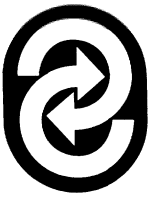
Fleet Management and Logistics



CENTRE FOR RESEARCH ON TRANSPORTATION
25TH ANNIVERSARY SERIES 1971 - 1996

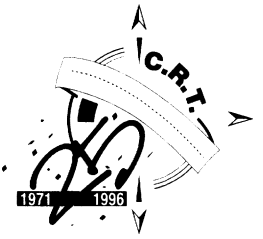
EQUILIBRIUM AND ADVANCED TRANSPORTATION MODELLING
edited by
Patrice Marcotte and Sang Nguyen

TELECOMMUNICATIONS NETWORK PLANNING
edited by
Brunilde Sansó and Patrick Soriano

FLEET MANAGEMENT AND LOGISTICS
edited by
Teodor Gabriel Crainic and Gilbert Laporte

AUTOMOBILE INSURANCE: Road Safety, New Drivers, Risks, Insurance Fraud and Regulation
edited by
Georges Dionne and Claire Laberge-Nadeau

TAKING STOCK OF AIR LIBERALIZATION
edited by
Marc Gaudry and Robert Mayes



Fleet Management and Logistics

Edited by
Teodor Gabriel Crainic
Université du Québec à Montréal
Gilbert Laporte
École des Hautes Études Commerciales de Montréal

Kluwer Academic Publishers
Boston/Dordrecht/London

Distributors for North, Central and South America:
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

Distributors for all other countries:
Kluwer Academic Publishers
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

Fleet management and logistics / edited by Teodor Gabriel Crainic, Gilbert Laporte.
p. cm. -- (Center for Research on Transportation 25th anniversary series, 1971 - 1996)
Includes bibliographical references (p.).
ISBN 0-7923-8161-0
1. Motor vehicle fleets--Planning--Mathematics. 2. Airlines--Planning--Mathematics. 3. Scheduling--Mathematics. 4. Business logistics--Mathematics. 5. Delivery of goods--Planning--Mathematics. 6. Hours of labor--Planning--Mathematics. I. Crainic, Teodor Gabriel. II. Laporte, Gilbert. III. Series.
TL165.F5249 1998
658.7'882--dc21 98-6780
CIP

Printed on acid-free paper.

Contents

Contributing Authors

ANN CAMPBELL
Georgia Institute of Technology
ann@isye.gatech.edu

I-MING CHAO
The Chinese Military Academy
8508341@ms.kscg.gov.tw

LLOYD CLARKE
Georgia Institute of Technology
lloyd.clarke@isye.gatech.edu

TEODOR GABRIEL CRAINIC
Université du Québec à Montréal
theo@crt.umontreal.ca

GUY DESAULNIERS
École Polytechnique de Montréal
guyd@crt.umontreal.ca

JACQUES DESROSIERS
École des HEC de Montréal
jacques.desrosiers@hec.ca

MICHEL GAMACHE
École Polytechnique de Montréal
michelga@crt.umontreal.ca

MICHEL GENDREAU
Université de Montréal
michelg@crt.umontreal.ca

BRUCE L. GOLDEN
University of Maryland, College Park
bgolden@umdacc.umd.edu

IRINA IOACHIM
TransQuest Inc., Atlanta
irina.ioachim@transquest.com

JAMES P. KELLY
University of Colorado, Boulder
james.kelly@colorado.edu

ANTON KLEYWEGT
Georgia Institute of Technology
anton.kleywegt@isye.gatech.edu

INMACULADA RODRÍGUEZ-MARTÍN
Universidad de La Laguna
irguez@ull.es

MARTINE LABBÉ
Université Libre de Bruxelles
mlabbe@smg.ulb.ac.be

GILBERT LAPORTE
École des HEC de Montréal
gilbert@crt.umontreal.ca

FRANÇOIS V. LOUVEAUX
Facultés U. Notre-Dame de la Paix, Namur
francois.louveaux@fundp.ac.be

JEAN-YVES POTVIN
Université de Montréal
potvin@iro.umontreal.ca

WARREN B. POWELL
Princeton University
powell@princeton.edu

MARTIN SAVELSBERGH
Georgia Institute of Technology
martin.savelsbergh@isye.gatech.edu

MARIUS M. SOLOMON
Northeastern University, Boston
solomon@neu.edu

FRANÇOIS SOUMIS
École Polytechnique de Montréal
soumis@crt.umontreal.ca

PAOLO TOTH
Universitá di Bologna
ptoth@deis.unibo.it

MICHEL TOULOUSE
University of Oklahoma
toulouse@bachman.cs.ou.edu

DANIELE VIGO
Universitá di Bologna
dvigo@deis.unibo.it

DANIEL VILLENEUVE
École Polytechnique de Montréal
danielv@crt.umontreal.ca

EDWARD A. WASIL
American University, Washington
ewasil@american.edu

Preface

TEODOR GABRIEL CRAINIC, DIRECTOR

The Centre for Research on Transportation (C.R.T.) was founded in 1971 by the Université de Montréal. From 1988 on, it is jointly managed by the Université de Montréal and its affiliated schools, the École des Hautes Études Commerciales and École Polytechnique. Professors, students and researchers from many institutions in the Montreal area join forces at the C.R.T. to analyze transportation, logistics and telecommunication systems from a multidisciplinary perspective.

The C.R.T. pursues three major, complementary objectives: training of high-level specialists; the advancement of knowledge and technology; the transfer of technology towards industry and the public sector. Its main field of expertise is the development of quantitative and computer-based models and methods for the analysis of urban, regional and intercity transportation networks, as well as telecommunication systems. This applies to the study of passenger and commodity flows, as well as to the socioeconomic aspects of transportation: policy, regulation, economics.

The twenty-fifth anniversary of the C.R.T. offered the opportunity to evaluate past accomplishments and to identify future trends and challenges. Five colloquia were thus organized on major research and application themes that also reflected our main research areas. They gathered together internationally renowned researchers who linked recent scientific and technological advances to modeling and methodological challenges waiting to be tackled, particularly concerning new problems and applications, and the increasingly widespread use of new technologies.

The present book, together with its four companions, is the result of these meetings. I wish to thank my colleagues who organized these colloquia and also edited the books: PATRICE MARCOTTE and SANG NGUYEN for **Equilibrium and Advanced Transportation Modelling**, BRUNILDE SANSÓ and PATRICK SORIANO for **Telecommunications Network Planning**, TEODOR GABRIEL CRAINIC and GILBERT LAPORTE for **Fleet Management and Logistics**, GEORGES DIONNE and CLAIRE LABERGE-NADEAU for **Automobile Insurance: Road Safety, New Drivers, Risks, Insurance Fraud and Regulation** and MARC GAUDRY and ROBERT MAYES for **Taking Stock of Air Liberalization**.

I also wish to take this opportunity to thank all companies and institutions who financially supported the celebration of our twenty-fifth anniversary and the publication of the five books: BELL, BUREAU D'ASSURANCE DU CANADA, CANADIAN PACIFIC RAILWAY, ÉCOLE DES HAUTES ÉTUDES COMMERCIALES DE MONTRÉAL, INRO CONSULTANTS INC., LES ENTREPRISES GIRO INC., MINISTÈRE DES TRANSPORTS DU QUÉBEC, SOCIÉTÉ DE L'ASSURANCE AUTOMOBILE DU QUÉBEC, TRANSPORTS CANADA and the UNIVERSITÉ DE MONTRÉAL.

Préface

TEODOR GABRIEL CRAINIC, DIRECTEUR

Le Centre de recherche sur les transports (C.R.T.) fut fondé en 1971 par l'Université de Montréal. En 1988, deux institutions affiliées, l'École des Hautes Études Commerciales et l'École Polytechnique, se sont jointes à celle-ci pour former un centre multidisciplinaire conjoint. Des professeurs, étudiants et chercheurs provenant principalement des universités de la région montréalaise s'y regroupent pour mettre en commun leurs compétences diverses afin d'analyser les systèmes de transport, logistiques et de télécommunication.

La mission du C.R.T. s'articule autour de trois axes complémentaires : la formation de spécialistes de haut niveau; l'avancement des connaissances et des technologies; le transfert de ces technologies vers l'industrie et les organismes publics. L'expertise du C.R.T. est principalement associée au développement de modèles et méthodes quantitatifs et informatiques d'analyse des réseaux de transport urbains, régionaux, interurbains et internationaux ainsi que des réseaux de télécommunication. Celle-ci s'applique tout autant au transport de passagers et de marchandises qu'aux aspects socioéconomiques : réglementation, sécurité, économie du transport.

L'année du vingt-cinquième anniversaire nous a fourni l'occasion de faire le point et de nous tourner vers l'avenir. Cinq colloques portant sur des thèmes actuels et reflétant les axes majeurs de recherche du C.R.T. sont issus de cette réflexion. Ces colloques, qui ont rassemblé des chercheurs de réputation internationale, ont permis de discerner des liens entre les réalisations récentes et les défis de modélisation et méthodologiques qui nous attendent, particulièrement dans les nouveaux champs de recherche et d'application, et dans l'utilisation grandissante de nouvelles technologies.

Ce livre et ses quatre compagnons sont le résultat tangible de ces colloques. Je remercie mes collègues qui les ont organisés et animés et qui ont également produit ces livres : PATRICE MARCOTTE et SANG NGUYEN pour **Equilibrium and Advanced Transportation Modelling**, BRUNILDE SANSÓ et PATRICK SORIANO pour **Telecommunications Network Planning**, TEODOR GABRIEL CRAINIC et GILBERT LAPORTE pour **Fleet Management and Logistics**, GEORGES DIONNE et CLAIRE LABERGE-NADEAU pour **Automobile Insurance : Road Safety, New Drivers, Risks, Insurance Fraud and Regulation** et MARC GAUDRY et ROBERT MAYES pour **Taking Stock of Air Liberalization**.

Je tiens également à remercier les compagnies et institutions qui nous ont appuyé financièrement dans la réalisation des célébrations du vingt-cinquième anniversaire et la publication des cinq livres : BELL, le BUREAU D'ASSURANCE DU CANADA, CANADIAN PACIFIC RAILWAY, L'ÉCOLE DES HAUTES ÉTUDES COMMERCIALES DE MONTRÉAL, LES CONSEILLERS INRO INC., LES ENTREPRISES GIRO INC., le MINISTÈRE DES TRANSPORTS DU QUÉBEC, la SOCIÉTÉ DE L'ASSURANCE AUTOMOBILE DU QUÉBEC, TRANSPORTS CANADA et l'UNIVERSITÉ DE MONTRÉAL.

INTRODUCTION

Teodor Gabriel Crainic and Gilbert Laporte

Transportation and logistics lie at the heart of human endeavour and sustain most social and economic activity. They also hold a central place in operations research. Over the years, their study has led to the development of several models and algorithms which have also been applied to a variety of other scientific and industrial fields.

Transportation is a complex field involving several players and decision levels, uncertainties and considerable capital expenditures. To remain competitive, this industry must more than ever rely on large amounts of data, sophisticated models and optimization techniques, as well as powerful computer and information technology. The variety and complexity of the field is reflected by the richness of research areas, models, methods, and software. The ten chapters of this book describe some of the most recent operations research advances in the field of transportation and logistics.

The first four contributions deal with various versions of the *Vehicle Routing Problem* (VRP), probably the most central model in distribution management. First, TOTH and VIGO review the exact solution methodologies for the VRP. In spite of more than two decades of research on exact algorithms for this difficult problem, it remains surprisingly hard to solve to optimality. Among the various approaches suggested in recent years, polyhedral methods seem to be highly promising. In the following chapter, GOLDEN, WASIL, KELLY and CHAO study the impact of metaheuristics for the approximate solution of the VRP. Such methods, tabu search in particular, have been highly successful on a large variety of combinatorial optimization problems. The development of powerful new domain exploration strategies for tabu search and the hybridization of this technique with evolutionary methods should prove a rich avenue of research in years to come. The next contribution, by DESAULNIERS, DESROSIERS, IOACHIM, SOLOMON, SOUMIS and VILLENEUVE, is an overview of solution methodologies for time constrained vehicle routing and crew scheduling problems. One of the most interesting advances in this area is the development of column generation methods capable of solving large-scale problems to optimality or near-optimality. These methods benefit several sectors of the economy, in particular the airline industry where any small percentage gain can translate into substantial savings. It is expected that column generation based methods can also be applied, with a similar impact, to the solution of routing and scheduling problems related to railway operations. In the fourth chapter, CAMPBELL, CLARKE, KLEYWEGT and SAVELSBERGH discuss the *Inventory Routing Problem*, a variant of the VRP that combines routing and customer

resupplying decisions. Growing competition in the area of logistics have forced several firms to pay closer attention to customer service and timeliness of deliveries. Solution methodologies for this important problem are not as developed as those that already exist for the standard VRP. Research in this area is only just emerging and should pose interesting challenges in the near future.

Dynamic real-time vehicle routing and dispatching is another fascinating area. It is central to several fast growing sectors such as emergency services, repairman dispatching, express courier delivery, dial-a-ride transportation, as well as truckload and container operations. In their article, GENDREAU and POTVIN report that interesting results have been derived through the application of artificial intelligence methods, some of which are based on neural networks. These fleet management issues are part of a much larger research domain, that of dynamic resource allocation and scheduling problems, which often account for the stochastic nature of events, operations, and their environment. Through the development of a "language" aimed at smoothing communications between industry, modelers, algorithm designers, and software specialists, POWELL initiates the development of a new paradigm to represent dynamic resources allocation problems, and to construct efficient implementation strategies. Most models used in the field of distribution management are assumed to be deterministic and known *a priori*. There exist, however, situations where some of the data is stochastic, which calls for different solution concepts. LAPORTE and LOUVEAUX show how a new optimization technique, the integer L-shaped method, has helped solve to optimality several classes of stochastic VRPs, an achievement that was deemed unrealistic only a few years ago.

In the following chapter, DESAULNIERS, DESROSIERS, GAMACHE and SOUMIS address the crew scheduling problem faced by all airlines. They present a unified formulation, as well as column based solution methods that fully exploit the network structure of the problem and rely on sophisticated branching strategies and cutting planes approaches. The next chapter, coauthored by LABBÉ, LAPORTE and RODRÍGUEZ-MARTÍN, studies combined location-routing problems in which a complex structure, such as a tree or a cycle, must be located in a graph, and some routing decisions must be made simultaneously. Such network design problems arise naturally in metro or motorway planning, for example. They are strategic in nature and have a long-lasting impact on subsequent operational decisions. In the last chapter, CRAINIC and TOULOUSE review the area of parallel metaheuristics. This field combines two key ingredients for the optimization of large-scale combinatorial problems of the type encountered in transportation planning. The authors provide new insight by studying commonalities among parallel implementations across several types of metaheuristics. Parallel metaheuristics should play an increasing role in our ability to develop workable and reliable support systems for real-time decision making in transportation.

We hope this book will benefit researchers and transportation scientists. Our warmest thanks and appreciation go to all authors and to the referees who provided valuable comments.

Montreal
February 1998

1 EXACT SOLUTION OF THE VEHICLE ROUTING PROBLEM

Paolo Toth
Daniele Vigo

1.1 INTRODUCTION

The Vehicle Routing Problem (VRP) is a hard and well-known combinatorial optimization problem which calls for the determination of the optimal routes used by a fleet of vehicles, based at one or more depots, to serve a set of customers. In practical applications of the VRP arising in the design and management of distribution systems, several operational constraints are imposed on the route construction. For example, the service may involve both deliveries and collections, the load along each route must not exceed the given capacity of the vehicles, the total length of each route must not be greater than a prescribed limit, the service of the customers must occur within given time windows, the fleet may contain heterogeneous vehicles, precedence relations may exist between the customers, the customer demands may not be completely known in advance, the service of a customer may be split among different vehicles, and some problem characteristics, as the demands or the travel times, may vary dynamically.

In this work we consider the static and deterministic basic version of the problem, known as the Capacitated VRP (CVRP). In the CVRP all the customers correspond to deliveries, the demands are known in advance and may not be split, the vehicles are identical and are based at a single central depot, only the capacity restrictions for the vehicles are imposed, and the objective is to minimize the total cost (i.e., the number of routes and/or their length or travel time). Generally, the travel cost between each pair of customer locations is the same in both directions, i.e., the resulting cost matrix is *symmetric*, whereas in some applications, as the distribution in urban areas with one-way directions imposed on the roads, the cost matrix is *asymmetric*.

The CVRP has been extensively studied since the early sixties and in the last years many new heuristic and exact approaches were presented. This attention to CVRP is motivated both by its practical relevance and by its considerable difficulty. In fact, the largest problems which can be consistently solved by the most effective exact algorithms proposed so far contain about 50 customers, whereas larger instances may be solved only in particular cases. So instances with hundreds of customers, as those arising in practical applications, may only be tackled with heuristic methods.

The CVRP is an extension of the well-known *Traveling Salesman Problem* (TSP), calling for the determination of the circuit with minimum cost visiting exactly once a given set of points. Therefore, many exact approaches for the CVRP were inherited from the huge and successful work done for the exact solution of the TSP.

Laporte and Nobert (1987) presented an extensive survey which was entirely devoted to exact methods for the VRP and gave a complete and detailed analysis of the state of the art up to the late eighties. The aim of the present work is to provide an update of that survey, describing the algorithms recently proposed for the exact solution of CVRP both for the case with symmetric and asymmetric cost matrices. Up to the end of the last decade the most effective exact approaches for the CVRP were mainly branch-and-bound algorithms using basic relaxations, as the assignment problem and the shortest spanning tree. Recently, more sophisticated bounds were proposed, as those based on Lagrangian relaxations or on the additive approach, which increased the size of the problems that can be solved to optimality by branch-and-bound. Moreover, following the success obtained by branch-and-cut methods for the TSP, encouraging results were obtained by using these algorithms for the CVRP.

In this work we treat separately problems with symmetric and asymmetric cost matrices. In fact, although the symmetric problems are special cases of the asymmetric ones, the latter were much less studied in the literature and the exact methods developed for them in general have a poor performance when applied to symmetric instances. Analogously, not all the approaches proposed for symmetric problems may be directly adapted to solve also asymmetric ones.

Other surveys covering exact algorithms, but often mainly devoted to heuristic methods, were proposed by Christofides, Mingozzi and Toth (1979), Magnanti (1981), Bodin *et al.* (1983), Christofides (1985a), Laporte (1992), and Fisher (1995). Annotated bibliographies were proposed by Christofides (1985b) and, recently, by Laporte (1997), whereas an extensive bibliography was presented by Laporte and Osman (1995). A book on the subject was edited by Golden and Assad (1988).

The work is organized as follows. In the next two sections we give a detailed description of CVRP as a graph theoretic problem, introduce the notation and describe some basic reduction rules. In Section 1.4 we consider the more general case where the cost matrix is asymmetric, illustrating the branch-and-bound algorithms proposed by Laporte, Mercure and Nobert (1986), and by Fischetti, Toth and Vigo (1994). We next present the exact methods proposed for the more widely studied CVRP with symmetric cost matrix. In Section 1.5 we examine the algorithms based on branch-and-bound. In particular, we discuss the basic relaxations based on K-tree and b-matching and their strengthening in a Lagrangian fashion proposed in Fisher (1994a) and in Miller (1995). In Section 1.6 we analyze the main ingredients of the branch-and-cut algorithm proposed by Cornuéjols and Harche (1993) and discuss some recent results in this field. In Section 1.7 we briefly examine the exact approaches

proposed by Agarwal, Mathur and Salkin (1989) and by Mingozzi, Christofides and Hadjiconstantinou (1995) based on set-partitioning formulations. Finally, we draw some conclusions and propose future directions of research.

The information about the performance, expressed in Mflops, of the computers used for testing the algorithms presented are taken (when available) from Dongarra (1996).

1.2 PROBLEM DEFINITION AND NOTATION

The CVRP may be defined as the following graph theoretic problem. Let $G = (V, A)$ be a complete graph where $V = \{0, \ldots, n\}$ is the vertex set and A is the arc set. Vertices $j = 1, \ldots, n$ correspond to the customers, each with a known nonnegative *demand*, d_j, to be delivered, whereas vertex 0 corresponds to the depot (with a fictitious demand $d_0 = 0$). Given a customer set $S \subseteq V$, let $d(S) = \sum_{j \in S} d_j$ denote the total demand of the set.

A nonnegative *cost*, c_{ij}, is associated with each arc $(i, j) \in A$ and represents the *travel cost* spent to go from vertex i to vertex j. Generally, the use of the loop arcs, (i, i), is not allowed and this is imposed by defining $c_{ii} = +\infty$ for all $i \in V$. If the cost matrix is asymmetric, A is a set of directed arcs and the corresponding problem is called *asymmetric* CVRP (henceforth indicated as ACVRP). Otherwise, i.e., when $c_{ij} = c_{ji}$ for all $i, j \in V$, the problem is called *symmetric* CVRP (for short indicated as CVRP) and the arc set A is often replaced by a set of undirected edges, E. In the following we denote the undirected edge set of graph G by A when edges are indicated by means of their endpoints $(i, j), i, j \in V$, and by E when edges are indicated through a single index e. Given a vertex set $S \subset V$, let $\delta(S)$ and $\sigma(S)$ denote the set of edges $e \in E$ (or arcs $(i, j) \in A$) which have only one or both endpoints in S, respectively. As usual, when single vertices $i \in V$ are considered, we write $\delta(i)$ rather than $\delta(\{i\})$.

In several practical cases the cost matrix satisfies the *triangle inequality*, $c_{ik} + c_{kj} \geq c_{ij}$ for all $i, j, k \in V$, i.e., it is not convenient to deviate from the direct link between two vertices i and j. The respect of the triangle inequality is sometimes required by the algorithms for CVRP and this may be obtained in an immediate way by adding a suitably large positive quantity M to the cost of each arc. However, the drastic distortion of the metric induced by this operation may produce very bad solutions with respect to the original costs. If G is strongly connected but not complete, it is always possible to obtain an equivalent complete graph where the cost of each arc (i, j) is defined as the cost of the shortest path from i to j, computed on the original graph. Note that in this case the complete graph satisfies the triangle inequality, therefore this may be seen also as a method for "triangularizing" complete graphs. Moreover, in some instances the vertices are associated with points of the plane with given coordinates and the cost c_{ij}, for all the arcs $(i, j) \in A$, is defined as the Euclidean distance between the two points corresponding to vertices i and j. In this case the cost matrix is symmetric and satisfies the triangle inequality, and the resulting problem is often called *Euclidean* CVRP. Observe that the frequently performed rounding to the nearest integer of the real-valued Euclidean arc costs may cause a violation of the triangular inequality, whereas this does not happen if the costs are rounded up.

A set of K identical vehicles, each with capacity C, is available at the depot. Each vehicle may perform at most one route, and we assume that K is not smaller than $K_{\min}$, where $K_{\min}$ is the minimum number of vehicles needed to serve all the

customers. The value of $K_{\min}$ may be determined by solving the *Bin Packing Problem* (BPP) associated with the CVRP, calling for the determination of the minimum number of bins, each with capacity C, required to load all the n items, each with nonnegative weight d_j, $j = 1, \ldots, n$. In spite of the fact that BPP is NP-hard in the strong sense, instances with hundreds of items can be optimally solved very effectively (see, e.g., Martello and Toth, 1990). In the following, given a set $S \subseteq V \setminus \{0\}$, we denote by $\gamma(S)$ the minimum number of vehicles needed to serve all the customers in S, i.e., the optimal solution value of the BPP with item set S. Note that $\gamma(V \setminus \{0\}) = K_{\min}$. Often $\gamma(S)$ is replaced by the trivial BPP lower bound $\lceil d(S)/C \rceil$. Moreover, to ensure feasibility we assume that $d_j \leq C$ for each $j = 1, \ldots, n$.

The CVRP consists of finding a collection of at most K simple *circuits* (corresponding to vehicle routes) with minimum cost, defined as the sum of the costs of the arcs belonging to the circuits, and such that:

i) each circuit visits vertex 0, i.e., the depot vertex;

ii) each vertex $j \in V \setminus \{0\}$ is visited by exactly one circuit;

iii) the sum of the demand of the vertices visited by a circuit does not exceed the vehicle capacity, C.

Often the additional objective requiring the minimization of the number of used circuits (i.e., vehicles) is added to that requiring the minimization of the total cost of the circuits. Normally the algorithms proposed in the literature do not consider this objective explicitly, however, depending on the characteristics of the algorithm used, there are different ways to take it into account. When the algorithm allows for the determination of solutions using a number of circuits smaller than K, this objective may be easily included by adding a large constant value (representing the fixed cost associated with the use of a vehicle) to the cost of the arcs leaving the depot. Thus, the optimal solution first minimizes the number of arcs leaving the depot (hence the number of circuits), then the cost of the other used arcs. If, as normally happens, the algorithm determines only solutions using all the K available vehicles, there are two possibilities. The first one is to compute $K_{\min}$ by solving the BPP associated with CVRP, and then to apply the algorithm with $K = K_{\min}$. The second possibility is to define an extended instance with graph $\overline{G} = (\overline{V}, \overline{A})$ obtained from G by adding $K - K_{\min}$ dummy vertices to V, each with demand $d_j = 0$. Let $W = \{n + 1, \ldots, n + K - K_{\min}\}$ be the set of these dummy vertices, the cost $\bar{c}_{ij}$ of the arcs $(i, j) \in \overline{A}$ is defined as:

$$\bar{c}_{ij} := \begin{cases} c_{ij} & \text{for } i, j \in V; \\ 0 & \text{for } i = 0,\ j \in W; \\ 0 & \text{for } i \in W,\ j = 0; \\ c_{0j} & \text{for } i \in W,\ j \in V \setminus \{0\}; \\ M & \text{for } i \in V \setminus \{0\},\ j \in W; \\ M & \text{for } i \in W,\ j \in W; \end{cases} \tag{1.1}$$

where M is a very large positive number. The optimal solution of the CVRP computed on the extended instance may contain "empty" routes made up by single dummy

vertices. Note that by adding a large constant to $\bar{c}_{0j}$, $j \in W$, the number of empty routes is maximized.

The CVRP is known to be NP-hard (in the strong sense), and generalizes the well-known Traveling Salesman Problem, in which $C \geq d(V)$ and $K = K_{\min} = 1$. Therefore, all the relaxations proposed for the TSP are valid for the CVRP. As already mentioned, the CVRP is also related to the Bin Packing Problem.

1.3 BASIC REDUCTION RULES

Several rules may be used to possibly remove arcs from A and consequently to speed-up the solution of CVRP. Many of them are inspired from the work done on the TSP. As previously mentioned, an alternative way to remove arcs from A (which does not sparsify the graph G) is obtained by setting the cost of these arcs to a very large positive value, say M. In the following we refer, for short, to the more general case of the asymmetric CVRP and we explicitly remove arcs from A.

The reduction rules may be used either for the original problem or for a subproblem where arcs of a given subset I are imposed in the solution, as happens in branch-and-bound and branch-and-cut algorithms. In this case the arcs of I define complete routes and also paths, some of which may enter or leave the depot. For reduction purposes, all the customers belonging to the $\rho \geq 0$ complete routes induced by I are removed from V. Let $\tilde{G} = (\tilde{V}, \tilde{A})$ be the subgraph of G induced by vertex set $\tilde{V}$ obtained from V by removing all the customers belonging to complete routes in I and let $\tilde{K} = K - \rho$. Moreover, let $\mathcal{P} = \{P_1, \ldots, P_r\}$ be the set of paths induced by I, each defined as an ordered set of vertices, and let h_i and t_i denote the first and the last vertex of path i, $i = 1, \ldots, r$. To simplify the notation each vertex i not covered by any arc in I is represented by a degenerate path $P_j \in \mathcal{P}$ made up by a singleton vertex, where $h_j = t_j = i$. Note that when $I = \emptyset$ then $r = |\mathcal{P}| = n$ and each path is degenerate.

The first type of reduction rules try to remove from $\tilde{A}$ all the arcs that, if used, would produce infeasible CVRP solutions. For example:

1. For each arc $(i, j) \in I$ we may clearly remove from $\tilde{A}$ all the arcs (i, p), $p \in \tilde{V}$ if $i \neq 0$, and (p, j), $p \in \tilde{V}$, if $j \neq 0$.

2. For each non-degenerate path P_i such that $h_i, t_i \neq 0$ we may remove all the arcs which would form a subtour disconnected from the depot. If $h_i = 0$ (resp., $t_i = 0$) we may remove arc $(t_i, 0)$, (resp., $(0, h_i)$) when

$$d(P_i) < C_{\min} = d(\tilde{V}) - (\tilde{K} - 1)C$$

   i.e., when on the remaining $\tilde{K} - 1$ vehicles there is not enough space to load the demand of the other customers.

3. For each pair of paths $P_i, P_j \in \mathcal{P}$ such that $d(P_i) + d(P_j) > C$, we may remove arc (t_i, h_j) from $\tilde{A}$, if $t_i, h_j \neq 0$.

