, β and γ^k, $k \in K$ associated with constraints (5.15)–(5.17), respectively, necessary for the solution of the subproblem. The marginal cost $\overline{c}_p^k$ of path p for vehicle $k \in K$ is given by:

$$\overline{c}_p^k = c_p^k - \sum_{i\in N} \alpha_i a_{ip}^k - \beta b_p^k - \gamma^k \tag{5.23}$$

$$= \sum_{(i,j)\in A^k} c_{ij}^k x_{ijp}^k - \sum_{i\in N} \alpha_i \Big(\sum_{j\in N\cup\{d(k)\}} x_{ijp}^k \Big) - \beta \Big(\sum_{j\in N} x_{o(k),jp}^k \Big) - \gamma^k \Big(\sum_{j\in N\cup\{d(k)\}} x_{o(k),j}^k \Big). \tag{5.24}$$

The arc set A^k, $k \in K$, can be partitioned the following way: $i \in N$ and $j \in N \cup \{d(k)\}$, or $i = o(k)$ and $j \in N$, or $i = o(k)$ and $j = d(k)$. Hence, the marginal cost $\overline{c}_p^k$ can be rewritten as

$$\sum_{i\in N} \sum_{j\in N\cup\{d(k)\}} (c_{ij}^k - \alpha_i) x_{ijp}^k + \sum_{j\in N} (c_{o(k),j}^k - \beta - \gamma^k) x_{o(k),jp}^k + (c_{o(k),d(k)}^k - \gamma^k) x_{o(k),d(k),p}^k \tag{5.25}$$

Therefore, we can define the marginal cost $\overline{c}_{ij}^k$, $(i,j) \in A^k$, $k \in K$, of an arc as:

$$\overline{c}_{ij}^k = \begin{cases} c_{ij}^k - \alpha_i & \text{if} \quad i \in N \\ c_{ij}^k - \beta - \gamma^k & \text{if} \quad i = o(k) \text{ and } j \in N \\ c_{ij}^k - \gamma^k & \text{if} \quad i = o(k) \text{ and } j = d(k). \end{cases} \tag{5.26}$$

Finding the minimum marginal cost column over the set Ω^k results in the following optimization problem:

$$\begin{aligned} \text{Minimize} \quad & \sum_{(i,j)\in A^k} \overline{c}_{ij}^k X_{ij}^k \\ \text{subject to:} \quad & (5.4)\text{–}(5.13). \end{aligned}$$

This optimization problem corresponds to an elementary shortest path problem with time and capacity windows. We have described solution strategies for relaxed versions of this problem in Section 4.5.

Solving the master problem, i.e., the linear program (5.14)–(5.18), is accelerated by generating several columns simultaneously, since the one-time solution of a subproblem by dynamic programming not only produces the minimum marginal cost column, but also many other columns of negative marginal cost.

Optimal Integer Solution: To obtain an optimal integer solution to the original multi-commodity flow formulation (5.1)–(5.13), one can try to find an integer solution to the master problem. This is a node-path formulation of the VRPTW equivalent to the original arc-node formulation. As pointed out by Appelgren (1969, 1971) in the context of vessel scheduling, as well as by Chvátal (1983, pp. 197–198) in the context of the Cutting-Stock Problem, it is not an easy task to find an optimal integer solution to a problem solved using a column generation scheme. All the cuts imposed along with the branching decisions taken must be compatible with the decomposition scheme, i.e., with the master and the subproblem structures. A branch-and-bound tree must be explored, where additional columns might be generated in each branch. The main difficulty stems from the fact that fixing a fractional optimal basic variable at zero results in the regeneration of the corresponding column. This can be avoided if the subproblem is allowed to generate second, third, ..., n^{th} best solutions (see Hansen, Jaumard and Poggi de Aragão 1991, and Maculan, Michelon and Plateau 1992).

The simplest approach to obtain an optimal integer solution is however to take all the decisions on the original formulation (5.1)–(5.13). Cuts and branching decisions can be taken on the number of vehicles used, on one or several binary flow variables, on time variables, or on load variables. The decomposition scheme is then reapplied on each branch. Local decisions, i.e., decisions which concern only a single path, such as fixing a flow variable at 0 or at 1, or dividing the time window of a time variable, are carried out directly on the subproblem networks without changing the shortest path solution approach. Global

decisions, i.e., those which concern more than one path, such as an integer cut on the total cost, or cuts or branching decisions on the number of vehicles used of each type at each depot, stay in the master problem definition. In other words, cuts and branch-and-bound decisions are taken on the multi-commodity flow model using flow and resource variables. These decisions are thereafter transferred to the adequate structure, i.e., the master problem or the subproblem network parameters.

Computational Results: This approach proved to be very successful on a variety of practical applications. In school bus fleet planning (the problem of assigning buses to trips), which is a m–TSPTW, problems with up to 151 trips were solved in Desrosiers, Soumis and Desrochers (1984), while the algorithm has obtained the optimal solution to a real world 279-trip problem (Girard 1990). These problems involved a single depot and a homogeneous fleet. In the context of the transportation of elderly and handicapped persons, a m-TSPTW model was also used as an approximate model after a grouping phase of the initial requests. The Dantzig-Wolfe decomposition/column generation algorithm was then used to solve problems of more than 300 trips (Dumas, Desrosiers and Soumis 1989, see also Section 6.3). In this case, more than thirty different initial conditions on the vehicles were treated, i.e., more than thirty different subproblems.

For the single depot VRPTW involving a homogeneous fleet, the algorithm found optimal solutions to a number of 100-customer problems (Desrochers, Desrosiers and Solomon 1992), mainly using branching strategies based on the value of the flow variables. This decomposition scheme has also been applied to the backhauling problem to produce optimal solutions to randomly generated problems with up to 100 customers (Gélinas, Desrochers, Desrosiers and Solomon 1992). In this case, the branching decisions were taken on the time variables, a strategy which reduced the size of the search tree as well as the computational time by a factor of two.

An empirically observed property is that the optimal integer solution value of the Set Partitioning Problem resulting from the use of a Dantzig-Wolfe decomposition or a column generation scheme is very close to its linear programming relaxation. Bramel and Simshi-Levi (1993) showed that the relative gap between fractional and integer solutions becomes arbitrarily small as the number of customers increases. This makes the branch-and-bound process very efficient.

The approach presented here has also been used on more complex problems. Those involving multiple depot sites and a heterogeneous vehicle fleet were solved using many subproblem structures, one for each depot/vehicle type (see Dumas, Desrosiers and Soumis

1989, and Haouari, Dejax and Desrochers 1991). By an adequate design of the network, the approach can also be generalized to include multiple time window constraints per customer, to determine coffee breaks or meal locations within flexible time periods, and to deal with many other practical restrictions (Sansó, Desrochers, Desrosiers, Dumas and Soumis 1990). For much more complex models involving more than two resources and solved using the Dantzig-Wolfe decomposition/column generation approach, the reader is referred to Section 7 on fleet planning, crew scheduling and crew rostering problems.

5.4 State-Space Relaxation and Integer Programming Relaxation

Kolen, Rinnooy Kan and Trienekens (1987) were the first to describe an optimization approach and to present computational results for the VRPTW, for the case of a single depot housing a homogeneous fleet. Their method was partially inspired by the work of Christofides, Mingozzi and Toth (1981a) on the classical VRP. The algorithm involves a branch-and-bound tree, where decisions are taken on fixing a flow variable at 0 or 1, and embeds state-space relaxation, which is an adaptation of the method described previously in Section 3.4 for the TSPTW.

Each node in the search tree corresponds to a set of fixed routes starting and finishing at the depot, or to at most one partial route starting at the depot, and to a set of customers that are forbidden to be inserted next on the partial route. In the branching step, a customer who does not already belong to one of the fixed routes or the partial route and is not forbidden to belong to the partial route is selected. This creates a binary decision on the corresponding flow variable consisting of whether or not to extend the partial route. When there is no current partial route, the decision is whether or not to initiate a new route with the selected customer as the first customer on the route.

The state-space relaxation method then calculates a lower bound on all feasible extensions of the incomplete solution. It relaxes the condition stating that each customer not yet on a route must be served exactly once. The least cost extension of the current incomplete solution to a set of routes is however computed so that the total load on all these routes is equal to $\sum_{i \in N} \ell_i$, and so that each route has a different last customer, i.e., the one visited just prior to the return to the depot.

A State-Space Relaxation: At the first node of the search tree, the states of the dynamic programming algorithm are of the form $STATE(i, \ell, k)$, $i \in N$, $0 \leq \ell \leq \sum_{i \in N} \ell_i$, $0 \leq k \leq v$ where each directed path from $STATE(0, 0, 0)$ to $STATE(i, \ell, k)$ corresponds

to a set of k routes with total load ℓ, and with different last customers, each one belonging to $\{1, \ldots, i\}$. The transition cost from $STATE(i, \ell, k)$ to $STATE(i+1, \ell, k)$ is zero: this is the case when customer $i+1$ is not covered by the k routes. The transition cost from $STATE(i, \ell, k)$ to $STATE(i+1, \ell+\ell', k+1)$ is computed using the value $F(i+1, \ell')$ of a shortest path problem with time windows originating at the depot node and having node $i+1$ as the last customer, and with total load on that path equal to ℓ'. The lower bound at the root node is given by

$$\min_{1 \leq k \leq v} STATE(n, \sum_{i \in N} \ell_i, k). \tag{5.27}$$

At the other nodes of the branch-and-bound tree, if there is no partial route currently fixed, then a set of routes is already fixed and the problem to be solved is identical to the root node problem, except that it is of a smaller dimension. In the case where a partial route is fixed, exactly one of the routes appearing in the lower bound must be an extension of the partial route. This implies that new states must be added to the original state space. These are defined in a similar manner and the reader is referred to the original paper for details.

Improvement of the lower bound is obtained using a time constrained shortest path algorithm with 2–cycle elimination and node penalties for uncovered customers. These penalties are obtained heuristically using subgradient optimization.

An Integer Programming Relaxation: This lower bound is closely related to the set partitioning type formulation (5.19)–(5.22) given in the case of a single depot and a homogeneous vehicle fleet. Rather than relaxing the binary conditions on the variables θ_p, $p \in \Omega$, Kolen, Rinnooy Kan and Trienekens (1987) relax the constraints (5.20) which require that each customer must be visited exactly once, and replace them by two types of constraints: a weighted row aggregation constraint on the one hand, and a different last customer constraint on the other hand. Define f_{ip}, $i \in N$, $p \in \Omega$ to be a binary constant taking value 1, if customer i is the last customer visited on route p, and 0, otherwise. Then, the integer programming relaxation is the following:

$$\text{Minimize} \sum_{p \in \Omega} c_p \theta_p \tag{5.28}$$

subject to:

$$\sum_{p \in \Omega} \Big(\sum_{i \in N} \ell_i a_{ip} \Big) \theta_p = \sum_{i \in N} \ell_i, \tag{5.29}$$

$$\sum_{p\in\Omega} f_{ip}\theta_p \leq 1, \qquad \forall\, i \in N \tag{5.30}$$

$$\sum_{p\in\Omega} \theta_p \leq v, \tag{5.31}$$

$$\theta_p \quad \text{binary}, \qquad \forall\, p \in \Omega. \tag{5.32}$$

Problem (5.28)–(5.32) is then solved to optimality, directly using dynamic programming at the first node of the tree. It is evident that the set Ω of feasible routes can be dramatically reduced, since only the least cost route among all those having the same load and last customer is retained in the state-space relaxation procedure. The optimal *integer* solution to (5.28)–(5.32) provides a lower bound on the VRPTW optimal solution cost. This solution might not be feasible since some customer may not be visited exactly once. Note that there is no dominance relation between the lower bounds provided by this state-space relaxation and by the linear programming relaxation of the set partitioning type formulation, respectively.

At the other nodes of the search tree, if there is no partial route currently fixed, the same formulation (5.28)–(5.32) is solved, except that some θ_p variables are already fixed at 1, while some others are removed because of a capacity constraint, or because some customers must be forbidden. In the case of a node in the search tree with a partial route already fixed, define $e_p, p \in \Omega$ a binary constant taking value 1, if route p is an extension of that partial route, and 0, otherwise. Then, the following supplementary constraint must be added to formulation (5.28)–(5.32):

$$\sum_{p\in\Omega} e_p\theta_p = 1. \tag{5.33}$$

This constraint claims that exactly one of the routes appearing in this branch must be an extension of the partial route.

The reader is also refered to Bianco, Mingozzi, Ricciardelli and Spadoni (1989) who use a similar integer programming relaxation solved by dynamic programming in the context of urban crew scheduling. Such a problem is further addressed in Section 7.

Computational Results: Optimal solutions are reported for problems with up to 15 customers. In our point of view, the limitations of this approach are essentially due to the size of the state space, even if this has been reduced by the use of a state-space relaxation procedure. Furthermore, this methodology cannot easily incorporate new constraints, multi-vehicle types or multi-depot problems, except by using an enlarged state space.

5.5 Lagrangean Relaxation Methods

Lagrangean relaxation can be applied to various VRPTW formulations in many ways. On the one hand, the difficult constraints (time window and capacity constraints) can be relaxed so that the resulting Lagrangean subproblem is easy to solve. On the other hand, part of the pure network flow constraints may be relaxed, thus retaining the complicating constraints in the Lagrangean subproblem. We will describe such approaches after we first present some theoretical aspects of Lagrangean Relaxation.

Theoretical aspects: Let IP denote the integer linear programming problem

$$\begin{aligned} Z_{IP} \;=\; & \text{minimize } \sum_j c_j X_j \\ & \text{subject to: } \sum_j a_{ij} X_j = b_i \text{ (or } \geq b_i), \quad \forall\, i \\ & \qquad\qquad X_j \in \mathcal{X}, \text{ integer }, \qquad \forall\, j, \end{aligned}$$

where $\mathcal{X}$ is a feasible region defined by a set of linear constraints. Then, problem LP is defined as the linear relaxation of problem IP obtained by dropping the integrality conditions and Z_{LP} is its optimal value. Let also subproblem $L(\lambda)$ be the Lagrangean relaxation of IP relative to $\sum_j a_{ij} X_{ij} = b_i$ (or $\geq b_i$), $\forall\, i$, with a vector of Lagrangean multipliers $\lambda = (\lambda_i)$, $\forall\, i$, unrestricted in sign (or $\lambda_i \geq 0$, $\forall\, i$), i.e.:

$$\begin{aligned} Z_{L(\lambda)} \;=\; & \text{minimize } \sum_j c_j X_j + \sum_i \lambda_i \Big(b_i - \sum_j a_{ij} X_j \Big) \\ & \text{subject to: } X_j \in \mathcal{X}, \text{ integer}, \qquad \forall\, j. \end{aligned}$$

Given that the Lagrangean Dual Problem is $Z_L = \max_\lambda Z_{L(\lambda)}$, it is well known (Geoffrion 1974) that

$$Z_{LP} \leq Z_L \leq Z_{IP}.$$

Finally, if the problem IP is feasible and the subproblem $L(\lambda)$ possesses the Integrality Property, i.e., the optimal value of $L(\lambda)$ when solved as a linear program is not altered by dropping the integrality constraints, then $Z_L = Z_{LP}$. This is a fundamental result which states that the lower bound Z_L on Z_{IP} provided by a Lagrangean relaxation approach is no better than the linear relaxation bound Z_{LP}, if the subproblem $L(\lambda)$ possesses the Integrality Property. If the integrality gap $Z_{IP} - Z_{LP}$ is small, the Integrality Property is a *good* property in the sense that subproblem $L(\lambda)$ is a linear problem easy to solve, and Z_L

provides a good lower bound on Z_{IP}. On the other hand, if the integrality gap $Z_{IP} - Z_{LP}$ is large, the Integrality Property is a *bad* property. To obtain satisfactorily good lower bounds with Lagragean relaxations in this case, integer Lagrangean subproblems $L(\lambda)$ without the Integrality Property must be solved, so that part of the integrality gap is explored in solving $L(\lambda)$. This approach is effective if the Lagrangean problems can be easily solved by exploiting their special structure. Examples include knapsack problems, shortest path problems with time window and resource constrained shortest path problems.

There are several possible approaches for finding the value Z_L, including subgradient optimization, dual ascent methods, and Dantzig-Wolfe decomposition. In the latter case, if one defines the master problem by retaining the constraints $\sum_j a_{ij} X_j = b_i$, (or $\leq b_i$) $\forall\, i$ and the objective function, the resulting subproblem is identical to the subproblem $L(\lambda)$. Lagrangean relaxation and Dantzig-Wolfe decomposition yield the same lower bound on Z_{IP} as long as the subproblem is the same for both. These two approaches are respectively dual and primal methods to obtain the same bound (see Magnanti, Shapiro and Wagner 1976, Magnanti 1981, Desrosiers, Sauvé and Soumis 1988, and Nemhauser and Wolsey 1988).

Lagrangean Relaxation of the Time and Capacity Constraints: Since constraints (5.7) and (5.9) can be linearized (see (5.7a) and (5.9a)), they can be relaxed in the ojective function (5.1) with the properly defined multipliers. The resulting Lagrangean subproblem then becomes a pure network flow problem, which possesses the Integrality Property. The best Lagrangean bound is thus equal to the linear programming bound described in Section 5.2. This approach is analyzed in Desrosiers, Sauvé and Soumis (1988) for the m–TSPTW. It was never tested numerically since it is known that the integrality gap in this case is generally too large to be explored by branch-and-bound.