The second type of reduction rules try to remove for $\tilde{A}$ the arcs that, if used, would not improve the currently best known solution. For example, let L and U be a lower and an upper bound on the optimal CVRP solution value, respectively. For each $(i, j) \in \tilde{A}$ let $\bar{c}_{ij}$ be the reduced cost of arc (i, j) associated with the lower

bound L. It is well-known that the reduced cost of an arc represents a lower bound on the increase of the optimal solution value if this arc is imposed. Therefore, for each $(i,j) \in \tilde{A}$ if $L + \bar{c}_{ij} \geq U$ we may remove (i,j) from $\tilde{A}$.

Whenever a customer has in $\tilde{A}$ only one entering or leaving arc we may impose that arc, add it to I, redefine the set of complete routes and paths in I and execute again steps 1. to 3. above.

1.4 THE ASYMMETRIC CVRP

In this section we examine the CVRP with asymmetric cost matrix (ACVRP). The first integer linear programming model we describe is a two-index vehicle flow formulation which uses $O(n^2)$ binary variables x, to indicate if a vehicle traverses or not an arc in the optimal solution. In other words, variable x_{ij} takes value 1 if arc $(i,j) \in A$ belongs to the optimal solution, and value 0 otherwise.

$$\text{(VRP1)} \qquad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1.2}$$

subject to

$$\sum_{i \in V} x_{ij} = 1 \qquad \text{for all } j \in V \setminus \{0\} \tag{1.3}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \text{for all } i \in V \setminus \{0\} \tag{1.4}$$

$$\sum_{i \in V} x_{i0} = K \tag{1.5}$$

$$\sum_{j \in V} x_{0j} = K \tag{1.6}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \gamma(S) \qquad \text{for all } S \subset V \setminus \{0\}, |S| \geq 2 \tag{1.7}$$

$$x_{ij} \in \{0,1\} \qquad \text{for all } i,j \in V. \tag{1.8}$$

The *indegree* and *outdegree* constraints (1.3) and (1.4) impose that exactly one arc enters and leaves each vertex associated with a customer, respectively. Analogously, constraints (1.5) and (1.6) impose the degree requirements for the depot vertex. The so-called *capacity-cut* constraints (1.7) impose both the connectivity of the solution and the vehicle capacity requirements. In fact, they stipulate that each cut $(V \setminus S, S)$ defined by a vertex set S is crossed by a number of arcs not smaller than $\gamma(S)$, i.e., the minimum number of vehicles needed to serve set S. The capacity-cut constraints remain valid also if $\gamma(S)$ is replaced by the trivial BPP lower bound (see, e.g., Cornuéjols and Harche, 1993).

Note that, because of the degree constraints (1.3)–(1.6), we have:

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} = \sum_{i \in S} \sum_{j \notin S} x_{ij} \qquad \text{for all } S \subset V \setminus \{0\}, S \neq \emptyset \tag{1.9}$$

in other words, each cut $(V \setminus S, S)$ is crossed in both directions the same number of times. From (1.9) we may also re-state (1.7) as:

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \gamma(V \setminus S) \qquad \text{for all } S \subset V, 0 \in S. \tag{1.10}$$

An alternative formulation may be obtained by transforming the capacity-cut constraints (1.7), by means of the degree constraints (1.3)–(1.6), into the well-known *generalized subtour elimination* constraints (GSEC):

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - \gamma(S) \qquad \text{for all } S \subset V \setminus \{0\}, S \neq \emptyset, \tag{1.11}$$

which impose that at least $\gamma(S)$ arcs leave each vertex set S.

Both families of constraints (1.7) and (1.11) have exponential cardinality. A possible way to partially overcome this drawback is to consider only a limited subset of them. This can be done by relaxing them in a Lagrangian fashion as done in Fisher (1994a) and in Miller (1995) (see Section 1.5.3) or by explicitly including them in the linear programming relaxation as done in branch-and-cut approaches (see Section 1.6).

Alternatively, an equivalent family of constraints with polynomial cardinality may be obtained by considering the subtour elimination constraints proposed for the TSP by Miller, Tucker and Zemlin (1960), and extending them to CVRP (see, e.g., Christofides, Mingozzi and Toth, 1979; and Kulkarni and Bhave, 1985):

$$u_i - u_j + C x_{ij} \leq C - d_j \qquad \text{for all } i, j \in V \setminus \{0\}, i \neq j, \text{ s.t. } d_i + d_j \leq C, \tag{1.12}$$

$$d_i \leq u_i \leq C \qquad \text{for all } i \in V \setminus \{0\}, \tag{1.13}$$

where u_i, $i \in V \setminus \{0\}$, is an additional continuous variable representing the load of the vehicle after visiting customer i. It is easy to see that constraints (1.12)–(1.13) impose the capacity requirements of CVRP. In fact, when $x_{ij} = 0$ the constraint is not binding since $u_i \leq C$ and $u_j \geq d_j$, whereas when $x_{ij} = 1$ they impose that $u_j \geq u_i + d_j$. These constraints may be strengthened by lifting some coefficients as illustrated by Desrochers and Laporte (1991).

Two exact algorithms, both based on the branch-and-bound approach, were proposed for ACVRP so far. The algorithm described by Laporte, Mercure and Nobert (1986) uses a lower bound based on the *Assignment Problem* (AP) relaxation of ACVRP. The algorithm proposed by Fischetti, Toth and Vigo (1994) combines, according to the so-called *additive approach*, the AP lower bound with a lower bound based on disjunction and one based on a min-cost flow relaxation. These bounds are briefly described in this section.

Other bounds for the ACVRP may be derived by generalizing the methods proposed for the symmetric case. For example, Fisher (1994a) proposed a way to extend to ACVRP the bounds based on K-tree he derived for the CVRP (described in Sections 1.5.1 and 1.5.3). In this extension the Lagrangian problem calls for the determination of an undirected K-tree on the undirected graph obtained by replacing each pair of directed arcs (i, j) and (j, i) with a single edge (i, j) with cost $c'_{ij} = \min\{c_{ij}, c_{ji}\}$. No computational testing for this bound was presented in Fisher

(1994a). Possibly better bounds may be obtained by explicitly considering the asymmetry of the problem, i.e., by using K-arborescences rather than K-trees and by strengthening the bound in a Lagrangian fashion as proposed by Fisher for the CVRP (see Toth and Vigo (1995) for an application to the Capacitated Shortest Spanning Arborescence problem).

1.4.1 The assignment lower bound

In analogy with what is done for the Asymmetric TSP (see, e.g., Carpaneto and Toth (1980), Laporte, Mercure, and Nobert (1986) proposed to relax model VRP1 by dropping the capacity-cut constraints (1.7). The resulting relaxation is a *Transportation Problem* (TP), calling for a min-cost collection of circuits of G visiting once all the vertices in $V \setminus \{0\}$, and K times vertex 0. This solution can be infeasible for CVRP since:

i) the total customer demand on a circuit can exceed the vehicle capacity;

ii) there may exist circuits not visiting vertex 0.

The solution of TP requires $O(n^3)$ time through a transportation algorithm. In practice, it is more effective to transform the problem into an *Assignment Problem* (AP) defined on the extended complete digraph $G' = (V', A')$, where $V' := V \cup \{n+1, \ldots, n+K-1\}$ contains $K-1$ additional copies of vertex 0, and the cost c'_{ij} of each arc in A' is defined as follows:

$$c'_{ij} := \begin{cases} c_{ij} & \text{for } i,j \in V \setminus \{0\}; \\ c_{i0} & \text{for } i \in V \setminus \{0\},\ j \in W; \\ c_{0j} & \text{for } i \in W,\ j \in V \setminus \{0\}; \\ \lambda & \text{for } i,j \in W; \end{cases} \tag{1.14}$$

where $\lambda = M >> 1$ and $W := \{0\} \cup \{n+1, \ldots, n+K-1\}$ is the set of the K vertices of G' associated with the depot. After this transformation, constraint (1.5) may be replaced by K constraints of type (1.3), one for each copy of the depot. Analogously, constraint (1.6) may be replaced by K constraints of type (1.4). This extension was originally proposed by Lenstra and Rinnooy Kan (1975), to transform into an ordinary TSP the m-TSP, which calls for the determination of a collection of m circuits visiting m times a distinguished vertex (i.e., the depot) and once all the remaining vertices. Observe that by defining λ in a different way we obtain an alternative transformation, with respect to that presented in Section 1.2, to obtain solutions using less than K vehicles. In particular, defining $\lambda = 0$ leads to the determination of the min-cost set of *at most* K routes, whereas defining $\lambda = -M$ leads to the determination of the min-cost set of $K_{\min}$ routes.

1.4.2 The disjunctive lower bound

The following two bounds were proposed by Fischetti, Toth and Vigo (1994). The first bound is based on a disjunction on infeasible arc subsets, whereas the second bound is based on a min-cost flow relaxation.

A given arc subset $B \subset A$ is called *infeasible* if no feasible solution to CVRP can use all its arcs, i.e., when

$$\sum_{(a,b)\in B} x_{ab} \leq |B| - 1 \tag{1.15}$$

is a valid inequality for ACVRP. For any given (minimal) infeasible arc subset $B \subset A$, the following logical disjunction holds for each $x \in F$, where F is the set of all the feasible solutions:

$$\bigvee_{(a,b)\in B} \left(x \in Q^{ab} := \left\{x \in \Re^A : x_{ab} = 0\right\}\right). \tag{1.16}$$

Then $|B|$ restricted problems are defined, each denoted as RP^{ab} and including the additional condition $x_{ab} = 0$ imposed for a different $(a,b) \in B$. For each RP^{ab}, a valid lower bound, ϑ^{ab}, is computed through the AP relaxation of the previous section (with $c_{ab} := +\infty$ to impose $x_{ab} = 0$). The disjunctive bound

$$L_D := \min\left\{\vartheta^{ab} : (a,b) \in B\right\}, \tag{1.17}$$

clearly dominates the AP lower bound, L_{AP}, since $\vartheta^{ab} \geq L_{AP}$ for all $(a,b) \in B$.

A possible way to determine infeasible arc subsets B is the following. First solve the AP relaxation with no additional constraints, and store the corresponding optimal solution $(x^*_{ij} : i,j \in V)$. If x^* is feasible for ACVRP, then clearly the lower bound L_{AP} cannot be improved. Otherwise, try to improve it by using a disjunction on a suitable infeasible arc subset B. Note that imposing $x_{ab} = 0$ for any $(a,b) \in A$ such that $x^*_{ab} = 0$ would produce $\vartheta^{ab} = L_{AP}$, hence a disjunctive bound $L_D = L_{AP}$. Therefore, B is chosen as a subset of $A^* := \left\{(i,j) \in A : x^*_{ij} = 1\right\}$, if any, corresponding to one of the following cases:

i) a circuit which is disconnected from the depot vertex,

ii) a sequence of customer vertices whose total demand exceeds C,

iii) a feasible circuit which leaves uncovered a set of customers, S, whose total demand cannot be served by the remaining $K-1$ vehicles, i.e., such that $\gamma(S) > K - 1$.

Different choices of the infeasible arc subset B lead to different lower bounds. Therefore, Fischetti, Toth and Vigo (1994) used an overall bounding procedure, called ADD_DISJ, based on the *additive approach* which considers, in sequence, different infeasible arc subsets so as to produce a possibly better overall lower bound.

The additive approach was proposed by Fischetti and Toth (1989) and allows for the combination of different lower bounding procedures, each exploiting different substructures of the considered problem. When applied to a minimization problem, each procedure returns a lower bound ρ and a *residual cost matrix*, $\tilde{c}$, such that:

$$\begin{aligned} \tilde{c} &\geq 0 \\ \rho + \tilde{c}x &\leq cx \quad \text{for all } x \in F. \end{aligned}$$

The entries of $\tilde{c}$ represent lower bounds on the increment of the optimal solution value if the corresponding arc is imposed in the solution. The different bounding

procedures are applied in sequence, and each of them uses as costs the residual cost matrix given by the previous procedure (obviously. the first procedure starts with the original cost matrix). The overall additive lower bound is the sum of the lower bounds obtained by each procedure. It can be easily shown that if the lower bounding procedures are based on linear programming relaxations, as those previously described for ACVRP, the reduced costs are valid residual costs. For further details see Fischetti, Toth and Vigo (1994), and Fischetti and Toth (1992).

Procedure ADD_DISJ starts by solving the AP relaxation with no additional constraints, and defines the initial lower bound LB as the optimal AP solution value, $v(AP)$, and the arc set A^* as the arcs used in the optimal AP solution. Then iteratively an infeasible subset B, if any, is chosen from A^* and used for the computation of the disjunctive lower bound, returning a lower bound L_D and the corresponding residual cost matrix. The current LB is increased by L_D, the set A^* is updated by removing from it all the arcs whose corresponding variables are not equal to 1 in the current optimal solution of the disjunctive bound. The process is iterated until A^* does not contain further infeasible arc subsets. Procedure ADD_DISJ can be implemented, through parametric techniques, so as to have an overall time complexity equal to $O(n^4)$.

1.4.3 The min-cost flow based lower bound

Let $\{S_1, \ldots, S_m\}$ be a given partition of V with $0 \in S_1$, and define

$$\begin{aligned} A_1 &:= \bigcup_{h=1}^{m} \{(i,j) \in A : i,j \in S_h\} \\ A_2 &:= A \setminus A_1. \end{aligned}$$

In other words, A is partitioned into $\{A_1, A_2\}$, where A_1 contains the arcs "internal" to the subsets S_h, and A_2 those connecting vertices belonging to different S_h's.

In the following, a lower bound L_P based on projection is described. The bound is given by $L_P := \vartheta_1 + \vartheta_2$, where $\vartheta_t, t = 1, 2$, is a lower bound on $\sum(c_{ij} : (i,j) \in A^* \cap A_t)$ for every (optimal) ACVRP solution $A^* \subset A$.

The contribution to L_P of the arcs internal to the given subsets S_h is initially neglected, i.e., ϑ_1 is set equal to 0. The rationale of this choice is clarified later. As to ϑ_2, this is computed by solving the following linear programming relaxation, called R1, obtained from model VRP1 by:

i) weakening degree equations (1.3)–(1.6) into inequalities, to take into account the removal of the arcs in A_1;

ii) imposing the capacity-cut constraints (1.7) and (1.10) only for the m subsets $S_h s$.

The model of R1 is:

$$(R1) \qquad \vartheta_2 = \min \sum_{(i,j) \in A_2} c_{ij} x_{ij} \qquad (1.18)$$

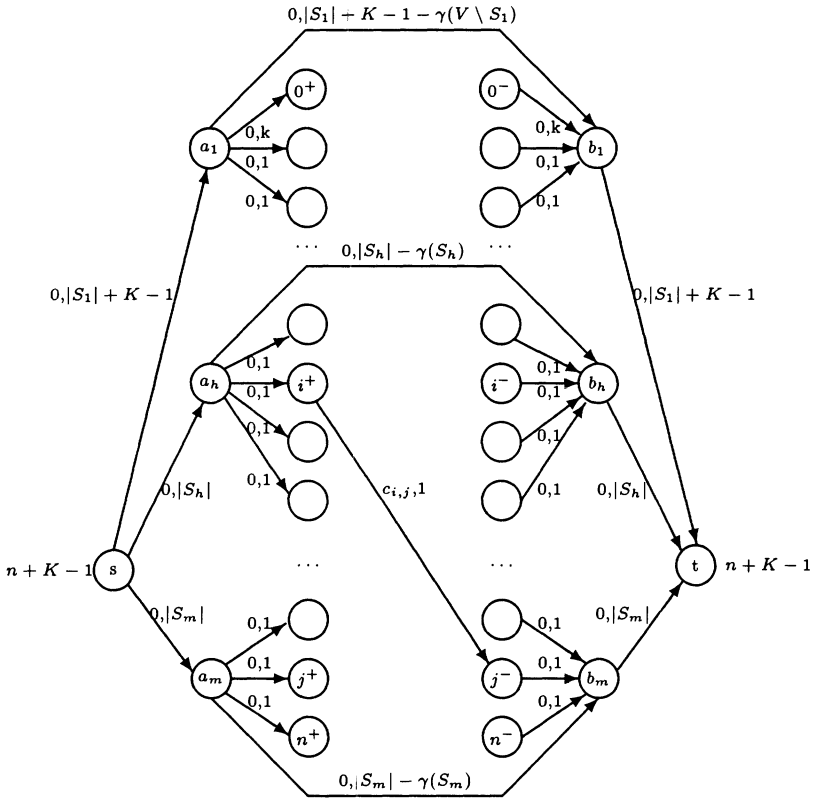


Figure 1.1 The auxiliary layered network for relaxation R1

subject to

$$\sum_{i \in V:(i,j) \in A_2} x_{ij} \leq \begin{cases} 1 & \text{for all } j \in V \setminus \{0\}; \\ K & \text{for } j = 0; \end{cases} \tag{1.19}$$

$$\sum_{j \in V:(i,j) \in A_2} x_{ij} \leq \begin{cases} 1 & \text{for all } i \in V \setminus \{0\}; \\ K & \text{for } i = 0; \end{cases} \tag{1.20}$$

$$\sum_{i \notin S_h} \sum_{j \in S_h} x_{ij} = \sum_{i \in S_h} \sum_{j \notin S_h} x_{ij} \geq \begin{cases} \gamma(V \setminus S_h) & \text{for } h = 1; \\ \gamma(S_h) & \text{for } h = 2, \ldots, m; \end{cases} \tag{1.21}$$

$$x_{ij} \in \{0,1\} \quad \text{for all } (i,j) \in A_2. \tag{1.22}$$

This model can be solved efficiently, since it can be viewed as an instance of a min-cost flow problem on an auxiliary layered network, as illustrated in Figure 1.1. The network contains $2(n + m + 1)$ vertices, namely:

- two vertices, say i^+ and i^-, for all $i \in V$;
- two vertices, say a_h and b_h, for all $h = 1, \ldots, m$;
- a source vertex, s, and a sink vertex, t.

The arcs in the network, and the associated capacities and costs, are:

- for all $(i, j) \in A_2$: arc (i^+, j^-) with cost c_{ij} and capacity $+\infty$;
- for all $h = 1, \ldots, m$: arcs (a_h, i^+) and (i^-, b_h) for all $i \in S_h$, with cost 0 and capacity 1 (if $i \neq 0$) or K (if $i = 0$);
- for all $h = 1, \ldots, m$: arc (a_h, b_h) with cost 0 and capacity $|S_h| - \gamma(S_h)$ (if $h \neq 1$) or $|S_1| + K - 1 - \gamma(V \setminus S_1)$ (if $h = 1$);
- for all $h = 1, \ldots, m$: arcs (s, a_h) and (b_h, t), both with cost 0 and capacity $|S_h|$ (if $h \neq 1$) or $|S_1| + K - 1$ (if $h = 1$).

It can be easily seen that finding the min-cost s-t flow of value $n+k-1$ on this network actually solves relaxation R1. The worst-case time complexity for the computation of ϑ_2, and of the corresponding residual costs, is $O(n^3)$ by using a specialized algorithm based on successive shortest path computations.

Different choices of the vertex partition $\{S_1, \ldots, S_m\}$ lead to different lower bounds. Note that choosing $S_h = \{h\}$ for all $h \in V$, produces a relaxation R1 that coincides with the AP relaxation of Section 1.4.1. When, on the other hand, non-singleton S_h's are present, relaxation R1 is capable of taking into account the associated capacity-cut constraints (that are, instead, neglected by AP), while loosing a possible contribution to the lower bound of the arcs inside S_h (which belong to A_1), and weakening the degree constraints of the vertices in S_h. Fischetti, Toth and Vigo (1994) used, in sequence, different partitions obtaining an overall additive procedure, called ADD_FLOW.

The procedure is initialized with the partition $S_h = \{h\}$ for all $h = 1, \ldots, m = n$ (i.e., with the AP relaxation). At each iteration of the additive scheme, relaxation R1 is solved, the current lower bound is increased, and the current costs are reduced accordingly. Then a convenient collection of subsets $S_{h_1}, \ldots, S_{h_r}$ (with $r \geq 2$) belonging to the current partition is selected and the subsets are replaced with their union, say S^*. The choice of this collection is made so as to produce an *infeasible* set S^*, i.e., a vertex set whose associated capacity-cut constraint is violated by the solution of the current relaxation R1. This hopefully produces an increase of the additive lower bound in the next iteration. The additive scheme ends when either $m = 1$, or no infeasible S^* is detected.

Procedure ADD_FLOW takes $O(n^4)$ time and the resulting additive lower bound clearly dominates the AP bound, which is used to initialize it. On the other hand no dominance relation exists between ADD_FLOW and procedure ADD_DISJ of the previous section. Therefore, Fischetti, Toth and Vigo proposed to apply ADD_DISJ and ADD_FLOW in sequence, again in an additive fashion. To reduce the average overall computing time, procedure ADD_FLOW was stopped when no increase of the current lower bound LB was observed for 5 consecutive iterations.

1.4.4 *Branch-and-bound algorithms for the asymmetric CVRP*

We now briefly describe the main ingredients of the branch-and-bound algorithms used for the exact solution of the ACVRP proposed by Laporte, Mercure and Nobert (1986) and by Fischetti, Toth and Vigo (1994). The two algorithms have the same basic structure, derived from that of the algorithm for the asymmetric TSP described in Carpaneto and Toth (1980) and originally proposed in Bellmore and Malone (1971): the first one uses as lower bound the AP relaxation of Section 1.4.1, whereas the second uses the two additive bounding procedures described in Sections 1.4.2 and 1.4.3.

The algorithms adopt a *breadth-first* search strategy, i.e., branching is always executed on the pending node of the branch-decision tree with the smallest lower bound value. This rule allows for the minimization of the number of subproblems solved at the expense of larger memory usage, and computationally proved to be more effective than the *depth-first* strategy, where the branching node is selected according to a last-in-first-out rule.

The partitioning rules used by both algorithms are related to the *subtour elimination* scheme used for the asymmetric TSP, and handle the relaxed constraints imposing the connectivity and the capacity requirements of the feasible ACVRP solutions. At a node ν of the branch-decision tree, let I_ν and F_ν contain the arcs imposed and forbidden in the current solution, respectively.

Given the set A^* of arcs belonging to the solution of the current relaxation, an arc sequence $B := \{(v_1, v_2), (v_2, v_3), \ldots, (v_h, v_{h+1})\} \subset A^*$ on which to branch is chosen.

Fischetti, Toth and Vigo defined B as the subset of A^* with the minimum number of non-imposed arcs, defining either a feasible circuit through vertex 0 (in this case, $v_1 = v_{h+1} = 0$), or a path which is infeasible according to the conditions of Section 1.4.2. Then $h = |B|$ descendant nodes are generated. The subproblem associated with node $\nu_i, i = 1, \ldots, h$ is defined by excluding the i-th arc of B and by imposing the arcs up to $i - 1$:

$$\begin{aligned} I_{\nu_i} &:= I_\nu \cup \{(v_1, v_2), \ldots, (v_{i-1}, v_i)\} \\ F_{\nu_i} &:= F_\nu \cup \{(v_i, v_{i+1})\} \end{aligned}$$

where $I_{\nu_1} := I_\nu$. In addition, when B is a feasible circuit, a further son node, ν_{h+1}, is generated with $I_{\nu_{h+1}} := I_\nu \cup B$, and $F_{\nu_{h+1}} := F_\nu$.

Laporte, Mercure and Nobert defined B as an infeasible subtour according to the conditions of Section 1.4.1, and used a more complex partitioning rule in which at each descendant node at most r arcs of B are simultaneously excluded, where $r := \lceil d(S)/C \rceil$ and S is the set of vertices spanned by B. In this case, since at most $\binom{|B|}{r}$ descendant nodes may be generated, the set B is chosen as the one minimizing $\binom{|B|}{r}$.

The performance of the branch-and-bound algorithms is enhanced by means of several additional procedures performing variable fixing, feasibility checks and dominance tests. The Fischetti, Toth and Vigo algorithm (FTV) at each node of the branch-decision tree uses a heuristic algorithm proposed by Vigo (1996) which starts from the infeasible solution associated with the current relaxation and tries to obtain a feasible solution through an insertion procedure and a post-optimization phase based on arc exchanges.

Laporte, Mercure and Nobert used their algorithm (LMN) to solve, on a VAX 11/780 computer (0.14 Mflops), test instances where demands d_j and costs c_{ij} were randomly generated from a uniform distribution in [0, 100], and rounded to the nearest integer. The vehicle capacity was defined as

$$C := (1 - \alpha) \max_{j \in V} \{d_j\} + \alpha d(V),$$

where α is a real parameter chosen in [0, 1]. The number of available vehicles was defined as $K = K_{\min}$, and computed by using the trivial BPP lower bound. Note that larger values of α produce larger C, and hence smaller K (when $\alpha = 1$, ACVRP reduces to the asymmetric TSP, since $K = 1$). No monotone correlation between α and the *average percentage load* of a vehicle, defined as $100\ d(V)/(K\ C)$, can instead be inferred. Laporte, Mercure, and Nobert considered $\alpha = 0.25$, 0.50, 0.75, and 1.0, producing $K = 4, 2, 2$, and 1, respectively.

For each pair (n,α), five instances were generated and algorithm LMN was run by imposing a limit on the total available memory. The LMN algorithm was able to solve instances with up to 90 vertices if $\alpha \geq 0.50$ (i.e., with $K \leq 2$). For the larger values of n only half or less of the instances were actually solved, while with $\alpha = 0.25$ only the instances with 10 vertices and one of those with 20 vertices were solved. The computing times for the most difficult instances solved were above 5000 seconds, whereas no statistics were reported for the non-solved instances. The algorithm was also tested on instances of the same type but with $K = K_{\min} + 2$ or $K = K_{\min} + 4$. These problems resulted much easier than the previous ones. Algorithm LMN was able to solve instances with up to 260 vertices. Finally, randomly generated Euclidean instances were considered and, as expected, algorithm LMN obtained poor results, being able to solve only some of the problems with 2 vehicles and up to 30 vertices.

Fischetti, Toth and Vigo tested their algorithm FTV on the same randomly generated instances used for LMN with $K = K_{\min}$. Algorithm FTV was able to solve all the instances with up to 300 vertices and up to 4 vehicles, within 1000 CPU seconds on a DECstation 5000/240 (5.3 Mflops). On these instances the additive lower bound considerably improved the AP value. Algorithm FTV was also tested on a class of more realistic problems where the cost matrices were obtained from those of the previous class by "triangularizing" the costs, i.e., by replacing each c_{ij} with the cost of the shortest path from i to j. The number of vehicles K and the average percentage vehicle load, say r, were fixed and the vehicle capacity was defined as $C := \lceil 100d(V)/(rK) \rceil$. Instances of this type with up to 300 vertices, 8 vehicles and with r equal to 80% and 90% were solved, those with $n \geq 150$ being easier than the smaller ones. Algorithm FTV was used to solve eight real-world instances with up to 70 vertices and 3 vehicles, coming from pharmaceutical and herbalist's product delivery in the center of an urban area with several one-way restictions imposed on the roads. These instances resulted more difficult than the randomly generated ones: the computing time and the number of nodes were higher than those required for analogous random instances. Moreover, the average gap, over the eight instances, of the additive bound with respect to the optimal solution value was about 5.5% (that of AP being 8.9%) whereas on random instances the gap was normally much smaller (1-2% for the additive bound and 2-5% for the AP). Finally, we recently applied algorithm FTV to randomly generated Euclidean instances and we obtained poor results analogous to those obtained by LMN.

1.5 BRANCH-AND-BOUND ALGORITHMS FOR THE SYMMETRIC CVRP

In this section we examine the exact algorithms based on branch-and-bound porposed for the symmetric version of CVRP proposed in by Fisher (1994a) and Miller (1995). We first give a general model for CVRP and describe the basic relaxations based on spanning trees and on b-matching. The strengthening of these basic relaxations in a Lagrangian fashion is then discussed and the overall branch-and-bound algorithms are described. The exact algorithm proposed by Hadjconstantinou, Christofides and Mingozzi (1995), although it actually is a branch-and-bound approach, will be briefly presented in Section 1.7 since it uses a lower bound based on the set partitioning formulation.

The model we consider is obtained, as proposed in Laporte, Nobert and Desrochers (1985), by adapting to CVRP the two-index vehicle flow formulation VRP1 of ACVRP. To this end it should be noted that in CVRP the routes are not oriented (i.e., the customers along a route may be visited indifferently clockwise or couter-clockwise). Therefore, it is not necessary to know in which direction edges are covered by the vehicles, and for each undirected edge $e \in E$ one integer variable x_e is used to indicate how many times the edge is covered in the optimal solution. In particular, if $e \notin \delta(0)$ then $x_e \in \{0,1\}$, whereas if $x_e \in \delta(0)$ then $x_e \in \{0,1,2\}$. The case $x_e = 2$ indicates that the endpoint customer of edge e, say j, is contained into the single-customer route $0 \rightarrow j \rightarrow 0$. The model reads:

$$\text{(VRP2)} \qquad \min \sum_{e \in E} c_e x_e \tag{1.23}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2 \qquad \text{for all } i \in V \setminus \{0\} \tag{1.24}$$

$$\sum_{e \in \delta(0)} x_e = 2K \tag{1.25}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\gamma(S) \qquad \text{for all } S \subset V \setminus \{0\}, |S| \geq 2 \tag{1.26}$$

$$x_e \in \{0,1,2\} \qquad \text{for all } e \in \delta(0) \tag{1.27}$$

$$x_e \in \{0,1\} \qquad \text{for all } e \notin \delta(0). \tag{1.28}$$

The *degree* constraints (1.24) and (1.25) impose that exactly two arcs are incident on each vertex associated with a customer, and 2K arcs are incident on the depot vertex, respectively. The capacity-cut constraints (1.26) impose both the connectivity of the solution and the vehicle capacity requirements, by forcing that a sufficient number of arcs enter each subset of vertices. Also in this case, due to (1.24), these constraints may be rewritten as the generalized subtour elimination constraints (GSECs):

$$\sum_{e \in \sigma(S)} x_e \leq |S| - \gamma(S) \qquad \text{for all } S \subset V \setminus \{0\}, |S| \geq 2 \tag{1.29}$$

where $\gamma(S)$ may be replaced by the trivial BPP lower bound.