Lagrangean Relaxations Based on Shortest Path Problems with Resource Constraints: Relaxing the constraints (5.2) on visiting each customer once, and the constraints (5.3) on the number of available vehicles, results in a set of Lagrangean subproblems, one for each specific vehicle. Given the set of multipliers $\alpha = (\alpha_i,\ i \in N)$, which are unrestricted in sign, and the multiplier $\beta \geq 0$, subproblem $L^k(\alpha, \beta)$ is defined as follows:

$$\text{Minimize} \qquad \sum_{(i,j)\in A^k} c_{ij}^k X_{ij}^k + \sum_{i\in N} \alpha_i (1 - \sum_{j \in N \cup \{d(k)\}} X_{ij}^k) + \beta (v - \sum_{j \in N} X_{o(k),j}^k)$$

$$\text{subject to:} \qquad (5.3)\text{–}(5.13).$$

The above subproblem is a shortest path problem with time window and capacity constraints. This approach has been tested in Desrosiers, Sauvé and Soumis (1988) using classical and augmented Lagrangean relaxation methods for the m–TSPTW. It has also been used by Fisher, Jörnsten and Madsen (1992) for the VRPTW in conjunction with variable splitting (Guignard and Kim 1987). This involved a semi-assignment problem, defined using constraint set (5.2) and solved by inspection, and a set of shortest path problems with time and capacity constraints, one for each available vehicle. In this case, variable splitting does not allow any improvement of the Lagrangean lower bound since the semi-assignment problem possesses the Integrality Property.

Variable splitting allows for various relaxation schemes, each one exploiting different solvable structures. In addition to the above scheme, Fisher, Jörnsten and Madsen (1992) also used one based on a K–tree structure, where K is the set of available vehicles. An additional approach has been considered by Jörnsten, Madsen and Sørensen (1986) who used the generalized assignment problem (which accounts for the vehicle capacity constraints) and a shortest path with only time window constraints. The reader is referred to the original papers and to Halse (1992) for further details.

Computational Results: For the m–TSPTW, the various Lagrangean relaxation schemes have not been computationally competitive with the Dantzig-Wolfe decomposition/column generation approach described in Section 5.3. In the context of school bus fleet planning, Lagrangean relaxation was used by Desrosiers, Sauvé and Soumis (1988) only to estimate the optimal fleet size. The best approach found was the augmented Lagrangean relaxation method (which uses Frank-Wolfe decomposition and the SPPTW). This approach was far better than the classical Lagrangean relaxation which uses subgradient optimization to adjust the multipliers. Problems with up to 223 nodes, where each node represented a trip, were solved in the context of school bus fleet assignment.

For the VRPTW, Lagrangean relaxation schemes have been used to minimize the total distance traveled for a fixed fleet size. Using the K-tree structure, Fisher, Jörnsten and Madsen (1992) report having solved a number of 25– and 50–customer problems and two 100–customer problems from the benchmark problem sets developed by Solomon (1983). This method did not perform as well as the Dantzig-Wolfe decomposition/column generation approach. It also did not prove competitive with the other variable splitting method proposed by the same authors and further developed and tested by Halse (1992). The two approaches were better than the K-tree method even on problems involving geographically clustered customers, where the latter approach should perform at its best.

The Dantzig-Wolfe decomposition/column generation algorithm and the variable splitting approach have exhibited similar computational results. The former method proved slightly superior in solution quality to the latter as it solved several more problems. In addition, the former method is much more efficient in terms of computational time. For example, the largest problem solved to optimality, consisting of 105 customers and 11 identical vehicles, has been solved by Halse (1992) in 33295 seconds (more than 9 hours) on a HP 9000/835 computer system and by Desrosiers and Gélinas (1993) in 142 seconds on a HP 9000/755 (the HP 9000/755 is 8–10 faster than the HP 9000/835).

There are several reasons which explain the performance of the Dantzig-Wolfe decomposition/column generation approach relative to that of the various Lagrangean relaxations on vehicle routing problems with time and capacity constraints. First, the simplex algorithm used to solve the master problem in the Dantzig-Wolfe decomposition makes use of a lot of information (e.g., all the added columns) and rapidly adjusts the multipliers, while subgradient optimization is dependent on only the previous iteration. Second, this method is primal and allows for the use of many heuristics to rapidly converge. For example, a small portion of the network may be used to increase the speed of convergence while the whole network is used only for final optimality tests. If such a reduction of the network size is used in the dual approach, the intermediate lower bounds are not valid. Third, the primal method provides more information to better design a successful branch-and-bound to explore the integrality gap. For example, branching decisions taken on the flow variables, on the time variables, or on global constraints were designed and implemented in the Dantzig-Wolfe decomposition approach. Most of these can be transferred to Lagrangean approaches, thus improving their performance. This is indeed the direction taken by Kohl and Madsen (1993) who report very promising results using bundle methods (see Lemaréchal 1989). They were able to dramatically reduce the CPU time required to solve a number of problems.

6 Pick-up and Delivery Problems with Time Windows

The Pick-up and Delivery Problem with Time Windows (PDPTW) involves the satisfaction of a set of transportation requests by a heterogeneous vehicle fleet housed at several depots. A transportation request consists of picking-up a certain number of customers at a predetermined pick-up location during a departure time interval and transporting them to a predetermined delivery location to be reached during an arrival time interval. The departure and arrival time windows are based on desired pick-up or delivery time requests specified by the customers. Loading or unloading times are incurred at each vehicle stop.

Depending on the context, the problem consists of minimizing two or three objectives, in a hierarchical fashion, subject to a variety of constraints. The first objective involves the minimization of the number of vehicles or the total vehicle fixed costs required to satisfy the transportation requests. For this minimum value, the second objective addresses the minimization of the total distance or travel time. These two objectives are sufficient to model the problem of transportation of goods. However, for the transportation of persons, a third objective is necessary. This minimizes the inconvenience created by pick-ups or deliveries performed either sooner or later than desired by the customers. This latter PDPTW context is called dial-a-ride.

The PDPTW involves a multitude of constraints: *visiting* constraints which ensure that each pick-up and delivery stop is visited exactly once; *time window* constraints to be satisfied at each stop; *capacity* constraints on the vehicles (this can even be a multi-dimensional parameter, as in the case of transportation for the handicapped, where vehicles have two capacity limits - one on the number of wheelchairs and the other for the ambulatory passengers); *depot* constraints guaranteeing that a vehicle returns to the same depot from which it has started after each service run; *coupling* constraints stating that for a given request the same vehicle must visit the pick-up and the delivery stops; *precedence* constraints imposing that each customer must be picked-up before it is dropped-off; finally, *resource* constraints on the availability of drivers and vehicle types. The PDPTW is a generalization of the VRPTW. In the pick-up version, the VRPTW is the particular case of the PDPTW where the destinations are all a common depot. In the delivery version, the VRPTW is the particular case of the PDPTW where the origins are all a common depot.

Several cases of the pick-up and delivery problem will not be treated here. One case is that of full-loads which can be modeled as a m–TSPTW. A second case is that involving simple backhauling, i.e., where all the deliveries take place before all the pick-ups. This can be modeled as a VRPTW problem with one resource for the time and one for the vehicle capacity, and has been treated in Section 5. In the general backhauling problem, where all goods to be delivered (pick-up) are routed from (to) the depot, and where goods may be picked-up anytime there is sufficient space in the vehicle, the capacity constraints require two resources. The first is the maximum load on the vehicle up to the present, while the second is the actual load on the vehicle at the last customer visited. Note that the former resource is a non-decreasing function, hence it can be used to forbid subtours even when time windows are absent (Halse 1992).

We will examine separately the one-vehicle case in Section 6.1 and the multi-vehicle case in Section 6.2. In addition we will present the related problem of schedule optimization to minimize the customer inconvenience for a fixed route in Section 6.3.

6.1 The Single Vehicle Pick-up and Delivery Problem

The dial-a-ride context constitutes the core of research on pick-up and delivery problems. The early work is due to Wilson et al. (1971, 1976, 1977) who sought real-time solutions. The Single Vehicle Pick-up and Delivery Problem (1–PDPTW) is a constrained TSPTW, where additional restrictions consist of capacity and precedence constraints. Psaraftis (1980, 1983) has exploited this in the development of the first algorithms to minimize the total customer inconvenience in the 1–PDPTW where time window constraints are defined by maximum position shifts. His forward and respectively backward dynamic programming algorithms have a $O(n^2 3^n)$ time complexity, where n is the number of requests. Consequently they were limited at that time to solving very small size problems (at most 10 requests).

The same objective function is examined by Sexton and Bodin (1985a,b) in the presence of one-sided time windows (i.e., specified maximum delivery times or minimum pick-up times, respectively). The overall problem is decoupled into a coordinating routing master problem, formulated as an integer program, and a scheduling subproblem for a fixed route, which admits a linear programming formulation. A heuristic version of Benders' decomposition is then used for solution, where only the scheduling subproblem is solved optimally. The optimal solution to the subproblem can be obtained very efficiently by a network flow algorithm (see Section 6.3). The efficiency of the scheduling algorithm makes the overall procedure computationally tractable. This allows the solution of medium-sized real problems ranging from 7 to 20 customers in an average of 18 seconds of UNIVAC 1100/81A CPU

time. Sexton and Choi (1986) use a similar methodology to minimize a linear combination of total vehicle operating time and total customer penalty due to missing any of the time windows for the single vehicle pick-up and delivery problem with soft time windows.

Recently, Van Der Bruggen, Lenstra and Schuur (1993) have developped a heuristic algorithm based on arc-exchange procedures to minimize the route duration. Tested on real-life problems, their method is shown to produce near-optimal solutions in a reasonable amount of computation time. They have also developped an alternative algorithm based on simulated annealing which finds high quality solutions in a relatively large CPU time.

An optimal algorithm for minimizing the total distance traveled has been designed by Desrosiers, Dumas and Soumis (1986). The notation and the mathematical formulation used, as well as some algorithmic details are presented next.

Notation: Consider a set of n requests. Associate to the pick-up location of request i a node i, and to his delivery location a node $n+i$. Note that different nodes may correspond to the same physical location. Let $N^P = \{1, \ldots, n\}$ be the set of pick-up nodes and $N^D = \{n+1, \ldots, 2n\}$ be the set of delivery nodes, and define $N = N^P \cup N^D$. The set of nodes of the network is $V = N \cup \{o, d\}$, where o is associated with the origin-depot location while d is associated with the destination-depot location.

Request i demands that p_i units (goods or customers) be transported from node $i \in N^P$ to the corresponding node $n+i \in N^D$. Let $\ell_i = p_i$, for $i \in N^P$, and $\ell_{n+i} = -p_i$, for $n+i \in N^D$. The capacity of the single vehicle is given by Q. For each pair of distinct nodes $i, j \in V$, t_{ij} represents the travel time, which includes the service time at node i, and c_{ij} denotes the travel cost. Let $[a_i, b_i]$ denote the time window associated to node i, $i \in N$. Retain in the set A only the arcs $(i, j), i, j \in V$, which satisfy a priori capacity and time constraints, and other restrictions such as those based on precedence, and the same location. Note that a preliminary step in the determination of the admissible arcs is a time window reduction process as described in Section 3.

Three types of variables are used in the formulation: binary flow variables X_{ij}, $(i, j) \in A$, time variables T_i, $i \in V$, and load variables L_i, $i \in V$. The flow variable X_{ij}, $(i, j) \in A$ equals 1, if the vehicle travels from node i to node j, and 0, otherwise. The time variable T_i, $i \in V$, represents the time at which service begins at node i. The load variable L_i, $i \in V$, gives the total load on the vehicle just after the service is completed at node i. It is assumed that the vehicle departs empty from the depot at a given fixed time, i.e., $L_o = 0$ and $a_o = b_o = T_o$.

Formulation: A nonlinear mathematical formulation of the Single Vehicle Pick-up and Delivery Problem with Time Windows is as follows:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{6.1}$$

subject to:

$$\sum_{j\in N\cup\{d\}} X_{ij} = 1, \qquad \forall\, i \in N \tag{6.2}$$

$$\sum_{j\in N^P} X_{o,j} = 1, \tag{6.3}$$

$$\sum_{i\in N\cup\{o\}} X_{ij} - \sum_{i\in N\cup\{d\}} X_{ji} = 0, \qquad \forall\, j \in N \tag{6.4}$$

$$\sum_{i\in N^D} X_{i,d} = 1, \tag{6.5}$$

$$X_{ij}(T_i + t_{ij} - T_j) \le 0, \qquad \forall\, (i,j) \in A \tag{6.6}$$

$$a_i \le T_i \le b_i, \qquad \forall\, i \in V \tag{6.7}$$

$$X_{ij}(L_i + \ell_j - L_j) = 0, \qquad \forall\, (i,j) \in A \tag{6.8}$$

$$\ell_i \le L_i \le Q, \qquad \forall\, i \in N^P \tag{6.9}$$

$$0 \le L_{n+i} \le Q - \ell_i, \qquad \forall\, n+i \in N^D \tag{6.10}$$

$$L_o = 0, \tag{6.11}$$

$$T_i + t_{i,n+i} \le T_{n+i}, \qquad \forall\, i \in N^P \tag{6.12}$$

$$X_{ij} \text{ binary}, \qquad \forall\, (i,j) \in A. \tag{6.13}$$

This formulation includes a $(2n+1)\times(2n+1)$ assignment problem structure (6.1)–(6.5) where each node is visited only once, time window constraints (6.7), capacity intervals at pick-up nodes (6.9) and at delivery nodes (6.10), and constraints on the initial load of the vehicle at the origin-depot node (6.11). Constraints (6.6) describe the relation between the flow variables and the time variables. They involve an inequality sign since waiting time is permitted before the start of service at a node. These constraints prevent subtour formation. Constraints (6.8) describe the relation between the flow variables and the vehicle load at each node. These constraints do not prevent subtour formation since the vehicle load may increase or decrease. Finally, constraints (6.12) impose the precedence relations on the associate pick-up and delivery nodes.

The objective function is not as general as those addressing user inconvenience issues. Nevertheless, such issues can be incorporated by defining the time windows appropriately. Furthermore, since the solution strategy is based on a forward dynamic programming approach, the linear objective function can easily be replaced by a more general nonlinear

function. For example, let $c(L_i) > 0$, $i \in V$, denote a non-decreasing function of the total load transported on the vehicle, just after the service is completed at node i. This function acts as a penalty factor on the travel cost and constraint (6.1) can be replaced by:

$$\text{Minimize} \quad \sum_{(i,j)\in A} c(L_i) c_{ij} X_{ij}. \tag{6.1a}$$

A Dynamic Programming Solution Approach: Desrosiers, Dumas and Soumis (1986) have developed an optimal forward dynamic programming algorithm which also capitalizes on the fact that the single vehicle PDPTW has a TSPTW structure with additional capacity and precedence constraints. The method is based on reductions of the state space and of the number of state transitions which are performed both a priori and during the execution of the algorithm, similar to the ones developed for the TSPTW algorithm discussed in Section 3.5.

The states are defined as follows: the state (S, i) is defined if there exists a feasible path which starts at node o, visits all the nodes in $S \subseteq N$ and ends at node $i \in S$. A path is feasible if it satisfies vehicle capacity, precedence and time window constraints. State (S, i) is said to be post-feasible if there is a path starting at i, visiting all the nodes not belonging to S and satisfying the above constraints. Several criteria are proposed which eliminate all non-feasible states and a number of not post-feasible states. An additional criterion involves the case of several customers situated at the same location. The authors show that when several customers have the same physical location, some arcs can a priori be eliminated without restricting the optimality of the solution. This criterion allows the a priori imposition of a servicing order for the customers.

For each state (S, i), two-dimensional labels are defined for each path from node o to node i. A two-dimensional label represents the arrival time of the path at node i and the total distance traveled on the path, respectively. Similarly to the dynamic programming algorithm for the TSPTW, a label is non-dominated if it is a minimal element in the set of labels of state (S, i) under the natural partial order over both time and cost. Several of the above criteria for state elimination are implemented through the use of labels. All the criteria developed lead to the elimination of a large number of states and state transitions. This is the main reason why the algorithm proved successful. It obtained the optimal solution to problems containing up to 40 customers (80 nodes) in less than 6 seconds on a CYBER 173.

An Application of the 1–PDPTW: This type of problem has seen an interesting application to the discharge of larvicide in rivers to combat the growth of the black fly larvae. The larvae are the root cause of millions of cases of onchocerciasis (river blindness) that occur annually in Africa and hence is the object of large-scale larvicide control programs. This multi-period 1–PDPTW involves the discharge by helicopters of up to 5 types of larvicide subject to many types of constraints. The actual pick-up points as well as refueling locations are chosen from a larger set during the solution process. Chalifour, Solomon, Boisvert and Desrosiers (1992) cluster sets of points where the same larvicide is to be used into small groups (mini-clusters) and use an insertion heuristic to solve the problem.

6.2 The Multiple Vehicle Pick-up and Delivery Problem

The general Multiple Vehicle Pick-up and Delivery Problem with Time Windows (m–PDPTW) has only recently received attention. The only optimal algorithm has been proposed in the context of goods transportation by Dumas, Desrosiers and Soumis (1991). This algorithm is based on a Dantzig-Wolfe decomposition approach embedded into a branch-and-bound search tree. The master problem results in the linear relaxation of a set partitioning type model, while feasible routes or columns are generated by a subproblem modeled as a shortest path problem with coupling, precedence, time window and capacity constraints.

Notation: The notation used is a direct extension of the one presented in the previous section. Let K, indexed by k, be the set of vehicles to be routed and scheduled. Then (V^k, A^k) is the network associated with a specific vehicle k. The set $V^k = N \cup \{o(k), d(k)\}$ is the set of nodes with $o(k)$ and $d(k)$ denoting the origin-depot and the destination-depot of vehicle k, respectively; the set A^k contains all feasible arcs, a subset of $V^k \times V^k$. The cost and time elements, the vehicle capacities and the three types of variables are defined adequately. Note that the index k serves to define a specific vehicle and a depot location, and more generally, the initial vehicle conditions of a heterogeneous fleet of vehicles. It is assumed that each vehicle k departs empty from its origin-depot, at a specified time value given by $T^k_{o(k)} = a_{o(k)} = b_{o(k)}$.

Formulation: We are now in a position to present a mathematical formulation of the multiple depot multiple vehicle types pick-up and delivery problem with time windows:

$$\text{Minimize} \sum_{k \in K} \sum_{(i,j) \in A^k} c^k(L^k_i) c^k_{ij} X^k_{ij} \tag{6.14}$$

subject to:

$$\sum_{k\in K}\sum_{j\in N\cup\{d(k)\}} X_{ij}^k = 1, \qquad \forall\, i \in N^P \tag{6.15}$$

$$\sum_{j\in N^P\cup\{d(k)\}} X_{o(k),j}^k = 1, \qquad \forall\, k \in K \tag{6.16}$$

$$\sum_{i\in N\cup\{o(k)\}} X_{ij}^k - \sum_{i\in N\cup\{d(k)\}} X_{ji}^k = 0, \qquad \forall\, k \in K,\ \forall\, j \in N \tag{6.17}$$

$$\sum_{i\in N^D\cup\{o(k)\}} X_{i,d(k)}^k = 1, \qquad \forall\, k \in K \tag{6.18}$$

$$X_{ij}^k(T_i^k + t_{ij}^k - T_j^k) \le 0, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{6.19}$$

$$a_i \le T_i^k \le b_i, \qquad \forall\, k \in K,\ \forall\, i \in V^k \tag{6.20}$$

$$X_{ij}^k(L_i^k + \ell_j - L_j^k) = 0, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{6.21}$$

$$\ell_i \le L_i^k \le Q^k, \qquad \forall\, k \in K,\ \forall\, i \in N^P \tag{6.22}$$

$$0 \le L_{n+i}^k \le Q^k - \ell_i, \qquad \forall\, k \in K,\ \forall\, n+i \in N^D \tag{6.23}$$

$$L_{o(k)}^k = 0, \qquad \forall\, k \in K \tag{6.24}$$

$$T_i^k + t_{i,n+i}^k \le T_{n+i}^k, \qquad \forall\, k \in K,\ \forall\, i \in N^P \tag{6.25}$$

$$\sum_{j\in N} X_{ij}^k - \sum_{j\in N} X_{j,n+i}^k = 0, \qquad \forall\, k \in K,\ \forall\, i \in N^P \tag{6.26}$$

$$X_{ij}^k \ge 0, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{6.27}$$

$$X_{ij}^k \text{ binary}, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k. \tag{6.28}$$

The nonlinear objective function minimizes the total travel cost. Note also that the fixed cost of a vehicle k is usually incorporated into the values $c_{o(k),j}^k$, for all $j \in N^P$. Constraints (6.15)–(6.18) and (6.28) form a multi-commodity flow problem. Next, constraints (6.19) describe the compatibility requirements between routes and schedules, while (6.20) are the time window constraints. Constraints (6.21) express the compatibility requirements between routes and vehicle loads, while (6.22) and (6.23) are the capacity intervals at pick-up and delivery nodes, respectively. These intervals are dependant of the vehicle used to satisfy a given request. Next, constraints (6.24) impose the initial load condition on each vehicle. Constraints (6.25), which are also vehicle dependent, are the precedence constraints forcing node i to be visited before node $n+i$, $i \in N^P$. Along with the coupling constraints (6.26) they ensure that the same vehicle k visits both nodes i and $n+i$, $i \in N^P$. Finally, the formulation involves non-negativity constraints on flow variables (6.27) and binary requirements (6.28).

A Dantzig-Wolfe Decomposition: The master problem, a linear program, retains the constraints (6.14)–(6.15), while the subproblem consists of a modified cost function and constraints (6.16)–(6.28). Note that the subproblem is separable by vehicle, i.e., one subproblem for each vehicle $k \in K$. As in the VRPTW case, the master problem will become the linear relaxation of a set partitioning type problem, while the subproblems are constrained shortest path problems.

Given the dual variables provided by the master problem, the arc costs are modified accordingly for a given vehicle k, $k \in K$ (see Section 5.3). An integer solution to the constrained shortest path problem is obtained by a forward dynamic programming in a manner similar to the algorithm presented for the single-vehicle PDPTW. The efficiency of the algorithm is derived from powerful label elimination criteria which greatly reduce the state space and the number of state transitions. These criteria generalize some of the corresponding criteria introduced for the single-vehicle version. For each path p starting at an origin-depot and ending at node i, denoted $\mathcal{P}_i^p$, labels of the form (set of visited nodes, time, cost) are defined. The three-dimensional label (S_i^p, T_i^p, Z_i^p) represents the set of nodes visited on the path p, the time the service starts at node i on path p and the total cost of path p, respectively. The information contained in the first component serves to determine the vehicle load at node i, i.e., $\sum_{j \in S_i^p} \ell_j$. A path is admissible, if it respects at each node the time window, priority, and capacity constraints. The coupling constraints must also be respected when the last visited node i is a destination-depot. Enumeration of all the feasible paths is impossible in practice. To reduce the number of labels, we associate with each admissible path $\mathcal{P}_i^p$ a label $(R(S_i^p), T_i^p, Z_i^p)$, where $R(S_i^p) \subseteq N^P$ is the subset of S_i^p containing only the pick-up nodes which have been visited, but whose corresponding delivery nodes have not yet been visited. The information provided by the first component is still sufficient to calculate the vehicle load at node i. However, the resulting optimal shortest path may contain a cycle, i.e., a path which satisfies the same *request* twice while respecting all the constraints. Since a request cycle must satisfy a sequence of at least 5 nodes (e.g., $i \rightarrow n+i \rightarrow j(\neq i) \rightarrow i \rightarrow n+i$, as arc $(n+i, i)$ never exists), such a cycle is not common in practice when time windows are small in comparison with the travel times. An admissible label associated with path $\mathcal{P}_i^p$, which cannot be extended from node i to a destination-depot due to time window or coupling constraints violations, is called a non post-feasible label. Many of these non post-feasible labels are eliminated during the solution process. In addition, labels are eliminated based on a dominance criterion, which extends the similar criterion for the single-vehicle case. A stronger dominance criterion is possible if the costs satisfy the triangle inequality.

Each time a subproblem is solved, if there exits a negative marginal cost path, it is added to the restricted set partitioning type formulation which is in turn solved by the simplex method. When no more negative marginal cost columns can be found from any subproblem, the solution to the restricted master problem constitutes the optimal solution to the master problem.

An integer solution to the original problem (6.14)–(6.28) is then obtained by exploring a branch-and-bound tree. The branching strategy directly exploits the pick-up sequence of the requests. This limits the size of the branch-and-bound tree when compared to other potential strategies. Order variables O_{ij}, $i, j \in N^P$ are set at 1 if pick-up j is the next pick-up after pick-up i, and set at 0 otherwise. To transfer the information provided by the order variables to the subproblem, a fourth component is added to each label in the shortest path algorithm. This component represents the last pick-up node visited. This additional component does not make the subproblem much harder to solve since the information it provides is closely related to the information provided by the first component.

The algorithm has been successful in solving two real life problems of sizes 19 and 30 requests, respectively, taken from Guinet (1984), as well as for randomly generated problems involving up to 55 requests with tight vehicle capacity constraints (Dumas, Desrosiers and Soumis 1991). Both real life problems were solved to optimality, without any branching. For all the other problems, the number of vehicles was always minimal after the addition of a cut at the first node, while the gap on travel cost ranged from 0.6% to 3.2%. Very good solutions were obtained within the exploration of a few nodes in the branch-and-bound tree, typically less than 10. The algorithm is appropriate for the PDPTW when the problem solutions require an average of up to 5 requests per vehicle, i.e., 10 pick-up and delivery stops. However it becomes ineffective in the dial-a-ride context due to the actual size of practical problems (over 3,000 requests) and the large number of requests per vehicle. Such problems have been solved by heuristic methods.

Heuristics for the Dial-a-Ride Problem with Time Windows: As in the single vehicle version, the research on the multi-vehicle PDPTW has been concentrated on the dial-a-ride context. A parallel insertion procedure was developed by Roy, Rousseau, Lapalme and Ferland (1984a, b). This algorithm simultaneously constructs routes for all vehicles starting at the beginning of the day by using proximity criteria. It can also insert new requests into a set of existing routes with the possibility of initializing new routes as needed.

A similar approach for minimizing a weighted combination of customer disutility and system costs was employed by Jaw, Odoni, Psaraftis and Wilson (1986). Simulated problems involving 250 customers and 10-14 vehicles took about 20 seconds each. A real problem involving about 2,617 customers and 20 vehicles (a much larger size than ever attempted before) was successfully solved in 12 minutes of VAX 11/750 CPU time. A comparison with a previous approach developed by the authors is provided in Psaraftis (1986). An improved version of the algorithm was implemented by Madsen, Ravn and Rygaard (1993) for a dynamic dial-a-ride problem with time windows. This real world problem with 300 requests and 24 vehicles was solved in less than 10 seconds on a HP 9000/720.

A traditional *cluster first, route second* approach was proposed by Bodin and Sexton (1986). The set of requests is partitioned into vehicle clusters and a single vehicle dial-a-ride problem solved for each cluster. The single-vehicle problems are solved by using the methodology described in Sexton and Bodin (1985a, 1985b). To further reduce total user inconvenience, requests are then relocated one at a time in vehicles different than that where they were originally assigned.

Dumas, Desrosiers and Soumis (1989) improve upon the above methodology by moving a part of the clustering problem into the routing problem. In the classical *cluster first, route second* philosophy, clusters consisting of the set of customers assigned to each vehicle are formed, and routing is performed separately in each cluster. However, it is very difficult to construct a good set of clusters, especially without relying on routing information. Hence, the authors have devised heuristic algorithms which group together customers who efficiently can be served by the same vehicle route segment to form mini-clusters. A 2,000-request problem is typically reduced to an approximately 800–task problem, where each mini-cluster defines a task to perform. This strategy moves part of the clustering process into the routing phase. The new routing problem, which constructs routes corresponding to drivers' workdays by stringing together the mini-clusters, is less difficult to solve than the original, since it is of smaller size and certain constraints have partly (in the case of time window and capacity restrictions) or totally (in the case of coupling and precedence constraints) been satisfied. In fact, the mini-clustering problem corresponds to a m–PDPTW while the routing problem is a m–TSPTW with multiple depots and a heterogeneous vehicle fleet for which a solution strategy has been described in Section 5.3. Figure 6.1 taken from Ioachim, Desrosiers, Dumas, and Solomon (1991) illustrates this two-phase solution approach.

The global routing solution, i.e., the m–TSPTW solution, can be approximated to permit the insertion of additional requests during the operation day. Such insertions are facilitated when periods of free time can be joined into longer non-interrupted periods. A nonlinear objective function which accomplishes this while also minimizing the operational costs is given by:

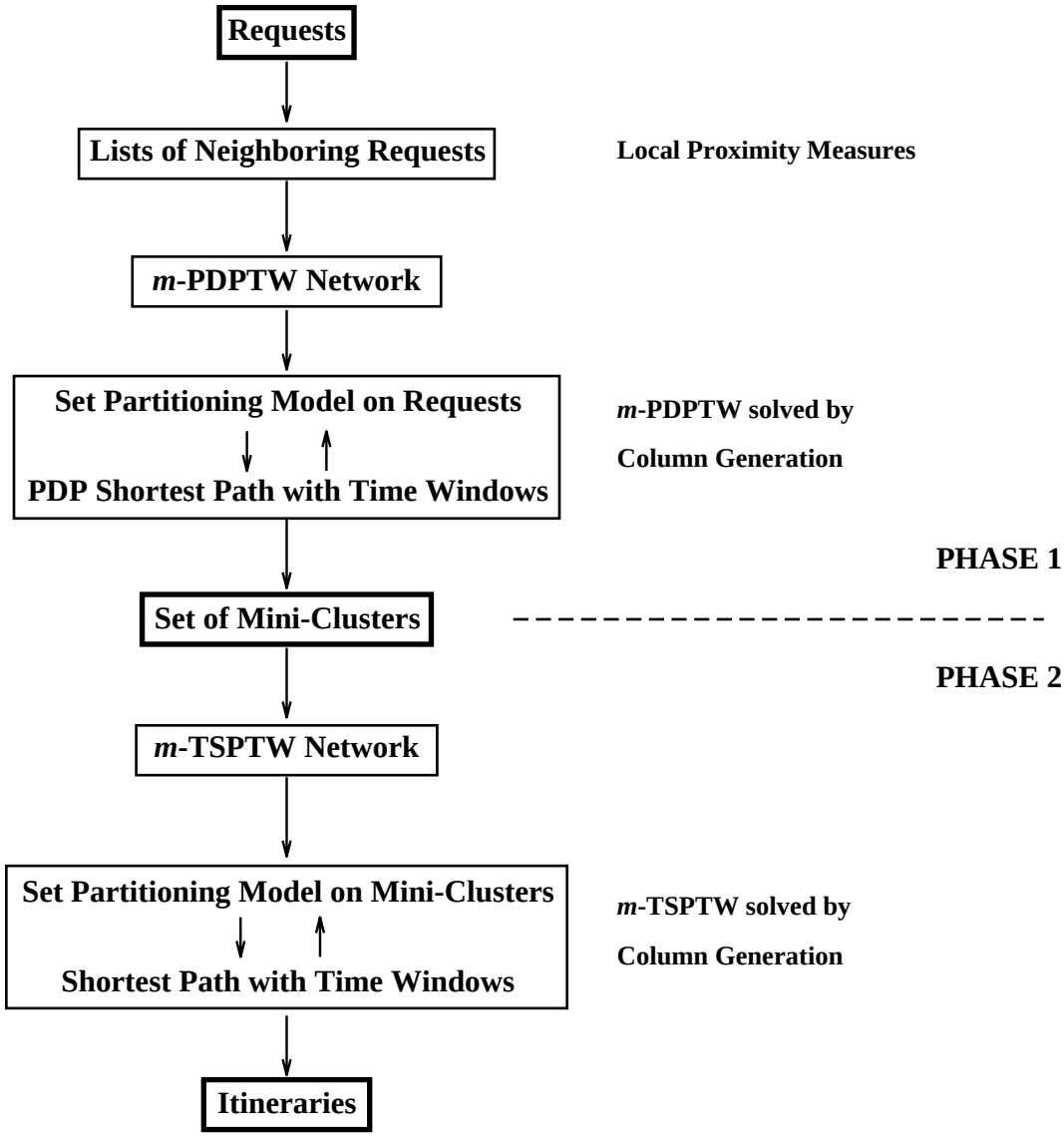


Figure 6.1: Mini-clustering / Route construction process for Dial-A-Ride Problems

$$\text{Maximize} \quad \sum_{k \in K} \sum_{(i,j) \in A^k} (T_j^k - T_i^k - t_{ij}^k)^\alpha X_{ij}^k \tag{6.29}$$

where $\alpha > 1$ is a constant, $T_j^k - T_i^k - t_{ij}^k$ is the free time between nodes i and j, and the nodes are either mini-clusters or depot pull-outs or pull-ins. Since the total schedule time is the sum of the total travel time and total free time, minimizing the first component of the sum is equivalent to maximizing the second when $\alpha = 1$. This approach can easily solve problems with up to 200 customers and 85 mini-clusters in less than 5 minutes on a CYBER 173. To solve larger problems, the day is partitioned into time slices and the algorithm is applied several times. A real-world application involving 880 customers and 282 mini-clusters has been solved in 22 minutes on the same computer. Recently, the algorithm has been utilized to solve problems involving up to 3,586 requests. It takes only one hour of CPU time on a HP–730 workstation to obtain a solution.

While Dumas, Desrosiers and Soumis (1989) use a sequential insertion heuristic to create the mini-clusters, Desrosiers, Dumas, Soumis, Taillefer and Villeneuve (1991) perform the insertions in parallel. Finally, Ioachim, Desrosiers, Dumas and Solomon (1991) utilize mathematical optimization techniques to globally define a set of mini-clusters. These are generated by solving a m–PDPTW by column generation using an enhanced version of the approach of Dumas, Desrosiers and Soumis (1991). The algorithm utilizes only the best feasible arcs between the pick-up and the drop-off nodes which satisfy several customer-suggested properties. A new initialization strategy is proposed for the constrained shortest path algorithm. This reduces by 40% the running time for the overall approach. The authors have also developed a heuristic to reduce the size of the network, while incurring only small losses in solution quality. Tests of the algorithm performed on problems with up to 250 requests (500 nodes) have shown a reduction of 10% in the travel time within the mini-clusters as compared to the above parallel insertion heuristic. For an actual problem involving 2,545 requests, the two-phase method, i.e., mini-clustering followed by global routing, produced a substantial 5.9% improvement in total traveling time over the same heuristic.