1.5.1 *The lower bounds based on trees*

Different relaxations based on spanning trees were presented for CVRP by extending the well-known 1-tree relaxation proposed by Held and Karp (1971) for the TSP.

Christofides, Mingozzi and Toth (1981) proposed a branch-and-bound algorithm based on the *k-degree center tree* (k-DCT) relaxation of CVRP. Given k, with $K \leq k \leq 2K$, the k-DCT is a min-cost spanning tree on G with degree k at the depot vertex. Then, K least cost arcs not in the tree are added, $2K - k$ of which are incident on depot, and the remaining $k - K$ are not incident on it. The bound was tightened by using Lagrangian penalties associated with the degree constraints. The branch-and-bound algorithm was able to solve problems from the literature with up to 25 vertices within 244 seconds on a CDC 7600 (2 Mflops).

Another tree-based relaxation was presented in Fisher (1994a), and requires the determination of a *K-tree*, defined as a min-cost set of $n + K$ edges spanning the graph. The approach used by Fisher is based on formulation VRP2 with the additional assumption that single-customer routes are not allowed. However, as he observed, in many cases this assumption is not constraining. In fact, customer j can be served alone in a route if and only if on the remaining $K - 1$ vehicles there is enough space to load the demand of the other customers, i.e., if $\gamma(V \setminus \{0, j\}) \leq K - 1$. By replacing $\gamma(\cdot)$ with the trivial BPP lower bound we may re-state the above condition as

$$d_j \geq C_{\min} = d(V) - (K - 1)C. \tag{1.30}$$

If, given a CVRP (or ACVRP) instance condition (1.30) is satisfied by no $j \in V$, then in any feasible solution no customer may be served alone in a route (hence the constraint preventing it is superfluous).

Fisher modeled the CVRP as the problem of determining a K-tree with degree equal to $2K$ at the depot vertex, with additional constraints imposing: (i) the vehicle capacity requirements, and (ii) that the degree of each customer vertex must be equal to 2. These additional constraints are relaxed in a Lagrangian fashion thus obtaining as Lagrangian problem the determination of a K-tree with degree $2K$ at the depot, which can be computed in $O(n^3)$ time (see Fisher, 1994b). This degree constrained K-tree relaxation may be easily obtained by considering formulation VRP2 and:

- removing the degree constraints (1.24);
- weakening the capacity-cut constraints (1.26) into connectivity constraints by replacing the right-hand side with 1.

In Figure 1.2(a) are illustrated the arcs belonging to the K-tree relaxation of an instance of Euclidean CVRP with $K = 2$. It can be seen that this K-tree solution is infeasible for CVRP because some vertices have degree different than two. Moreover, the demand associated with the branches leaving the depot may exceed the vehicle capacity.

The basic lower bound corresponding to the degree constrained K-tree relaxation is quite poor. In fact, for the 7 Euclidean instances with $n \leq 100$ solved in Fisher (1994a) its percentage ratio with respect to the optimal solution value ranges between 55.8% and 77.6% and is on average 67.9%.

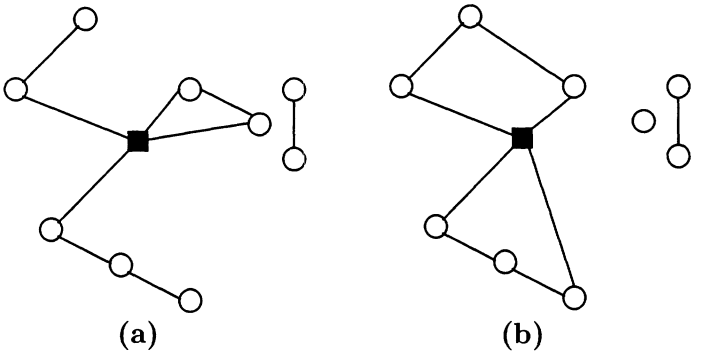


Figure 1.2 Basic relaxations of an Euclidean CVRP instance with $K = 2$; (a) the K-tree relaxation; (b) the b-matching relaxation.

1.5.2 The lower bound based on matching

The b-matching is a natural relaxation for CVRP and is the counterpart for the symmetric problem of the assignment relaxation for ACVRP described in Section 1.4.1. However, only recently this relaxation came on the scene, due to the work by Miller (1995), after the development of efficient codes for the b-matching problem (see, e.g., Miller and Pekny, 1995).

The b-matching relaxation of CVRP may be obtained by considering model VRP2 and removing the capacity-cut constraints (1.26). The resulting relaxed problem requires the determination of a subset of arcs covering all the vertices and such that the degree of each customer vertex is equal to two, while the degree of the depot is equal to $2K$. Figure 1.2(b) illustrates the arcs belonging to the b-matching relaxation corresponding to the same Euclidean instance of Figure 1.2(a). By observing the figure it can be noted that the b-matching solution may be infeasible for CVRP since: (i) some connected components (i.e., subtours) may be disconnected from the depot, and (ii) the demand associated with a subtour may exceed the vehicle capacity. As for the AP relaxation for ACVRP, it is possible to obtain an equivalent 2-matching relaxation by adding $K - 1$ copies of the depot (see, Section 1.4.1 for further details and Pekny and Miller (1994) for an effective 2-matching algorithm). In Miller (1995) some of the GSECs (1.29) are relaxed in a Lagrangian fashion obtaining as Lagrangian problem the determination of a min-cost b-matching.

The average ratio of the lower bound corresponding to the basic b-matching relaxation computed for some Euclidean instances with up to 50 vertices ranges between 71.7% and 92.9% and is, on average, equal to 85.3% of the optimal solution value (Miller, 1997).

1.5.3 The Lagrangian lower bounds

The relaxations of CVRP presented in the previous sections have in general a poor quality. Thus, both Fisher (1994a) and Miller (1995) strengthened them by dualizing,

in a Lagrangian fashion, some of the relaxed constraints. In particular, Fisher included in the objective function the degree constraints (1.24) and some of the capacity-cut constraints (1.26), whereas Miller included some of the GSECs (1.29). It should be also remembered that Fisher did not allow single-customer routes. As in related problems, good values for the Lagrangian multipliers associated with the relaxed constraints are determined by using a standard subgradient optimization procedure (see, e.g., Fisher, 1985).

The main difficulty associated with this relaxation is represented by the exponential cardinality of the set of relaxed constraints (i.e., the capacity-cuts and the GSECs) which does not allow for the explicit inclusion of all of them in the objective function. To this end, both authors proposed to include only a limited family $\mathcal{F}$ of relaxed capacity-cut or GSEC constraints and to iteratively add to the Lagrangian relaxation the constraints which are violated by the current solution of the Lagrangian problem. In particular, at each iteration of the subgradient optimization procedure the arcs incident on the depot in the current Lagrangian solution are removed. Violated constraints (i.e., capacity-cuts or GSECs, depending on the approach), if any, are detected by examining the connected components obtained in this way. This detection routine is exact. In other words, if a constraint associated with, say, vertex set S, is violated by the current Lagrangian solution, then there is a connected component of that solution spanning all the vertices in S.

The new constraints are added to the Lagrangian problem, i.e., to $\mathcal{F}$, with an associated multiplier and the process is iterated until no violated constraint is detected (hence the Lagrangian solution is feasible) or a prefixed number of subgradient iterations has been executed. Slack constraints are periodically purged from $\mathcal{F}$.

Fisher (1994a) initialized $\mathcal{F}$ with an explicit set of constraints containing the customer subsets nested around $K+3$ *seed* customers. The seeds were chosen as the K customers farthest from the depot in the routes corresponding to an initial feasible solution, whereas the last three were the customers maximally distant from the depot and the other seeds. For each seed, 60 sets were generated by including customers according to increasing distances from the seed. After 50 subgradient iterations new sets were added to $\mathcal{F}$ by identifying violated capacity-cuts in the current Lagrangian solution as previously explained. The step size used in the subgradient optimization method was initially set to 2 and reduced by a factor of 0.75 if the lower bound was not improved in the last 30 iterations. The number of iterations of the subgradient optimization procedure performed at the root node of the branch-and-bound algorithm ranged between 2000 and 3000. The overall Lagrangian bound considerably improved the basic K-tree relaxation and was, on average, 99% of the optimal solution for the Euclidean instances with $n \leq 100$ solved in Fisher (1994a).

Miller (1995) initialized $\mathcal{F}$ as the empty set and at each iteration of the subgradient procedure detected violated GSECs and additional constraints belonging to the following two classes. The first type of constraints is given by additional GSECs which were added when the current Lagrangian solution $\bar{x}$ contains two or more overloaded routes. The customer set of these new GSECs is the union of the sets $S_1, \ldots, S_k$ associated with the GSECs violated by $\bar{x}$. This increases the possibility that arcs connecting customers belonging to the overloaded routes to those in sets $S_1, \ldots, S_k$ are selected by the b-matching solution. The second type of constraints was added when $\bar{x}$ contained routes which were *underloaded*, i.e., whose associated load was

smaller than the minimum vehicle load $C_{\min}$ defined by (1.30). In this case for each such set S, with $0 \in S$, a constraint of the form

$$\sum_{e \in \sigma(S)} x_e \leq |S| - 1 \tag{1.31}$$

which breaks the current underloaded route in $\bar{x}$, was added to $\mathcal{F}$. The procedure was iterated until no improvement was obtained since 50 subgradient iterations. The step size is modified in an adaptive way every five subgradient iterations to produce a slight oscillation in lower bound values during the progress of the subgradient procedure. If the lower bound is monotonically increasing, the step size is increased by 50%; if the oscillation of the lower bound value is greater than 2%, the step size is reduced by 20%, and when the oscillation is smaller than 0.5% it is increased by 10%. The final Lagrangian bound of Miller is considerably tight, being on average 98% of the optimal solution value for the 8 problems with $n \leq 50$ solved in Miller (1995).

1.5.4 Branching schemes and overall algorithms

Many branching schemes were used for CVRP and almost all are extensions of those used for the TSP.

The first scheme we consider, proposed in Christofides, Mingozzi and Toth (1981), is known as *branching on arcs* and proceeds by extending partial paths, starting from the depot and finishing at a given vertex. At each node of the branch-decision tree an arc (i, j) is selected to extend the current partial path and two descendant nodes are generated: the first node is associated with the inclusion of the selected arc in the solution (i.e., $x_{ij} = 1$), while in the second node the arc is excluded (i.e., $x_{ij} = 0$).

Miller (1995) used this branching scheme, where the arc to branch with is selected by examining the solution obtained by the Lagrangian relaxation based on b-matching described in Sections 1.5.3. When a partial path is present in the current subproblem ending, say, with vertex v, the arc (v, h) belonging to the current Lagrangian solution is selected. If the current subproblem does not contain a partially fixed path, e.g., at the root node or when a ruote has been closed by the last imposed arc, the arc connecting the depot with the unrouted customer j with the largest demand is selected for branching. In this case a third descendant node is also created, by imposing $x_{0j} = 2$, i.e., by considering, if feasible, the route containing only customer j. The resulting branch-and-bound algorithm was applied to Euclidean CVRP instances from the literature, where the edge costs are computed as the Euclidean distances between the customers and rounded to the nearest integer. The algorithm was able to solve problems with up to 50 customers within 15,000 seconds on a Sun Sparc 2 (4 Mflops).

Fisher (1994a) used a mixed scheme where branching on arcs is used when no partial path is present in the current subproblem. In this case the currently unserved customer i with the largest demand is chosen and the arc (i, j) is used for branching, where j is the unserved customer closest to i. At the node where arc (i, j) is excluded from the solution, branching on arcs is again used, whereas at the second node the scheme known as *branching on customers* is used. One of the two ending customers, say v, of the currently imposed sequence of customers is chosen, and branching is performed by enumerating the customers which may be appendend to that end of the sequence. A subset T of currently unserved customers is selected, e.g., that including the unserved customers closest to v, and $|T| + 1$ nodes are generated. Each of the first

$|T|$ nodes corresponds to the inclusion in the solution of a different customer $j \in T$, while in the last node all the arcs $(v, j), j \in T$ are excluded.

The performance of this branching scheme may be enhanced by means of a dominance test proposed by Christofides, Mingozzi and Toth (1981). A node of the branch-decision tree where a partial sequence of customers $v, \ldots, w$ is fixed, can be fathomed if there exists a lower cost ordering of the customers in the sequence starting with v and ending with w. The improved ordering may be heuristically determined, e.g., by means of exchange procedures as those proposed in Lin and Kernighan (1973).

The mixed branching scheme with the described dominance rule was used by Fisher to attempt the solution of Euclidean CVRP instances with real distances and about 100 customers, but proved unsuccessful. In fact, Fisher observed that in instances where many small clusters of close customers exist (as is the case of several instances form the literature) any solution in which these customers are served contiguously in the same route have almost the same cost. Thus, when the sequence of these customers have to be determined through branching, unless an extremely tight bound is used, it would be very difficult to fathom many of the resulting nodes. Therefore, in Fisher (1994a) an alternative branching scheme is proposed, aiming at exploiting macroproperties of the optimal solution whose violation would have a large impact on the cost, thus allowing the fathoming of the corresponding nodes. To this end a subset T of currently unserved customers is selected and two descendant nodes are created: at the first node the additional constraint $\sum_{e \in \delta(T)} x_e = 2\lceil d(T)/C \rceil$ is added to the current problem, while at the second node the constraint $\sum_{e \in \delta(T)} x_e \geq 2\lceil d(T)/C \rceil + 2$ is imposed. Some ways of identifying suitable subsets as well as additional dominance rules are described in Fisher (1994a). This second branch-and-bound algorithm was successfully applied to some Euclidean CVRP instances with real distances and with no single customer route allowed. The largest solved instance included 100 customers and was solved within less than 60,000 seconds on a small Apollo Domain 3000 computer (0.071 Mflops).

1.6 BRANCH-AND-CUT ALGORITHMS FOR THE SYMMETRIC CVRP

Polyhedral combinatorics has since the last two decades established itself as a powerful framework for the solution of hard combinatorial problems. The impressive results obtained by branch-and-cut algorithms for the solution of the TSP and the strong structural interconnections existing between TSP and CVRP, motivated the investigation in this direction.

Laporte, Nobert and Desrochers (1985) used the linear programming (LP) relaxation of model VRP2 without (1.26) as a base for an exact algorithm for the VRP with both capacity and maximum distance restrictions (DCVRP). This algorithm may be considered a forerunner of the branch-and-cut approaches for CVRP. The initial LP relaxation contains only degree constraints, (1.24) and (1.25), and

$$x_e \geq 0 \qquad \text{for all } e \in E \tag{1.32}$$

$$x_e \leq 2 \qquad \text{for all } e \in \delta'(0) \tag{1.33}$$

$$x_e \leq 1 \qquad \text{for all } e \notin \delta'(0) \tag{1.34}$$

where $\delta'(0) = \{e \in \delta(0) : c_e \leq \frac{1}{2}\text{max. route length}\}$. Note that $\delta'(0)$ may be further reduced by removing the edges connecting the depot with customers whose demand

is smaller than the minimum vehicle load $C_{\min}$ (see (1.30)). This initial relaxation is iteratively strengthened by adding constraints of three types. First *subtour prevention constraints* are derived by considering the connected components induced by the variables fixed to one in the current subproblem. These constraints prevent the formation of illegal subtours, i.e., subtours disconnected from the depot or which violate the capacity or distance requirements. This is done by imposing to be not greater than zero the sum of all the variables which would induce an illegal subtour if taking value one. Observe that this addition of constraints is equivalent to some of the reductions described in Section 1.3, and usually applied to CVRP. The second family of constraints considered is made up by the GSECs (1.29) imposing the capacity (and distance) requirements of the vehicles (where the trivial BPP lower bound is used for $\gamma(\cdot)$ in the capacity constraints and an analogous lower bound is used in the distance constraints). Violated constraints of this type are detected by considering the connected components induced by the set of variables which assume value greater than zero in the current LP solution. Finally, at the root node of the branch-decision tree also Gomory cuts (see Gomory (1963)were added to the current LP relaxation. This algorithm was able to solve randomly generated Euclidean and non-Euclidean instances of CVRP (i.e., DCVRP instances in which the distance constraint was not active) with two or three vehicles and up to 60 customers within 500 seconds on a CYBER 173 computer.

The polyhedral structure of the special case of CVRP where all the customers have demand equal to one was studied by Campos, Corberán and Mota (1991), and a branch-and-cut for this problem was presented by Araque *et al.* (1994). Cornuéjols and Harche (1993) published the first work entirely devoted to the polyhedral study of the CVRP, where they analyzed the standard 2-index CVRP model, VRP2, in which single customer routes are allowed and all the K available vehicles must be used. Due to the presence of equalities (1.24) and (1.25) the polyhedron of CVRP is never full-dimensional and this complicates the polyhedral analysis. Therefore, as was done for the TSP, Cornuéjols and Harche first considered the full-dimensional polyhedron, containing the CVRP polyhedron as a face, associated with the so-called *Graphical* VRP (GVRP) where each customer may be visited more than once (see, e.g., Fleischmann (1988) for a description of the graphical relaxation of the TSP). The basic properties of the GVRP polyhedron are investigated and conditions under which the nonnegativity constraints (1.32), and the capacity constraints (1.26) define facets of the GVRP and CVRP polyhedra are determined.

Additional families of valid inequalities are derived, based on the so-called *path* inequalities of the graphical TSP. Given a connected and not necessarily complete graph $G(V,E)$, an odd integer $s \geq 3$ and a partition of V into A, Z, B_j^i, $i = 1,\ldots,s$, $j = 1,\ldots,n_i \geq 2$ such that:

i) B_j^i is nonempty and the subgraph of G induced by B_j^i is connected, for $i = 1,\ldots,s$, $j = 0,\ldots,n_i+1$ (where $B_0^i \equiv A$ and $B_{n_i+1}^i \equiv Z$ for $i = 1,\ldots,s$);

ii) the edge set (B_j^i, B_{j+1}^i), connecting vertices of the two subsets B_j^i and B_{j+1}^i, is nonempty for $i = 1,\ldots,s$, $j = 0,\ldots,n_i$.

The *path inequality* corresponding to such partition is

$$\sum_{e\in E} f_e x_e \geq f_0 \tag{1.35}$$

where $f_0 = 1 + \sum_{i=1}^{s} \frac{n_i+1}{n_i-1}$ and for each $e \in E$ the coefficient is

$$f_e = \begin{cases} 1 & \text{for } e \in (A, Z) \\ \dfrac{|j-p|}{n_i - 1} & \text{for } e \in (B_j^i, B_p^i), i = 1, \ldots s, \\ & \quad \text{and } j \neq p \text{ such that } |j-p| \leq n_i \\ \max\left\{ \dfrac{p-j+2}{n_l-1}, \dfrac{j-p+2}{n_i-1} \right\} & \text{for } e \in (B_j^i, B_p^l), i \neq l \\ & \quad j = 1, \ldots, n_i \text{ and } p = 1, \ldots, n_l \\ 0 & \text{otherwise.} \end{cases}$$

Related inequalities are the *wheelbarrow* and the *bicycle* inequalities, arising when $Z = \emptyset$ and $Z = A = \emptyset$, respectively. The validity of these inequalities for GVRP and CVRP follows from their validity for the graphical TSP, whose polyhedron contains that of GVRP and CVRP. Furthermore, new classes of inequalities (*capacitated* path, wheelbarrow and bicycle inequalities) taking into account capacity restrictions are described. Also in this case sufficient conditions under which these inequalities define facets of GVRP and CVRP are derived.

Finally, Cornuéjols and Harche considered the *comb* inequalities described by Chvátal (1973) and by Grötschel and Padberg (1979) for the TSP. We are given a complete graph G, a value $s \geq 3$ odd, and vertex subsets $W_0, W_1, \ldots, W_s$ such that W_i and W_j are disjoint for each $i \neq j \neq 0$, $|W_i \setminus W_0| \geq 1$ and $|W_i \cap W_0| \geq 1$ for $i = 1, \ldots, s$. Subset W_0 is called *handle*, whereas subsets $W_1, \ldots, W_s$ are called *teeth* of the comb inequality

$$\sum_{i=0}^{s} \sum_{e \in \sigma(W_i)} x_e \leq \left(\sum_{i=0}^{s} |W_i| \right) - \frac{3s+1}{2} + \alpha(K-1) \tag{1.36}$$

where

$$\alpha = \begin{cases} 0 & \text{if } 0 \notin \bigcup_{i=0}^{s} W_i \\ 1 & \text{if } 0 \in W_0 \setminus \bigcup_{i=1}^{s} W_i \text{ or } 0 \in W_j \setminus W_0 \text{ for some } j = 1, \ldots, s \\ 2 & \text{if } 0 \in W_j \cap W_0 \text{ for some } j = 1, \ldots, s. \end{cases}$$

The case where the depot does not belong to the handle nor to the teeth was already considered by Laporte and Nobert (1984). Cornuéjols and Harche also described an extension of comb inequalities in which several teeth may intersect, called *capacitated comb inequalities*, which are shown to be valid for CVRP and under suitable assumptions also facet-defining.

The derived inequalities are used by Cornuéjols and Harche as cutting planes to solve two instances of CVRP, with 18 and 50 customers, within a branch-and-cut algorithm. The identification of violated inequalities was performed by Cornuéjols and Harche by hand from the current optimal LP solution. The 18 customers problem was solved at the root node, whereas branching was needed for the solution of the 50 customers one. For this problem the lower bound value obtained at the root node was 99.12% of the optimal solution value.

Effective branch-and-cut algorithms for CVRP were recently proposed by Augerat *et al.* (1995), by Hill (1995), and by Achutan, Caccetta and Hill (1995) and (1996). Augerat *et al.* (1995) described several heuristic separation procedures for the classes of valid inequalities proposed by Cornuéjols and Harche. Their branch-and-cut algorithm was able to solve many CVRP instances from the literature containing up to 134 customers within about 18,000 seconds on a Sun Sparc 10 (10 Mflops). Achuthan, Caccetta and Hill used valid inequalities for CVRP based on subtour elimination and on properties of feasibile solutions for which they also give heuristic separation procedures. Their algorithm was able to solve within one hour of CPU time on a Sun Sparc 2 (4 Mflops) some of the instances from the literature with up to 100 customers and randomly generated instances with two or three vehicles and up to 100 customers.

Recently Fischetti, Salazar and Toth (1995) used a different vehicle flow formulation as a basis for a branch-and-cut algorithm. The model uses $O(n^2K)$ binary variables ξ: variable ξ_{ek} counts the number of times edge $e \in E$ is traversed by vehicle k $(k = 1, \ldots, K)$ in the optimal solution. In addition, there are $O(nK)$ binary variables y: variable y_{ik} $(i = 0, \ldots, n;\ k = 1, \ldots, K)$ takes value 1 if customer i is served by vehicle k in the optimal solution, and 0 otherwise.

$$\text{(VRP3)} \qquad \min \sum_{e \in E} c_e \sum_{k=1}^{K} \xi_{ek} \tag{1.37}$$

subject to

$$\sum_{k=1}^{K} y_{ik} = 1 \qquad \text{for each } i \in V \setminus \{0\} \tag{1.38}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{1.39}$$

$$\sum_{e \in \delta(i)} \xi_{ek} = 2y_{ek} \qquad \text{for each } i \in V, k = 1, \ldots, K \tag{1.40}$$

$$\sum_{i \in V} d_i y_{ik} \leq C \qquad \text{for each } k = 1, \ldots, K \tag{1.41}$$

$$\sum_{e \in \delta(S)} \xi_{ek} \geq 2y_{ik} \qquad \text{for each } S \subseteq V \setminus \{0\}, i \in S, k = 1, \ldots, K \tag{1.42}$$

$$y_{ik} \in \{0, 1\} \qquad \text{for each } i \in V, k = 1, \ldots, K \tag{1.43}$$

$$\xi_{ek} \in \{0, 1\} \qquad \text{for each } e \notin \delta(0), k = 1, \ldots, K \tag{1.44}$$

$$\xi_{ek} \in \{0, 1, 2\} \qquad \text{for each } e \in \delta(0), k = 1, \ldots, K. \tag{1.45}$$

Constraints (1.38)–(1.40) impose that each vertex is visited exactly once, that K vehicles leave the depot and that the same vehicle enters and leaves a given customer, respectively. Constraints (1.41) are the capacity restriction for each vehicle, whereas (1.42) impose the connectivity of the solution. Models of this type in which the endpoints of each edge are explicitly indicated are also known as *three-index* formulations and were extensively used as a base for successful exact approaches for the VRP with time windows and other variants of the VRP (see Desrosiers *et al.*, 1995). The main drawback of this model is represented by the increased number of variables. On the

other hand, it allows both the direct use of all the inequalities proposed for model VRP2 (by simply defining $x_e = \sum_{k=1}^{K} \xi_{ek}$ for all $e \in E$), and the development of additional and stronger cuts. Preliminary computational results, obtained by using a simple branch-and-cut framework, showed that instances with up to 30 vertices can be solved to optimality in short computing time.

1.7 THE SET PARTITIONING APPROACH

The *set partitioning* (SP) formulation of the VRP was originally proposed by Balinski and Quandt (1964) and uses an exponential number of binary variables, each associated with a different feasible circuit of G. More specifically, let $\mathcal{H} = \{H_1, \ldots, H_M\}$ denote the collection of all the circuits of G each corresponding to a feasible route, with $M = |\mathcal{H}|$. Each circuit H_j has an associated optimal cost c_j, and let a_{ij} be a binary coefficient which takes value 1 if vertex i is covered (i.e., visited) by route H_j. The binary variable x_j, $j = 1, \ldots, M$, is equal to 1 if and only if circuit H_j is selected in the optimal solution. The model is:

$$\text{(VRP4)} \qquad \min \sum_{j=1}^{M} c_j x_j \tag{1.46}$$

subject to

$$\sum_{j=1}^{M} a_{ij} x_j = 1 \qquad i \in V \setminus \{0\} \tag{1.47}$$

$$\sum_{j=1}^{M} x_j = K \tag{1.48}$$

$$x_j \in \{0, 1\} \qquad j = 1, \ldots, M. \tag{1.49}$$

This is a very general model which may easily take into account several constraints as, for example, time windows, since route feasibility is implictly considered in the definition of set $\mathcal{H}$. Agarwal, Mathur and Salkin (1989) proposed an exact algorithm for CVRP based on the set partitioning approach, whereas several successful applications of this technique to tightly constrained VRPs are reported in Desrosiers *et al.* (1995). Moreover, the linear programming relaxation of this formulation is typically very tight (see also Bramel and Simchi-Levi (1997) which gives a detailed probabilistic analysis of the quality of the linear programming relaxation of the set partitioning formulation). Finally, observe that if the cost matrix satisfies the triangle inequality then the set partitioning model may be transformed into an equivalent *set covering* (SC) model VRP4' by writing (1.47) as

$$\sum_{j=1}^{M} a_{ij} x_j \geq 1 \qquad i \in V \setminus \{0\}. \tag{1.50}$$

Any feasible solution to model VRP4 is also feasible for VRP4' and any feasible solution to VRP4' may be transformed into a feasible solution of VRP4 of not greater cost. In fact if the VRP4' solution is infeasible for VRP4 this means that one or

more customers are visited more than once. Then, these customers may be removed by applying shortcuts from all the routes where they are included but one; since the triangle inequality holds each such shortcut would not increase the solution cost. The main advantage of using the SC formulation with respect to the SP one is that in the former only inclusion-maximal feasible circuits, among those with the same cost, need be considered in the definition of $\mathcal{H}$. This considerably reduces the number M of variables. In addition, when using SC formulation the dual solution space is considerably reduced since dual variables are restricted to non negative values only.

One of the main drawbacks of SP and SC models is represented by the huge number of variables, which in non tightly-constrained instances with tenths of customers may easily run into the billions. The explicit generation of all the feasible circuits (columns) is thus normally impractical, and one has to resort to a *column generation* approach to solve the linear programming relaxation of model VRP4. In this approach a restricted problem, explicitly containing only a small subset of the possible columns, is iteratively solved, thus obtaining primal and dual solutions $\overline{x}$ and π, w, respectively (where π is the vector of dual variables associated with constraints (1.47) and w is the dual variable associated with constraint (1.48)). To check if the current solution $\overline{x}$ is optimal for the complete problem, the column r with smallest linear programming reduced cost $\overline{c}_r$ is computed, where

$$\overline{c}_r = \min_{j=1,\ldots,M} \{c_j - \sum_{i=1}^{n} \pi_i a_{ij} - w\}. \tag{1.51}$$

If $\overline{c}_r$ is nonnegative, then the solution of the current restricted problem is optimal also for the complete problem; otherwise column r is added to the problem and a new iteration is executed until the optimal solution of the LP relaxation is determined.

In CVRP the so-called column generation subproblem, i.e., the problem of determinig the column with smallest reduced cost, is difficult to be solved. More precisely, let y be a set of n binary variables, one for each customer, taking value 1 if the customer is selected in the circuit with the smallest reduced cost, and value 0 otherwise. The column generation subproblem is:

$$\text{(CG)} \qquad \min f(y) - \sum_{i=1}^{n} \pi_i y_i - w \tag{1.52}$$

subject to

$$\sum_{i=1}^{n} d_i y_i \leq C \tag{1.53}$$

$$y_i \in \{0,1\} \qquad i = 1,\ldots,n \tag{1.54}$$

where $f(y)$ is the cost of the optimal circuit visiting the depot and all the customers for which $y_i = 1$. Observe that $f(y)$ is a nonlinear function of y whose value may only be determined by solving a TSP. Note also that the proposed scheme, with (1.46)–(1.49) as master problem and (1.52)–(1.54) as subproblem may be obtained by applying Dantzig-Wolfe decomposition to an appropriate three-index formulation of CVRP (see, Desaulniers *et al.*, 1998).

A weaker scheme can be obtained by solving a relaxed version of the subproblem, calling for the solution of a constrained shortest path problem. This type of approach has been succesfully used to solve the VRP with Time Windows and other versions of VRP (see, e.g., Desrochers, Desrosiers and Solomon, 1992; and Desrosiers *et al.*, 1995).

Agarwal, Mathur and Salkin (1989) considered a relaxation, called RVRP4, of model VRP4 where constraint (1.48) is not present, and developed an exact algorithm for the Euclidean case of CVRP. They solved the column generation subproblem by using a branch-and-bound algorithm which uses as lower bound a linear approximation of $f(y)$ of the type $f(y) = \sum_{i=1}^{n} p_i y_i$, thus reducing problem CG to a knapsack problem which is relatively much easier to solve (see, e.g., Martello and Toth, 1990). The performance of the column generation scheme is improved by means of additional ingredients such as the *a-priori* estimation of optimal values of the dual variables as a function of customer attributes, as the demand and the proximity to other customers. Upper bounds on the dual variables are also imposed to reduce excessive fluctuations of their values.

The overall algorithm by Agarwal, Mathur and Salkin used the final (fractional) LP solution of RVRP4, with value Z_{LP}, to derive good heuristic solutions for the CVRP. Starting from the columns in the optimal LP basis, they heuristically solved the set covering problem involving such columns and then transformed the resulting solution into a set partitioning one, with value Z_U, by applying shortcuts. This heuristic algorithm was tested on Euclidean instances from the literature and showed that the quality of the obtained solution is generally good.

The optimal solution to CVRP is obtained by determining the optimal integer solution of the set partitioning problem containing all the columns which have reduced cost not greater than $Z_U - Z_{LP}$. However, when this gap is not small the number of columns generated may be still extremely large. Thus, Agarwal, Mathur and Salkin proposed to circumvent this problem by iteratively solving a set partitioning problem which includes only the columns with reduced cost not greater than $t \leq Z_U - Z_{LP}$. If the gap between the optimal solution value Z_{SP} of such reduced problem and Z_{LP} is smaller than t then the solution found is optimal also for the unrestricted problem and possibly required a much smaller computational effort. Otherwise all the columns with reduced cost not greater than $Z_{SP} - Z_{LP}$ are generated and the set partitioning problem is solved again thus obtaining the final optimal solution to the unrestricted problem. To guarantee that the first set of generated columns contains at least one feasible CVRP solution, the value of t was defined as the largest reduced cost of a route belonging to a known feasible solution.

Agarwal, Mathur and Salkin successfully applied their algorithm to seven Euclidean CVRP instances with up to 25 customers, where the arc costs were defined as the Euclidean distances rounded to the nearest integer. The algorithm required at most 103 seconds on an IBM 370/4281, which according to the authors is about 7.5 times slower that the CDC 7600 used in Christofides, Mingozzi and Toth (1981). On average the set partitioning based algorithm resulted considerably faster than the branch-and-bound approach proposed by Christofides, Mingozzi and Toth.

Recently, Hadjcostantinou, Christofides and Mingozzi (1995) proposed a branch-and-bound algorithm where the lower bound is computed by heuristically solving the dual of the linear programming relaxation of model VRP4. The heuristic dual

solutions are obtained by combining two relaxations of the original problem: the q-path relaxation proposed in Christofides, Mingozzi and Toth (1981), and the k-shortest path relaxation proposed in Christofides and Mingozzi (1989). The proposed approach was able to solve randomly generated Euclidean instances with up to 30 vertices and instances proposed in the literature with up to 50 vertices, within a time limit of 12 hours on a Silicon Grapics Indigo R4000 (12 Mflops).

1.8 CONCLUSIONS

In this paper we reviewed the most important exact algorithms proposed during the last decade for the Capacitated Vehicle Routing Problem with either symmetric or asymmetric cost matrix. The progress made with these algorithms with respect to those of the previous generation is considerable: the dimension of the largest instances solved has been increased from about 20 to more than 100 customers. However, the CVRP is still far from being a closed chapter in the combinatorial optimization book. In fact some Euclidean problems from the literature with 75 customers are still unsolved and, in our opinion, the size of the problems which may be actually solved in a systematic way by the present approaches is limited to few tenths of customers.

Several possible directions of research are still almost uncovered, e.g., Dantzig-Wolfe decomposition based approaches (also known as *branch-and-price* approaches), but also a more deep investigation and understanding of the capabilities of the available techniques is strongly needed. As an example, we may mention that a direct computational evaluation and comparison of the effectiveness of the algorithms presented in this paper for the symmetric case is not possible. In fact, each author either considered a slightly different problem (e.g., in Fisher (1994a) single customers routes were not allowed, whereas Miller (1995) allowed them) or solved a completely different set of instances. The only instance which has been tackled by almost all the authors we considered is the 50 customers Euclidean problem described in Christofides and Eilon (1969). However, for this instance Fisher (1994a) used a real-valued cost matrix with Euclidean distances, Miller (1995), and Augerat *et al.* (1995) used an integer cost matrix with Euclidean distances rounded to the nearest integer. As to Hadjconstantinou, Christofides and Mingozzi (1995), and Achuthan, Caccetta and Hill (1995) they used a hybrid solution where the integer cost of each arc is defined as the Euclidean distance between its endpoints multiplied by 10^4 and 10^3, respectively, and then rounded to the nearest integer. Another research issue which may lead to interesting results is represented by the adaptation to the symmetric CVRP of the exact approaches developed for the asymmetric case, and vice versa.

Ackowledgments

This work has been supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST), and by Consiglio Nazionale delle Ricerche (CNR), Italy. The authors gratefully thank Alberto Caprara, Jacques Desrosiers, Gilbert Laporte and Juan Josè Salazar for their useful comments.

References

Achuthan, N.R., L. Caccetta and S.P. Hill. (1995). An Improved Branch and Cut Algorithm for the Capacitated Vehicle Routing Problem. Technical report, School of Mathematics and Statistics, Curtin University of Technology, Perth, Australia.

Achuthan, N.R., L. Caccetta and S.P. Hill. (1996). A New Subtour Elimination Constraint for the Vehicle Routing Problem. *European Journal of Operational Research*, 91:573–586.

Agarwal, Y., K. Mathur and H.M. Salkin. (1989). A Set-Partitioning-Based Exact Algorithm for the Vehicle Routing Problem. *Networks*, 19:731–749.

Araque, J.R., G. Kudva, T.L. Morin and J.F. Pekny. (1994). A Branch-and-Cut for Vehicle Routing Problems. *Annals of Operations Research*, 50:37–59.

Augerat, P., J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef and G. Rinaldi. (1995). Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem. Technical Report RR949-M, ARTEMIS-IMAG, Grenoble, France.

Balinski, M. and R. Quandt. (1964). On an Integer Program for a Delivery Problem. *Operations Research*, 12:300–304.

Bellmore, M. and J.C. Malone. (1971). Pathology of Travelling Salesman Subtour-Elimination Algorithms. *Operations Research*, 19:278–307.

Bodin, L., B.L. Golden, A.A. Assad and M.O. Ball. (1983). Routing and Scheduling of Vehicles and Crews, the State of the Art. *Computers and Operations Research*, 10(2):63–212.

Bramel, J. and D. Simchi-Levi. (1997). On the Effectiveness of the Set Partitioning Formulation for the Vehicle Routing Problem. *Operations Research*, 45:295–301.

Campos, V., A. Corberán and E. Mota. (1991). Polyhedral Results for a Vehicle Routing Problem. *European Journal of Operational Research*, 52:75–85.

Carpaneto, G. and P. Toth. (1980). Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem. *Management Science*, 26:736–743.

Christofides, N. (1985a). Vehicle Routing. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, editors, *The Traveling Salesman Problem*, pages 431–448. Wiley, Chichester.

Christofides, N. (1985b). Vehicle Routing. In J.K. Lenstra and A.H.G. Rinnooy Kan, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 148–163. Wiley, Chichester.

Christofides, N. and S. Eilon. (1969). An Algorithm for the Vehicle Dispatching Problem. *Operational Research Quarterly*, 20:309–318.

Christofides, N. and A. Mingozzi. (1989). Vehicle Routing: Practical and Algorithmic Aspects. In C.F.H. van Rijn, editor, *Logistics: Where Ends Have to Meet*, pages 30–48. Pergamon Press.

Christofides, N., A. Mingozzi and P. Toth. (1979). The Vehicle Routing Problem. In N. Christofides, A. Mingozzi, P. Toth and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. Wiley, Chichester.

Christofides, N., A. Mingozzi and P. Toth. (1981). Exact Algorithms for the Vehicle Routing Problem Based on the Spanning Tree and Shortest Path Relaxations. *Mathematical Programming*, 20:255–282.

Chvátal, V. (1973). Edmonds Polytopes and Weakly Hamiltonian Graphs. *Mathematical Programming*, 5:29–40.

Cornuéjols, G. and F. Harche. (1993). Polyhedral Study of the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 60(1):21–52.

Desaulniers, G., J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis and D. Villeneuve. (1997). A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93, Kluwer, Norwell, MA.

Desrochers, M., J. Desrosiers and M.M. Solomon. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40:342–354.

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

Desrosiers, J., Y. Dumas, M.M. Solomon and F. Soumis. (1995). Time Constrained Routing and Scheduling. In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhuser, editors, *Network Routing*, Handbooks in Operations Research and Management Science, pages 35–139, North-Holland, Amsterdam.

Dongarra, J.J. (1996). Performance of Various Computers using Standard Linear Equations Software. Technical Report CS–89–85, University of Tennessee, Knoxville.

Fischetti, M., J.J. Salazar-Gonzalez and P. Toth. (1995). Experiments with a Multi-Commodity Formulation for the Symmetric Capacitated Vehicle Routing Problem. In *Proceedings of the 3rd Meeting of the EURO Working Group on Transportation*, Barcelona, Spain. Universitat Politècnica de Catalunya.

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

Fischetti, M. and P. Toth. (1992). An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem. *Mathematical Programming*, 53:173–197.

Fischetti, M., P. Toth and D. Vigo. (1994). A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs. *Operations Research*, 42(5):846–859.

Fisher, M.L. (1985). An Application Oriented Guide to Lagrangian Relaxation. *Interfaces*, 15:10–21.

Fisher, M.L. (1994a). Optimal Solution of Vehicle Routing Problems using Minimum k-Trees. *Operations Research*, 42:626–642.

Fisher, M.L. (1994b). A Polynomial Algorithm for the Degree Constrained k-Tree Problem. *Operations Research*, 42(4):776–780.

Fisher, M.L. (1995). Vehicle Routing. In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, editors, *Network Routing*, Handbooks in Operations Research and Management Science, pages 1–33. North-Holland, Amsterdam.

Fisher, M.L. and R. Jaikumar. (1981). A Generalized Assignment Heuristic for the Vehicle Routing Problem. *Networks*, 11:109–124.

Fleischmann, B. (1988). A New Class of Cutting Planes for the Symmetric Traveling Salesman Problem. *Mathematical Programming*, 40:225–246.

Golden, B.L. and A.A. Assad. (1988). *Vehicle Routing: Methods and Studies*. North-Holland, Amsterdam.

Gomory, R. (1963). An Algorithm for Integer Solution to Linear Programs. In *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York.

Grötschel, M. and M.W. Padberg. (1979). On the Symmetric Traveling Salesman Problem: I-II. *Mathematical Programming*, 16:265–302.

Hadjiconstantinou, E., N. Christofides and A. Mingozzi. (1995). A New Exact Algorithm for the Vehicle Routing Problem Based on q-Paths and k-Shortest Paths Relaxations. *Annals of Operations Research*, 61:21–43.

Held, M. and R.M. Karp. (1971). The Traveling Salesman Problem and Minimum Spanning Trees: Part II. *Mathematical Programming*, 1:6–25.

Hill, S.P. (1995). *Branch-and-Cut Methods for the Symmetric Capacitated Vehicle Routing Problem.* PhD thesis, Curtin University of Technology, Australia.

Kulkarni, R.V. and P.V. Bhave. (1985). Integer Programming Formulations of Vehicle Routing Problems. *European Journal of Operational Research*, 20:58–67.

Laporte, G. (1992). The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research*, 59:345–358.

Laporte, G. (1997). Vehicle Routing. In M. Dell'Amico, F. Maffioli and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, Wiley, Chichester.

Laporte, G., H. Mercure and Y. Nobert. (1986). An Exact Algorithm for the Asymmetrical Capacitated Vehicle Routing Problem. *Networks*, 16:33–46.

Laporte, G. and Y. Nobert. (1984). Comb Inequalities for the Vehicle Routing Problem. *Methods of Operations Research*, 51:271–276.

Laporte, G., Y. Nobert and M. Desrochers. (1985). Optimal Routing under Capacity and Distance Restrictions. *Operations Research*, 33:1050–1073.

Laporte, G. and Y. Nobert. (1987). Exact Algorithms for the Vehicle Routing Problem. *Annals of Discrete Mathematics*, 31:147–184.

Laporte, G. and I.H. Osman. (1995). Routing Problems: A Bibliography. *Annals of Operations Research*, 61:227–262.

Lenstra, J.K. and A.H.G. Rinnooy Kan. (1975). Some Simple Applications of the Traveling Salesman Problem. *Operational Research Quarterly*, 26:717–734.

Lin, S. and B.W. Kernighan. (1973). An Effective Heuristic Algorithm for the Travelling Salesman Problem. *Operations Research*, 21:498–516.

Magnanti, T.L. (1981). Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects. *Networks*, 11:179–214.

Martello, S. and P. Toth. (1990). *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Chichester.

Miller, C.E., A.W. Tucker and R.A. Zemlin. (1960). Integer Programming Formulations and Traveling Salesman Problems. *Journal of the ACM*, 7:326–329.

Miller, D.L. (1995). A Matching Based Exact Algorithm for Capacitated Vehicle Routing Problems. *ORSA Journal on Computing*, 7(1):1–9.

Miller, D.L. (1997). personal communication.

Miller, D.L. and J.F. Pekny. (1995). A Staged Primal-Dual Algorithm for Perfect b-Matching with Edge Capacities. *ORSA Journal on Computing*, 7:298–320.

Pekny, J.F. and D.L. Miller. (1994). A Staged Primal-Dual Algorithm for Finding a Minimum Cost Perfect Two-Matching in an Undirected Graph. *ORSA Journal on Computing*, 6:68–81.

Toth, P. and D. Vigo. (1995). An Exact Algorithm for the Capacitated Shortest Spanning Arborescence. *Annals of Operations Research*, 61:121–142.

Vigo, D. (1996). A Heuristic Algorithm for the Asymmetric Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, 89:108–126.

2 THE IMPACT OF METAHEURISTICS ON SOLVING THE VEHICLE ROUTING PROBLEM: ALGORITHMS, PROBLEM SETS, AND COMPUTATIONAL RESULTS

Bruce L. Golden
Edward A. Wasil
James P. Kelly
I-Ming Chao

2.1 INTRODUCTION

In the standard, capacitated vehicle routing problem (VRP), a homogeneous fleet of vehicles services a set of customers from a single depot. Each vehicle has a fixed capacity that cannot be exceeded and each customer has a known demand that must be satisfied. Each customer must be serviced by exactly one visit of a single vehicle and each vehicle must leave and return to the depot. There may be route-length restrictions that limit the distance traveled by each vehicle. The objective is to generate a sequence of deliveries for each vehicle so that all customers are serviced and the total distance traveled by the fleet is minimized.

Since it was first introduced by Dantzig and Ramser (1959), the standard VRP and its variants have been studied extensively in the MS/OR literature. The survey articles by Bodin *et al.* (1983) and Laporte (1992) and the volumes edited by Golden and Assad (1988) and Ball *et al.* (1995) provide excellent background material on VRP models and solution procedures. The article by Hall and Partyka (1997) surveys the vehicle routing software industry and describes 20 packages in detail.

In Table 2.1, we give a brief history of vehicle routing. In the 1960s and 1970s, VRP research focused on route-building, route-improvement, and two-phase heuristics.

In the 1980s, mathematical programming-based heuristics were developed. These heuristics required more computational effort, but produced very high-quality solutions. At the end of this three-decade period, some problems with approximately 50 customers could be solved to optimality. In the 1990s, the research focus shifted to applying general-purpose metaheuristics to the VRP (including simulated annealing, deterministic annealing, genetic algorithms, neural networks, and tabu search) and some of these methods generated highly accurate solutions to well-known problem sets. For background information on metaheuristics, Osman and Kelly (1996) present an accessible overview. In an edited volume, Reeves (1993) and his co-authors give a comprehensive review of metaheuristics for combinatorial problems. Fisher (1995) describes the "three generations" of VRP research on heuristics including the current focus on simulated annealing and tabu search. Gendreau, Laporte, and Potvin (1997) survey the latest developments in the application of local search algorithms to the VRP. In Section 2.2, we review three metaheuristics – tabu search, simulated

Table 2.1 A brief history of vehicle routing.

Decade	*Events*
1950s	The VRP is formulated as an integer program Some small problems (10 to 20 customers) are solved
1960s	Early route-building heuristics (e.g., Clarke and Wright, 1964) are proposed 2-opt and 3-opt are applied to the VRP (Christofides and Eilon, 1969) Some problems with 30 to 100 customers are solved
1970s	A number of two-phase heuristics are proposed (e.g., Gillett and Miller, 1974) Computational efficiency becomes an issue (e.g., Golden, Magnanti, and Nguyen, 1977) Some larger problems (100 to 1,000 customers) are solved Some problems with 25 to 30 customers can be solved using optimal methods
1980s	Math programming-based procedures are proposed (e.g., Fisher and Jaikumar, 1981) Interactive (man-machine) heuristics are developed (e.g., Cullen, Jarvis, and Ratliff, 1981) Some problems with approximately 50 customers can be solved using optimal methods
1990s	Metaheuristics are applied to the VRP Some problems with 50 to 100 customers can be solved to optimality (Fisher, 1995)

annealing, and deterministic annealing – that have been used to solve the standard VRP. In Section 2.3, we present a detailed discussion of the computational experience with VRP metaheuristics that focuses on test problems and gives results that have appeared in the literature over the last six years. We also present issues related to fine-tuning an algorithm. In Section 2.4, we develop a new set of 20 large-scale vehicle routing problems that have 200 to 483 customers and generate solutions to these problems using tabu search and a procedure based on deterministic annealing. We conclude the paper with comments on future directions in vehicle routing research.

2.2 OVERVIEW OF THREE METAHEURISTICS

Traditionally, local improvement methods for the VRP explore the neighborhood of a current solution and only select new solutions that strictly decrease the total distance

traveled by the fleet. Tabu search, simulated annealing, and deterministic annealing can select (under certain conditions) new solutions that increase the total distance and, in this way, they can avoid becoming trapped in poor local optima. In this section, we present an overview of these three metaheuristics.

2.2.1 Tabu Search

Typically, tabu search-based algorithms for the VRP incorporate several common components that form the basis for this class of metaheuristics. These components include a neighborhood definition, short-term memory, long-term memory, and an aspiration criterion. In addition to these standard components, developers often include more elaborate adaptive programming techniques such as strategic oscillation (see the article by Glover (1997) and Chapter 3 in Reeves (1993) for descriptions of these techniques). In this section, we describe the basic tabu search strategies that have been incorporated into several successful VRP algorithms. We summarize these strategies and the tabu search-based VRP algorithms in Table 2.2. We point out that subtle differences in algorithm implementation and structure can produce significant differences in solution quality.

Neighborhood

A fundamental building block of any tabu search algorithm is the definition of the search neighborhood. The neighborhood is defined in terms of a move that transforms one solution to another. Typically, VRP algorithms use an insertion move where a customer is removed from one route and inserted into another route. Moves that involve exchanging or swapping customers between two routes are also considered by some algorithms. Finally, in one algorithm, a series of insertions are constructed to form an ejection chain (customer A is moved to a new position occupied by customer B thereby ejecting customer B, and so on; see Rego and Roucairol (1996) and Chapter 3 in Reeves (1993) for more details). Each type of move creates a neighborhood around the current solution from which a new solution is selected. The selection of the new solution is biased by the short-term and long-term memory functions. In addition, the question of feasibility is used to further define the neighborhood structure. Four algorithms listed in Table 2.2 allow infeasible solutions that are penalized in proportion to their degree of infeasibility. For example, a capacity constraint on a route may be violated. In this case, the distance measure for the route is inflated by an amount that is proportional to the capacity overflow. Thus, infeasible solutions, although penalized, may be selected from the neighborhood.

Short-term Memory

The component of tabu search that is used most often in VRP algorithms is short-term memory. Usually, short-term memory is imposed to restrict the search from revisiting solutions that were considered previously and to discourage the search process from cycling between subsets of solutions. Typically, developers of algorithms restrict the neighborhood to exclude solutions that contain attributes that are designated tabu. In five algorithms listed in Table 2.2, short-term memory is used to prohibit move reversal

Table 2.2 Tabu search components found in six VRP algorithms.

Reference	*Neighborhood*	*Short-term Memory*	*Long-term Memory*	*Comments*
Semet and Taillard (1993)	Insertion moves. Allows for infeasibility (unmet demand) by dynamic penalty in objective function.	Prohibits customers from being assigned to routes from which they were recently removed. Tabu time is selected from a uniform distribution.	None.	Heterogeneous fleet with complex side constraints. Candidate list based on a systematic sampling of all possible moves. Intensification/diversification accomplished via full or approximate evaluation.
Taillard (1993)	Swap moves. Only feasible moves considered.	Prohibits customers from being assigned to routes from which they were recently removed. Tabu time is selected from a uniform distribution.	Frequency-based penalty term is used to discourage frequently made moves.	Decomposition approach.
Gendreau, Hertz, and Laporte (1994)	Insertion moves. Allows for infeasibility (capacity or length) controlled by dynamic penalty in objective function.	Prohibits customers from being assigned to routes from which they were recently removed. Tabu time is selected from a uniform distribution.	Frequency-based penalty term is used to discourage movement of frequently reassigned customers.	Efficient insertion procedures limit candidate moves to those that contain close customers.

Note: The aspiration criterion for all the procedures cited here is the same: the tabu status is overridden if the solution value is better than the best solution found so far.

Table 2.2 *(continued)* Tabu search components found in six VRP algorithms.

Reference	*Neighborhood*	*Short-term Memory*	*Long-term Memory*	*Comments*
Rochat and Taillard (1995)	Swap moves. Only feasible moves considered.	Prohibits customers from being assigned to routes from which they were recently removed. Tabu time is selected from a uniform distribution.	Frequency-based penalty term is used to discourage frequently made moves.	Intensification and diversification strategy for creating initial solutions based on previous solutions.
Rego and Roucairol (1996)	Ejection chains of sequential insertions and displacements of customers. Allows for infeasible solutions.	Prohibits individual ejections (insertions) from being reversed. Tabu time is selected from a uniform distribution.	Frequency-based penalty term is used to discourage movement of frequently reassigned customers.	Parallel implementation.
Xu and Kelly (1996)	Insertion and swap moves controlled by a network flow model. Allows for infeasibility (capacity) by dynamic penalty factors that are used to define network flow costs.	Prohibits customers from being reassigned to routes from which they were recently removed. Prohibits customers that have been recently moved to be moved again. Tabu time is selected from a uniform distribution.	Frequency-based penalty term is used to discourage movement of frequently reassigned customers.	Strategic oscillation between swaps and insertions.

Note: The aspiration criterion for all the procedures cited here is the same: the tabu status is overridden if the solution value is better than the best solution found so far.

by temporarily disallowing customer movement from one route to another. The number of iterations that a customer's mobility is restricted is often referred to as the tabu time. Frequently, the tabu time is selected from a random uniform distribution.

Long-term Memory

Long-term memory is used to diversify the search. Usually, it is implemented to discourage frequently made moves and to encourage the search process to explore new regions of the solution space. Five algorithms listed in Table 2.2 use a frequency-based measure to penalize moves that involve often-moved customers. This strategy makes the movement of seldom-moved customers more attractive and encourages the search to explore new solutions. There is common agreement among developers of tabu search-based algorithms that long-term memory is essential for producing the highest-quality solutions.

Aspiration Criterion

The aspiration criterion is a mechanism for overriding the short-term and long-term memory functions. All six algorithms listed in Table 2.2 use a simple aspiration criterion: if a solution is better than the best seen so far, then the solution should be selected regardless of its tabu status.

2.2.2 Simulated Annealing

In order to alleviate the problem of becoming trapped at a local optimum, simulated annealing allows the selection of some uphill moves in a controlled manner. We now briefly describe the simulated annealing procedure (see the overview article by Osman and Kelly (1996) and Chapter 2 in Reeves (1993) for more details).

Let S be the current solution and $N(S)$ be a neighborhood of S that contains alternative solutions that are in the "vicinity" of S. We randomly select $S' \in N(S)$ and compute the difference $D = f(S') - f(S)$, where $f(S)$ is the objective function value of S. If $D < 0$, then S' is selected as the new solution (that is, downhill moves are always accepted). If $D > 0$ and $e^{-D/T} > q$ (where $0 < q < 1$ is randomly generated from a uniform distribution), then S is selected as the new solution (that is, an uphill move may be accepted). T is simply a control parameter known as the temperature. Typically, the temperature is gradually lowered during the search process so that the probability of accepting an uphill move steadily decreases. The simulated annealing procedure continues until a stopping condition is met.

We point out that two of the earliest papers to apply simulated annealing to the vehicle routing problem were by Robusté, Daganzo, and Souleyrette (1990) and Alfa, Heragu, and Chen (1991). In the first paper, Robusté, Daganzo, and Souleyrette develop an algorithm in which routes are improved with three types of customer moves. They test their algorithm on four problems that contain 80, 100, 120, and 500 customers. In the second paper, Alfa, Heragu, and Chen combine simulated annealing with the 3-opt heuristic and test their algorithm on three problems from Christofides and Eilon (1969) that contain 30, 50, and 75 customers.

2.2.3 *Deterministic Annealing*

Dueck (1993) develops several deterministic variants of simulated annealing, including threshold accepting and record-to-record travel. In threshold accepting (see Dueck and Scheurer (1990) and Osman and Kelly (1996) for more details), a threshold value T is specified as the upper bound on the amount of objective function increase allowed from one move to the next. If $f(S') < f(S)+T$, then S' is selected as the new solution. Thus, uphill moves can be made.

In record-to-record travel, *Record* is defined as the total distance of the best solution observed so far. *Deviation* is defined as $k\% \times Record$. If $S' < Record + Deviation$, then S' is selected as the new solution. Again, uphill moves can be made. We point out that record-to-record travel has been used by Chao, Golden, and Wasil (1993, 1995) in heuristics for the multi-depot VRP and the period VRP.

2.3 COMPUTATIONAL EXPERIENCE

In this section, we describe the set of vehicle routing test problems and the computational results that have appeared in the literature.

2.3.1 *Test Problems*

The most widely used vehicle routing test problems are the 14 benchmark problems that are described in a 1979 article by Christofides, Mingozzi, and Toth (1979). The characteristics of these test problems are summarized in Table 2.3. In the remainder of this paper, we refer to these as the benchmark problems. The first three problems were first described in a 1969 article by Christofides and Eilon (1969). Problem 4 was produced by combining the customers of problems 1 and 3 with the depot of problem 3, and problem 5 was produced by combining the first 49 customers of problem 2 with the customers and depot of problem 4. Problems 6 to 10 were produced by including a route-length constraint (with a service time for each customer) in problems 1 to 5. In problems 1 to 10, the locations of the customers were generated by a random uniform distribution. In problems 11 and 12, the customers were clustered to create problems similar to those observed in practice and problems 13 and 14 were produced by including a route-length constraint in problems 11 and 12.

Although most of the computational testing over the last 20 years or so has focused on the benchmark problems, there are other vehicle routing test problems that have been presented or published in the literature. We summarize a representative set of sources for test problems in Table 2.4. These sources provide a wide variety of problems with different characteristics and some of the problem sets are available on the Internet.