Additional m–PDPTW Applications: Heuristics for the general PDPTW have been proposed in the context of military air- or sealift. These problems involve the movement of cargo and/or passengers by plane (ship) between a number of bases (ports) which optimize a prescribed measure of system performance such that the time window constraints on the pick-up and delivery times and the other problem constraints are satisfied. Peacetime or wartime air- or sealift involve planning horizons of up to several months. Solanki and Southworth (1991) present an application of Solomon's (1987) insertion heuristic which

adjusts an existing schedule execution planning for military airlift. This is a dynamic environment where the existing airlift operations plan may need to be modified daily. The authors report having solved a problem involving 42 aircrafts and 10 airfields for seven days in less than two minutes on a SUN 4 workstation.

Another heuristic, the Airlift Planning Algorithm (APA) was devised by Rappoport, Levi, Golden and Toussaint (1992) for a much longer planning horizon of up to 90 days. To simplify the problem, the assignment of payload to aircraft is treated before the routes and schedules are tested for feasibility. The concept of *best fit* drives the former problem. The latter problem is solved using shortest paths. To obtain feasibility, the APA may necessitate several iterations through the two phases. Even then however, due to the complexity of the many types of constraints, feasibility is not guaranteed. For a 90-day scheduling horizon problem, involving 13 bases and 285 planes, APA was able to move 95.9% of cargo (61.2% on time) and 99.9% of passengers (92.3% on time) in 696 seconds on a SUN SPARC 1 workstation. The algorithm can be used to initialize the Solanki and Southworth (1991) method. It has also been enhanced by a preprocessor that sequences the requirements to be moved more intelligently. This involves the solution of a transportation problem and is described in Rappoport, Levy, Toussaint and Golden (1992).

The sealift problem has been examined by Psaraftis, Orlin, Bienstock and Thompson (1985). The authors propose a heuristic approach to be used in military emergencies. The heuristic is initialized with a one-to-one assignment of some cargoes to some ships. The scheduling horizon is then partitioned into time slices, numbered from the earliest to the latest. At iteration k, the heuristic temporarily assigns the cargoes whose earliest pick-up times fall within the k^{th} time slice. A transportation problem where each arc cost is a measure of the utility of the corresponding cargo/ship assignment is used for this purpose. Certain cargoes are then permanently assigned, allowing for cargo splitting if necessary. The horizon is then rolled to the next time slice. The authors have examined scenarios involving 7 ships which must pick-up and deliver 20 cargoes over a scheduling horizon of 98 days divided into bi-weekly intervals. This rolling horizon heuristic has been used by Thompson and Psaraftis (1983) to provide good initial solutions to be improved by cyclic transfer algorithms. These have proved very successful in decreasing total tardiness.

While the above approaches permit each ship to carry many cargoes at the same time, Fisher and Rosenwein (1989) consider the pick-up and delivery of bulk cargoes. Their optimization algorithm is based on research conducted earlier for a similar truck scheduling problem, which is presented in Fisher, Greenfield, Jaikumar and Kedia (1982), and Bell, Dalberto, Fisher, Greenfield, Jaikumar, Kedia, Mack and Prutzman (1983). The first phase of the approach is the generation of a list of candidate schedules for each ship. All, or a limited number of schedules, can be generated based on an input parameter that controls

the amount of unloaded travel permitted. The schedule selection phase is a Set Packing Problem that is solved with the branch-and-bound procedure described in Fisher and Kedia (1990), where bounds are obtained through Lagrangean relaxation. Fisher and Rosenwein (1989) report having optimally solved a realistic problem involving 28 cargoes to be lifted by a fleet of 17 ships in 50 seconds of VAX 8600 CPU time. Finally, the transport of a single bulk product between production and consumption ports is being addressed by Christiansen (1993). The load to be picked-up or discharged is either fixed or variable leading to an inventory–PDPTW model.

6.3 Schedule Optimization for a Fixed Route

Any solution approach to the VRPTW or the PDPTW must examine three aspects of the problem: the clustering of the nodes (or the assignment of customers to vehicles), the routing of the vehicles and the development of each vehicle's schedule. Approaches differ in terms of the simultaneous or hierarchical treatment of these issues. In all cases, however, the schedule associated with the route determined for each vehicle must be optimized. As time windows generally are fairly narrow in these problems, the service times obtained in the travel cost minimization part of the problem will provide a certain quality of service. To maximize the service quality, a final explicit optimization can be performed with respect to the set of desired service times at nodes. Hence, beginning service at a node at a less desirable time incurs a penalty or inconvenience cost.

Notation and Formulation: Given a fixed route which visits, in sequence, nodes $1, 2, \ldots, n$, let N be the set of these nodes. Let also $t_{i,i+1}$, $i \in N \setminus \{n\}$ be the travel time from node i to node $i+1$. For each node $i \in N$, define the variable T_i as the time service begins at node i. Define also the convex function $f_i(T_i)$ as the inconvenience function. Then, the schedule optimization problem for a fixed route is:

$$\text{Minimize} \sum_{i \in N} f_i(T_i) \tag{6.30}$$

subject to:

$$T_i + t_{i,i+1} \leq T_{i+1}, \qquad \forall\, i \in N \setminus \{n\} \tag{6.31}$$

$$a_i \leq T_i \leq b_i, \qquad \forall\, i \in N. \tag{6.32}$$

We seek to minimize the total inconvenience such that the schedule is compatible with the fixed route (6.31) and the service times fall within the time windows (6.32).

Literature: Three approaches to this problem have been presented to date (see Figure 6.2). In the first approach, Sexton and Bodin (1985a) consider time windows of the form $(-\infty, b]$ and a linear decreasing inconvenience function $f_i(T_i)$ such that $f_i(b_i) = 0$. In a related study, Sexton and Choi (1986) consider a piecewise linear function with $f_i(T_i) = 0$, if $T_i \in [a_i, b_i]$, $f_i(T_i)$ decreasing for $T_i \in (-\infty, a_i)$ and increasing for $T_i \in (b_i, +\infty)$. The two proposed algorithms have $O(n)$ complexity.

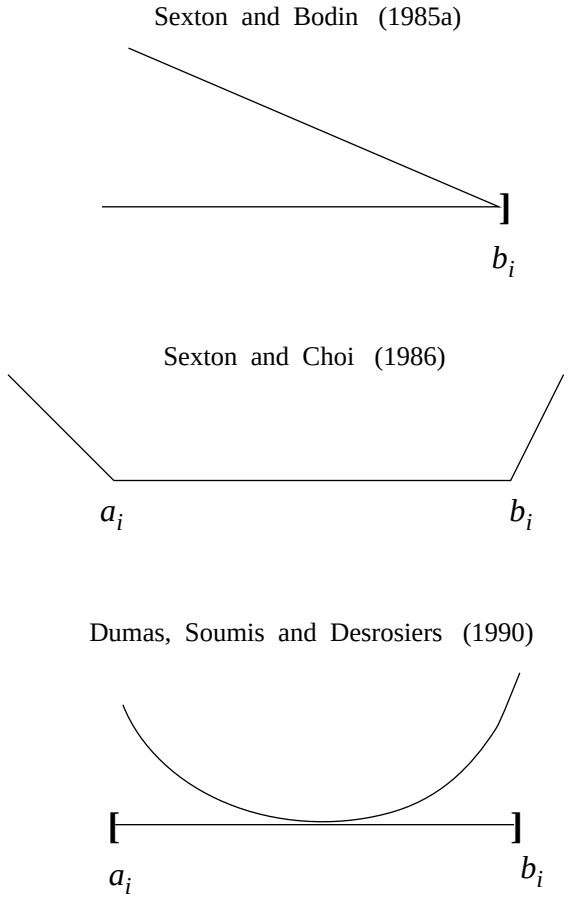


Figure 6.2: Examples of inconvenience functions

The third and most general approach to date is that of Dumas, Soumis and Desrosiers (1990) who consider a general convex function on the time window $[a_i, b_i]$. The authors propose a dual approach based on relaxing constraints (6.31). The violated constraints are then reintroduced one by one until they are all satisfied. The complexity of the algorithm is of the order of n unidimensional minimizations. For time windows, the use of convex inconvenience functions is more interesting than the use of linear functions since it allows

solutions at the interior of the feasible region. In addition, the model admits two simple enhancements. The first consists of adding a linear cost to waiting time. This is of particular interest when the waiting time causes an inconvenience. A second generalization entails discretizing the service time variables. These extensions do not imply any modifications to the algorithm. An adaptation of the algorithm has also found an application in statistics for the estimation of a step function in a nonlinear regression model (Dumas, Desrosiers and Soumis 1990). This occurs when the dependent variable of a regression model is discrete while the optimization determines where the steps take place. In the above mentioned application, the algorithm is used to estimate a travel time function in an urban context where the function values are only multiples of 5 minutes. In that case, decision variables are linked by ratio constraints, i.e., $X_i \leq R_{i,i+1} X_{i+1}$, where $R_{i,i+1} > 0$ for all $i \in N \setminus \{n\}$.

Two special cases of the schedule optimization for a fixed route are worth noting. The first involves a quadratic inconvenience function, for which the authors proved that the complexity is linear in the number of elementary operations. The second deals with a linear inconvenience function. In this case, the optimization is even easier as the dual can be modeled as a minimum cost flow problem, a result previously obtained by Sexton and Bodin (1985a). The linear case is presented next.

The Linear Case: Let us rewrite problem (6.30)–(6.32) with linear inconvenience functions:

$$\text{Minimize} \sum_{i \in N} \alpha_i T_i + \beta_i \tag{6.33}$$

subject to:

$$T_i - T_{i+1} \leq -t_{i,i+1}, \qquad \forall\, i \in N \setminus \{n\} \quad [-X_{i,i+1}] \tag{6.34}$$

$$-T_i \leq -a_i, \qquad \forall\, i \in N \quad [-X_{n+1,i}] \tag{6.35}$$

$$T_i \leq b_i, \qquad \forall\, i \in N \quad [-X_{i,n+1}] \tag{6.36}$$

The variables between brackets indicate the corresponding dual variables. Ignoring the constants β_i, $i \in N$, the dual problem can be formulated as follows:

$$\text{Minimize} - \sum_{i \in N \setminus \{n\}} t_{i,i+1} X_{i,i+1} - \sum_{i \in N} a_i X_{n+1,i} + \sum_{i \in N} b_i X_{i,n+1} \tag{6.37}$$

subject to:

$$\begin{array}{cccccll} X_{1,2} & +X_{1,n+1} & & -X_{n+1,1} & = -\alpha_1, & & \\ X_{i,i+1} & +X_{i,n+1} & -X_{i-1,i} & -X_{n+1,i} & = -\alpha_i, & \forall\, i \in N \setminus \{1,n\} & (6.38) \\ & +X_{n,n+1} & -X_{n-1,n} & -X_{n+1,n} & = -\alpha_n, & & \\ & & & X_{i,i+1} \geq 0, & & \forall\, i \in N \setminus \{n\} & (6.39) \\ & & & X_{n+1,i} \geq 0, & X_{i,n+1} \geq 0, & \forall\, i \in N. & (6.40) \end{array}$$

The dual is a transportation problem with $n+1$ nodes and $3n-1$ arcs. One node is associated with each variable T_i, $i \in N$, with a demand of α_i units, and an artificial node $n+1$, with a supply $\alpha_{n+1} = \sum_{i \in N} \alpha_i$. This formulation has also been efficiently treated by Sexton and Bodin (1985a) using an $O(n)$ algorithm. However, the number of arcs can further be reduced a priori to only $2n-1$ arcs.

Note first that the linear function $f_i(T_i) = \alpha_i T_i + \beta_i$, $a_i \leq T_i \leq b_i$, reaches its minimum at a_i, if $\alpha_i > 0$, and at b_i, otherwise. Assume next that the time windows have been reduced to eliminate the portions which are never used. This can be done recursively by means of the following simple relations:

$$a_{i+1} := \max\{a_{i+1}, a_i + t_{i,i+1}\}, \qquad i = 1, \ldots, n-1 \tag{6.41}$$

$$b_i := \min\{b_i, b_{i+1} - t_{i,i+1}\}, \qquad i = n-1, \ldots, 1. \tag{6.42}$$

The complementary slackness conditions help to show that $X_{n+1,i} X_{i,n+1} = 0$, $\forall\, i \in N$, i.e., that at least one of the two variables is zero. Dumas, Soumis and Desrosiers (1990) then prove that the variables set at zero can in fact be specified:

$$\begin{aligned} X_{n+1,i} &= 0, \qquad \forall\, i \in \{i | \alpha_i < 0\} \\ X_{i,n+1} &= 0, \qquad \forall\, i \in \{i | \alpha_i \geq 0\}. \end{aligned} \tag{6.43}$$

The dual variables associated with the network nodes are the variables T_i of the original problem (the additional variable T_{n+1} is set at zero). The solution process starts with a basis consisting of variables $X_{i,n+1}$, if $\alpha_i < 0$, and $X_{n+1,i}$, if $\alpha_i \geq 0$, $i \in N$, and uses as entry criterion, the variable $X_{i,i+1}$ of smallest index i with negative marginal cost, i.e., $-t_{i,i+1} - T_i + T_{i+1} < 0$. The entry of one of these arcs into the basis corresponds to setting at equality the ith constraint in (6.34). The solution approach decides sequentially which constraint is set to equality, and previous decisions are not reconsidered. The algorithm takes at most $n-1$ iterations. Hence it is of linear complexity.

7 A Unified Framework for Fleet and Crew Scheduling Problems

Network models and algorithms have been used for many years in the field of vehicle fleet planning and crew scheduling problems. For example, the survey paper by Carraresi and Gallo (1984) reviews the network-based optimization algorithms and their use in the context of Urban Mass Transit vehicle routing and crew scheduling applications. In the same context, many other papers also appear in the very recent proceedings of the workshop on computer-aided transit scheduling (Desrochers and Rousseau 1992).

In Section 2, we have considered a number of fixed schedule problems. The simplest ones were formulated and solved using classical network models and algorithms. On the other hand, the multi-depot vehicle scheduling problem was formulated as a multi-commodity network flow problem. The best known optimal solution approach uses a Dantzig-Wolfe decomposition/column generation scheme embedded in a branch-and-bound enumeration tree. Its subproblem/column generator is as a classical shortest path problem on an acyclic network.

Section 4 has introduced the shortest path problem with time window constraints (SPPTW), its generalized version which involves multi-resource constraints (SPPRC) and their corresponding solution approaches based on dynamic programming algorithms. In the vehicle routing applications of Section 5, one and two resource constrained shortest path algorithms were again used as subproblems or feasible route generators in a Dantzig-Wolfe decomposition/column generation optimal solution approach to the m-TSPTW and to the VRPTW. This approach has been presented for multiple depots and heterogeneous vehicle fleet cases. Yet it can also be extended to the simple backhauling problem, to multiple time window intervals at each customer, to models which incorporate rest and meal periods, and most other realistic situations. Even more sophisticated shortest path algorithms, with additional precedence and coupling constraints, have been introduced in Section 6 to design an optimal approach for pick-up and delivery routing problems. At that time, we have also introduced nonlinear non-decreasing objective functions, (6.14) for example, which have been optimized using dynamic programming algorithms in the subproblems. All the previous problem definitions have involved at most two resources: elapsed time and cumulative load; these resources came from time window and capacity restrictions.

In this section, we consider vehicle fleet planning and crew scheduling problems which involve many resource constraints. These constraints relate mostly to maintenance schedules in vehicle fleet planning, and worker collective agreement regulations in crew scheduling. There are many important application domains for fixed or variable schedule problems with resource constraints: vehicle fleet assignment for buses, aircraft, ships and locomotives; crew pairing construction and monthly crew workload assignment for urban mass transit, airline, naval and rail companies. They all share the fact that feasible solutions correspond to paths on properly designed multi-commodity networks.

In the airline context, the crew costs represent a large share of these companies' total costs and any improvement translates into sizable savings. For example, at Northwest Airlines, the total crew costs represented $1050 millions in 1989. Thus an improvement of 5% in the solution cost would amount to annual savings of more than $50 millions (Barutt and Hull 1990). This explains the existence of several specialized software packages for solving these problems.

We will focus this section on an optimal solution approach for two fixed schedule pairing construction problems: the urban bus driver scheduling problem, and the classical airline crew scheduling problem. Nevertheless, the multi-commodity network flow model with resource constraints, as well as the Dantzig-Wolfe decomposition/column generation solution approach described below, have also been applied to aircraft fleet planning (from daily to multiple month schedules), to train driver pairing construction, and also to monthly crew workload assignment problems, again in the airline context. Furthermore, the model and methodology is by no means restricted to fixed schedule problems as time window restrictions involve only a supplementary resource. This has been recently applied to aircraft fleet planning in conjunction with a passenger demand model on a set of Air France test problems where benefits obtained went up to 20% on some scenarios.

7.1 A Multi-Commodity Network Flow Model with Resource Constraints

Notation: Let F, indexed by f, represent the set of operational tasks to be performed exactly once. For example, F may represent the set of operational flight legs in an airline context. Define K as the set of commodities. With each commodity $k \in K$, associate the graph $G^k = (V^k, A^k)$, where V^k is the set of nodes and $A^k \subseteq V^k \times V^k$ is the set of arcs. Let $o(k)$ and $d(k)$, $k \in K$, denote the origin-depot and the destination-depot nodes, respectively. Let X_{ij}^k be the flow of type k through arc $(i,j) \in A^k$.

A given commodity $k, k \in K$, corresponds to a single vehicle in vehicle routing problems, to a specific crew in crew pairing problems, and to individual workers in monthly workload assignment problems. Graphs (V^k, A^k), for all $k \in K$, are oriented time-space networks, where nodes correspond to real locations at different times, while arcs represent for example a vehicle, a crew or a worker activity such as briefing, debriefing, an operational task, or a deadhead task, a rest period, a meal, a connection, or a partial or a full workday, or, finally, a partial or even a full schedule. Arcs may represent single or multiple activities. Define $A_f^k \subseteq A^k, k \in K, f \in F$, the subset of arcs on graph G^k such that each of these arcs contains an operational task. Description of some specialized time-space networks can be found in Section 2.2, and in Barnhart, Johnson, Anbil and Hatay (1991), Desrochers, Gilbert, Sauvé and Soumis (1992), Desrosiers, Dumas, Desrochers, Soumis, Sansó and Trudeau (1991), Lamatsch (1992), and Lavoie, Minoux and Odier (1988), to name a few.

Define R as the set of resources. The intervals $[a_i^{kr}, b_i^{kr}]$, $k \in K$, $i \in V^k$, $r \in R$, restrict the amounts of resources used to reach node $i \in V^k$, and t_{ij}^{kr} is the number of units of resource $r \in R$ used on arc $(i,j) \in A^k$. The resource variables are given by T_i^{kr}, $k \in K$, $i \in V^k$, $r \in R$. Assume that initial conditions are given by $T_{o(k)}^{kr} = a_{o(k)}^{kr} = b_{o(k)}^{kr}$, for all $r \in R$, at the origin-node $o(k)$, for all $k \in K$.

Finally let M, indexed by m, be a set of global constraints. Global constraints do not involve just a single commodity path, but many or even all commodities. In a simplified model, for a given $m \in M$, let $B_m^k \subseteq A^k, k \in K$, be the subset of arcs on graph (V^k, A^k) included in the constraint definition. Specific coefficients $b_{mij}^k, m \in M$ are assigned to arcs $(i,j) \in B_m^k, k \in K$. Let $\underline{b}_m$ and $\overline{b}_m$, $m \in M$ be the lower and upper bound values on a global constraint m.

Note that an arc $(i,j) \in A^k, k \in K$ is infeasible if it is not possible to go from node i to node j while respecting the resource intervals at both nodes. All infeasible arcs are then excluded from the sets A^k, for all $k \in K$. Conversely, all arcs $(i,j) \in A^k$ respect the following conditions:

$$a_i^{kr} + t_{ij}^{kr} \leq b_j^{kr}, \quad \forall\, k \in K, \quad \forall\, r \in R.$$

Formulation: The mathematical formulation given below makes use of binary network flow variables $\mathbf{X}^k = (X_{ij}^k, (i,j) \in A^k), k \in K$, and resource variables $\mathbf{T}^k = (T_i^{kr}, r \in R, i \in V^k), k \in K$.

Vehicle fleet planning and crew scheduling problems refer to a large class of optimization problems in which vehicles and crews must be assigned to fixed time-tabled tasks in such a way that:

- a properly defined cost function $\sum_{k \in K} c^k(\mathbf{X}^k, \mathbf{T}^k)$ is minimized;
- each task in F is carried out once;
- a given set of global and local constraints are satisfied.

Following the notation used in Desrosiers, Dumas, Solomon and Soumis (1993), the multicommodity network flow model with resource constraints is:

$$\text{Minimize} \sum_{k \in K} c^k(\mathbf{X}^k, \mathbf{T}^k) \tag{7.1}$$

subject to:

$$\sum_{k \in K} \sum_{(i,j) \in A^k_f} X^k_{ij} = 1, \qquad \forall\, f \in F \tag{7.2}$$

$$\underline{b}_m \leq \sum_{k \in K} \sum_{(i,j) \in B^k_m} b^k_{mij}\, X^k_{ij} \leq \overline{b}_m, \qquad \forall\, m \in M \tag{7.3}$$

$$\sum_{j:(o(k),j) \in A^k} X^k_{o(k),j} = 1, \qquad \forall\, k \in K \tag{7.4}$$

$$\sum_{i:(i,j) \in A^k} X^k_{ij} - \sum_{i:(j,i) \in A^k} X^k_{ji} = 0, \qquad \forall\, k \in K,\ \forall\, j \in V^k \setminus \{o(k), d(k)\} \tag{7.5}$$

$$\sum_{i:(i,d(k)) \in A^k} X^k_{i,d(k)} = 1, \qquad \forall\, k \in K \tag{7.6}$$

$$X^k_{ij}(T^{kr}_i + t^{kr}_{ij} - T^{kr}_j) \leq 0, \qquad \forall\, k \in K,\ \forall\, r \in R,\ \forall\, (i,j) \in A^k \tag{7.7}$$

$$a^{kr}_i \leq T^{kr}_i \leq b^{kr}_i, \qquad \forall\, k \in K,\ \forall\, i \in V^k,\ \forall\, r \in R \tag{7.8}$$

$$X^k_{ij} \geq 0, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{7.9}$$

$$X^k_{ij} \quad \text{binary}, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k. \tag{7.10}$$

The above formulation generalizes the previous ones given in Sections 2 and 5. The cost function (7.1) may be linear, such as $\sum_{k \in K} \sum_{(i,j) \in A^k} c^k_{ij} X^k_{ij}$, or be a non-decreasing non-linear function using the operators $\sum$, min and max. Relations (7.2) require that each operational task be performed once. Relations (7.3) describe the global set of constraints, where coefficient values $\underline{b}_m$, $\overline{b}_m$ and b^k_{mij} may be restricted to integer values. Special cases of (7.3) are the fleet size constraints (2.3), (2.11) and (5.3) of the previous sections. This type of constraints can also represent global constraints on the number of full time and part time bus drivers in a feasible schedule, or airline base constraints related to the number of available crews. Constraint sets (7.4)–(7.10) describe the solution space of a shortest path problem with resource constraints introduced in Section 4.

Other types of local constraints, such as constraint sets (6.26)–(6.27) for the precedence and coupling requirements already encountered in pick-up and delivery problems, can be introduced in the proposed model. The dynamic programming solution approach must therefore be adapted accordingly to these new restrictions.

The previous model is easily generalized in two additional ways. First, the cost component can be regarded as a resource which is cumulated along the path structure of a subproblem. Since this kind of subproblem is solved using a forward dynamic programming algorithm, nonlinear cost functions are easily treated. The basic condition on a valid cost function is that it be non-decreasing (Desrochers, Gilbert, Sauvé and Soumis 1992, and Desrosiers, Dumas, Solomon and Soumis 1993). Therefore nonlinear resource functions can be used to model very complex conditions. The second extension concerns the set of global constraints which can include constraints written in terms of the resource variables. Such restrictions may appear, for example, in aircraft routing problems in a situation where start times of flights are flexible within time intervals and, additionally, there must be a delay of at least 45 minutes between consecutive flight legs with the same origin-destination pair. Another example is the imposition of separate time windows on school starting and ending times which imply an adjustment of the time windows assigned to the arrival and starting times of bus-trips (Ferland and Fortin 1989). These restrictions, which may involve more than a single path, therefore appear as global constraints.

A Dantzig-Wolfe Decomposition: Following the same principles described in Sections 2.3, 5.3 and 6.2, the decomposition scheme is applied to (7.1)–(7.10), i.e., the master problem includes relations (7.1)–(7.3), while the column generators or subproblems come from (7.4)–(7.10) and a modified cost function described below.

Each column $p \in \Omega^k$ of the master problem can be represented by a path satisfying constraint sets (7.4)–(7.10). This path p is in turn described by binary flow values x^k_{ijp}, $(i,j) \in A^k$, $k \in K$. Any solution X^k_{ij} to the master problem can be expressed as a non-negative convex combination of a finite number of paths, i.e.,

$$
\begin{aligned}
X^k_{ij} = \sum_{p \in \Omega^k} x^k_{ijp} \theta^k_p, \qquad & \forall\, k \in K,\ \forall\, (i,j) \in A^k \\
\sum_{p \in \Omega^k} \theta^k_p = 1, \qquad & \forall\, k \in K \\
\theta^k_p \geq 0, \qquad & \forall\, k \in K,\ \forall\, p \in \Omega^k.
\end{aligned}
$$

Define next the constants a_{fp}^k and b_{mp}^k:

$$a_{fp}^k = \sum_{(i,j)\in A_f^k} x_{ijp}^k, \qquad \forall\, k \in K, \forall\, f \in F,\ \forall\, p \in \Omega^k, \tag{7.11}$$

$$b_{mp}^k = \sum_{(i,j)\in B_m^k} b_{mij}^k x_{ijp}^k, \qquad \forall\, k \in K,\ \forall\, m \in M,\ \forall\, p \in \Omega^k. \tag{7.12}$$

The binary constant a_{fp}^k takes value 1, if path $p \in \Omega^k$, $k \in K$ covers task $f \in F$ and 0, otherwise, while the constant b_{mp}^k takes integer values. Given the flow values describing a path $p \in \Omega^k$, $k \in K$, i.e.,

$$\mathbf{x}_p^k = (x_{ijp}^k,\ (i,j) \in A^k), \qquad k \in K,\ p \in \Omega^k,$$

and the resource values along that path, i.e.,

$$\mathbf{t}_p^k = (t_{ip}^{kr},\ i \in V^k,\ r \in R), \qquad k \in K,\ p \in \Omega^k,$$

the cost of such a path p is evaluated as

$$c_p^k = c(\mathbf{x}_p^k, \mathbf{t}_p^k), \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k. \tag{7.13}$$

Making the substitutions for X_{ij}^k in (7.1)–(7.3) gives a linear program which is close to the linear relaxation of a set partitioning type formulation:

$$\text{Minimize} \sum_{k\in K} \sum_{p\in\Omega^k} c_p^k \theta_p^k \tag{7.14}$$

subject to:

$$\sum_{k\in K} \sum_{p\in\Omega^k} a_{fp}^k \theta_p^k = 1, \qquad \forall\, f \in F \tag{7.15}$$

$$\underline{b}_m \le \sum_{k\in K} \sum_{p\in\Omega^k} b_{mp}^k \theta_p^k \le \overline{b}_m, \qquad \forall\, m \in M \tag{7.16}$$

$$\sum_{p\in\Omega^k} \theta_p^k = 1, \qquad \forall\, k \in K \tag{7.17}$$

$$\theta_p^k \ge 0, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k. \tag{7.18}$$

As previously presented in Section 5.3, column coefficients of the master problem formulation are generated by solving resource constrained shortest path problems (see Section 4 for their solution through dynamic programming algorithms). Given the dual variables α_f, $f \in F$, β_m, $m \in M$, and γ^k, $k \in K$, associated respectively with (7.15), (7.16) and

(7.17) in the linear program above, the marginal cost $\overline{c}_p^k$, $k \in K$, $p \in \Omega^k$ of a path is given by

$$\overline{c}_p^k = c^k(\mathbf{x}_p^k, \mathbf{t}_p^k) - \sum_{f \in F} \alpha_f a_{fp}^k - \sum_{m \in M} \beta_m b_{mp}^k - \gamma^k. \tag{7.19}$$

Considering the coefficient definitions (7.11) and (7.12), the dual variables must be properly assigned to the adequate arcs in the set A^k. The marginal cost of a path is thus defined from all the arc parameters, i.e., resources consumed, arc costs, and dual variables. In the case of more complex models involving nonlinearities, the marginal costs are computed using the $\sum$, min and max operators. The nonlinearities introduced by the min and max operators are treated by using a multi-dimensional linear cost during the paths' construction, and by applying min and max operators once the paths are completed.

Optimal Integer Solution: The Dantzig-Wolfe decomposition/column generation scheme is embedded in a branch-and-bound search tree. Integer solutions are related to the original formulation (7.1)–(7.10). Branch-and-bound strategies can then be applied to the flow variables X_{ij}^k, $(i,j) \in A^k$, $k \in K$, to the resource variables T_i^{kr}, $i \in V^k$, $k \in K$, $r \in R$, and to any sets of constraints added to the set M. Every decision taken on the original formulation (7.1)–(7.10) can be transferred, either to the master problem or to the subproblems, and thus it is easily usable in the proposed Dantzig-Wolfe decomposition/column generation approach for integer programs.

7.2 The Bus Driver Scheduling Problem

The Bus Driver Scheduling Problem consists of covering every part of the bus schedule by bus drivers at minimum cost. The bus schedule defines the vehicle blocks, each vehicle block (or simply block) being a bus trip starting at the depot and going back to the depot. Along such a block, there are relief points where there can be a change of driver. The portion of a block which falls between two consecutive relief points, and which is thus always served by a single driver, is a task. A piece of work consists of several consecutive tasks in a block to be performed by a single driver.

The internal regulations of a bus company, and the collective agreement between the drivers' union and the transit operators define all of the details relevant to duty (workday) and schedule feasibility. A duty consists of one or more pieces of work executed by the same driver. In cases where a duty is composed of more than one piece of work, breaks or unworked periods are inserted between the pieces. The feasibility of a duty depends both on the length of the pieces of work and on the length of the breaks. It is also restricted by

constraints such as limits on the number of pieces of work in a duty, the total worked time, the paid time, or the total spread. The collective agreement may also classify the duties into types and define constraints on the global manpower schedule.

Literature: Three main types of algorithms have been developed to solve the urban transit crew scheduling problem and later implemented: the runcutting method, HASTUS, and set partitioning methods. These methods and others have been described in detail in the proceedings of the last four workshops on computer-aided transit scheduling (Wren 1981, Rousseau 1985, Daduna and Wren 1988, and Desrochers and Rousseau 1992) and in the survey of Carraresi and Gallo (1984). Here we restrict our attention to optimal approaches: the set partitioning method and its variants. The reader is however referred to branch-and-bound algorithms for the exact solution of particular cases of the Bus Driver Scheduling Problem arising only when the *spread time constraint* or the *working time constraint* are imposed. These are given in Fischetti, Martello and Toth (1987) and (1989), respectively.

The set partitioning method uses a set partitioning type problem (see (7.14)–(7.18) above) to choose a set of feasible duties (columns) which cover all the tasks (rows) in such a way that a minimal cost schedule is obtained. The variable θ_p^k takes the value 1, if duty $p \in \Omega^k$, $k \in K$ is chosen, and 0, otherwise, where the cost of duty $p \in \Omega^k$ is c_p^k, $k \in K$. The partitioning constraints (7.15) guarantee that each task $f \in F$ will be covered by one duty p, a_{fp}^k being equal to 1, if the duty $p \in \Omega^k$, $k \in K$ covers the task f, and 0, otherwise. The additional constraints (7.16) restrict the proportion of duties of a given type in the schedule. Convex combination constraints (7.17) are aggregated whenever initial conditions are identical for a k-group of indices, and these aggregated constraints are integrated to the set of global constraints (7.16). Except for very small size problems, it is impossible to enumerate all the feasible columns, so that heuristic rules are used for the elimination of improbable relief points and less efficient workdays (Heurgon 1975, Manington and Wren 1975, Ryan and Forster 1981, Mitra and Darby-Dowman 1985, and Wren, Smith and Miller 1985). The reader is also referred to Falkner and Ryan (1992) for a recent successful implementation of a set partitioning approach for Bus Crew Scheduling in Christchurch, New Zealand.

The Dantzig-Wolfe decomposition/column generation approach can be used to avoid the explicit use of the extremely large number of columns in the set partitioning type formulation of the crew scheduling problem. It permits to implicitly consider all the feasible columns using a subproblem generator. This subproblem consists of finding the minimal marginal cost feasible duty for a driver. An optimal implementation of this approach was developed by Desrochers and Soumis (1989). The subproblems are shortest path problems

with resource constraints, where the resource constraints insure that each feasible path corresponds to a feasible workday. Details on how to model the subproblem are presented in Desrochers, Gilbert, Sauvé and Soumis (1992). The paper describes the design of the time-space network, the resources to account for the number of pieces of work and the spread, and the modelling of more complicated situations, such as overtime and spread premiums, and constraints on the duration of a half-day or on workday types.

Computational Results: Results on real life problems involving 167 and 235 tasks and requiring 3 and 5 resources are reported in Desrochers and Soumis (1989). Integer solutions with a gap of .05% and .07% with respect to the linear programming lower bounds were obtained by branch-and-bound. The commercial software package CREW-OPT based on this approach has been integrated into the HASTUS System (Hamer and Séguin 1992) and is in operation in the cities of Lyon and Toulouse, France. The main advantage of the Dantzig-Wolfe decomposition/column generation solution approach is that it always produces very high quality feasible schedules which do not require any manual adjustments. This column generation method has also been used in a successful benchmark for train driver assignment with up to 15.3% improvement over manual solutions at East Japan Rail. Tested problems ranged from 384 to 779 tasks.

7.3 The Airline Crew Pairing Construction Problem

In recent years, several surveys have been published on the classical airline crew scheduling problem, i.e., the crew pairing construction problem (Etschmaier and Mathaisel 1985, Gershkoff 1989, and Barutt and Hull 1990). The subject is also devoted a chapter in the book by Dušan (1988). They express the renewed current interest in this problem.