2.3.2 *Computational Results*

Since 1991, various metaheuristics have been applied to the 14 benchmark problems. For the most part, research has concentrated on using tabu search and to a much lesser extent simulated annealing and neural networks. In Table 2.5, we present a representative set of papers that solve the benchmark problems with metaheuristics. In these papers, the authors focused, to a large extent, on the quality of the solution as the measure of computational performance. With the exception of VB95, all of

Table 2.3 Characteristics of the vehicle routing test problems taken from Christofides, Mingozzi, and Toth (1979).

Test Problem	*Number of Customers*	*Number of Vehicles*	*Vehicle Capacity*	*Max. Route Length*	*Service Time*
1	50	5	160		
2	75	10	140		
3	100	8	200		
4	150	12	200		
5	199	17	200		
6	50	6	160	200	10
7	75	11	140	160	10
8	100	9	200	230	10
9	150	14	200	200	10
10	199	18	200	200	10
11	120	7	200		
12	100	10	200		
13	120	11	200	720	50
14	100	11	200	1040	90

Table 2.4 Sources of vehicle routing test problems in the literature.

Source	*Description of Test Problems*	*URL*
Beasley (1990)	OR-Library contains a variety of vehicle routing problems including the 14 benchmark test problems	http://mscmga.ms.ic.ac.uk/
Fisher (1994)	Data and optimal solutions for three problems with 44, 71, and 134 customers	
Russell (1977)	Problems 2 and 3 from Christofides, Mingozzi, and Toth with different vehicle capacities and numbers of vehicles	
Taillard (1993)	Problem with 385 customers and randomly generated grid problems with 64, 144, 256, 324, and 1,024 customers	http://www.idsia.ch/~eric
Van Breedam (1994)	420 problems with 100 customers, including problems with time windows, pickups, and heterogeneous demand	http://www.ruca.ua.ac.be

the papers listed in Table 2.5 report results using real distances. VB95 uses rounded distances, G91 uses real, truncated, and rounded distances, and XK96 uses real and integer distances.

Table 2.5 Papers that report computational experience in solving the fourteen benchmark vehicle routing test problems.

Abbreviation	*Authors*	*Year*	*Source*	*Comments*
Tabu Search				
G91	Gendreau *et al.* (1991)	91	WP	Uses real, truncated, and rounded distances
T92	Taillard (1992)	92	WP	
T93	Taillard (1993)	93	NW	
OF93	Osman (1993)	93	AOR	Algorithm uses first-best-admissible strategy
OB93	Osman (1993)	93	AOR	Algorithm uses best-admissible strategy
GS94	Gendreau *et al.* (1994)	94	MS	Single pass of TABUROUTE algorithm
GB94	Gendreau *et al.* (1994)	94	MS	Best solution in multiple passes of TABUROUTE
RT95	Rochat and Taillard (1995)	95	JH	
RRS96	Rego and Roucairol (1996)	96	MH	Sequential algorithm
RRP96	Rego and Roucairol (1996)	96	MH	Parallel algorithm
XK96	Xu and Kelly (1996)	96	TS	Uses real and integer distances
XK97	Xu and Kelly (1997)	97	PC	Parameter settings as in XK96
Simulated Annealing				
OS93	Osman (1993)	93	AOR	
VB95	Van Breedam (1995)	95	EJOR	Uses rounded distances
Neural Networks				
GH96	Ghaziri (1996)	96	MH	Hierarchical deformable network algorithm
GS96	Ghaziri (1996)	96	MH	Supervised hierarchical deformable network

AOR	*Annals of Operations Research*	NW	*Networks*
EJOR	*European Journal of Operational Research*	PC	Personal communication
JH	*Journal of Heuristics*	TS	*Transportation Science*
MH	*Meta-heuristics: Theory and Applications*	WP	Working Paper
MS	*Management Science*		

In Table 2.6, we provide the solution values to the benchmark problems that are generated by the tabu search algorithms that use real distances. These are the solution values that are reported in each paper and they give rise to minor discrepancies (e.g., for problem 6, is 555.44 reported by OF93 the same as 555.43 reported by G91?). Four papers (G91, GB94, OF93, and XK96) give vehicle routes so that we were able

Table 2.6 Solution values to fourteen benchmark problems generated by tabu search algorithms.

Pr	*n*	*G91*	*T92*	*T93*	*OF93*	*OB93*	*GS94*
1	50	**524.61***	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**
2	75	836.37	835.32	**835.26**	844	844	835.77
3	100	**826.14**	828.98	**826.14**	838	835	829.45
4	150	1034.90	1029.64	**1028.42**	1044.35	1052	1036.16
5	199	1329.29	1300.89	1298.79	1334.16	1354	1322.65
6	50	**555.43**	**555.43**	**555.43**	**555.44**	**555.44**	**555.43**
7	75	913.23	**909.67**	**909.68**	911	913	913.23
8	100	**865.94**	**865.94**	**865.94**	878	866.75	**865.94**
9	150	1189.79	1164.24	**1162.55**	1184	1188	1177.76
10	199	1421.88	1403.21	1397.94	1441	1422	1418.51
11	120	1043.94	1073.05	**1042.11**	1043	**1042.11**	1073.47
12	100	822.85	**819.56***	**819.56**	819.59	819.59	**819.56**
13	120	1551.63	1550.15	**1541.14**	1545.98	1547	1573.81
14	100	**866.37**	**866.37**	**866.37**	**866.35**	**866.35**	**866.37**

Pr	*n*	*GB94*	*RT95*	*RRS96*	*RRP96*	*XK96*	*XK97*
1	50	**524.61**		**524.61**	**524.61**	**524.61**	
2	75	835.32		837.50	835.32	**835.26**	
3	100	**826.14**		827.53	827.53	**826.14**	
4	150	1031.07		1054.29	1044.35	1029.56	
5	199	1311.35	**1291.45**	1338.49	1334.55	1298.58	
6	50	**555.43**		**555.43**	**555.43**		**555.43**
7	75	**909.68**		**909.68**	**909.68**		965.62
8	100	**865.94**		868.29	866.75		881.38
9	150	1162.89		1178.84	1164.12		No solution
10	199	1404.75	**1395.85**	1420.84	1420.84		1439.29
11	120	**1042.11**		1043.54	**1042.11**	**1042.11**	
12	100	**819.56**		**819.56**	**819.56**	**819.56**	
13	120	1545.93		1550.17	1550.17		1618.55
14	100	**866.37**		866.53	**866.37**		915.24

Bold Best-known solution value
* Optimal solution value
Pr Test problem number
n Number of customers

to resolve all discrepancies (except one) by comparing the sequence of customers on each route. In the case of problem 7 for T92, since no routes are provided, we treat 909.67 the same as 909.68.

The accuracy of the tabu search algorithms, as measured by percent above the best-known solution, is given in Table 2.7. As a class, the tabu search algorithms generate high-quality solutions to the benchmark problems. To give some idea of the efficiency of tabu search algorithms, we show the running times for five algorithms on different computing platforms in Table 2.8.

With the exception of XK97, on average, each tabu search algorithm generates a solution value that is less than 1% above the best-known solution. The algorithm of Taillard (T93 and RT95) performs the best – this algorithm produces the best-known solution value to each of the 14 benchmark problems. We point out that Taillard (1993) does not provide running times since "... it is very difficult to give them as our algorithm does not find the best-known solution of every problem at each run." We have more to say about computational experimentation and reporting computational results in Section 2.5.

In Table 2.9, we present the solution value and the percent above the best-known solution for simulated annealing and neural network algorithms. Clearly, these procedures are not competitive with the tabu search procedures on the benchmark problems.

2.3.3 Parameters and Their Settings

All of the metaheuristics that have been used to solve the vehicle routing problem contain several parameters (roughly anywhere from 5 parameters to more than 25 parameters). The values of the parameters need to be set before the metaheuristic is run. In Table 2.10, we display the parameters for two tabu search heuristics (XK96 and GS94) and one simulated annealing heuristic (VB95). In XK96, there are 28 penalty, time-related, and control parameters. The control parameters include tolerances of solution, gap, capacity violations, and capacity thresholds. Furthermore, in XK96, there are four parameters that determine how the tabu search algorithm solves a problem. For example, Xu and Kelly terminate a search at 100,000 iterations, invoke their restart/recovery procedure at iteration 20,000 and for every 750 iterations thereafter, and store 37 elite solutions.

2.3.4 Fine-tuning an Algorithm

It is not an easy task to determine the appropriate values for parameters found in VRP algorithms. The ways to set a parameter's value have ranged from simple trial-and-error to sophisticated sensitivity analysis. For example, Van Breedam (1995) states: "The values that have to be assigned to all these technical parameters are for the greater part determined by trial-and-error-like experiments." Gendreau, Hertz, and Laporte (1991) state: "This algorithm contains several parameters and a number of variants can easily be envisaged. Several tests and a considerable amount of fine tuning were carried out in order to arrive at the current version." We point out that, once appropriate values for parameters have been identified, it is standard practice in the literature to report results generated by running an algorithm with the parameters

Table 2.7 Accuracy of tabu search algorithms: Percent above best-known solution.

Pr	*n*	*G91*	*T92*	*T93*	*OF93*	*OB93*	*GS94*	*GB94*	*RT95*	*RRS96*	*RRP96*	*XK96*	*XK97*
1	50	0	0	0	0	0	0	0	0	0	0	0	
2	75	0.133	0.007	0	1.046	1.046	0.061	0.007		0.268	0.007	0	
3	100	0	0.344	0	1.436	1.072	0.401	0		0.168	0.168	0	
4	150	0.630	0.119	0	1.549	2.293	0.753	0.258		2.516	1.549	0.111	
5	199	2.930	0.731	0.568	3.307	4.843	2.416	1.541	0	3.642	3.337	0.552	
6	50	0	0	0	0	0	0	0		0	0		0
7	75	0.390	0	0	0.145	0.365	0.390	0		0	0		6.149
8	100	0	0	0	1.393	0.094	0	0		0.271	0.094		1.783
9	150	2.343	0.145	0	1.845	2.189	1.308	0.029		1.401	0.135		No solution
10	199	1.865	0.527	0.149	3.235	1.873	1.623	0.638	0	1.790	1.790		3.112
11	120	0.176	2.969	0	0.085	0	3.009	0		0.137	0	0	
12	100	0.401	0	0	0.004	0.004	0	0		0	0	0	
13	120	0.681	0.585	0	0.314	0.380	2.119	0.311		0.586	0.586		5.023
14	100	0	0	0	0	0	0	0		0.018	0		5.641
Average		0.682	0.388	0.051	1.026	1.011	0.863	0.199	0	0.771	0.548	0.095	3.618*

* Does not include problem 9

Table 2.8 Running times (in minutes) for tabu search algorithms.

Pr	*n*	*OF93*	*OB93*	*GS94*	*XK96*	*XK97*
1	50	1.90	1.12	6.0	29.92	
2	75	2.98	1.18	53.8	48.80	
3	100	25.72	11.25	18.4	71.93	
4	150	59.33	51.25	58.8	149.90	
5	199	54.10	32.88	90.9	272.52	
6	50	2.88	2.34	13.5		30.67
7	75	17.62	3.38	54.6		102.13
8	100	49.97	20.00	25.6		98.15
9	150	79.26	40.73	71.0		168.08
10	199	76.02	55.17	99.8		368.37
11	120	24.09	23.31	22.2	91.23	
12	100	14.87	6.79	16.0	56.61	
13	120	47.23	22.38	59.2		201.75
14	100	19.59	92.98	65.7		152.98
Average		33.97	26.05	46.82	102.99	160.30

OF93, OB93 CPU time to completion on a Vax 8600 computer
GS94 Total computation time on a Silicon Graphics Workstation (36 MHz, 5.7 Mflops)
XK96, XK97 Total time on a DEC ALPHA Workstation (DEC OSF/1 v3.0)

fixed at these values (for example, the single pass version of TABUROUTE in GS94). However, it is also common practice to report the best solution found during the course of performing sensitivity analysis (for example, the multiple passes of TABUROUTE in GS94).

Xu, Chiu, and Glover (1996), in the context of the Steiner Tree-Star problem, develop a procedure for fine-tuning five key factors in a tabu search algorithm (see Table 2.11). Using two statistical tests in a very small number of experiments, they are able to find a set of values for the five factors that generates improved results in nearly three-quarters of their test problems. Their procedure could be used to find appropriate values for parameters found in VRP metaheuristics. Furthermore, a fractional factorial experiment could also be used to find "best" settings for parameters (see Robertson *et al.* (1998) for a finance application in which fractional factorial experiments are used to construct neural network models).

Xu and Kelly (1996) try to identify the relative contributions of five different components of their tabu search algorithm (network flow moves, swap moves, tabu short-term memory, restart/recovery strategy, a simple tabu search procedure (TSTSP) to find the best sequence of customers on a route). They disable each component one at a time, execute their algorithm on seven VRPs with real distances, and compare the solutions of the five different strategies. Xu and Kelly conclude: "... the TS [tabu search] memory and restart/recovery strategy effectively help to locate extremely good solutions and TSTSP provides an effective enhancement over 3-opt ...." Furthermore, the authors run their algorithm with different frequencies for the swap

Table 2.9 Solution values and accuracy of simulated annealing and neural network algorithms.

		Solution Value			*% Above Best-known Solution*		
Pr	*n*	*OS93*	*GH96*	*GS96*	*OS93*	*GH96*	*GS96*
1	50	528	545.1	539.2	0.646	3.906	2.781
2	75	838.62			0.402		
3	100	829.18	911.2	893.4	0.368	10.296	8.141
4	150	1058	1133.2	1084.7	2.876	10.188	5.472
5	199	1376	1421.5	1401.4	6.547	10.070	8.514
6	50	**555.44**	567.3	561.3	0	2.135	1.055
7	75	**909.68**			0		
8	100	866.75	902.4	894.3	0.094	4.211	3.275
9	150	1164.12	1304.5	1264	0.135	12.210	8.727
10	199	1417.85	1604.3	1580.4	1.576	14.933	13.221
11	120	1176	1113.4	1102.4	12.848	6.841	5.785
12	100	826	826.6	825.1	0.786	0.859	0.676
13	120	1545.98	1645.8	1608.5	0.314	6.791	4.371
14	100	890	881.5	879.8	2.727	1.746	1.550
				Average	2.094	7.016	5.297

Bold Best-known solution value

moves – every two, three, five, and six iterations – and conclude: "... the performance of our algorithm is sensitive to the frequency ... [and it] performs best using a medium frequency."

Van Breedam (1996) tries to determine the significant effects of parameters for a genetic algorithm (GA) procedure and a simulated annealing (SA) procedure for the VRP. He uses analysis of variance (the Automatic Interaction Detection technique) to determine the structure of the relation between total travel time and seven GA parameters (including population size, number of generations, type of local improvement operator, and quality of initial solution) and eight SA parameters (including cooling rate and type of move). Van Breedam applies GA and SA to 15 test problems with 100 customers each (four problems have time windows, two have pickups, and three have heterogeneous demand). He concludes that certain parameters have "consistent significant effect for all problems." For example, in the case of GA, not using a local improvement operator gives worse solutions and using good initial solutions produces better final solutions.

2.3.5 Summary of Observations from the Recent Literature

In reviewing the recent computational experience with metaheuristics for solving the VRP, we make the following observations in three areas.

Overall Performance

Approaches based on neural networks are not competitive. Tabu search outperforms simulated annealing while simulated annealing outperforms genetic algorithms.

Table 2.10 Parameters found in three metaheuristics.

Tabu Search

XK96
Nine Penalty Parameters
ρ_1 ρ_2 ρ_3 τ_1 τ_2 ρ_4 $\bar{\rho}_4$ $\dot{\rho}_4$ ρ_5

Nineteen Time-related and Control Parameters
n_1 n_2 n_3 n_4 n_5 n_6 n_7 n_8 n_9 n_{10} s_1 L_1 H_1
τ_1 τ_2 τ_3 ϵ λ_1 λ_2

Other Parameters
Maximum number of iterations
Iteration to begin restart/recovery strategy
Number of stored elite solutions
Number of network moves

GS94
Eleven Parameters
W q p_1 p_2 g h λ θ_{min} θ_{max} α β

Other Parameters
Maximum number of iterations

Simulated Annealing

VB95
Six Technical Parameters
T_{begin} R_c N_{tot} N_{temp} *Acceptance Ratio* T_{end}

Other Parameters
Distance limit(DL)
Number of stops relocated or exchanged(SL)

Tabu search seems to work best when infeasible intermediate solutions are allowed via penalty functions.

Computational Experimentation

In general, solution quality has been emphasized at the expense of running time and recent papers use real distances between customers. Tabu search procedures perform well, but involve a large number of parameters. For the most part, the values of parameters are adjusted in an ad hoc way. Some tabu search procedures have been parallelized.

Computational Reporting

There are two ways to report results. Some papers report only the best results, while other papers report single-pass results as well as the best results. The single-pass results are generated with a single set of parameter values. In some papers, during the course of fine-tuning an algorithm, parameters are varied over a range of

Table 2.11 Procedure to fine-tune a metaheuristic developed by Xu, Chiu, and Glover (1996).

Fine-tuning a Tabu Search Algorithm

Xu, Chiu, and Glover (1996) develop a tabu search algorithm for solving the Steiner Tree-Star problem. Using statistical tests, they improve the performance of their algorithm by finding more appropriate values for five key factors. The key factors are parameters or options whose values need to be determined. Their procedure for fine-tuning the tabu search algorithm is described below.

Fine-tuning procedure

Order the five factors by importance.
1 Tabu tenures 2 Base probability for move selection 3 Fine-tuned probability for move selection
4 Recovery strategy 5 Frequency of activating swap moves

Start with first factor: Tabu tenures.
18 Candidate Settings → Execute on 19 problems → Use nonparametric tests to eliminate inferior settings
(18 runs) (Friedman test, Wilcoxon signed rank test)

→ Two runs survive for future consideration

Go to second factor: Base probability for move selection.
6 Candidate Settings → Execute on 19 problems → Use nonparametric tests to eliminate inferior settings
(6 runs for each survivor) (Friedman test, Wilcoxon signed rank test)

→ One run survives for future consideration

...
Go to fifth factor: Frequency of activating swap moves.

Authors examine only 44 runs out of 9,270 possible runs and conclude
"... Run 33, the fine-tuned algorithm, significantly improves upon
the base algorithm ... it yields improved solutions in fourteen cases ..."

Table 2.12 Projected running times for large-scale VRPs.

		Actual Total Time (mins)	
Pr	*n*	*GS94**	*XK96***
1	50	6.0	29.92
2	75	53.8	48.80
3	100	18.4	71.93
4	150	58.8	149.90
5	199	90.9	272.52
11	120	22.2	91.23
12	100	16.0	56.61

* Silicon Graphics Workstation (36 MHz, 5.7 Mflops)
** DEC ALPHA Workstation (DEC OSF/1 v3.0)
roughly five times as fast as an SG Workstation

GS94
Estimated Total Time = $0.001784\ n^{2.025}$
$R^2 = 0.95$ (with second point removed)

XK96
Estimated Total Time = $0.049688\ n^{1.591}$
$R^2 = 0.95$

	Projected Total Time (in minutes)	
n	*GS94*	*XK96*
250	128.0	324.6
300	185.2	433.9
350	253.0	554.5
400	331.6	685.7
450	420.9	827.0
500	521.0	977.9

values and the best results are recorded. Some papers include the routes of the best solutions.

There are four ways to report the running time of an algorithm that we have come across in the literature: (1) time to obtain the best solution, (2) time to obtain a solution within 1% or 5% of the best-known solution, (3) total computation time, and (4) no running time.

Table 2.13 Characteristics of the large-scale vehicle routing problems.

Test Problem	*Number of Customers*	*Vehicle Capacity*	*Max. Route Length*
1	240	550	650
2	320	700	900
3	400	900	1200
4	480	1000	1600
5	200	900	1800
6	280	900	1500
7	360	900	1300
8	440	900	1200
9	255	1000	∞
10	323	1000	∞
11	399	1000	∞
12	483	1000	∞
13	252	1000	∞
14	320	1000	∞
15	396	1000	∞
16	480	1000	∞
17	240	200	∞
18	300	200	∞
19	360	200	∞
20	420	200	∞

2.4 LARGE-SCALE VEHICLE ROUTING PROBLEMS

In soft drink and beer distribution, the average number of customers visited daily is approximately 600. In sanitation (node) routing, the number of sites visited daily is often between 200 and 1,000. Many vehicle routing problems are large-scale and, in practice, we have observed that most real-world applications of vehicle routing software involve at least 200 stops per day. This is due, in large part, to the fact that the cost of running vehicle routing software is high. In addition to the software itself, data (e.g., the street database and the need to address match customers), consulting and customization (e.g., special constraints and the need to interface the software with the billing system), and internal project management are expensive. The total implementation cost might range from three to six times the cost of the vehicle routing software. If a savings of approximately 5% is expected, then the number of trucks and, therefore, the number of stops per day need to be large enough to justify the purchase of vehicle routing software.

With such a large number of stops, heuristics remain the only practical option to solve these VRPs, yet there are few large-scale problems on which to test the performance of heuristics. In addition, it is not clear that the tabu search heuristics compared in Tables 2.6, 2.7, and 2.8 are fast enough to solve large-scale problems. To begin to explore this issue, we used regression analysis to project running times for GS94 and XK96 on large-scale VRPs. In particular, we built regression models based

Table 2.14 Record-to-record heuristic for the large-scale vehicle routing problem.

Step 1. Initialization
Apply the Clarke and Wright algorithm
Obtain an initial solution
Step 2. One-point Movement
Make feasible one-point moves
Use record-to-record travel (uphill moves are allowed)
Step 3. Two-point Exchange
Exchange points on different routes
Feasibility must be maintained
Use record-to-record travel (uphill moves are allowed)
Step 4. Clean-up
Apply one-point movement (only downhill moves)
Apply two-point exchange (only downhill moves)
Apply 2-opt to each route (only downhill moves)
Step 5. Local Reinitialization
Repeat R1 times
Resequence individual routes
Go to Step 2
Step 6. Global Reinitialization
Repeat R2 times
Perturb the current best solution
Go to Step 2

on Table 2.8 and projected total running time for n = 250, 300, 350, 400, 450, and 500. The results are presented in Table 2.12. Although the projections are rough, they clearly indicate the need for new, more computationally efficient metaheuristics.

2.4.1 New Test Problems and Results

In this section, we generate a new set of 20 large-scale vehicle routing problems (LSVRPs), apply a tabu search heuristic and a heuristic based on record-to-record travel to each problem, and then compare the heuristic solutions to an estimated solution and a solution produced by the Clarke and Wright algorithm.

The characteristics of the LSVRPs are shown in Table 2.13. The problems have 200 to 483 customers. Eight problems have route-length restrictions. Problems 1 to 8 have customers located in concentric circles around the depot. Problems 9 to 12

Table 2.15 Solution values to 20 LSVRPs.

Pr	*n*	*Estimated Solution*	*C & W*	*XK*	*Running Time**	*RTR*	*Running Time***
1	240	5859.62	5974.23	**5646.46**	802.87	5834.60	3.68
2	320	**8566.04**	9257.23	8570.28	898.53	9002.26	22.66
3	400	**11649.06**	12282.90	11880.37	1749.27	11879.95	40.04
4	480	15108.68	16054.22	15250.78	2432.42	**14639.32**	122.61
5	200	8631.64	7230.47	7361.29	591.40	**6702.73**	11.24
6	280	9843.01	9372.33	9088.66	913.70	**9016.93**	18.79
7	360	**11047.69**	11901.54	11411.85	1062.73	11213.31	22.55
8	440	**12250.06**	12982.65	12825.00	1586.20	12514.20	111.37
9	255	678.82	663.57	589.10	340.20	**587.09**	23.01
10	323	865.50	838.92	**746.56**	501.82	749.15	31.49
11	399	1074.80	1052.12	**932.68**	852.72	934.33	69.19
12	483	1306.73	1270.99	1140.72	1151.10	**1137.18**	101.09
13	252	956.04	952.74	881.07	1465.77	**881.04**	6.01
14	320	1220.27	1221.69	1118.09	1577.30	**1103.69**	21.83
15	396	1516.48	1512.66	1377.79	4340.07	**1364.23**	32.62
16	480	1844.67	1774.68	**1656.66**	8943.45	1657.93	47.55
17	240	796.13	771.70	**666.84**	2314.00	720.44	5.69
18	300	1133.25	1069.29	**973.60**	4101.02	1029.21	8.15
19	360	1554.64	1467.16	**1338.74**	5718.38	1403.05	12.42
20	420	2081.38	1963.38	**1831.62**	10839.73	1875.17	31.05

Bold Best-known solution value
* Minutes on a DEC ALPHA Workstation
** Minutes on a 100 MHz Pentium-based PC

Solution Procedures
C&W Clarke and Wright algorithm
RTR Record-to-record travel algorithm
XK Tabu search algorithm of Xu and Kelly

have customers located in concentric squares with the depot located in one corner. Problems 13 to 18 have customers located in concentric squares around the depot. The data for all 20 problems are available at
http://www-Bus.colorado.edu/Publications/workingpapers/kelly.

We now apply three heuristics to each problem: a Clarke and Wright algorithm, the tabu search algorithm of Xu and Kelly (1996), and a variant of record-to-record travel. We also generate an estimated solution to each LSVRP. The tabu search algorithm is adapted from Xu and Kelly (1996), except that the TSTSP function was disabled for problems 4 and 8 since memory problems were encountered. The record-to-record travel algorithm is adapted from Chao, Golden, and Wasil (1993) and sketched in Table 2.14. The solutions to the 20 new test problems and the running times are

shown in Table 2.15. The routes of the best solution to each problem are available at http://www-Bus.colorado.edu/Publications/workingpapers/kelly.

In examining Table 2.15, we see that XK produces the best solution to eight problems and RTR produces the best solution to eight problems. The estimated solutions are the best solutions to four problems. We point out that there is wide difference in running times between XK and RTR. The running times for RTR range from approximately 4 minutes to 123 minutes, while the running times for XK range from approximately 341 minutes to a staggering 10,840 minutes.

2.5 FUTURE DIRECTIONS IN VEHICLE ROUTING RESEARCH

For the most part, over the last 20 years or so, VRP heuristics have been tested on the same 14 benchmark problems. Meanwhile, computers have become much faster, heuristics have become more complex, and the number of parameters that need to be set has increased significantly. In fact, some current heuristics suffer from inelegant design: they have more parameters than the number of benchmark problems! Furthermore, we suspect that heuristics are being overfit on the benchmark problems.

In the future, the goal should be to design VRP heuristics that are capable of producing high-quality, near-optimal solutions. Heuristics should be lean and parsimonious and contain few parameters. They should be computationally efficient, robust, and simple.

Authors should explicitly list and define parameters, so as to facilitate reproducibility. The parameter values should be determined in a systematic way (e.g., via experimental design) and a single set of parameter values should be obtained. Alternatively, a heuristic might search over a set of internal (hidden) parameters to find the best solution for each problem.

With the new, larger VRPs, there are now 34 benchmark problems and the largest of these has nearly 500 customers. Borrowing from the neural network literature, a subset of 20 or so benchmark problems can be used for "training." The resulting heuristic can then be applied to the 14 or so remaining benchmark problems for "testing." The total computation time and the solution (routes) should be reported for each problem.

Over the next five years, we believe that VRP metaheuristics will continue to grow in use and in computational power. Researchers will need to be increasingly attentive to issues of elegance and parsimony and to comprehensive testing and reporting when developing new algorithms. We hope that our large-scale problems and our suggestions for producing a sound computational design will help researchers in developing new, high-quality VRP metaheuristics. The outlook is bright for incorporating them into real-world routing software.

Acknowledgments

We thank Jiefeng Xu for running the network flow-based tabu search heuristic of Xu and Kelly on the 14 benchmark problems and the 20 LSVRPs. We also thank Balasubramanian Rangaswamy for his skillful help in producing this paper.

References

Alfa, A., S. Heragu and M. Chen. (1991). A 3-opt Based Simulated Annealing Algorithm for Vehicle Routing Problems. *Computers & Industrial Engineering*, 21(1–4), 635–639.

Ball, M., T. Magnanti, C. Monma and G. Nemhauser (editors). (1995). *Network Routing*. Handbooks in Operations Research and Management Science, Volume 8, Elsevier, Amsterdam, The Netherlands.

Beasley, J. (1990). OR-Library: Distributing Test Problems by Electronic Mail, *Journal of the Operational Research Society*, 41(11), 1069–1072.

Bodin, L., B. Golden, A. Assad and M. Ball. (1983). Routing and Scheduling of Vehicles and Crews. *Computers & Operations Research*, 10 (2), 63–211.

Chao, I-M., B. Golden and E. Wasil. (1993). A New Heuristic for the Multi-depot Vehicle Routing Problem that Improves Upon Best-known Solutions. *American Journal of Mathematical and Management Sciences*, 13(3 & 4), 371–406.

Chao, I-M., B. Golden and E. Wasil. (1995). An Improved Heuristic for the Period Vehicle Routing Problem. *Networks*, 26, 25–44.

Christofides, N. and S. Eilon. (1969). An Algorithm for the Vehicle Dispatching Problem. *Operational Research Quarterly*, 20, 309–318.