Let us first describe some terms used in the airline industry. A flight leg (or leg) is the portion of a flight between a take-off and the subsequent landing. A duty starts with a briefing period, which is then followed by several legs separated by connections and it ends with a debriefing period. A deadhead leg is a leg where a crew member is assigned to travel as a passenger. It is used to transfer a crew member to a city where there is an undersupply, or to return a crew member to its base. A pairing is a trip from the base and back to the base, composed of one or more duties separated by rest periods.

The purpose of the classical airline crew scheduling problem is to construct a set of minimal cost crew pairings covering all the legs and satisfying all the rules in the workers convention and in the air traffic regulations. Some constraints, such as minimum and

maximum connection time, maximum time flown, and maximum time in operation relate to duty construction. Other constraints, such as minimum and maximum rest period, the 8 in 24 rule which forbids to allocate more than 8 hours of flight time in 24 to a pilot without extra compensation, restrict the pairing construction. There can also be global constraints on the crew schedule to insure a proper allocation of the pairings to the bases.

One of the most important aspects of the crew pairing construction problem is cost evaluation. The costs can be divided into three components: crew costs, accommodation costs, and penalty costs. The crew costs include the basic crew salaries, and the various guarantees on minimum flight time per duty, on the duty duration and on the pairing duration. The deadhead costs are also included in the crew costs, while the post-pairing cost may either be evaluated or not. The accommodation costs include the per diem or meal allowance, and the hotel and transportation costs incurred every time a crew is spending a rest period away from its base. The penalty costs are used mainly to penalize undesirable but legal pairings, or to increase the robustness of the crew schedule. For example, there often is a penalty on aircraft change during a duty, to reduce the consequences of a potential plane delay on the whole system.

Literature: The airline crew pairing construction problem can be formulated as a set partitioning type problem similarly to the bus crew scheduling problem. In air transportation, rows correspond to flight legs and the columns correspond to feasible pairings (see Arabeyre, Fearnley, Steiger and Teather 1972). Some interesting results on problems with relatively few feasible pairings have been obtained by Marsten, Muller and Killion (1979), Marsten and Shepardson (1981) and Ryan and Garner (1985). Several recent optimization systems based on the a priori generation of a subset of pairings are described in Baker, Bodin and Fisher (1985), Gershkoff (1989), Anbil, Gelman, Patty and Tanga (1991), and Graves, McBride, Gershkoff, Anderson and Mahidhara (1993). These systems all make use of the following three main components: a generator, an optimizer and a local improvement module.

Hoffman and Padberg (1993) have presented a branch-and-cut approach to optimally solving set partitioning problems and set partitioning problems with base constraints. The branch-and-cut solver is applied to problems for which feasible pairings had been given a priori. It generates cutting planes based on the underlying structure of the polytope and incorporates these cuts into a tree-search algorithm that uses automatic reformulation procedures, heuristics and linear programming technology to assist in the solution.

Wedelin (1993) presents an approximation algorithm for 0–1 integer programming problems, which has been used as a solver for a set covering formulation with constraints. The algorithm, originally developed in a probabilistic framework, can be interpreted in this context as a simple ascent algorithm for the dual LP-relaxation, along with an approximation scheme which manipulates the reduced costs to produce integer solutions. This method is used in a feedback process in conjunction with a heuristic pairing generator. Computational tests indicate that this method solves faster than other LP-software packages, such as CPLEX (CPLEX 1992), large problems up to $2 \cdot 10^6$ pairings while providing solution quality similar.

Problems with a huge number of feasible pairings have been solved by column generation. A heuristic implementation of the column generation approach was developed by Crainic and Rousseau (1987). The new negative marginal cost columns were generated using a partial enumeration method. An integer solution was produced using a heuristic single branch enumeration method. Another implementation of the column generation approach was developed by Lavoie, Minoux and Odier (1988). The columns were generated by solving a classical shortest path problem in an expanded network which integrates all the constraints. The reduced cost of a pairing was evaluated by a linear approximation. This method produced significant savings (over 5%) at Air France compared to previous solutions on medium and long haul problems. The problems on which it was applied involved up to 1100 flight legs. The same approach was recently used on long haul problems by Barnhart, Johnson, Anbil and Hatay (1991). Three of the previous authors, Barnhart, Hatay and Johnson (1992), also developed a new methodology to improve crew pairing solutions through the efficient selection of deadhead flights from other airline companies. Their methodology uses the dual information determined in solving the linear programming relaxation of the crew pairing problem to build arrival and departure cost profiles at each station. These profiles provide a mechanism to price-out potential deadhead flights. Preliminary numerical results indicate significant reductions in crew costs.

An exact algorithm based on Dantzig-Wolfe decomposition/column generation is presented in Desrosiers, Dumas, Desrochers, Soumis, Sansó and Trudeau (1991). The minimal marginal cost columns are generated with resource constrained shortest path subproblems. Many subproblem networks have been considered, where flight legs or duties have appeared either as nodes or as arcs. Networks representing duties as nodes or as arcs insure feasibility relatively to duty restrictions by network construction. Networks representing flight legs on nodes or on arcs are smaller but need more resources. A detailed description of the design of the networks and the resources, as well as some complex cost functions are given in Desrosiers, Dumas, Solomon and Soumis (1993).

Computational Results: Very recent results presented in Desrosiers, Dumas, Solomon and Soumis (1993) show the strength of the approach used to solve optimally model (7.1)–(7.10) (see also Desrosiers, Dumas, Desrochers, Soumis, Sansó and Trudeau 1991). A 6–resource model was applied to a regional carrier's data set. The weekly schedule obtained for a 986 leg problem resulted in a 5.3% improvement over the sum of the costs of the optimal solutions obtained separately for each of the five fleet involved. On seven medium haul test problems from Air France, of sizes 154, 280, 342, 392, 477 legs (April'90 data set), 566 and 701 legs (January'92 data set) respectively, a 4–resource model allowed an average improvement of 5.9% over previous solutions.

Finally, on two test problems from a North American carrier involving 313 and 282 legs, respectively, and also over 2,000 deadhead legs, a 5–resource model resulted in savings of respectively 4.0% and 9.6% over other computerized solutions. If the objective considered is the excess crew cost (also called pay-and-credit), than the savings are of 31.5% and of 37.4%, respectively. In both test problems, the number of pairings generated was less then 20,000. This represents a very small fraction of the total number of feasible pairings, which was estimated at $190 * 10^{12}$, i.e., 190 million of millions pairings in the case of the 282 leg problem.

The multi-commodity flow model with resource constraints solved using a Dantzig-Wolfe decomposition/column generation approach embedded into a branch-and-bound search tree has proved to be a very strong mathematical tool. Eight of the fifteen test problems mentioned above have been solved to optimality, while the optimality gap for the seven others was always within 1%. For example, the 9.6% improvement on the above 282 leg problem is almost optimal, the integrality gap being of \$68 over the LP lower bound of \$206,651. This primal approach, which can implicitly consider an extraordinarily huge number of pairings, is the only one which can also provides the optimality gap value on a feasible schedule solution.

During the writing of this survey, an optimization system based on this approach was used to solve test problems from several airline companies. The larger test problems involved more than 4,000 flight legs and they were all solved on workstations. For these, we have designed a solution procedure using time slice decomposition with overlapping time periods, and hence, we had to apply the algorithm several times. This procedure is similar to the one mentioned in the previous section and used for large scale dial-a-ride problems. On each of the test problems, this system has obtained the best optimized solutions compared to any other system. This approach has proved usable in practice and it has opened the way for optimal and near optimal solutions to many other applications in crew scheduling and vehicle fleet planning problems.

8 Conclusions and Perspectives

In this paper we have tried to illustrate the spectacular growth of the field of time constrained routing and scheduling during the past fifteen years. We have provided an extensive overview of the models and algorithms developed, focusing on optimization methods. While heuristics will remain a viable tool for very large-scale and/or complex problems, optimization methods are becoming effective for solving practical size problems. The size of problems solved by such methods increases constantly. This new generation of optimal algorithms blends the effectiveness of advanced optimization methods, tailored to take advantage of special problem structures, with the efficiency of advanced computer science methods, and the computing power of workstations. Several models and algorithms developed over this period have already proved themselves in a number of successful implementations, while challenging new applications are being attacked every year.

For complex problems with fixed or flexible schedules, optimal algorithms based on Dantzig-Wolfe decomposition/column generation schemes have emerged as the most performant solution methodologies. The equivalent dual schemes based on Lagrangean Relaxation are currently showing very encouraging results. Dynamic programming has proved as one of the fundamental building blocks of these approaches. Furthermore, for simpler and sufficiently constrained problem structures, it constitutes a straightforward method for obtaining an optimal solution to realistic size problems. In addition, it is also flexible in that it allows the utilization of a variety of objective functions. All the above methods are also very flexible from the perspective that they can be transformed into optimization-based heuristics. The success of these optimal methodologies are an expression of the maturity level reached in this field. They constitute a solid platform for future research developments.

The work on mathematical models amenable to be solved by optimal methods in different time constrainted areas has led to a first unified framework. This is a multi-commodity network flow model with additional resource constraints. Dantzig-Wolfe decomposition can be applied to it to provide the classical set partitioning formulation which in turn is solved by column generation. Resource variables help to manage complex nonlinear cost functions and difficult local constraints (e.g., time windows, vehicle capacity, and union rules). Forward dynamic programming algorithms can handle these resources in shortest path computations. These developments have eventually provided insight into the design of the branch-and-bound search tree: all decisions must relate to the original multi-commodity model. This

methodology can be extended far beyond the routing and scheduling field. It can be applied to develop optimal approaches to integer programs solved by column generation, an open problem since the early '60s. While the framework has been used in many applications, it can still be generalized. One way could be through new global constraints on resources. This could be complemented by more complexly constrained shortest paths.

A more specific research direction concerns the VRPTW area. We believe that further advances will be generated by promising research on polyhedral structures. For example, the linear programming bound in the Dantzig-Wolfe decomposition approach to the VRPTW could be enhanced by means of cuts. These have been instrumental for the solution of loosely constrained problems, while at the other extreme, dynamic programming embedded in decomposition approaches has solved sufficiently constrained problems. Combining these methodologies could lead to methods capable of solving moderately tight problems. The interface with artificial intelligence could provide means for selecting an appropriate method for a given problem structure. Problem characteristics such as routing structure, time window tightness and distribution, vehicle capacity, length of the scheduling horizon and the number of customers per route are important knowledge acquisition tools that could be embedded in expert systems for the VRPTW.

Future research should also continue to address computational efficiency issues, as optimal methods are being used for increasingly larger problems. We expect that more powerful algorithms will come from the interface with computer science. Specifically, new data structures and parallel implementations could lead the way.

As companies are making a substantial effort toward the computer integration of their operations, much more knowledge should be transferred between logistics and manufacturing. In particular, the application of time constrained routing models and algorithms to manufacturing is an important research direction that has barely been tapped.

The increased capability of optimal methods has lead to some very interesting applications. We expect this trend to continue. There are a number of strategies that run across the successful implementations. One set involves preserving as much realism as possible in the mathematical model. This translates into having a global perspective on the real world problem at hand. Decomposition techniques can then partition the overall problem into models to be solved by powerful algorithms. Feedback mechanisms between these models allow the return to the overall problem. In addition, the underlying networks need to be close to the real life settings. Over the last decade we have witnessed the use of increasingly realistic network structures. This was and will continue to be a product of the augmenting strength of optimal solution techniques. Nevertheless, one may need to choose among different network representations. A related idea is the use of realistic objective functions. In

the current customer oriented business environment, lateness oriented objectives are such examples for routing problems. A second set involves simplification strategies. They will continue to play a central role in lifting the ceiling of solvable problem size. Mini-clustering is a good example of such a strategy.

The improved efficiency of optimal algorithms has substantially cut the planning lead time. By being run closer to the actual decision time, the algorithms can use more accurate data, and hence produce more accurate results. This is leading to a wider acceptance of these planning tools in practice. Other important research directions that we hope to see developing in the next few years, such as sensitivity analysis and reoptimization capabilities, could also significantly increase their acceptance. The recent work on soft time windows is a small step in this direction.

In addition to the further advances for optimization methods, there are a number of other key factors that are needed for the widespread utilization of these methodologies in practice. These are as valid for the routing and scheduling field as they are for any O. R. area. First, O. R. implementations should continue to integrate the new developments in mathematics, computer science and technology. Better information systems, data bases and computer graphics should partner optimization methods. So should powerful computers such as workstations and parallel machines. Teams are naturally the next ingredient. They should include researchers, programmers and business people. To be successful, the researchers on a team should rediscover the interdisciplinary nature of O. R. Too often, their narrow specialization has hindered the implementation process. As computer implementation strategies are an integral component of bringing to light the power of an optimization method, a team should also take advantage of the new generation of programmers. They have been around computers since early childhood and have been trained in computer science. Hence, they can harness the power of computers much better than their predecessors. Business people should be another element of such teams. They have the know-how necessary to sell O. R. solutions to interested companies. In addition they can serve the users' need to talk to non-mathematical people. Overall, each team member needs to also be a generalist in order to facilitate communication among the team members. A final factor is the necessary cooperation between O.R. people (teams) and users. An important element of this cooperation is the integration of the O.R. solutions within the current business practices of the company implementing the methodology.

We hope that this paper has offered a comprehensive view of this rapidly moving field. While it has reached a certain level of maturity, many important problems remain open. We can only hope that this paper has steered sufficient interest that many of its readers will embark on or continue their research in this field. The growth of the field can only be furthered by challenging competition among different research teams.

9 References

AFRATI, F., S. COSMADAKIS, C. PAPADIMITRIOU, G. PAPAGEORGIOU and N. PAPAKOSTANTINOU (1986), The Complexity of the Traveling Repairman Problem. *Theoretical Informatics and Applications 20,* 79–87.

AHUJA, R.K., T.L. MAGNANTI and J.B. ORLIN (1989), Network Flows, in *Handbooks in Operations Research and Management Science, volume 1: Optimization,* (G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todol, eds.), North-Holland, Amsterdam, 211–369.

ANBIL, R. E. GELMAN, B. PATTY and R. TANGA (1991), Recent Advances in Crew Pairing Optimization at American Airlines, *Interfaces 21*, 62–74.

ANEJA, Y.P., V. AGGARWAL and K.P.K. NAIR (1983), Shortest Chain Subject to Side Constraints. *Networks 13,* 295–302.

APPELGREN, L.H. (1969), A Column Generation Algorithm for a Ship Scheduling Problem. *Transportation Science 3,* 53–68.

APPELGREN, L.H. (1971), Integer Programming Methods for a Vessel Scheduling Problem. *Transportation Science 5,* 64–78.

ARABEYRE, J.P., J. FEARNLEY, C. STEIGER and W. TEATHER (1972), The Airline Crew Scheduling Problem: A Survey, *Transportation Science 3*, 140–163.

BAKER, E. (1983), An Exact Algorithm for the Time Constrained Traveling Salesman Problem. *Operations Research 31,* 938–945.

BAKER, E.K., L.D. BODIN and M. FISHER (1985), The Development of a Heuristic Set Covering Based System for Air Crew Scheduling, *Transportation Policy Decision Making 3*, 95–110.

BAKER, E. and J. SCHAFFER (1986), Computational Experience with Branch Exchange Heuristics for Vehicle Routing Problems with Time Window Constraints. *American Journal of Mathematical and Management Sciences 6,* 261–300.

BARNHART, C., L. HATAY and E.L. JOHNSON (1992), *Deadhead Selection for the Long-Haul Crew Pairing Problem.* Working Paper COC 92-01, School of Industrial and Systems Engineering, Georgia Tech., Atlanta, 34 pages.

BARNHART, C., E. JOHNSON, R. ANBIL and L. HATAY (1991), *A Column Generation Technique for the Long-Haul Crew Assignment Problem.* Industrial and Systems Engineering Reports Series, COC-91-01, Georgia Institute of Technology, Atlanta, Georgia, 30332.

BARUTT, J. and T. HULL (1990), Airline Crew Scheduling: Supercomputers and Algorithms. *SIAM News 23,* no 6, pages 1, 20–22.

BELL, W., L. DALBERTO, M. FISHER, A GREENFIELD, R. JAIKUMAR, P. KEDIA, R. MACK and P. PRUTZMAN (1983), Improving the Distribution of Industrial Gases with an Online Computerized Routing and Scheduling Optimizer. *Interfaces 13,* 4–23.

BELLMAN, R.E. (1958), On a Routing Problem. *Quarterly of Applied Mathematics 16,* 87–90.

BERTOSSI, A.A., P. CARRARESI and G. GALLO (1987), On Some Matching Problems Arising in Vehicle Scheduling Models. *Networks 17,* 271–281.

BIANCO, L., A. MINGOZZI, S. RICCIARDELLI and M. SPADONI (1989), *Algorithms for the Crew Scheduling Problem Based on the Set Partitioning Formulation*, Istituto di Analisi dei Sistemi ed Informatica del CNR, R. 280, 00185 Roma, Italy.

BLAIS, J.-Y., J. LAMONT, and J.-M. ROUSSEAU (1990), The HASTUS Vehicle and Manpower Scheduling System at the Société de Transport de la Communauté Urbaine de Montréal. *Interfaces 20,* 26–42.

BODIN, L. and L. BERMAN (1979), Routing and Scheduling of School Buses by Computer. *Transportation Science 13,* 113–129.

BODIN, L. and B. GOLDEN (1981), Classification in Vehicle Routing and Scheduling. *Networks 11,* 97–108.

BODIN, L., B. GOLDEN, A. ASSAD and M. BALL (1983), Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers and Operations Research 10,* 62–212.

BODIN, L., D. ROSENFIELD and A. KYDES (1978), UCOST: A Micro Approach to a Transit Planning Problem. *Journal of Urban Analysis 5,* 46–69.

BODIN, L. and T. SEXTON (1986), The Multi-Vehicle Subscriber Dial-a-Ride Problem. *TIMS Studies in the Management Sciences 26,* 73–86.

BÖKINGE, U. and D. HASSELSTRÖM (1980), Improved Vehicle Scheduling in Public Transport through Systematic Changes in Time-table. *European Journal of Operational Research 5,* 388–395.

BRAMEL, J. and D. SIMSHI-LEVI (1993), *On the Effectiveness of Set Partitioning Formulations for the Vehicle Routing Problem*. Working Paper, Graduate School of BUsiness, Columbia University.

CARPANETO D., M. DELL'AMICO, M. FISCHETTI and P. TOTH (1989), A Branch and Bound Algorithm for the Multiple Vehicle Scheduling Problem. *Networks 19,* 531–548.

CARPANETO, G. and P. TOTH (1980), Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem. *Management Science 26,* 736–743.

CARRARESI, P. and G. GALLO (1984), Network Models for Vehicle and Crew Scheduling. *European Journal of Operational Research 16,* 139–151.

CASCO, D., B. GOLDEN and E. WASIL (1988), Vehicle Routing with Backhauls: Models, Algorithms and Case Studies, in *Vehicle Routing: Methods and Studies*, (B.L. Golden, A.A. Assad, eds.), North-Holland, Amsterdam, 127–147.

CEDER, A. and H. I. STERN (1981), Deficit Function Bus Scheduling with Deadheading Trip Insertions for Fleet Size Reduction. *Transportation Science 15,* 338–363.

CHRISTIANSEN, M. (1993), *Ship Routing and Scheduling* (preliminary version), Ph.D. Dissertation, Department of Economics, The Norwegian Institute of Technology, Trondheim, Norway.

CHRISTOFIDES, N., A. MINGOZZI and P. TOTH (1981a), Exact Algorithms for the Vehicle Routing Problem Based on the Spanning Tree and Shortest Path Relaxations. *Mathematical Programming 20,* 255–282.

CHRISTOFIDES, N., A. MINGOZZI and P. TOTH (1981b), State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks 11,* 145–164.

CHVÁTAL, V. (1983), *Linear Programming*, Freeman.

CPLEX Reference Manual (1992), *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, CPLEX Optimization, Inc., Incline Village, NV 89451-9436, U.S.A.

CYRUS J.P. (1988), *The Vehicle Scheduling Problem: Models, Complexity and Algorithms.* Ph.D. Dissertation, Technical University of Nova Scotia, Halifax, Canada.

DADUNA, J.R. and A. WREN (eds.) (1988), *Computer-Aided Transit Scheduling.* Lecture Notes in Economic and Mathematical Systems 308, Springer Verlag, Berlin Heildelberg.

DANTZIG, G. and D. FULKERSON (1954), Minimizing the Number of Tankers to Meet a Fixed Schedule. *Naval Research Logistics Quarterly 1,* 217–222.

DANTZIG, G., D. FULKERSON and S. JOHNSON (1954), Solution of Large-Scale Traveling-Salesman Problem. *Operations Research 2,* 393–410.

DANTZIG, G.B. and P. WOLFE (1960), The Decomposition Algorithm for Linear Programming. *Operations Research 8* (also in *Econometrica 29,* 1961).

DEIF, I. and L. BODIN (1984), Extension of the Clarke and Wright Algorithm for Solving the Vehicle Routing Problem with Backhauling, in *Conference on Computer Software Uses in Transportation and Logistics Management*, (A.E. Kidder, ed.), Babson Park, MA, 75–96.

DELL'AMICO, M., M. FISCHETTI and P. TOTH (1993), Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem. *Management Science* (to appear).

DENARDO, E.V. and B.L. FOX (1979), Shortest-Route Methods: 1. Reaching, Pruning and Buckets. *Operations Research 27,* 161–186.

DESROCHERS, M. (1988), *An Algorithm for the Shortest Path Problem with Resource Constraints.* Cahiers du GERAD G–88–27, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6.

DESROCHERS, M., J. DESROSIERS and M. SOLOMON (1992), A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research 40,* 342–354.

DESROCHERS, M., J. GILBERT, M. SAUVÉ and F. SOUMIS (1992), CREW-OPT: Subproblem Modeling in a Column Generation Approach to Urban Crew Scheduling. in *Computer-Aided Transit Scheduling,* (M. Desrochers and J.-M. Rousseau, eds.), Lecture Notes in Economics and Mathematical Systems 386, Springer Verlag, Berlin Heidelberg, 395–406.

DESROCHERS, M. and G. LAPORTE (1991), Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints. *Operations Research Letters 10,* 27–36.

DESROCHERS, M., J.K. LENSTRA, M.W.P. SAVELSBERGH and F. SOUMIS (1988), Vehicle Routing with Time Windows: Optimization and Approximation, in *Vehicle Routing: Methods and Studies,* (B. Golden and A.A. Assad, eds.), North-Holland, Amsterdam, 65–84.

DESROCHERS, M. and J.-M. ROUSSEAU (eds.) (1992), *Computer-Aided Transit Scheduling.* Lecture Notes in Economics and Mathematical Systems 386, Springer Verlag, Berlin Heidelberg.

DESROCHERS, M. and F. SOUMIS (1988a), A Generalized Permanent Labeling Algorithm for the Shortest Path Problem with Time Windows, *INFOR 26,* 191–212.

DESROCHERS, M. and F. SOUMIS (1988b), A Reoptimization Algorithm for the Shortest Path Problem with Time Windows. *European Journal of Operational Research 35,* 242–254.

DESROCHERS, M. and F. SOUMIS (1989), A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science 23,* 1–13.

DESROSIERS, J., Y. DUMAS, M. DESROCHERS, F. SOUMIS, B. SANSÓ and P. TRUDEAU (1991), A Breakthrough in Airline Crew Scheduling. *Proceedings of the 26th Annual Meeting of the Canadian Transportation Research Forum*, Quebec City, 464–478.

DESROSIERS, J., Y. DUMAS, M. SOLOMON and F. SOUMIS (1993), *The Airline Crew Pairing Construction Problem.* Working Paper, GERAD, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6.

DESROSIERS, J., Y. DUMAS and F. SOUMIS (1986), A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows. *The American Journal of Mathematical and Management Sciences 6,* 301–325.

DESROSIERS, J., Y. DUMAS, F. SOUMIS, S. TAILLEFER and D. VILLENEUVE (1991), *An Algorithm for Mini-Clustering in Handicapped Transport.* Cahiers du GERAD G–91–02, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6.

DESROSIERS, J., J.-A. FERLAND, J.-M. ROUSSEAU, G. LAPALME and L. CHAPLEAU (1986), TRANSCOL: A Multi-Period School Bus Routing and Scheduling System. *TIMS Studies in the Management Sciences 22,* 47–71.

DESROSIERS, J. and É. GÉLINAS (1993), NorFA Research Course, Narvik, Norway, June 21–28.

DESROSIERS, J., P. PELLETIER, and F. SOUMIS (1983), Plus Court Chemin avec Contraintes d'Horaires, *RAIRO 17,* 357–377. (in French)

DESROSIERS, J., M. SAUVÉ and F. SOUMIS (1988), Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows. *Management Science 34,* 1005–1022.

DESROSIERS J., F. SOUMIS and M. DESROCHERS (1982), Routes sur un réseau espace-temps. *Comptes-rendus du congrès de l'ASAC, Recherche Opérationnelle 3,* 28–44. (in French)

DESROSIERS, J., F. SOUMIS and M. DESROCHERS (1984), Routing with Time Windows by Column Generation. *Networks 14,* 545–565.

DESROSIERS J., F. SOUMIS, M. DESROCHERS and M. SAUVÉ (1986), Methods for Routing with Time Windows. *European Journal of Operational Research 23,* 235–245.

DIJKSTRA, E. (1959), A Note on Two Problems in Connection with Graphs. *Numerische Mathematik 1,* 269–271.

DILWORTH, R. (1950), A Decomposition Theorem for Partially Ordered Sets. *Ann. of Math. 51,* 161–166.

DROR, M. (1994), *Note on the Complexity of the Shortest Path Models for Column Generation in the VRPTW*. Working Paper, Decision Sciences, College of Business, The University of Arizona, Tucson, Arizona 85721, 7 pages. Forthcoming in *Operations Research*.

DUMAS, Y., J. DESROSIERS, E. GÉLINAS and M.M. SOLOMON (1991), *An Optimal Algorithm for the Traveling Salesman Problem with Time Windows*. Cahiers du GERAD G–91–51, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6. Forthcoming in *Operations Research*.

DUMAS, Y., J. DESROSIERS and F. SOUMIS (1989), *Large Scale Multi-Vehicle Dial-a-Ride Problems*. Cahiers du GERAD G–89–30, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6.

DUMAS, Y., J. DESROSIERS and F. SOUMIS (1989), Minimisation d'une Fonction Convexe Séparable avec Contraintes de Rapport entre les Variables. *RAIRO 23,* 305–317. (in French)

DUMAS, Y., J. DESROSIERS and F. SOUMIS (1991), The Pick-up and Delivery Problem with Time Windows. *European Journal of Operational Research 54,* 7–22.

DUMAS, Y., M. SALOMON, M.M. SOLOMON, and L.N. VAN WASSENHOVE (1993), Discrete Lotsizing and Scheduling with Sequence Dependent Setup Times and Setup Costs, Working Paper, INSEAD, Fontainebleau, France.

DUMAS, Y., F. SOUMIS and J. DESROSIERS (1990), Optimizing the Schedule for a Fixed Vehicle Path with Convex Inconvenience Costs. *Transportation Science 24,* 145–152.

DUŠAN, T. (1988), *Airlines Operations Research*. Gordon and Breach, New York.

EL-AZM, A. (1985), The Minimum Fleet Size Problem and Its Applications to Bus Scheduling, in *Computer Scheduling of Public Transport 2* (J.-M. Rousseau, ed.), North-Holland, Amsterdam, 493–512.

ETSCHMAIER, M.M. and D.F.X. MATHAISEL (1985), Airline Scheduling: an Overview. *Transportation Science 19,* 127–138.

FALKNER, J.C. and D.M. RYAN (1992), EXPRESS: Set Partitioning for Bus Crew Scheduling in Christchurch, in *Computer-Aided Transit Scheduling,* (M. Desrochers and J.-M. Rousseau, eds.), Lecture Notes in Economics and Mathematical Systems 386, Springer Verlag, Berlin Heidelberg, 359–378.

FERLAND, J.A. and L. FORTIN (1989), Vehicle Routing with Sliding Time-Windows. *European Journal of Operational Research 38*, 213–226.

FINKE, G., A. CLAUS and E. GUNN (1984), A Two-Commodity Network Flow Approach to the Traveling Salesman Problem. *Congressus Numerantium 41,* 167–178.

FISCHETTI, M., S. MARTELLO and P. TOTH (1987), The Fixed Job Schedule Problem with Spread-Time Constraints. *Operations Research 35,* 849–858.

FISCHETTI, M., S. MARTELLO and P. TOTH (1989), The Fixed Job Schedule Problem with Working-Time Constraints. *Operations Research 37,* 395–403.

FISCHETTI, M. and P. TOTH (1989), An Additive Bounding Procedure for Combinatorial Optimization Problems. *Operations Research 37*, 319–328.

FISHER, M. (1989), *Optimal Solution of Vehicle Routing Problems using Minimum K-Trees.* Researh Report 89-12-13, Department of Decision Sciences, The Wharton School, University of Pennsylvania, Philadelphia.

FISHER, M., A. GREENFIELD, R. JAIKUMAR and P. KEDIA (1982), *Real-Time Scheduling of Bulk Delivery Fleet: Practical Application of Lagrangian Relaxation.* WP 82-10-11, Department of Decision Sciences, University of Pennsylvania, Philadelphia.

FISHER, M. and R. JAIKUMAR (1981), A Generalized Assignment Heuristic for Vehicle Routing. *Networks 11,* 109–124.

FISHER, M., K.O. JÖRNSTEN and O.B.G. MADSEN (1992), *Vehicle Routing with Time Windows.* Research Report 4C/1991, IMSOR, The Technical University of Denmark, Lyngby, DK-2800, Denmark.

FISHER, M. and P. KEDIA (1990), Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics. *Management Science 36*, 674–688.

FISHER, M.L. and M.B. ROSENWEIN (1989), An Interactive Optimization System for Bulk-Cargo Ship Scheduling, *Naval Research Logistics 36,* 27–42.

FORBES, M.A., J.N. HOLT and A.M. WATTS (1991), Exact Solution of Locomotive Scheduling Problems. *Journal of the Operational Research Society 42*, 825–831.

FORD, L.R. (1956), Network Flow Theory, *The Rand Corporation* 293.

FORD, L. and D.R. FULKERSON (1962), *Flows in Networks.* Princeton University Press, New Jersey.

FRIIS, J. (1985), *Rutelægning med Tidsvinduer: Metoder og Algoritmer*, Master Thesis, Research Report 11/85, IMSOR, The Technical University of Denmark, Lyngby, DK-2800, Denmark. (in Danish)

FULKERSON D.R. (1972), Flow Networks and Combinatorial Operations Research, in *Perspective on Optimisation*, (A.M. Geoffrion, ed.), Addison-Wesley, Reading, Mass., 139–171.

GAREY, M.R. and D.S. JOHNSON (1979), *Computer and Intractability: a Guide to the Theory of NP-completeness.* Freeman, San Francisco.

GÉLINAS, S. (1993), Private communication.

GÉLINAS, S., M. DESROCHERS, J. DESROSIERS and M.M. SOLOMON (1992), *A New Branching Strategy for Time Constrained Routing Problems with Application to Backhauling.* Cahiers du GERAD G–92–13, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6. Revised April 1994.

GEOFFRION, A.M. (1974), Lagrangian Relaxation and Its Uses in Linear Programming. *Mathematical Programming Study 2,* 82–114.

GENDREAU, M., G. LAPORTE and M.M. SOLOMON (1992), Single-Vehicle Routing and Scheduling to Minimize the Number of Missed Deadlines. Forthcoming in *Transportation Science*.

GERSHKOFF, I. (1989), Optimizing Flight Crew Schedules. *Interfaces 19,* 29–43.

GERTSBAKH, I. and H. STERN (1978), Minimal Resources for Fixed and Variable Job Schedules. *Operations Research 26,* 68–85.

GIRARD, P. (1990), *Conversion de Données et Comparaison des Résultats de TRANSCOL et GENCOL.* Département de génie électrique et informatique, Ecole Polytechnique de Montréal, Canada.

GOETSCHALCKX, M. and C. JACOBS-BLECHA (1989), The Vehicle Routing Problem with Backhauls. *European Journal of Operational Research 42,* 39–51.

GOLDEN, B. and A. ASSAD (1984), *The Simple Backhaul Problem.* Internal memo, Distribution Systems Technologies Inc., Columbia, Maryland.

GRAHAM, D. and H. NUTTLE (1986), A Comparison of Heuristics for a School Bus Scheduling Problem. *Transportation Research 20B,* 175–182.

GRAVES, G.W., R.D. McBRIDE, I. GERSHKOFF, D. ANDERSION and D. MAHIDHARA (1993), Flight Crew Scheduling, *Management Science 39*, 736–745.

GUIGNARD, M. and S. KIM (1987), Lagrangean Decomposition: A Model Yielding Stronger Lagrangean Bounds. *Mathematical Programming 39*, 215–228.

GUINET, A. (1984), *Le système T.I.R.: un système d'établissement de tournées industrielles routières*, Ph.D. dissertation, Université Claude Bernard, Lyon, France.

HALSE, K. (1992), *Modeling and Solving Complex Vehicle Routing Problems,* Ph.D. Dissertation no. 60, IMSOR, Technical University of Denmark, Lyngby, 372 pages.

HAMER, N. and L. SEGUIN (1992), The HASTUS System: New Algorithms and Modules for the 90s, in *Computer-Aided Transit Scheduling,* (M. Desrochers and J.-M. Rousseau, eds.), Lecture Notes in Economics and Mathematical Systems 386, Springer Verlag, Berlin Heidelberg, 17–29.

HANSEN, P. (1980), Bicriterion path problem, in *Lecture Notes in Economics and Mathematical Systems 177,* (G. Fandel and T. Gal, eds.) Springer-Verlag, Heidelberg, 109–127.

HANSEN, P., B. JAUMARD and M. POGGI de ARAGÃO (1991), Un algorithme Primal de Programmation Linéaire Généralisée pour les Programmes Mixtes. *Comptes Rendus de l'Académie des Sciences de Paris 313*, 557–560.

HAOUARI, M., P. DEJAX and M. DESROCHERS (1991), *Modelling and Solving Complex Vehicle Routing Problems using Column Generation.* Working Paper, LEIS, École Centrale, Paris.

HEURGON, E. (1972) Un Problème de Recouvrement: L'Habillage des Horaires d'une Ligne d'Autobus. *RAIRO 6*, 13–29. (in French)

HEURGON, E. (1975), *Preparing Duty Rosters for Bus Routes by Computer*, in Preprint Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services, Chicago.