Christofides, N., A. Mingozzi and P. Toth. (1979). The Vehicle Routing Problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pages 315-338, John Wiley & Sons, Chichester, UK.

Clarke, G. and J. Wright. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12, 568–581.

Cullen, F., J. Jarvis and D. Ratliff. (1981). Set Partitioning Based Heuristics for Interactive Routing. *Networks*, 11, 125–144.

Dantzig, G. and J. Ramser. (1959). The Truck Dispatching Problem. *Management Science*, 6, 81–91.

Dueck, G. (1993). New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, 104, 86–92.

Dueck, G. and T. Scheurer. (1990). Threshold Accepting: A General Purpose Optimization Algorithm. *Journal of Computational Physics*, 90, 161–175.

Fisher, M. (1994). Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees. *Operations Research*, 42, 626–642.

Fisher, M. (1995). Vehicle Routing. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Routing*, Handbooks in Operations Research and Management Science, Volume 8, pages 1–33, Elsevier, Amsterdam, The Netherlands.

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

Gendreau, M., A. Hertz and G. Laporte. (1991). A Tabu Search Heuristic for the Vehicle Routing Problem. *CRT-777*, Centre de Recherche sur les Transports, Université de Montréal, Montréal, Canada.

Gendreau, M., A. Hertz and G. Laporte. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40, 1276–1290.

Gendreau, M., G. Laporte and J-Y. Potvin. (1997). Vehicle Routing: Modern Heuristics. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 311-336, John Wiley & Sons Ltd., London, England.

Ghaziri, H. (1996). Supervision in the Self-organizing Feature Map: Application to the Vehicle Routing Problem, In I. Osman and J. Kelly, editors, *Meta-heuristics: Theory and Applications*, pages 651-660, Kluwer Academic Publishers, Boston, Massachusetts.

Gillett, B. and L. Miller. (1974). A Heuristic Algorithm for the Vehicle Dispatch Problem. *Operations Research*, 22, 340–349.

Glover, F. (1997). Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges, In R. Barr, R. Helgason and J. Kennington, editors, *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*, pages 1–75, Kluwer Academic Publishers, Boston, Massachusetts.

Golden, B. and A. Assad (editors). (1988). *Vehicle Routing: Methods and Studies.* Studies in Management Science and Systems, Volume 16, North-Holland, Amsterdam, The Netherlands.

Golden, B., T. Magnanti and H. Nguyen. (1997). Implementing Vehicle Routing Algorithms. *Networks*, 7, 113–148.

Hall, R. and J. Partyka. (1997). On the Road to Efficiency. *OR/MS Today*, 24(3), 38–47.

Laporte, G. (1992). The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research*, 59, 345–358.

Osman, I. (1993). Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. *Annals of Operations Research*, 41, 421–451.

Osman, I. and J. Kelly. (1996). Meta-heuristics: An Overview. In I. Osman and J. Kelly, editors, *Meta-heuristics: Theory and Applications*, pages 1-21, Kluwer Academic Publishers, Boston, Massachusetts.

Reeves, C. (editor). (1993). *Modern Heuristic Techniques for Combinatorial Problems.* Halsted Press, New York.

Rego, C. and C. Roucairol. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, in *Meta-heuristics: Theory and Applications*, edited by I. Osman and J. Kelly, 661–675, Kluwer Academic Publishers, Boston, Massachusetts.

Robertson, S., B. Golden, G. Runger and E. Wasil. (1998). Neural Network Models for Initial Public Offerings. forthcoming in *Neurocomputing.*

Robusté, F., C. Daganzo and R. Souleyrette II. (1990). Implementing Vehicle Routing Models. *Transportation Research*, 24B(4), 263–286.

Rochat, Y. and E. Taillard. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1, 147–167.

Russell, R. (1977). An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Conditions. *Operations Research*, 25, 517–524.

Semet, F. and E. Taillard. (1993). Solving Real-life Vehicle Routing Problems Efficiently Using Tabu Search. *Annals of Operations Research*, 41, 469–488.

Taillard, E. (1992). Parallel Iterative Search Methods for Vehicle Routing Problem. *ORWP 92/03*, École Polytechnique Fédérale de Lausanne, Département de Mathématiques, CH-1015, Lausanne, Switzerland.

Taillard, E. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23, 661–673.

Van Breedam, A. (1994). An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selection of Problems with Vehicle-related, Customer-related, and Time-related Constraints. Ph.D. Dissertation, Faculty of Applied Economics, University of Antwerp – RUCA, Antwerp, Belgium.

Van Breedam, A. (1995). Improvement Heuristics for the Vehicle Routing Problem Based on Simulated Annealing. *European Journal of Operational Research*, 86, 480–490.

Van Breedam, A. (1996). An Analysis of the Effect of Local Improvement Operators in Genetic Algorithms and Simulated Annealing for the Vehicle Routing Problem. *RUCA Working Paper 96/14*, Faculty of Applied Economics, University of Antwerp, Antwerp, Belgium.

Xu, J., S. Chiu and F. Glover. (1996). Fine-tuning a Tabu Search Algorithm with Statistical Tests. Working Paper, Graduate School of Business, University of Colorado, Boulder, Colorado.

Xu, J. and J. Kelly. (1996). A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. *Transportation Science*, 30, 379–393.

Xu, J. and J. Kelly. (1997). Personal communication.

3 A UNIFIED FRAMEWORK FOR DETERMINISTIC TIME CONSTRAINED VEHICLE ROUTING AND CREW SCHEDULING PROBLEMS

Guy Desaulniers, Jacques Desrosiers
Irina Ioachim, Marius M. Solomon
François Soumis, Daniel Villeneuve

3.1 INTRODUCTION

Time constrained routing and scheduling is of significant importance across land, air and water transportation. These problems are also encountered in a variety of manufacturing, warehousing and service sector environments. Their mathematical complexity and the magnitude of the potential cost savings to be achieved by utilizing O.R. methodologies have attracted researchers since the early days of the field. Witness to this are the pioneering efforts of Dantzig and Fulkerson (1954), Ford and Fulkerson (1962), Appelgren (1969, 1971), Levin (1971), Madsen (1976) and Orloff (1976). Much of the methodology developed has made extensive use of network models and algorithms.

The eighties mark a turning point in this field, as important new developments have lead to the solution of more general and realistic problems. These advances were captured in a number of extensive and insightful surveys. Beginning with Magnanti (1981), several authors including Bodin *et al.* (1983), Carraresi and Gallo (1984), Solomon and Desrosiers (1988), Desrochers *et al.* (1988), Laporte (1992) and Fisher (1995) have provided timely reviews of the field. Two areas that have greatly benefited from these developments are urban mass transit and the airline industry. The proceedings of the workshops on computer-aided transit scheduling (see, for example,

Desrochers and Rousseau, 1992; and Daduna, Branco and Paixão, 1995), those of the AGIFORS meetings, and the surveys conducted by Etschmaier and Mathaisel (1984, 1985) highlight the spectacular progress made in these contexts.

These achievements have fueled an even stronger interest in vehicle fleet planning and crew scheduling problems in the nineties. This has been reflected by a large number of articles dedicated to the subject. While most of them have been covered in the recent comprehensive survey conducted by Desrosiers *et al.* (1995), several notable advances have since been reported in the airline context. These concern fleet applications - Hane *et al.* (1995), Desaulniers *et al.* (1997b) and Gu *et al.* (1994); crew applications - Barnhart, Hatay and Johnson (1995), Barnhart *et al.* (1994a), Hoffman and Padberg (1993), Graves *et al.* (1993), Wedelin (1995) and Desaulniers *et al.* (1997a); rostering applications - Gamache and Soumis (1998), Gamache *et al.* (1998a), Gamache *et al.* (1998b) and Ryan (1992); and day-to-day operations - Stojković, Soumis and Desrosiers (1998). Additional very recent developments will be presented in later sections.

The need for an integrating framework for vehicle routing and crew scheduling problems has been apparent for some time. Several attempts have been made in this direction. The first descriptive taxonomy has been proposed by Bodin and Golden (1981). Later, Desrochers, Lenstra and Savelsbergh (1990) have suggested a formal classification scheme. Recently, Desrosiers *et al.* (1995) have introduced the first general model for vehicle routing and crew scheduling problems. While it encompasses a variety of real-world environments, it stops short of modeling several important practical complexities.

This paper provides important contributions on both the modeling and algorithmic dimensions. On the modeling level, it presents a more general model than Desrosiers *et al.* (1995) which integrates all the different time constrained vehicle routing and crew scheduling problem types examined so far. In particular, it includes all the deterministic problems classified by Desrochers, Lenstra and Savelsbergh (1990). The model extends well-known generic formulations to allow the modeling of all real-world circumstances encountered to date in this environment. In addition, it enables the reader to understand the common structure of time constrained problems. It also allows to perceive the relations between the various problems, the different forms of the model used previously in the literature, and assorted applications across a unified formulation. This finally permits the reader to remark the diversity of specialized algorithms which have been designed to solve these, and especially to comprehend the difficulties inherent in certain modeling aspects.

At the algorithmic level, the paper presents a branch-and-bound framework to solve the nonlinear multi-commodity network flow models in this class. It shows that a variety of strategies and algorithms can be utilized for the computation of lower bounds and for devising branching schemes. The lower bounds are derived by using a decomposition approach. We focus on an extension of the Dantzig-Wolfe decomposition principle and establish that this is valid even for nonlinear objective functions and constraints. We also illustrate that it embeds, as special cases, the column generation-based methods using Set Partitioning formulations previously suggested in the literature. The branching module used to obtain integer solutions compatible with column generation is more general, but yet simpler than other prior strategies. Finally, we examine the constrained Shortest Path Problems that appear

at the subproblem level of the decomposition. The paper permits to see the variety of specialized dynamic programming algorithms that have been developed to solve these and more general single commodity problems and some of the aspects which have not yet received attention.

3.2 RELATIONS BETWEEN TIME CONSTRAINED PROBLEMS

The formulations presented in this section model various types of vehicle routing and crew scheduling problems. They all have an underlying time-space network structure. The tasks to be covered are associated with nodes, which in turn represent customers to service, sites to visit or activities to perform (e.g., flight legs). The arcs represent inter-task activities such as traveling and deadheading. Starting from the Shortest Path Problem with time windows, we gradually add practically important constraints to model more complex problems such as the Vehicle Routing Problem with Time Windows (VRPTW) and its extensions involving pickup and delivery requests, precedence constraints and nonlinear objective functions. This hierarchical model construction establishes clear relations among the various time constrained routing and scheduling problems. It also leads to a unified formulation given in Section 3.3.

3.2.1 The VRPTW and Related Problems

Let $G = (V, A)$ be a directed graph, where V is the node set and A is the arc set. A time window $[a_i, b_i]$, $i \in V$ is associated with each node. Each arc $(i, j) \in A$ is characterized by a real cost c_{ij} and a positive duration t_{ij}, and it satisfies the feasibility condition $a_i + t_{ij} \leq b_j$. We assume that the service time at node i is included in the time value t_{ij} and that waiting is allowed before the start of a time window. The set of nodes V is composed of $N \cup \{o, d\}$, where N consists of nodes that can be visited on paths originating at the source node o and finishing at the sink node d.

Shortest Path Problems with Time Windows: They constitute the backbone of the problem structures to be discussed as they are embedded in all time window constrained routing and scheduling problems. We begin by providing a nonlinear formulation for the Elementary Shortest Path Problem with Time Windows (ESPTW). Recall that an elementary path is a path without cycles. This problem involves binary flow variables X_{ij}, $(i, j) \in A$ which are equal to 1 for the arcs in the solution path and 0 otherwise, and time variables T_i, $i \in V$ which represent the start of service at node i. Formally, ESPTW seeks to

$$Minimize \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{3.1}$$

subject to:

$$\sum_{j:(o,j)\in A} X_{oj} = \sum_{i:(i,d)\in A} X_{id} = 1\,, \tag{3.2}$$

$$\sum_{j:(i,j)\in A} X_{ij} - \sum_{j:(j,i)\in A} X_{ji} = 0\,, \qquad \forall\, i \in N \tag{3.3}$$

$$X_{ij}(T_i + t_{ij} - T_j) \leq 0\,, \qquad \forall\,(i,j) \in A \tag{3.4}$$

$$a_i \leq T_i \leq b_i\,, \qquad \forall\, i \in V \tag{3.5}$$

$$X_{ij} \ \textit{binary}, \qquad \forall\,(i,j) \in A\,. \tag{3.6}$$

The objective function (3.1) minimizes the sum of the arc costs. Constraints (3.2)–(3.3) define the classical network flow constraints for a path originating at source node o and ending at sink node d. The nonlinear constraints (3.4) express the compatibility requirements between flow and time variables. Their structure is very close to the usual complementary slackness conditions for linear programs. Finally, the time window constraints are given by (3.5), while constraints (3.6) impose binary values for the flow variables.

The ESPTW is NP-hard as it includes the binary and the classical Knapsack Problems as special cases. Negative cost cycles render the ESPTW NP-hard in the strong sense (Dror, 1994). Hence, the literature has focused on its relaxation, the Shortest Path Problem with Time Windows (SPTW), which allows for multiple visits at the same node. Even though a node may be visited several times, the time window constraints and the positive arc durations t_{ij} guarantee the finiteness of paths. Indeed, as any cycle will have a positive duration, the network can be transformed into an acyclic time-oriented network by creating multiple copies of each original node that have disjoint time windows. Hence, a path in the new network may visit several copies of an original node, but at different times. The node copies are formally described in Section 3.2.2. Therefore, the SPTW is an ESPTW on an acyclic underlying network, as is the case for Knapsack Problems. Several efficient pseudo-polynomial dynamic programming algorithms have been proposed to solve this NP-hard problem in Desrosiers, Pelletier and Soumis (1983) and in Desrochers and Soumis (1988a, 1988b). These algorithms can be accelerated through data preprocessing (i.e., *a priori* time window reduction) as described in Desrosiers *et al.* (1995). When $a_i = b_i$ for all $i \in V$, both the ESPTW and the SPTW reduce to a classical Shortest Path Problem on an acyclic graph, denoted SPA, which can be solved in polynomial time (Fulkerson, 1972) for any real arc cost values.

We consider next the inclusion of vehicle capacity constraints in the ESPTW and the SPTW and denote the resulting Shortest Path Problems with time windows and capacity restrictions by ESPTWQ and SPTWQ, respectively. In order to formulate the vehicle capacity constraints, let Q denote the capacity, l_i, $i \in V$ the demand at node i and L_i, $i \in V$ the load variables representing the accumulated vehicle load at node i. Without loss of generality, let $l_o = l_d = 0$. Given that the arcs (i,j) with $l_i + l_j > Q$ have been removed from the arc set A, the additional capacity constraints can be written as:

$$X_{ij}(L_i + l_j - L_j) \leq 0\,, \qquad \forall\,(i,j) \in A\,, \tag{3.7}$$

$$l_i \leq L_i \leq Q\,, \qquad \forall\, i \in V\,. \tag{3.8}$$

Even though relations (3.7) can be written as equalities, we present them in inequality format to emphasize their similarity to relations (3.4). As previously, the elementary path version ESPTWQ is NP-hard in the strong sense, while a pseudo-polynomial dynamic programming algorithm for SPTWQ is described in Desrochers (1986).

The Traveling Salesman Problem with Time Windows (TSPTW): The ESPTW is a relaxation of the TSPTW. Indeed, by imposing the additional constraints:

$$\sum_{j:(i,j)\in A} X_{ij} = 1\,,\ \forall\, i \in N \tag{3.9}$$

on the ESPTW model (3.1)–(3.6), the resulting model is a TSPTW. It represents finding a least cost path satisfying the time windows and visiting each node of the network exactly once. Solution techniques based on dynamic programming have been proposed for this type of problem by Baker (1983), Mingozzi, Bianco and Ricciardelli (1993), Christofides, Mingozzi and Toth (1981), Dumas *et al.* (1995) and Psaraftis (1980, 1983). These have performed well whenever the time windows were relatively narrow. An easy case of this problem is obtained when the width of the time windows is bounded by a constant value which is independent of the problem size; for this case, Gélinas and Soumis (1998) has proposed a dynamic programming algorithm of polynomial complexity. Additionally, if all time windows reduce to a fixed time-tabled schedule, i.e., $a_i = b_i$ for all $i \in V$, and if the problem is feasible, then the unique solution path simply visits all nodes in increasing time order. Finally, in the presence of capacity constraints, there is no need to introduce a TSP type model since the capacity requirements will be satisfied if $\sum_{i\in V} l_i \leq Q$ and this can be verified *a priori*.

The Vehicle Routing Problem with Time Windows (VRPTW): The TSPTW formulation given by (3.1)–(3.6) and (3.9) can be extended to the Multiple Traveling Salesman Problem with Time Windows, usually denoted m-TSPTW. The aim of the m-TSPTW is to cover a set of tasks exactly once using at most m minimum cost itineraries. To formulate this problem, constraints (3.2) are replaced by:

$$\sum_{j:(o,j)\in A} X_{oj} = \sum_{i:(i,d)\in A} X_{id} \leq m\,. \tag{3.10}$$

The single depot, homogeneous vehicle fleet VRPTW model is further obtained by adding the capacity constraints (3.7)–(3.8) to this m-TSPTW formulation (see Kolen, Rinnooy Kan and Trienekens, 1987; Desrochers, Desrosiers and Solomon, 1992; Fisher, Jörnsten and Madsen, 1997; Kohl and Madsen, 1997).

For the multiple-depot, heterogeneous vehicle fleet version of the VRPTW, we present a multi-commodity type model. To account for vehicles of different types and different depot locations, each vehicle defines a specific commodity. Let K be the set of vehicles, with $|K| = m$, and associate with each of them a graph $G^k = (V^k, A^k)$. Let $V^k = N^k \cup \{o(k),\, d(k)\}$, where $o(k)$ and $d(k)$ denote the source and sink depots corresponding to the initial and final locations of vehicle k, and N^k its restricted set of nodes such that $N = \cup_{k\in K} N^k$. Assume next that arcs $(o(k), d(k))$ with zero cost and duration are included in A^k to account for unused vehicles. Finally, to consider the individual commodities, all variables and parameters of the previous formulation are indexed by k, except for the node demand parameters l_i, for $i \in V^k$, $k \in K$. Then, the formulation is as follows:

$$Minimize \sum_{k\in K} \sum_{(i,j)\in A^k} c_{ij}^k X_{ij}^k \tag{3.11}$$

subject to:

$$\sum_{k \in K} \sum_{j:(i,j) \in A^k} X_{ij}^k = 1\,, \qquad \forall\, i \in N \tag{3.12}$$

$$\sum_{j:(o(k),j) \in A^k} X_{o(k),j}^k = \sum_{i:(i,d(k)) \in A^k} X_{i,d(k)}^k = 1\,, \qquad \forall\, k \in K \tag{3.13}$$

$$\sum_{j:(i,j) \in A^k} X_{ij}^k - \sum_{j:(j,i) \in A^k} X_{ji}^k = 0\,, \qquad \forall\, k \in K,\ \forall\, i \in N^k \tag{3.14}$$

$$X_{ij}^k (T_i^k + t_{ij}^k - T_j^k) \leq 0\,, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{3.15}$$

$$a_i^k \leq T_i^k \leq b_i^k\,, \qquad \forall\, k \in K,\ \forall\, i \in V^k \tag{3.16}$$

$$X_{ij}^k (L_i^k + l_j - L_j^k) \leq 0\,, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{3.17}$$

$$l_i \leq L_i^k \leq Q^k\,, \qquad \forall\, k \in K,\ \forall\, i \in V^k \tag{3.18}$$

$$X_{ij}^k \ \textit{binary}, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k\,. \tag{3.19}$$

Constraints (3.13)–(3.19) are separable by vehicle and define a SPTWQ structure for each index $k \in K$. Linking constraints (3.12) ensure that each node is visited exactly once, and hence assigned to a single vehicle, while the objective function (3.11) minimizes the total arc cost. Note that if several vehicles of the same type are located at the same depot, it is more convenient to use a single commodity for all these vehicles. Nevertheless, the above formulation is general enough for practical applications requiring specific initial and final conditions for each vehicle, situations that may occur in an operating rather than a planning mode (for example, see Stojković, Soumis and Desrosiers, 1998).

Other formulations can be given for the VRPTW. For example, Fisher (1994) exploits a tree structure for the classical Vehicle Routing Problem and extends it to time constrained routing problems. The capacity, the time windows and the constraints requiring that each customer be visited once are dualized to obtain a Lagrangean problem that provides lower bounds in a branch-and-bound algorithm.

Consider now some special cases of formulation (3.11)–(3.19). First, the multiple-depot, heterogeneous vehicle fleet m-TSPTW is obtained by eliminating the capacity constraints (3.17)–(3.18). Second, the Multiple Depot Vehicle Scheduling Problem (see Fischetti and Toth, 1989; Ribeiro and Soumis, 1994), denoted by MDVSP, where the vehicles have to be assigned to a set of fixed time-tabled tasks, is obtained by removing the time and capacity constraints (3.15)–(3.18). Even though the resulting commodity networks are acyclic for all $k \in K$ and a SPA structure is embedded into the MDVSP formulation, the MDVSP remains an NP-hard problem (Carraresi and Gallo, 1984). However, its corresponding single depot version, namely the SDVSP, turns out to be a classical transportation problem solvable in polynomial time. For a specialized quasi-assignment algorithm, see also Paixão and Branco (1987). The simplified MDVSP version requiring only the minimization of the number of routed vehicles reduces to the SDVSP since the costs are independent of the depot locations. For a complexity analysis of these types of models in the context of aircraft fleet assignment problems, the reader is referred to the work of Gu *et al.* (1994).

3.2.2 *Extensions*

In this section we present five extensions to the previous formulations which model additional problem features. Time and capacity are often referred to as constrained resources (see Desrosiers *et al.*, 1995) and the first extension concerns models which require more than two resources. The second introduces new types of single path constraints, while the third is used to model multiple path linking constraints in terms of flow and resource variables. The fourth extension concerns new forms of the objective function while soft time windows are examined in the last one.

Multiple Resource Problems: Several resources, other than time and capacity, may be specified for a given problem. For example, crew scheduling problems in urban transit, rail or airline contexts involve many other resource restrictions stemming from the respective collective agreement specifications. Let R^k represent the resource set for commodity k, $k \in K$. Several parameters and problem variables are resource dependent. Thus, let $T_i^{kr} \in [a_i^{kr}, b_i^{kr}]$ denote the resource variable specifying the value of resource $r \in R^k$ accumulated at node $i \in V^k$ on commodity network k and let t_{ij}^{kr} be the consumption of that resource on the arc $(i,j) \in A^k$. When replacing the time and capacity constraints (3.15)–(3.18) by the set of resource constraints:

$$X_{ij}^k (T_i^{kr} + t_{ij}^{kr} - T_j^{kr}) \leq 0\,, \qquad \forall\, k \in K,\ \forall\, r \in R^k,\ \forall\, (i,j) \in A^k \tag{3.20}$$

$$a_i^{kr} \leq T_i^{kr} \leq b_i^{kr}\,, \qquad \forall\, k \in K,\ \forall\, r \in R^k,\ \forall\, i \in V^k, \tag{3.21}$$

we obtain a formulation of the Multiple Traveling Salesman Problem with Resource Constraints (m-TSPR). For $m = 1$, one can easily derive the Traveling Salesman Problem with Resource Constraints (TSPR). Similarly, the resource constrained Shortest Path Problems ESPR and SPR can be obtained as extensions of the ESPTWQ and SPTWQ. The SPR is NP-hard and the pseudo-polynomial dynamic programming algorithm described in Desrochers (1986) can be used for its solution. Note that in most crew scheduling applications, the underlying networks G^k are acyclic, since the schedules are requested for fixed time-tabled tasks. In such cases, the ESPR and SPR are equivalent as the solution of the latter problem directly provides an elementary path.

The relations between all previous problem types are illustrated in Figure 3.1. The first row depicts time constrained Shortest Path Problems. The second row contains only two single-vehicle TSP type problems. Finally, the last two rows illustrate multi-vehicle traveling salesman problems with side constraints involving single/multiple depots and a homogeneous/ heterogeneous vehicle fleet. Acronyms in italics denote problems solvable in polynomial time (the SPA and SDVSP), while the problems ESPTW, ESPTWQ, ESPR, TSPTW, TSPR, MDVSP, m-TSPTW, VRPTW and m-TSPR are NP-hard in the strong sense. Pseudo-polynomial algorithms have been developed for three NP-hard time constrained Shortest Path Problems on acyclic graphs, namely the SPTW, SPTWQ and SPR, which are written in parenthesis in Figure 3.1. Arrows from left to right and from top to bottom portray an increasing set of constraints.

Single Path Constraints: The resource constraints (3.20)–(3.21) are called single path constraints because they are related to the feasibility of a single path, i.e., to a single commodity $k \in K$. Other types of single path constraints have been utilized in

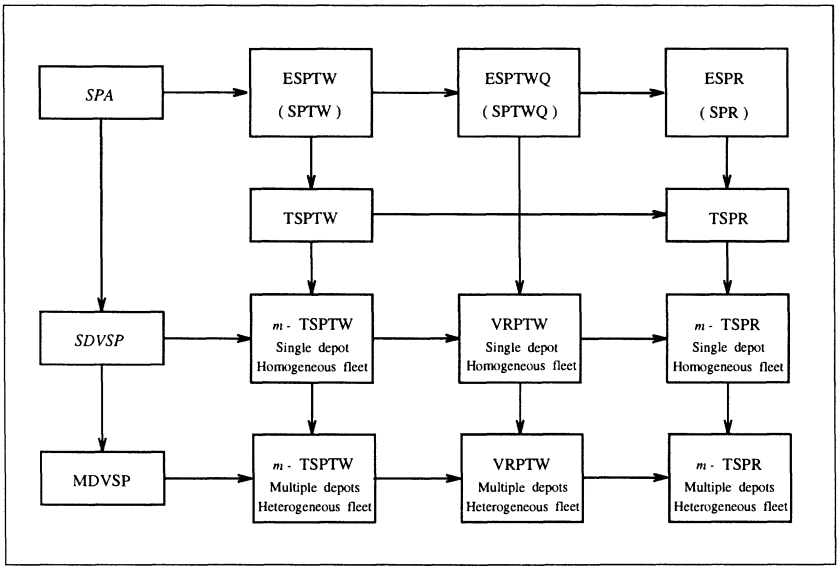


Figure 3.1 Relations between Time Constrained Problems

the literature and we chose two important examples to discuss next. The first arises in pickup and delivery problems. The second is introduced to partially, or completely, eliminate cycles in non-elementary shortest path problems. Finally, drawing from these examples, we propose a general form of such single path constraints.

Consider the following pickup and delivery problem for which a set of transportation requests has been specified in advance. Each transportation request has two associated nodes, one for the pickup and the other for the delivery. Without loss of generality, we can assume that pickup and delivery nodes are different from the depot nodes. Let $N = N^P \cup N^D$, where N^P denotes the set of pickup nodes and N^D denotes the set of delivery nodes, with $|N^P| = |N^D| = n$. Next, for $i \in N^P$, let $n + i \in N^D$ denote the associated delivery node. Time windows and demands are specified for all the nodes. Pickup nodes have positive demands l_i, delivery nodes have negative demands $l_{n+i} = -l_i$, $i \in N^P$ and, without loss of generality, we assume that $l_o = l_d = 0$. Vehicles have limited capacities, Q^k, $k \in K$. Finally, let $N^k = N^{Pk} \cup N^{Dk}$ and let $V^k = N^k \cup \{o(k), d(k)\}$ be the node set of vehicle (i.e. commodity) k.

The aim of the problem is to find a minimum cost assignment for the vehicles in set K which respects the time windows, the vehicle capacities and services all requests. In addition to the time window and capacity constraints of the VRPTW formulation (3.11)–(3.19), two additional single path constraints characterize this pickup and delivery problem: the coupling of and the precedence relation between the pickup and delivery nodes i and $n + i$ of the same request. These constraints can be written as follows (Solomon and Desrosiers, 1988):

$$\sum_{j:(i,j)\in A^k} X_{ij}^k - \sum_{j:(j,n+i)\in A^k} X_{j,n+i}^k = 0\,, \qquad \forall\, k \in K,\ \forall\, i \in N^{Pk} \tag{3.22}$$

$$T_i^k + t_{i,n+i}^k - T_{n+i}^k \leq 0\,, \qquad \forall\, k \in K,\ \forall\, i \in N^{Pk}\,. \tag{3.23}$$

The coupling constraints (3.22) ensure that the pickup and delivery nodes corresponding to each request are both visited by the same vehicle, while the precedence constraints forcing each pickup to occur before the respective delivery are expressed by (3.23).

Relations (3.22)–(3.23) together with the VRPTW formulation (3.11)–(3.19) provide a formulation for the Vehicle Routing Problem with Time Windows, Pickups and Deliveries (VRPTWPD). To eliminate redundancy, the covering constraints (3.12) and the capacity limits (3.18) need only be imposed for pickup nodes $i \in N^P$ and $i \in N^{Pk}$, respectively. For $m = 1$, we derive the single vehicle version (TSPTWPD) which can be efficiently solved optimally by dynamic programming whenever the time or capacity windows are relatively narrow (Desrosiers, Dumas and Soumis, 1986). Obviously, the Shortest Path Problems ESPTWPD and SPTWPD can be similarly obtained as extensions of the ESPTWQ and SPTWQ, respectively. An optimal dynamic programming algorithm for the SPTWPD is provided in Dumas, Desrosiers and Soumis (1991).

The second example of additional single path constraints are the cycle elimination constraints. When negative cost cycles are present in the network, several copies of a task node can be created in order to eliminate the cycles. These copies have disjoint time windows obtained by partitioning the original time windows. Let now $w \in N$ represent a task to be covered, and $COPY^k(w)$ denote the set of copy nodes of task w in G^k. There are at most $b_w^k - a_w^k + 1$ such copies if the time related problem data is integer. To eliminate 2-cycles, that is, path segments of the form $i_1 \rightarrow j \rightarrow i_2$ with $i_1, i_2 \in COPY^k(w)$, we must add the following constraints to the path structure (3.13)–(3.19):

$$\sum_{i\in COPY^k(w)} (X_{ij}^k + X_{ji}^k) \leq 1\,, \quad \forall\, k \in K,\ \forall\, w, j \in N^k\,. \tag{3.24}$$

Appending these constraints to a resource constrained shortest path problem formulation will help reduce the gap between the solution of the respective Shortest Path Problem and the corresponding elementary Shortest Path Problem. Several papers report on algorithms and results obtained by using this type of relations, such as Kolen, Rinnooy Kan and Trienekens (1987), Desrochers, Desrosiers and Solomon (1992) and Kohl and Madsen (1997) for the SPTWQ and Houck *et al.* (1980) for the classical Traveling Salesman Problem.

To achieve complete cycle elimination and to provide a formulation for the ESPTW on the augmented network, created by adding all necessary node copies, the following stronger constraint set must be satisfied:

$$\sum_{i\in COPY^k(w)} \sum_{j:(i,j)\in A^k} X_{ij}^k \leq 1\,, \qquad \forall\, k \in K,\ \forall\, w \in N^k\,. \tag{3.25}$$

The use of the augmented network forces other constraints to be modified accordingly. For example, for the task covering constraints, relations (3.12) must be replaced by bundle type constraints over the copies of an original task-node:

$$\sum_{k\in K}\sum_{i\in COPY^k(w)}\sum_{j:(i,j)\in A^k} X_{ij}^k = 1\,, \qquad \forall\, w \in N\,. \tag{3.26}$$

Note that due to the computational burden involved, no efficient algorithms using this type of cycle elimination constraints have been developed to date.

From the above examples we can develop a general form of these single path constraints that we give next. Let L^k, indexed by l, denote the set of single path constraints other than the resource constraints (3.20)–(3.21) and the usual flow conservation equations (3.13)–(3.14). Let d_{li}^{kr} and $d_{l,ij}^k$ denote the coefficients of the resource and flow variables, respectively, and d_l the right hand side value. General single path constraints can then be expressed as:

$$\sum_{(i,j)\in A^k} d_{l,ij}^k X_{ij}^k + \sum_{i\in V^k}\sum_{r\in R^k} d_{li}^{kr} T_i^{kr} \le d_l\,, \qquad \forall\, k \in K,\, \forall\, l \in L^k\,. \tag{3.27}$$

The flow variables permit these constraints to embed the cycle elimination constraints (3.24), (3.25) and the coupling constraints (3.22), while the time variables allow constraints (3.27) to contain the precedence constraints (3.23) as a special case. Other types of precedence relations such as those required in Mingozzi, Bianco and Ricciardelli (1993) for TSP type problems can be modelled using these general single path constraints. Finally, note that constraints of the same form as (3.27) were obtained by several authors when linearizing the compatibility constraints between the flow and resource variables (3.20).

Multiple Path Linking Constraints: The third extension focuses on constraints involving several commodities $k \in K$, called linking constraints. Let H, indexed by h, denote the set of these additional constraints other than the usual covering constraints (3.12). Further letting $b_{h,ij}^k$ denote the coefficient of variable X_{ij}^k in constraint h and b_h its right hand side value, a special case of these constraints can be formulated as:

$$\sum_{k\in K}\sum_{(i,j)\in A^k} b_{h,ij}^k X_{ij}^k \le b_h\,, \qquad \forall\, h \in H\,. \tag{3.28}$$

This type of constraint is needed to model multiple path flow requirements. For example, if cycle elimination constraints (3.24) or (3.25) are not imposed in the path structure corresponding to each commodity k, then the classical subtour elimination constraints:

$$\sum_{k\in K}\sum_{i\in N'}\sum_{j\in N\setminus N'} X_{ij}^k \ge 1\,, \qquad \forall\, N' \subset N,\, 2 \le |N'| \le |N| - 1 \tag{3.29}$$

can be used. Such constraints have been used by Langevin *et al.* (1993) for the TSPTW and by Fisher (1994) and Kohl *et al.* (1998) for the VRPTW. Additionally, such multiple path constraints are used to restrict the fleet size (see Desrosiers *et al.*

1995) or the fleet composition in vehicle routing problems, and as base constraints related to the number of available crews in the airline context.

A more general form of constraints (3.28) links together the flow and resource variables. Given that b_{hi}^{kr} denotes the coefficient of variable T_i^{kr} in constraint h, these constraints can be written as:

$$\sum_{k\in K}\sum_{(i,j)\in A^k} b_{h,ij}^k X_{ij}^k + \sum_{k\in K}\sum_{i\in V^k}\sum_{r\in R^k} b_{hi}^{kr} T_i^{kr} \le b_h, \qquad \forall\, h \in H\,. \tag{3.30}$$

Such constraints are useful in periodic aircraft routing and scheduling with time windows (Desaulniers *et al.*, 1997b), in airline schedule synchronization (Ioachim *et al.*, 1994) and in applications with sliding time windows (Ferland and Fortin, 1989). The use of constraints (3.30) forces the resource window restrictions (3.21) to be replaced by relations:

$$a_i^{kr} \le T_i^{kr} \le b_i^{kr}\,, \quad \forall k \in K, \forall r \in R^k, \forall i \in \{o(k), d(k)\} \tag{3.31}$$

$$a_i^{kr}\Big(\sum_{j:(i,j)\in A^k} X_{ij}^k\Big) \le T_i^{kr} \le b_i^{kr}\Big(\sum_{j:(i,j)\in A^k} X_{ij}^k\Big)\,, \quad \forall k \in K, \forall r \in R^k, \forall i \in N^k\,. \tag{3.32}$$

For a given k, constraints (3.32) impose that $T_i^{kr} = 0, \quad \forall\, r \in R^k$ whenever $\sum_{j:(i,j)\in A^k} X_{ij}^k = 0$, $k \in K$ and $i \in N^k$, that is, only the nodes within a single solution path in G^k contribute to resource coefficients in the constraint set (3.30).

Objective Functions: More general objective functions have been reported in the literature, especially for Shortest Path Problems. Several examples include (3.33)–(3.36) given below.

$$\sum_{r\in R^k}\sum_{(i,j)\in A^k} g(T_i^{kr}) c_{ij}^k X_{ij}^k\,, \tag{3.33}$$

where c_{ij}^k are real parameters, g is a positive non-decreasing function and T_i^{kr} are resource variables; for example, $g(L_i^k)$ denotes a function of the total load transported in a vehicle k when it leaves node i in a VRPTWPD model (Dumas, Desrosiers and Soumis, 1991).

$$\sum_{(i,j)\in A^k} c_{ij}^k X_{ij}^k + \sum_{i\in V^k}\sum_{r\in R^k} c_i^{kr} T_i^{kr}, \tag{3.34}$$

where c_{ij}^k and c_i^{kr} are real parameters and T_i^{kr} are resource variables. Such an objective function may be encountered when linking constraints (3.30) together with (3.12) are Lagrangean relaxed. See Desaulniers *et al.* (1997b) for applications in periodic aircraft routing and scheduling problems with time windows. When a single resource is used, namely time, an optimal dynamic programming algorithm for the Shortest Path Problem with Time Windows and Time Costs has been proposed in Ioachim *et al.* (1998). For the special case of this problem where the path is fixed, several polynomial algorithms have been suggested to find the optimal schedule by Sexton and Bodin (1985), Sexton and Choi (1986) and Dumas, Desrosiers and Soumis (1990). Finally, using this type of objective function for the TSPTW, Gélinas (1998) presents an application to job-shop scheduling.

$$\sum_{(i,j)\in A^k} c_{ij}^k X_{ij}^k + g(T_{d(k)}^{k1}, \ldots, T_{d(k)}^{k|R^k|}), \tag{3.35}$$

where c_{ij}^k are real parameters and g is a non-decreasing function of the resource variables $T_{d(k)}^{kr}$, $r \in R^k$, at the sink node $d(k)$. The cost of a crew duty in bus driver scheduling problems falls in this category (Desrochers *et al.*, 1992).

$$\sum_{(i,j)\in A^k} c_{ij}^k X_{ij}^k + \sum_{i\in V^k} \sum_{r\in R^k} g_i^{kr}(T_i^{kr}) \sum_{j:(i,j)\in A^k} X_{ij}^k, \tag{3.36}$$

where c_{ij}^k are real parameters, g_i^{kr} are non-decreasing functions and T_i^{kr} are resource variables. An example is provided by cost functions in airline crew pairing problems. Since a pairing involves several duty periods, the cost may depend on resource values at the end of each duty period, in addition to depending on resource values at the end of the pairing.

Soft Time Windows: Several papers (Koskosidis, Powell and Solomon, 1992; Sexton and Choi, 1986) have considered soft time windows to derive feasible solutions to problems where hard time windows are too restrictive. In such cases the fixed interval for a given resource variable T_i^{kr} is replaced by a penalty function g_i^{kr} which determines the penalty to be incurred whenever the resource variable is not within a target value range at node $i \in N^k$. For shortest path type problems, if the penalty functions are piecewise linear, an optimal solution can be found using the algorithm presented in Ioachim *et al.* (1998). Nondecreasing functions on nodes and arcs constitute another simple case since the solution is given by the smallest feasible resource values at each node on a path (see Section 3.4.2). Hence, for an arbitrary nonlinear penalty function, only the intervals where the function is decreasing pose a problem. If they are reasonably narrow, these can be approximated by linear functions or discretized for optimal solutions.

3.3 A UNIFIED FORMULATION

In the previous section, starting from simple scheduling problems we gradually introduced additional constraints to model more complex situations. In this section we present a formulation which encompasses all these constraints. To describe this model we consider a different type of time-space network where the tasks to be covered are positioned on the arcs. An arc may represent a single activity or a series of activities. Furthermore, the same task can appear on several arcs. As examples, consider network representations of crew pairing problems appearing in airline and urban transit contexts where a task (flight leg/trip leg) can be found on several duty arcs (Lavoie, Minoux and Odier, 1988; Desrochers *et al.*, 1992; Desaulniers *et al.*, 1997a). The nodes represent sites at different moments in time such as the beginning and the end of any activity. The task-on-node problem formulations introduced in Section 3.2 can be transformed into task-on-arc formulations by simply associating a given task with all the arcs entering the respective task node.

On the one hand, the more general task-on-arc formulation presented next permits to include in the network model some constraints on the path structure required by an application. To do so, it is sufficient to include in the network all the arcs corresponding to the task sequences satisfying these constraints and to exclude all other arcs covering these same tasks. On the other hand, in some cases, a large subnetwork derived from path enumeration can be replaced by an aggregated subnetwork of

smaller size which uses less activities by arc and less intermediate nodes. Examples are provided by the time-line networks used for crew pairing and aircraft routing applications where the number of arcs becomes linear in terms of the number of tasks to cover (Levin, 1971; Soumis, Ferland and Rousseau, 1980; Barnhart *et al.*, 1994a; and Desrosiers *et al.*, 1995). For each application, the trade off between path enumeration and network aggregation is the determining factor for the computational efficiency of the solution algorithms.

Notation and Formulation: Let N, indexed by w, represent the set of tasks to cover and n_w the number of times that task w must be covered. Applications where a task has to be covered more than once are encountered, for example, in rostering, bidline and locomotive assignment contexts (see Gamache *et al.*, 1998a; Ziarati *et al.*, 1997). Let again K represent the set of commodities and $G^k = (V^k, A^k)$ be an acyclic network associated with $k \in K$, where $V^k = N^k \cup \{o(k), d(k)\}$ is the node set and A^k is the arc set. Note that with the task-on-arc network representation, nodes in N^k no longer represent tasks to cover, but only time-space sites. Let further R^k be the set of resources specific to commodity k. Using the previously introduced flow and resource variables, let $\boldsymbol{X}^k = (X_{ij}^k \mid (i,j) \in A^k)$ denote the flow variable array, and $\boldsymbol{T}_i^k = (T_i^{kr} \mid r \in R^k)$, $i \in V^k$ the resource variable arrays. Next, let $a_{w,ij}^k$ designate the binary coefficient of the flow variable X_{ij}^k, 1 if arc $(i,j) \in A^k$ covers task w and 0 otherwise, for $k \in K$. Finally, let S denote a set of additional variables Y_s taking values in $[a_s, b_s]$, for $s \in S$. In addition to the notation used in the previous section, new definitions are needed to model the more general constraints of the unified model: $T_{d(k)}^{k0}$ is the cost function for commodity k evaluated by using a resource indexed $r = 0$ in network G^k, $k \in K$; a_{ws}, b_{hs} are the coefficients of the variable Y_s, $s \in S$ in the task covering constraint $w \in N$ and in the linking constraint $h \in H$, respectively; f_{ij}^{kr} is a function used for the resource extension from node i to node j, $(i,j) \in A^k$ for resource $r \in R^k$ and flow of type $k \in K$. The importance and utilization of this type of function is given below.

With this notation, the unified model can be presented as a nonlinear mixed-integer program:

$$Minimize \sum_{k \in K} T_{d(k)}^{k0} + \sum_{s \in S} c_s Y_s \tag{3.37}$$

subject to:

$$\sum_{k \in K} \sum_{(i,j) \in A^k} a_{w,ij}^k X_{ij}^k + \sum_{s \in S} a_{ws} Y_s = n_w, \quad \forall\, w \in N \tag{3.38}$$

$$\sum_{k \in K} \sum_{(i,j) \in A^k} b_{h,ij}^k X_{ij}^k + \sum_{k \in K} \sum_{i \in V^k} \sum_{r \in R^k} b_{hi}^{kr} T_i^{kr} + \sum_{s \in S} b_{hs} Y_s = b_h, \quad \forall\, h \in H \tag{3.39}$$

$$a_s \leq Y_s \leq b_s, \quad \forall s \in S \tag{3.40}$$

$$\sum_{j:(o(k),j) \in A^k} X_{o(k),j}^k = 1, \quad \forall\, k \in K \tag{3.41}$$

$$\sum_{j:(i,j) \in A^k} X_{ij}^k - \sum_{j:(j,i) \in A^k} X_{ji}^k = 0, \quad \forall\, k \in K, \forall\, i \in N^k \tag{3.42}$$

$$\sum_{j:(j,d(k))\in A^k} X^k_{j,d(k)} = 1, \quad \forall\, k \in K \tag{3.43}$$

$$X^k_{ij}(f^{kr}_{ij}(\boldsymbol{T}^k_i, \boldsymbol{T}^k_j) - T^{kr}_j) \leq 0, \quad \forall\, k \in K,\ \forall\, r \in R^k,\ \forall\, (i,j) \in A^k \tag{3.44}$$

$$a^{kr}_i \leq T^{kr}_i \leq b^{kr}_i, \quad \forall\, k \in K,\ \forall\, r \in R^k,\ \forall\, i \in \{o(k), d(k)\} \tag{3.45}$$

$$a^{kr}_i\Big(\sum_{j:(i,j)\in A^k} X^k_{ij}\Big) \leq T^{kr}_i \leq b^{kr}_i\Big(\sum_{j:(i,j)\in A^k} X^k_{ij}\Big), \quad \forall\, k \in K,\ \forall\, r \in R^k,\ \forall\, i \in N^k \tag{3.46}$$

$$\sum_{(i,j)\in A^k} d^k_{l,ij} X^k_{ij} + \sum_{i\in V^k}\sum_{r\in R^k} d^{kr}_{li} T^{kr}_i \leq d_l, \quad \forall\, k \in K,\ \forall\, l \in L \tag{3.47}$$

$$X^k_{ij} \ \textit{binary}, \quad \forall\, k \in K,\ \forall\, (i,j) \in A^k\,. \tag{3.48}$$

The analysis of this formulation focuses on three new aspects: the objective function, the new variables and the resource extension functions. The objective function (3.37) seeks to minimize the total cost incurred. Examples of both linear and nonlinear functions have been presented in Section 3.2.2. Given the above definition of the cost function, its value is completely computed at the sink node $d(k)$. Therefore, all nonlinearities are moved into the subproblem structure and are now part of the resource extension functions f^{k0}_{ij}. Additional aspects of the cost function are discussed in Section 3.4.2.

Several applications use the additional variables Y_s, $s \in S$, appearing in constraints (3.38)–(3.40). In constraint set (3.38), these may represent slack variables used for task under or over covering (Barnhart *et al.*, 1994a; Graves *et al.*, 1993; Desaulniers *et al.*, 1997a; Ziarati *et al.*, 1997) and associated penalty costs may be specified in the objective function. Constraints of type (3.39) are a more general version of constraints (3.28) and (3.30) which also involve the additional variables Y_s. Consider an example where constraints (3.28) are used to control the fleet size; one additional variable Y_s can be used to represent the number of vehicles required in a solution. Thus, this variable Y_s can be used during the solution process to impose lower and upper integer bounds on the fleet size. As another example, constraints linking time variables and additional variables Y_s were introduced for a weekly fleet assignment problem with flight synchronization constraints in Ioachim *et al.* (1994). These variables were used to model special time restrictions stating that each occurrence of a given origin-destination flight leg had to start at the same time, within a certain time window, each day of the week. Note that constraints (3.39) are a more general form of the covering constraints (3.38) and, depending on the application, only one of these two sets may be needed. Finally, constraints (3.40) give the set of admissible values for the additional variables for which integrality requirements may also be imposed.

The resource extension function introduced in this formulation leads to a new expression (3.44) of the previous resource constraints (3.20). This function, $f^{kr}_{ij}(\boldsymbol{T}^k_i, \boldsymbol{T}^k_j)$, may be linear or nonlinear and it may depend of several resources. The particular case involving a single resource variable, that is, $f^{kr}_{ij}(T^{kr}_i) = T^{kr}_i + t^{kr}_{ij}$ for all $r \in R^k$, has been discussed in Section 3.2.2. An example of a nonlinear resource extension function is presented next. Other than (3.44), the remainder of the constraints were discussed in Section 3.2.

The VRPTW with Simultaneous Pickups and Deliveries: We consider the simultaneous pickup and delivery problem (Min, 1989) to describe a nonlinear form of a resource extension function f_{ij}^{kr} dependent of resource vector $\boldsymbol{T}_i^k$. In this problem, vehicles starting with an unknown load at the source depot must visit nodes having both pickup and delivery requests. We focus our analysis on the capacity constraints. For this, we define non-negative load values l_i^P for pickups and l_i^D for deliveries, for all nodes $i \in N$. Two types of resource variables can be associated with the capacity constraints (see Gélinas, 1990; and Halse, 1992): L_i^k, which denotes the usual load accumulated at the visited pickup nodes from the source node $o(k)$ up to node i, and $MaxL_i^k$, which denotes the maximum load carried at some point by the vehicle since its departure from the source depot up to node i, for all $k \in K$ and $i \in N^k$. These variables have the following bounds:

$$l_i^P \leq L_i^k \leq Q^k\,, \qquad \forall k \in K,\, \forall\, i \in N^k \tag{3.49}$$

$$\max(l_i^P, l_i^D) \leq MaxL_i^k \leq Q^k\,, \qquad \forall\, k \in K,\, \forall\, i \in N^k\,. \tag{3.50}$$

Let f_{ij}^{k1} and f_{ij}^{k2} denote the resource extension functions, for the two resources, respectively. The load accumulated in the vehicle at node j is computed using solely the first resource variable:

$$f_{ij}^{k1}(L_i^k, MaxL_i^k) = f_{ij}^{k1}(L_i^k) = L_i^k + l_j^P, \qquad \forall\, k \in K\,,\, \forall\,(i,j) \in A^k\,. \tag{3.51}$$

However, the maximal load ever accumulated in the vehicle must also account for deliveries. Hence its resource extension function must depend on both capacity resource variables:

$$f_{ij}^{k2}(L_i^k, MaxL_i^k) = \max\{L_i^k + l_j^P, MaxL_i^k + l_j^D\}, \quad \forall\, k \in K\,,\, \forall\,(i,j) \in A^k\,. \tag{3.52}$$

A special case of this problem is the VRPTW with backhauls. Its associated resource extension function can be easily derived from (3.51)–(3.52) since the nodes are either pickups or deliveries. In addition, for simple backhauling where a vehicle first performs all the deliveries before starting to visit the pickup nodes, only one capacity variable is required as the problem can be modeled as a VRPTW (Gélinas *et al.*, 1995).

For simplicity, we have used the simultaneous pickup and delivery problem to illustrate the use of resource extension functions dependent of multiple resource variables. However, these are often encountered for crew scheduling problems arising in airline, urban and rail transit situations where the cost function depends on several resources. In such cases, the cost itself can be defined as a resource and the incurred cost of a path is evaluated using resource extension functions.

The above unified model is general enough to include all the deterministic problem types presented in the classification scheme of Desrochers, Lenstra and Savelsbergh (1990) as special cases. It also includes problems for which the solution can be represented as a set of paths on a graph of all possible states of the commodities. Therefore, applications involving a Knapsack Problem structure can be modelled by using this unified formulation. Such applications include the Bin Packing Problem, i.e., the binary Cutting-Stock Problem (Vance *et al.*, 1994), the Generalized Assignment Problem (Savelsbergh, 1993), graph coloring problems (Mehrorta and Trick, 1993), and even the classical Cutting-Stock Problem (Gilmore and Gomory, 1961). This last problem can also be formulated as a Vehicle Routing Problem in which the vehicles

and their capacities correspond to the large rolls to be cut into smaller rolls (called items) and their widths, respectively; the customers and their demands correspond to the items and their widths, respectively; and the vehicle routes are the cutting patterns. These patterns are obtained by solving a Knapsack Problem which is formulated as a constrained Shortest Path Problem on an acyclic graph with a capacity window on the sink node. As a further illustration, we next show how a formulation for the VRPTW with split deliveries can be derived from our model.

The VRPTW with Split Deliveries: In this problem, each node's demand may be satisfied by one or several partial deliveries performed by different vehicles (Dror and Trudeau, 1989). To formulate the problem, consider a set K of vehicles that start at the depot node, and let again l_i denote the demand at node i. Further let resource variable L_i^k represent the total load delivered using vehicle k from the depot node up to node i. Additional resource variables D_i^k are also needed to designate the quantity (i.e., partial or full delivery) delivered by vehicle k at node i. The VRPTW formulation (3.11)–(3.19) can then be modified to account for split deliveries as follows. The capacity limits (3.18) need to be replaced by the two following constraint sets:

$$0 \leq D_i^k \leq l_i(\sum_{j:(i,j)\in A^k} X_{ij}^k) \leq L_i^k \leq Q^k(\sum_{j:(i,j)\in A^k} X_{ij}^k), \quad \forall k \in K, \forall i \in N^k \tag{3.53}$$

$$0 \leq D_i^k \leq l_i \leq L_i^k \leq Q^k, \qquad \forall k \in K, \forall i \in \{o(k), d(k)\}, \tag{3.54}$$

where, without loss of generality, $l_{o(k)} = l_{d(k)} = 0$. Furthermore, the covering constraints (3.12) have to be replaced by:

$$\sum_{k\in K} D_i^k = l_i, \qquad \forall i \in N. \tag{3.55}$$

These constraints ensure that the node demand is completely satisfied by the sum of the partial deliveries. Note that constraints (3.55) are a special case of constraints (3.39) and that for this formulation they replace both constraints (3.38) and (3.39). Finally, the compatibility requirements (3.17) between the flow and resource variables are expressed as:

$$X_{ij}^k(L_i^k + D_j^k - L_j^k) \leq 0, \qquad \forall k \in K, \forall (i,j) \in A^k. \tag{3.56}$$

Observe that no such compatibility constraints are necessary between the flow and the resource representing the quantity D_j^k delivered by vehicle k at node j since they are given by $X_{ij}^k(0 - D_j^k) \leq 0, \forall k \in K, \forall (i,j) \in A^k$. That is, non-negativity requirements $D_j^k \geq 0$ which are already given in the capacity limit constraint set (3.53)–(3.54). For this problem, the resource extension functions f_{ij}^{kL} and f_{ij}^{kD} for the total load and the quantity delivered, respectively, are given by:

$$f_{ij}^{kL}(L_i^k, D_j^k) = L_i^k + D_j^k, \qquad \forall k \in K, \forall (i,j) \in A^k \tag{3.57}$$

$$f_{ij}^{kD}(D_i^k) = 0, \qquad \forall k \in K, \forall (i,j) \in A^k. \tag{3.58}$$

3.4 SOLUTION METHODOLOGY

In this section, we present an approach to solve the unified model (3.37)–(3.48) for a number of commodities greater than one. Otherwise, the problem reduces to a TSP

type problem and can be solved using dynamic programming whenever the resource window constraints are tight enough.

Since the objective function (3.37) and the constraint sets (3.41)–(3.48) are separable by commodity, the unified model has a block-angular structure with linking constraints (3.38)–(3.40). Therefore a natural way to solve it is to use a decomposition approach such as Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) or Lagrangean Relaxation (see Geoffrion, 1974) to obtain lower bounds to be used within a branch-and-bound framework. Even though these are equivalent primal and dual decomposition approaches, we have chosen to present an extension of the former. This is because this column generation type approach is the most widely encountered methodology in the time constrained vehicle routing and scheduling literature.

3.4.1 An Extension of the Dantzig-Wolfe Decomposition Principle

In the spirit of the linear programming Dantzig-Wolfe decomposition principle, formulation (3.37)–(3.48) is divided into a master problem and a subproblem. The master problem consists of (3.37)–(3.40), i.e., the objective function, the covering constraints, the additional multiple path linking constraints and the constraint sets on the supplementary variables. Note that the master problem retains all constraints involving more than one commodity. On the other hand, the subproblem considers a marginal cost function presented later and the remaining constraints (3.41)–(3.48). The subproblem is separable by commodity $k \in K$ and can therefore be seen as $|K|$ subproblems, one for each commodity. Each of these subproblems is a constrained Shortest Path Problem.

Extreme Points of the Subproblem: Since the subproblem's constraints (3.41)–(3.48) define a path structure (on an acyclic graph) used to send one flow unit between $o(k)$ and $d(k)$, its extreme points correspond to elementary paths in G^k and are described by flow and resource vectors:

$$(\boldsymbol{x}_p^k, \boldsymbol{\tau}_p^k) = (x_{ijp}^k, \tau_{ip}^{kr}),\ k \in K,\ p \in \Omega^k,\ (i,j) \in A^k,\ r \in R^k\,,$$

where Ω^k, indexed by p, defines the set of extreme points. Any solution X_{ij}^k and T_i^{kr} satisfying the constraints (3.41)–(3.48) can then be expressed as a non-negative convex combination of these extreme points and must consist of binary X_{ij}^k values, i.e.,

$$X_{ij}^k = \sum_{p \in \Omega^k} x_{ijp}^k \theta_p^k, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{3.59}$$

$$X_{ij}^k \ \textit{binary}, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{3.60}$$

$$T_i^{kr} = \sum_{p \in \Omega^k} \tau_{ip}^{kr} \theta_p^k, \qquad \forall\, k \in K,\ \forall\, r \in R^k,\ \forall\, i \in V^k \tag{3.61}$$

$$\sum_{p \in \Omega^k} \theta_p^k = 1, \qquad \forall\, k \in K \tag{3.62}$$

$$\theta_p^k \geq 0, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k. \tag{3.63}$$

The new variables θ_p^k, $k \in K$, $p \in \Omega^k$, are said to be path variables since each of them is associated with a path in G^k. Note that, for each $k \in K$, there is a path variable corresponding to the one-arc path between $o(k)$ and $d(k)$ that allows the commodity to be idle. If these empty path variables were removed, the constraints in set (3.62) could then be written as inequalities.