HOFFMAN, K.L. and M. PADBERG (1993), Solving Airline Crew Scheduling Problems by Branch-and Cut. *Management Science 39*, 657–682.

HOUCK Jr., D.J., J.-C. PICARD, M. QUEYRANNE and R.R. VEMUGANTI (1980), The Travelling Salesman Problem as a Constrained Shortest Path Problem: Theory and Computational Experience. *Opsearch 17,* 93–109.

IOACHIM, I., S. GÉLINAS, J. DESROSIERS and F. SOUMIS (1993), *An Algorithm for the Shortest Path Problem with Time Windows and Linear Inconvenience Costs.* Working Paper, GERAD, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6.

IOACHIM, I., J. DESROSIERS, Y. DUMAS and M.M. SOLOMON (1991), *A Request Clustering Algorithm in Door-to-Door Transportation.* Cahiers du GERAD G–91–50, École des Hautes Études Commerciales, Montréal, Canada, H3T 1V6.

JAFFE, J.M. (1984), Algorithms for Finding Paths with Multiple Constraints. *Networks 14,* 95–116.

JAW, J., A. ODONI, H. PSARAFTIS and N. WILSON (1986), A Heuristic Algorithm for the Multi-Vehicle Advance-Request Dial-A-Ride Problem with Time Windows. *Transportation Research 20B,* 243–257.

JOKSCH, H.C. (1966), The Shortest Route Problem with Constraints. *J. Math. Analysis Applic. 14,* 191–197.

JÖRNSTEN, K., O.B.G. MADSEN and B. SØRENSEN (1986), *Exact Solution of the Vehicle Routing and Scheduling Problem With Time Windows by Variable Splitting.* Research Report 5/86, IMSOR, The Technical University of Denmark, Lyngby, DK-2800, Denmark.

KIM, T. (1985), *Solution Algorithms for Sealift Routing and Scheduling Problems.* Ph.D. Dissertation, M.I.T., Boston.

KNIGHT, K. and J. HOFER (1968), Vehicle Scheduling with Timed and Connected Calls: A Case Study. *Operational Research Quarterly 19,* 299–310.

KOHL, N. and O.B.G. MADSEN (1993), *An Optimization Algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangean Relaxation.* Research Report 17/1993, IMSOR, The Technical University of Denmark, Lynejby, DK–2800, Denmark.

KONTORAVDIS, G. and J.F. BARD (1992), *Improved Heuristics for the Vehicle Routing Problem with Time Windows*, Working Paper, Department of Mechanical Engineering, The University of Texas, Austin, TX 78712–1063.

KOSKOSIDIS, Y.A., W.B. POWELL and M.M. SOLOMON (1992), An Optimization Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints. *Transportation Science 26*, 69–85.

KOLEN, A.W.J., A.H.G. RINNOOY KAN and H.W.J.M. TRIENEKENS (1987), Vehicle Routing with Time Windows. *Operations Research 35,* 266–273.

LAMATSCH, A. (1992), An Approach to Vehicle Scheduling with Depot Capacity Constraints, in *Computer-Aided Transit Scheduling,* (M. Desrochers and J.-M. Rousseau, eds), Lecture Notes in Economics and Mathematical Systems 386, Springer Verlag, Berlin Heidelberg, 181–195.

LANGEVIN, A., M. DESROCHERS, J. DESROSIERS and F. SOUMIS (1993), A Two-Commodity Flow Formulation for the Traveling Salesman and Makespan Problems with Time Windows. *Networks 23,* 631–640.

LAPORTE, G. (1992), The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research 59,* 345–358.

LAPORTE, G. and Y. NOBERT (1987), Exact Algorithms for the Vehicle Routing Problem. *Annals of Discrete Mathematics 31,* 147–184.

LASDON, L.S. (1970), *Optimization Theory for Large Systems*. MacMillan, New York.

LAVOIE, S., M. MINOUX and E. ODIER (1988), A New Approach of Crew Pairing Problems by Column Generation and Application to Air Transport. *European Journal of Operational Research 35,* 45–58.

LEMARÉCHAL, C. (1989), Non Differentiable Optimization, in *Handbooks in Operations Research and Management Science, volume 1: Optimization*, (G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todol, eds.), North-Holland, Amsterdam, 529–572.

LENSTRA, J.K. and A.H.G. RINNOOY KAN (1981), Complexity of Vehicle Routing and Scheduling Problems. *Networks 11,* 221–228.

LEVIN, A. (1971), Scheduling and Fleet Routing Models for Transportation Systems. *Transportation Science 5,* 232–255.

LIN, S. (1965), Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal 44,* 2245–2269.

LIN, S, and B. KERNIGHAN (1973), An effective heuristic algorithm for the traveling salesman problem. *Operations Research 21,* 498–516.

LUCENA, A. (1986), *Exact Solution Approaches for the Vehicle Routing Problem*. Ph.D. Thesis, Department of Management Science, Imperial College of Science and Technology, University of London.

MACULAN, N., P. MICHELON and G. PLATEAU (1992), *Column-Generation in Linear Programming with Bounding Variable Constraints and its Application in Integer Programming*, Working paper ES–268/92, Federal University of Rio de Janeiro, P.O. Box 68511, 21945 Rio de Janeiro, Brazil, 13 pages.

MADSEN, O.B.G. (1976), Optimal Scheduling of Trucks - A Routing Problem with Tight Due Times for Delivery, in *Optimization Applied to Transportation Systems*, (H. Strobel, R. Genser and M. Etschmaier, eds.), IIASA, International Institute for Applied System Analysis, CP-77-7, Laxenburgh, Austria, 126–136.

MADSEN, O., H. RAVN and J.R. RYGAARD (1993), *REBUS, A System for Dynamic Vehicle Routing for the Copenhagen Fire Fighting Company*. Research Report 2/1993, IMSOR, the Technical University of Denmark, Lyngby, DK-2800, Denmark.

MAGNANTI, T. (1981), Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects. *Networks 11,* 179–214.

MAGNANTI, T.L., J.F. SHAPIRO and M.H. WAGNER (1976), Generalized Linear Programming Solves the Dual. *Management Science 22,* 1195–1203.

MALANDRAKI, C. and M. DASKIN (1989), *A Cutting Plane Heuristic Algorithm for the Time Dependent Traveling Salesman Problem.* Working Paper, Department of Civil Engineering, Northwestern University, Evanston, Illinois.

MARSTEN, R.E., M.R. MULLER and C.L. KILLION (1979), Crew Planning at Flying Tiger: A Successful Application of Integer Programming. *Management Science 25*, 1175–1183.

MARSTEN, R.E. and F. SHEPARDSON (1981), Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications. *Networks 11,* 165–177.

MARTIN, G. (1981), Aircraft Scheduling Considered as an N-Task, M-Parallel Machine Problem with Start-Times and Deadlines. *INFOR 19,* 152–161.

MARTINS, E.Q.V. (1984), On a Multicriteria Shortest Path Problem. *European Journal of Operational Research 16,* 236–245.

MESQUITA, M. and J. PAIXÃO (1992), Multiple Depot Vehicle Scheduling Problem: A New Heuristic Based on Quasi-Assignment Algorithms, in *Computer-Aided Transit Scheduling,* (M. Desrochers and J.-M. Rousseau, eds.), Lecture Notes in Economics and Mathematical Systems 386, Springer Verlag, Berlin Heidelberg, 181–195.

MILLER, C., A. TUCKER and R. ZEMLIN (1960), Integer Programming Formulations and Traveling Salesman Problems. *J.A.C.M. 7,* 326–329.

MINOUX, M. (1975), Plus Court Chemin avec Contraintes: Algorithmes et Applications. *Ann. Télécom. 30,* 1–12. (in French)

MITRA, G. and K. DARBY-DOWMAN (1983), CRU-SCHED: A Computer Based Bus Crew Scheduling System using Integer Programming, in *Computer Scheduling of Public Transport 2*, (J.-M. Rousseau, ed.), North-Holland, Amsterdam, 223–232.

MOORE, E.F. (1959), *The Shortest Path Trough a Maze,* Proc. of International Symposium on the Theory of Switching, Part II, April 2-5, 1957, Harvard University Press, Cambridge, MA.

NEMHAUSER G.L. and L.A. WOLSEY (1988), *Integer and Combinatorial Optimisation.* John Wiley & Sons.

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

ORLOFF, C. (1976), Route Constrained Fleet Scheduling. *Transportation Science 10,* 149–168.

PAIXÃO, J. and I.M. BRANCO (1987), A Quasi-Assignment Algorithm for Bus Scheduling. *Networks 17,* 249–270.

PALLOTTINO, S. (1984), Shortest Path Methods: Complexity, Interrelations and New Propositions. *Networks 14,* 257–267.

PAPADIMITRIOU, C. (1977), The Euclidean Traveling Salesman Problem is NP-Complete. *Theoretical Computer Science 4,* 237–244.

PAPE, U. (1980), Algorithm 562: Shortest Path Lengths. *ACM Transactions on Mathematical Software 6,* 450–455.

POTVIN, J.-Y., T. KERVAHUT and J.-M. ROUSSEAU (1992), *A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows.* Publication 855, Centre de Recherche sur les Transports, Université de Montréal, Canada H3C 3J7.

POTVIN, J.-Y., G. LAPALME and J.-M. ROUSSEAU (1989), A Generalized K-opt Exchange Procedure for the MTSP. *INFOR 27,* 474–481.

POTVIN, J.-Y. and J.-M. ROUSSEAU (1993), A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows. *European Journal of Operational Research 66,* 331–340.

PSARAFTIS, H. (1980), A Dynamic Programming Solution to the Single-Vehicle, Many-to-Many, Immediate Request Dial-A-Ride Problem. *Transportation Science 14,* 130–154.

PSARAFTIS, H. (1983), An Exact Algorithm for the Single-Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows. *Transportation Science 17,* 351–357.

PSARAFTIS, H. (1986), Scheduling Large-Scale Advance-Request Dial-A-Ride Systems. *American Journal of Mathematical and Management Sciences 6,* 327–367.

PSARAFTIS, H.N., J.B. ORLIN, D. BIENSTOCK and P.M. THOMPSON (1985), *Analysis and Solution Algorithms of Sealift Routing and Scheduling Problems: Final Report,* Working Paper 1700-85, M.I.T., Sloan School of Management, MA 02139.

PSARAFTIS, H., M. SOLOMON, T. MAGNANTI and T. KIM (1990), Routing and Scheduling on the Shoreline with Release Times. *Management Science 36,* 212–223.

PULLEN, H. and M. WEBB (1967), A Computer Application to a Transport Scheduling Problem. *Computer Journal 10,* 10–13.

RAPPOPORT, H.K., L.S. LEVY, B.L. GOLDEN and K. TOUSSAINT (1992), A Planning Heuristic for Military Airlift. *Interfaces 22*, 73–87.

RAPPOPORT, H.K., L.S. LEVY, K. TOUSSAINT and B.L. GOLDEN (1992), *A Transportation Problem Formulation for the MAC Airlift Planning Problem,* Working Paper, University of Maryland, College Park, Maryland 20742.

RIBEIRO, C. and F. SOUMIS (1994), A Column Generation Approach to the Multiple Depot Vehicle Scheduling Problem, *Operations Research 42*, 41–52.

ROUSSEAU, J.-M. (ed.) (1985), *Computer Scheduling of Public Transport 2.* North-Holland, Amsterdam.

ROY, S., J.-M. ROUSSEAU, G. LAPALME and J. A. FERLAND (1984a), *Routing and Scheduling for the Transportation of Disabled Persons - The Algorithm.* Working Paper TP 5596E, Transport Canada, Transport Development Center, Montreal, Canada.

ROY, S., J.-M. ROUSSEAU, G. LAPALME and J. A. FERLAND (1984b), *Routing and Scheduling for the Transportation of Disabled Persons - The Tests.* Working Paper TP 5598E, Transport Canada, Transport Development Center, Montreal, Canada.

RYAN, D.M. and B.A. FORSTER (1981), An Integer Programming Approach to Scheduling, in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, (A. Wren, ed.), North-Holland, Amsterdam, 269–280.

RYAN, D.M. and K.M. GARNER (1985), The Solution of Air-Crew Scheduling Problems for Air New Zealand. *Proceedings of the* 21^{st} *Annual Conference of O.R.S.N.Z.,* 42–48.

RUSSELL, R. (1977), An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Conditions. *Operations Research 25,* 517–524.

SANSÓ, B., M. DESROCHERS, J. DESROSIERS, Y. DUMAS and F. SOUMIS (1990), *Modeling and Solving Routing and Scheduling Problems: GENCOL User Guide.* GERAD, 5255 Decelles, Montréal, Canada, H3T 1V6.

SAVELSBERGH, M.W.P. (1985), Local Search in Routing Problems with Time Windows. *Annals of Operations Research 4,* 285–305.

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

SAVELSBERGH, M.W.P. (1989), *The Vehicle Routing Problem with Time Windows: Minimizing Route Duration,* Working Paper, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.

SEXTON, T. and L. BODIN (1985a), Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling. *Transportation Science 19,* 378–410.

SEXTON, T. and L. BODIN (1985b), Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: II. Routing. *Transportation Science 19,* 411–435.

SEXTON, T. and Y. CHOI (1986), Pickup and Delivery of Partial Loads with Time Windows. *American Journal of Mathematical and Management Sciences 6,* 369–398.

SMITH, B. and A. WREN (1981), VAMPIRES and TASC: Two Successfully Applied Bus Scheduling Programs, in *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling.* (A. Wren, ed.), North-Holland, Amsterdam, 97–124.

SMITH, B.M. and A. WREN (1988), A Bus Crew Scheduling System using a Set Covering Formulation. *Transportation Research 22A,* 97–108.

SOLANKI, R.S. and F. SOUTHWORTH (1991), An Execution Planning Algorithm for Military Airlift. *Interfaces 21,* 121–131.

SOLOMON, M. (1983), *The Vehicle Routing and Scheduling Problem with Time Window Constraints: Models and Algorithms.* Ph.D. Dissertation, Department of Decision Sciences, University of Pennsylvania, Philadelphia.

SOLOMON, M. (1986), On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Networks 16,* 161–174.

SOLOMON, M. (1987), Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research 35,* 254–265.

SOLOMON, M., E. BAKER and J. SCHAFFER (1988), Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures, in *Vehicle Routing: Methods and Studies*, (B.L. Golden, A.A. Assad, eds.), North-Holland, Amsterdam, 85–106.

SOLOMON, M., A. CHALIFOUR, J. DESROSIERS and J. BOISVERT (1992), An Application of Vehicle-Routing Methodology to Large-Scale Larvicide Control Programs. *Interfaces 22*, 88–99.

SOLOMON, M.M. and J. DESROSIERS (1988), Time Window Constrained Routing and Scheduling Problems. *Transportation Science 22,* 1–13.

SOUMIS F., J. FERLAND and J.-M. ROUSSEAU (1980), A Large Scale Model for Airline Fleet Planning and Scheduling Problem. *Transportation Research 14B,* 191–201.

SOUMIS, F., J. DESROSIERS and M. DESROCHERS (1985), Optimal Urban Bus Routing with Scheduling Flexibilities, in *Lecture Notes in Control and Information Sciences 59,* (P. Thoft Christensen, ed.), Springer-Verlag, Berlin Heidelberg, 155–165.

SWERSEY, A. and W. BALLARD (1983), Scheduling School Buses. *Management Science 30,* 844–853.

THOMPSON, P.M. and H.N. PSARAFTIS (1989), *Cyclic Transfer Algorithms for Multi-Vehicle Routing and Scheduling Problems*, Working Paper 89-008, Leavey School of Business and Administration, Santa Clara University, CA 95053.

TSITSIKLIS, J. (1992), Special Cases of the Traveling Salesman and Repairman Problems with Time Windows. *Networks 22*, 263–282.

VAN DER BRUGGEN, L.J.J., J.K. LENSTRA and P.C. SCHUUR (1993), Variable-Depth Search for the Single-Vehicle Pickup and Delivery Problem with Time Windows, *Transportation Science 27*, 298–311.

WEDELIN, D. (1993), *Efficient Algorithms for Probabilistic Inference, Combinatorial Optimization and the Discovery of Causal Structure from Data*, Ph.D. Dissertation, Department of Computer Sciences, Chalmers University of Technology, Göteborg, Sweden.

WILSON, H., J. SUSSMAN, H. WANG and B. HIGONNET (1971), *Scheduling Algorithms for Dial-a-Ride Systems.* Urban Systems Laboratory Report USL TR-70-13, M.I.T., Boston.

WILSON, H. and H. WEISSBERG (1976), *Advanced Dial-a-Ride Algorithms Research Project: Final Report.* Report R76-20, Dept. of Civil Engineering, M.I.T., Boston.

WILSON, H. and N. COLVIN (1977), *Computer Control of the Rochester Dial-a-Ride System.* Report R77-31, Dept. of Civil Engineering, M.I.T., Boston.

WREN, A., B.M. SMITH and A.J. MILLER (1985), Complementary Approaches to Crew Scheduling, in *Computer Scheduling of Public Transport 2*, (J.-M. Rousseau, ed.), North-Holland, Amsterdam, 263–278.

WREN, A. (ed.) (1981). *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling.* North-Holland, Amsterdam.

YANO, C., T. CHAN, L. RICHTER, T. CUTLER, K. MURTY and D. McGETTIGAN (1987), Vehicle Routing at Quality Stores. *Interfaces 17,* 52–63.