Extreme Point Substitution: Substituting (3.59)–(3.63) in (3.37)–(3.40) and rearranging the summation order, the integer master problem is transformed into:

$$\textit{Minimize} \sum_{k \in K} \tau_{d(k)p}^{k0} \theta_p^k + \sum_{s \in S} c_s Y_s \tag{3.64}$$

subject to:

$$\sum_{k \in K} \sum_{p \in \Omega^k} \Big(\sum_{(i,j) \in A^k} a_{w,ij}^k x_{ijp}^k \Big) \theta_p^k + \sum_{s \in S} a_{ws} Y_s = n_w, \qquad \forall\, w \in N \tag{3.65}$$

$$\sum_{k \in K} \sum_{p \in \Omega^k} \Big(\sum_{(i,j) \in A^k} b_{h,ij}^k x_{ijp}^k + \sum_{i \in V^k} \sum_{r \in R^k} b_{hi}^{kr} \tau_{ip}^{kr} \Big) \theta_p^k + \sum_{s \in S} b_{hs} Y_s = b_h, \qquad \forall\, h \in H \tag{3.66}$$

$$a_s \le Y_s \le b_s, \qquad \forall s \in S \tag{3.67}$$

$$\sum_{p \in \Omega^k} \theta_p^k = 1, \qquad \forall\, k \in K \tag{3.68}$$

$$\theta_p^k \ge 0, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k \tag{3.69}$$

$$X_{ij}^k = \sum_{p \in \Omega^k} x_{ijp}^k \theta_p^k, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k \tag{3.70}$$

$$X_{ij}^k \ \textit{binary}, \qquad \forall\, k \in K,\ \forall\, (i,j) \in A^k. \tag{3.71}$$

The Master Problem: Defining new coefficients as:

$$c_p^k = \tau_{d(k)p}^{k0}, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k \tag{3.72}$$

$$a_{wp}^k = \sum_{(i,j) \in A^k} a_{w,ij}^k x_{ijp}^k, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k,\ \forall\, w \in N, \tag{3.73}$$

$$b_{hp}^k = \sum_{(i,j) \in A^k} b_{h,ij}^k x_{ijp}^k + \sum_{i \in V^k} \sum_{r \in R^k} b_{hi}^{kr} \tau_{ip}^{kr}, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k,\ \forall\, h \in H, \tag{3.74}$$

and substituting them in (3.64)–(3.66), the master problem can be written as:

$$\textit{Minimize} \sum_{k \in K} \sum_{p \in \Omega^k} c_p^k \theta_p^k + \sum_{s \in S} c_s Y_s \tag{3.75}$$

subject to:

$$\sum_{k\in K}\sum_{p\in\Omega^k} a^k_{wp}\theta^k_p + \sum_{s\in S} a_{ws}Y_s = n_w, \qquad \forall\, w \in N \tag{3.76}$$

$$\sum_{k\in K}\sum_{p\in\Omega^k} b^k_{hp}\theta^k_p + \sum_{s\in S} b_{hs}Y_s = b_h, \qquad \forall\, h \in H \tag{3.77}$$

$$a_s \le Y_s \le b_s, \qquad \forall s \in S \tag{3.78}$$

$$\sum_{p\in\Omega^k} \theta^k_p = 1, \qquad \forall\, k \in K \tag{3.79}$$

$$\theta^k_p \ge 0, \qquad \forall\, k \in K,\ \forall\, p \in \Omega^k \tag{3.80}$$

$$X^k_{ij} = \sum_{p\in\Omega^k} x^k_{ijp}\theta^k_p, \qquad \forall\, k \in K,\ \forall\,(i,j) \in A^k \tag{3.81}$$

$$X^k_{ij}\ \textit{binary}, \qquad \forall\, k \in K,\ \forall\,(i,j) \in A^k. \tag{3.82}$$

In general, it is not true that binary restrictions on the X^k_{ij} arc variables can be replaced by binary restrictions on the θ^k_p path variables. However this statement holds when the coefficients b^k_{hp} are expressed solely in terms of the arc variable values, i.e., $b^k_{hp} = \sum_{(i,j)\in A^k} b^k_{h,ij}x^k_{ijp}$. In this case, since the definition of the original master problem solution space (3.37)–(3.40) involves only arc variables, binary requirements on these variables are equivalent to binary requirements on path variables. Constraints (3.81) and (3.82) can therefore be removed.

In the general case, that is, when some coefficients b^{kr}_{hi} in (3.74) are different from zero, the solution part associated with commodity k may actually be fractional in terms of the path variables while it corresponds to the same path with different resource values. Hence, the arc variables take binary values equal to 1 for all arcs in the selected path and 0 otherwise, and no branching decisions are necessary. The optimal solution in terms of the resource variables can therefore be computed as a non binary convex combination of the path variables using relations (3.61).

An example of this situation is presented in Ioachim *et al.* (1994) for an aircraft routing problem that considers the synchronization of flight departures within their respective time window. The branch-and-bound process is stopped as soon as the arc variables take binary values even though there still remain fractional-valued path variables. Time variable optimal values are then derived from (3.61). It appears that the optimal solutions for some commodities cannot be obtained as subproblem extreme points.

Objective Function of the Subproblem: To describe the objective function of subproblem $k \in K$, let $\boldsymbol{\alpha} = \{\alpha_w \mid w \in N\}$, $\boldsymbol{\beta} = \{\beta_h \mid h \in H\}$ and $\boldsymbol{\gamma} = \{\gamma^k \mid k \in K\}$, be the vectors of the dual variables associated with constraint sets (3.76), (3.77) and (3.79), respectively. The reduced cost $\bar{c}^k_p(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ of any path variable θ^k_p in function of vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ is given by:

$$\begin{aligned}\bar{c}_p^k(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) &= c_p^k - \sum_{w\in N} a_{wp}^k \alpha_w - \sum_{h\in H} b_{hp}^k \beta_h - \gamma^k \\ &= \tau_{d(k)p}^{k0} - \sum_{(i,j)\in A^k} \Big(\sum_{w\in N} a_{w,ij}^k \alpha_w + \sum_{h\in H} b_{h,ij}^k \beta_h\Big) x_{ijp}^k \\ &\quad - \sum_{i\in V^k} \sum_{r\in R^k} \Big(\sum_{h\in H} b_{hi}^{kr} \beta_h\Big) \tau_{ip}^{kr} - \gamma^k.\end{aligned}$$

This relation can then be used to write the objective function of subproblem k in terms of the original flow and resource variables satisfying (3.41)–(3.48):

$$\textit{Minimize } T_{d(k)}^{k0} - \sum_{(i,j)\in A^k} \Big(\sum_{w\in N} a_{w,ij}^k \alpha_w + \sum_{h\in H} b_{h,ij}^k \beta_h\Big) X_{ij}^k - \sum_{i\in V^k} \sum_{r\in R^k} \Big(\sum_{h\in H} b_{h,i}^{kr} \beta_h\Big) T_i^{kr} - \gamma^k. \tag{3.83}$$

3.4.2 Algorithmic Issues

As described above, the unified model (3.37)–(3.48) can be transformed into the master problem (3.75)–(3.82) and the subproblem (3.41)–(3.48) and (3.83). This transformation permits instead of directly solving the former model, to solve the latter using a branch-and-bound algorithm. The two key elements in the master problem solution process are the computation of lower bounds and the selection of branching strategies. Regarding the subproblem solution, we address special structures for the cost and resource extension functions. Finally, commodity aggregation for identical subproblems is examined.

Master Problem Lower Bounds: At each branching node, a lower bound can be found by solving the linear relaxation (3.75)–(3.80) of the master problem (3.75)–(3.82) using a column generation technique. This consists of alternately solving a restricted linear master problem and the subproblems until the solution of the latter proves that optimality has been reached for the former. The role of the restricted linear master problem is to find the best solution to the master problem while considering a relatively small number of variables. Hence, path variables to be introduced in the restricted master problem are generated as needed. The dual variables corresponding to this solution are then transferred to the subproblems. The role of the subproblems is to find new path variables with negative reduced costs taking into account the dual variables provided by the restricted linear master problem. If these exist, the coefficient columns associated with a subset of them are generated and transferred to the restricted linear master problem that will be solved again. If no such path variables exist, the solution process stops.

The restricted linear master problem is often degenerate in some applications and the dual simplex method could be used in place of the primal simplex algorithm to accelerate the solution process. Alternatively, a right hand side perturbation and other anti-cycling techniques can be used to avoid degeneracy difficulties (see Wolfe, 1963; Desrosiers, Soumis and Desrochers, 1984; and Falkner and Ryan, 1988). Recent stabilization techniques involving both perturbation and penalization strategies are

described in du Merle *et al.* (1997) and Hansen *et al.* (1998). Interior point methods can also be used to solve the restricted linear master problem. The analytic center cutting plane method developed by (Goffin and Vial, 1996; Goffin *et al.*, 1997) is compatible with column generation approaches in the sense that a warm start is possible when new variables are added. This method provides more stable dual variables than the simplex algorithm (see also du Merle, 1995). These methods have a tendency to provide more stable dual variables. Since the master problem has a block-angular structure, it can also be solved using specialized solution procedures such as the generalized upper bounding procedure of Dantzig and Van Slyke (1967). This method performs all the iterations of the simplex method on a working basis of reduced size. A detailed description of it, in the context of multi-commodity flow models, is given in Barnhart *et al.* (1991).

A lower bound can also be found using Lagrangean Relaxation. In this context, the Lagrangean relaxed problem corresponds to the subproblem, while the Lagrangean Dual problem has the same role as the restricted linear master problem in the Dantzig-Wolfe decomposition. Their essential purpose is to provide dual multipliers for the subproblems. In a Lagrangean Relaxation approach, the Lagrangean dual can be solved by either subgradient optimization or by a bundle method. See LeMaréchal (1989) for a general description of the method, and Kohl and Madsen (1997) for an application of it to the VRPTW. In the latter case the dual variables are adjusted by quadratic programming. When they are solved to optimality, these two dual methods produce the same lower bound on the master problem as the Dantzig-Wolfe decomposition scheme. However, in practice, these dual methods are stopped early to avoid expensive tailing-off behavior.

To reduce the tailing-off effect that can occur when using a Dantzig-Wolfe decomposition or a Lagrangean Relaxation approach, one may want to stop the iterative process before reaching the optimality criteria. In both cases, given a set of multipliers $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$, a lower bound for the current branching node is computed as follows (see Lasdon, 1970):

$$\begin{aligned} LB(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \quad = \quad & \sum_{w \in N} n_w \alpha_w + \sum_{h \in H} b_h \beta_h + \sum_{k \in K} \gamma^k + \sum_{k \in K} \min_{p \in \Omega^k} \bar{c}_p^k(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) \\ & + \sum_{s \in S : \bar{c}_s > 0} a_s \bar{c}_s + \sum_{s \in S : \bar{c}_s < 0} b_s \bar{c}_s \,, \end{aligned} \tag{3.84}$$

where $\bar{c}_s$ denotes the reduced cost of supplementary variable Y_s. The last two components of (3.84) account for the contribution of the supplementary variables to the lower bound. The reader can also observe that the sum of the first two components is equal to the value of the objective function for the current restricted master problem. Note that this is the usual lower bound computed in a Lagrangean Relaxation approach since the $\boldsymbol{\gamma}$ multipliers will cancel out between the third and fourth components of (3.84). At a given branching node, the best lower bound is computed as the maximum over all the lower bounds obtained at this node.

When the objective function consists solely in minimizing the number of commodities used and no supplementary variables are involved in the formulation, the expression of the lower bound takes a very simple form. Let n be the number of commodities used, z_{LP} the objective value of the current restricted master problem

and $\bar{c}_{min}$ the minimum path reduced cost. Therefore, relation $n \geq z_{LP} + |K|\bar{c}_{min}$ can be favorably replaced by $n \geq z_{LP} + n\bar{c}_{min}$ to yield a better valid lower bound:

$$n \geq \lceil z_{LP}/(1 - \bar{c}_{min}) \rceil . \tag{3.85}$$

This bounding procedure has been used successfully by Farley (1990) and Vance *et al.* (1994) for various cutting stock problems.

A result on the quality of the master problem lower bound has been obtained when the model reduces to a Set Partitioning Problem. In this case, Bramel and Simchi-Levi (1997) showed that the relative gap between fractional and integer solutions becomes arbitrarily small as the number of tasks to cover increases. This could make the branch-and-bound process more efficient.

Finally, we emphasize that the solution of a relaxed version of a constrained Shortest Path Problem does not invalidate the solution process. For example, using the SPTW instead of the ESPTW as a subproblem (or equivalently, using an expanded acyclic graph) only results in a deterioration of the master problem lower bound. Columns which contain several copies of the same tasks may appear in the master problem structure, but the branch-and-bound decisions and the cutting strategies must eliminate them, as they cannot be part of any optimal solution.

Branching and Cutting Strategies: Several papers report on the use of column generation techniques to optimally solve integer programs. Methods finding k-best solutions were proposed by Sweeney and Murphy (1979), Hansen, Minoux and Labbé (1987), Hansen, Jaumard and Poggi de Aragão (1991, 1992) and Maculan, Michelon and Plateau (1992). The paper of Holm and Tind (1988) looks in another direction and generalizes the Dantzig-Wolfe decomposition principle to linear integer programs by means of concave, polyhedral price functions. This method however requires to solve an integer restricted master program at each main iteration.

Branch-and-bound, cuts and column generation have been combined several times in various vehicle routing applications: Desrosiers, Soumis and Desrochers (1984) for the m-TSPTW, Dumas, Desrosiers and Soumis (1991) for the VRPTWPD, Desrochers, Desrosiers and Solomon (1992) for the VRPTW, and Gélinas *et al.* (1995) for the simple backhauling problem with time windows. Other applications include bus drivers assignment (Desrochers and Soumis, 1989), frame decomposition for telecommunication by satellite (Ribeiro, Minoux and Penna, 1989), urban bus assignment (Ribeiro and Soumis, 1994). Recent papers are those on the binary cutting-stock problems (Vance *et al.*, 1994), the generalized assignment problem (Savelsbergh, 1993), graph coloring problems (Mehrorta and Trick, 1993), the airline crew pairing problem (Desaulniers *et al.*, 1997a), the aircraft routing problem (Desaulniers *et al.*, 1997b) and the locomotive assignment problem (Ziarati *et al.*, 1997). Vanderbeck and Wolsey (1996) presents a general framework which extends a branching scheme proposed by Ryan and Foster (1981) for the Set Partitioning Problem and additional results reported in Barnhart *et al.* (1994b). In this paper, Vanderbeck and Wolsey develop a combined branching and subproblem modification scheme: branching decisions are taken on the variables of the column generation master problem while the subproblem necessitates the addition of new variables and constraints to account for these decisions.

We propose a general and yet simple branching and cutting framework based on flow, resource or supplementary variables, or on any weighted sum of these variables. When the unified model (3.37)–(3.48) is solved by Lagrangean Relaxation (a dual approach), branching decisions are naturally taken on flow X_{ij}^k, and when integrality is required, on resource T_i^{kr} or on supplementary Y_s variables. Flow-based branching decisions occur, for example, in Fisher (1994) for the classical Vehicle Routing Problem and in Kohl and Madsen (1997) for the VRPTW. When a Dantzig-Wolfe decomposition approach is applied, i.e., an equivalent primal approach, the same branching and cutting strategies can be utilized. Furthermore, designing branching decisions and cutting planes which are based on the original problem variables becomes crucially important when general integer problems are solved using a Dantzig-Wolfe decomposition scheme. As shown in Desrosiers *et al.* (1994), imposing integrality requirements on the master problem variables does not always result in a valid formulation.

In our unified model, general strategies based on the original variables are as follows. On the one hand, if a decision involves more than one commodity, then this multiple path decision can be written in the form of the linking constraints (3.39). As previously described, such a constraint is transferred to constraint set (3.77) of the master problem, using substitution equations (3.59) and (3.61). Through relation (3.77), the multiplier associated with this constraint is adequately transferred to the arcs and nodes of the relevant subproblem networks as coefficient for the flow and resource variables. On the other hand, if a decision involves a single commodity, then it is a single path decision. Such a decision can be restricted to a single subproblem. However, if it modifies the subproblem structure, a new specialized algorithm has to be used for solving it. This new algorithm may be one of the several algorithms presented in Section 3.2 or may yet have to be designed. Alternatively, any cumbersome single path constraint in set (3.47) may be left at the master problem level since it is a special case of a constraint in set (3.39).

Next, we present several examples of branching and cutting decisions compatible with the column generation approach proposed to solve the unified model. We first present some special multiple path linking decisions that can be treated at the subproblem level. We then provide examples of branching decisions and cutting planes to be inserted at the master problem level. We also give an example of a very strong cutting plane that acts locally at the subproblem level. Finally, decisions involving resource and supplementary variables are discussed.

Consider the case where the tasks are represented by nodes and must be covered exactly once. While several other possibilities exist, branching decisions can be taken on the following linear combinations of flow variables:

$$
\begin{aligned}
& X_{ij}^k, && \forall\, k \in K \text{ and } \forall\, i,j \in N \\
X_{i,J} = \sum_{k \in K} \sum_{j \in J:(i,j) \in A^k} & X_{ij}^k, && \forall\, i \in N \text{ and } \forall\, J \subset N \\
X_i^{K'} = \sum_{k \in K'} \sum_{j:(i,j) \in A^k} & X_{ij}^k, && \forall\, i \in N \text{ and } \forall\, K' \subset K.
\end{aligned}
$$

Variables X_{ij}^k and the linear combinations $X_{i,J}$ and $X_i^{K'}$ must take binary values at optimality since the tasks must be covered exactly once. Therefore, any such fractional-valued binary variable can be set to 0 on one branch and to 1 on the other

branch. Fixing X_{ij}^k at 1 corresponds to consecutively assigning tasks i and j to commodity k. Setting $X_{i,J}$ to 1 specifies that a task of subset J must be performed immediately after task i without identifying the commodity. In particular, if J contains a single task, then this task must immediately follow task i. Finally, fixing $X_i^{K'}$ at 1 implies that task i must be covered by a commodity of subset K'. In particular, if K' contains a single commodity, then this commodity must carry out task i. Branching decisions on these multiple path linking constraints can be taken into account in the subproblems without changing their mathematical structure. For example, if variable $X_{i,\{j\}}$ is fixed at 1, arcs $(i,j') \in A^k$, such as $j' \neq j$ and $(i',j) \in A^k$ such as $i' \neq i$ are removed from network G^k, $\forall k \in K$. If variable $X_{i,\{j\}}$ is fixed at 0, the arc $(i,j) \in A^k$ is removed from network G^k, $\forall k \in K$.

Branching strategies and cuts involving flow variables cannot always be taken into account at the subproblem level. For example, the linear combination

$$X^{K'} = \sum_{k \in K'} \sum_{\substack{j:(o(k),j) \in A^k \\ j \neq d(k)}} X_{o(k),j}^k, \quad \forall K' \subseteq K,$$

computes the number of commodities used in subset K' and must take an integer value at optimality. When such a variable takes a fractional value $x^{K'}$, it can be restricted to take a value less than or equal to $\lfloor x^{K'} \rfloor$ on one branch and greater than or equal to $\lceil x^{K'} \rceil$ on the other branch. Alternatively, if the objective function primarily minimizes the number of commodities used, a cut on this number can be imposed. These decisions are integrated in constraint set (3.39). Another example can occur when the objective function consists of a linear combination of flow variables and the cost coefficients are integer. In this case, the following cut can be added to constraint set (3.39) whenever the objective function takes a fractional value z:

$$\sum_{k \in K} \sum_{(i,j) \in A^k} c_{ij}^k X_{ij}^k \geq \lceil z \rceil. \tag{3.86}$$

When a lower bound on the cost of paths associated with a single commodity k can be computed, then a resulting cut of the same type as (3.86) can be imposed locally in subproblem k (Gamache *et al.*, 1998b). Even though the previous cut (3.86) may appear simple, in fact it possesses the general structure of any cut written as a weighted sum of the flow variables. Special cases of it are the classical subtour elimination constraints mentioned in Section 3.2.2 and the generalized subtour elimination constraints

$$\sum_{k \in K} \sum_{i \in N'} \sum_{j \in N \setminus N'} X_{ij}^k \geq v(N'), \quad \forall N' \subset N,\ 2 \leq |N'| \leq |N| - 1, \tag{3.87}$$

where $v(N')$ is a lower bound on the number of paths required to visit all nodes of N'. Such cuts have been used by several authors to solve the VRP (see Laporte, 1992; and Fisher, 1994) and its extension involving time windows (see Kontoravdis and Bard, 1992; and Kohl *et al.*, 1998).

Branching decisions on resource variables can also be devised when appropriate. For instance, if an integer resource variable T_i^{kr} takes a fractional value τ_i^{kr}, then one may restrict this variable to be less than or equal to $\lfloor \tau_i^{kr} \rfloor$ on one branch and

greater than or equal to $\lceil \tau_i^{kr} \rceil$ on the other. These decisions can be easily treated in the subproblems. The application of such strategies to the simple backhauling problem with time windows are presented in Gélinas *et al.* (1995). A stronger type of resource branching decision is derived for the aircraft schedule synchronization problem described in Ioachim *et al.* (1994). In this case, supplementary variables are defined as $Y_i^r = \sum_{k \in K} T_i^{kr}$ and when such a variable Y_i^r takes a fractional value y_i^r, then two branches are created as above using $\lfloor y_i^r \rfloor$ and $\lceil y_i^r \rceil$. Since variable Y_i^r is linked with a specific resource variable r, these decisions are locally transferred at the subproblem level to all the corresponding T_i^{kr} variables, i.e., $T_i^{kr} \leq \lfloor y_i^r \rfloor$ on one branch and $T_i^{kr} \geq \lceil y_i^r \rceil$ on the other. Moreover, this type of decision also applies for an integer value of Y_i^r if y_i^r is obtained as a convex combination of several columns with different τ_{ip}^{kr} values: the two branches are then given by $Y_i^r \leq y_i^r - 1$ and $Y_i^r \geq y_i^r$.

Branching decisions involving resource variables which cannot be taken into account in the subproblems can also be devised, since these decisions have the general form of constraints (3.39). However, if the objective function of the subproblem did not contain any resource variable before the addition of this type of constraint, the subproblem structure will be modified since its objective function will now contain resource variables (see Subproblem Solution below).

In the unified model (3.37)–(3.48) where the tasks are represented by arcs on time-space networks, most of the branching decisions and cuts described above are directly applicable or need simple adaptations. Taking into account the specifics of each application, one can devise other efficient branching rules and cuts.

Cost and Resource Extension Functions: In formulation (3.37)-(3.48), a resource is used to compute the cost of a path, hence transferring all the inherent non linearity of the cost function to the subproblem structure. This paragraph states some desirable properties for cost functions $c^k(\boldsymbol{X}^k, \boldsymbol{T}^k)$, $k \in K$, expressed in terms of flow and resource arrays $\boldsymbol{X}^k = (X_{ij}^k \mid (i,j) \in A^k)$ and $\boldsymbol{T}^k = (T_i^{kr} \mid i \in V^k,\ r \in R^k)$, as well as for all other resource extension functions.

For a commodity $k \in K$, define the cost function as:

$$c^k(\boldsymbol{X}^k, \boldsymbol{T}^k) = \sum_{(i,j) \in A^k} g_{ij}^k(\boldsymbol{T}_i^k) X_{ij}^k + \sum_{i \in V^k} g_i^k(\boldsymbol{T}_i^k) \sum_{j:(i,j) \in A^k} X_{ij}^k \,, \tag{3.88}$$

where we assume, for the next paragraphs, that $\boldsymbol{T}_i^k$ does not include the cost resource variable T_i^{k0}. The function $c^k(\boldsymbol{X}^k, \boldsymbol{T}^k)$ separates into two parts: the arc and the node components. If arc $(i,j) \in A_k$ is used on a path (i.e., if $X_{ij}^k = 1$), the cost of this arc is given by $g_{ij}^k(\boldsymbol{T}_i^k)$, a function of the resource vector $\boldsymbol{T}_i^k$. If node $i \in V^k$ is visited on a path (i.e., if $\sum_{j:(i,j) \in A^k} X_{ij}^k = 1$), the cost incurred at this node is evaluated by the function $g_i^k(\boldsymbol{T}_i^k)$. Therefore, the cost extension function $f_{ij}^{k0}(\boldsymbol{T}_i^k, \boldsymbol{T}_j^k)$ on arc $(i,j) \in A_k$, can be defined as:

$$f_{ij}^{k0}(\boldsymbol{T}_i^k, \boldsymbol{T}_j^k) = g_{ij}^k(\boldsymbol{T}_i^k) + g_j^k(\boldsymbol{T}_j^k) \tag{3.89}$$

and $T_{o(k)}^{k0}$, the cost resource variable, can be set to $g_{o(k)}^k(\boldsymbol{T}_{o(k)}^k)$ at the source node $o(k)$ when the value of all resource variables $T_{o(k)}^{kr}$ have been initialized.

Like the cost extension functions, other resource extension functions f_{ij}^{kr}, $r \in R^k$, have been defined as functions of the resource variables at both nodes i and j. However, to facilitate the design of efficient algorithms for the subproblem solution, all extension functions should possess the following two desirable properties: all resource extension functions on arc (i,j) depend solely on the resource vector $\boldsymbol{T}_i^k$ and all cost and resource extension functions are non-decreasing. The first property permits the computation of intermediate values $f_{ij}^{kr}(\boldsymbol{T}_i^k, \cdot)$, $r \in R^k$, at node j. These values provide lower bounds on the resource variables T_j^{kr}. From the second property, it follows that, if the extension on arc (i,j) is feasible (i.e., if $f_{ij}^{kr}(\boldsymbol{T}_i^k, \cdot) \leq b_j^{kr}$, $\forall r \in R^k$), the lowest cost at node j is always attained at the lowest feasible resource values. Consequently, the values of the resource variables can be computed as:

$$T_j^{kr} = \max \{a_{ij}^{kr}, f_{ij}^{kr}(\boldsymbol{T}_i^k, \cdot)\}, \quad \forall r \in R^k. \tag{3.90}$$

The reader can observe that the maximum function of (3.90) is a composition of two non-decreasing functions and hence, it is also a non-decreasing function. The resource extension functions can therefore be directly defined as the above right-hand side function. Finally, note that non-decreasing cost functions are very common in practice and that it suffices that this property be shared by the functions g_{ij}^k and g_i^k.

Let $\boldsymbol{\tau}_i^k$ and $\boldsymbol{\tau}_i'^k$ be two vectors of cost and resource variable values at node i such that $\boldsymbol{\tau}_i^k \leq \boldsymbol{\tau}_i'^k$. When all extension functions f_{ij}^{kr} possess the above properties, extending $\boldsymbol{\tau}_i^k$ and $\boldsymbol{\tau}_i'^k$ on arc (i,j) yields two vectors of resource variable values $\boldsymbol{\tau}_j^k$ and $\boldsymbol{\tau}_j'^k$ at node j (when both extensions are feasible) that satisfy $\boldsymbol{\tau}_j^k \leq \boldsymbol{\tau}_j'^k$. This obvious result is useful in the subproblem solution process considered in the next paragraph.

Subproblem Solution: In the following, we assume that all extension functions satisfy the previous two properties and we denote by $\bar{c}_{ij}^k(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ the reduced cost of an arc $(i,j) \in A^k$ which is calculated as:

$$\bar{c}_{ij}^k(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = T_d^{k0}(k) I(j) - (\sum_{w \in N} a_{w,ij}^k \alpha_w + \sum_{h \in H} b_{h,ij}^k \beta_h) - \sum_{r \in R^k} (\sum_{h \in H} b_{hi}^{kr} \beta_h) T_i^{kr}, \tag{3.91}$$

where the indicator function $I(j) = 1$ if $j = d(k)$, and 0 otherwise. To solve a subproblem on graph G^k, define additional resource variables $\bar{Z}_i^k$, $i \in V^k$, each of them representing the accumulated reduced cost from the source node $o(k)$ up to node i. The reduced cost extension function $\bar{z}_{ij}^k(\cdot)$ on arc $(i,j) \in A^k$ is defined by:

$$\bar{z}_{ij}^k(\bar{Z}_i^k, \boldsymbol{T}_i^k, \boldsymbol{T}_j^k) = \bar{Z}_i^k + \bar{c}_{ij}^k(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}), \tag{3.92}$$

where $\bar{Z}_{o(k)}^k$ is initialized at $-\gamma^k$.

With this new notation, two features of the subproblem's formulation on G^k are modified. First, the objective function simply becomes:

$$Minimize \ \bar{Z}_{d(k)}^k \tag{3.93}$$

and second, the following set of constraints is added to the initial structure (3.41)–(3.48):

$$X_{ij}^k (z_{ij}^k(\bar{Z}_i^k, \boldsymbol{T}_i^k, \boldsymbol{T}_j^k) - \bar{Z}_j^k) \leq 0, \qquad \forall (i,j) \in A^k \tag{3.94}$$

$$\bar{Z}_{o(k)}^k = -\gamma^k . \tag{3.95}$$

In (3.93), the reduced cost of the shortest path for subproblem k is computed by minimizing the resource variable $\bar{Z}^k_{d(k)}$ at the sink node. Compatibility requirements between flow and resource variables are given in (3.94) in a form similar to (3.44), except that the reduced cost extension function $\bar{z}^k_{ij}$ depends on the new resource variables $\bar{Z}^k_i$ as well as the original resource variable vectors $\boldsymbol{T}^k_i$ and $\boldsymbol{T}^k_j$ at both nodes i and j, respectively. No bounds are imposed on the reduced cost resource except for the value of $\bar{Z}^k_{o(k)}$ at the source node which is given by (3.95). Note however that an upper bound of zero can be imposed on $\bar{Z}^k_{d(k)}$ since the subproblem searches for negative reduced cost paths.

Even though the original cost function $T^k_{d(k)}$ and the subproblem objective function $\bar{Z}^k_{d(k)}$ are obviously non-decreasing, the reduced cost extension functions $\bar{z}^k_{ij}$ are not necessarily non-decreasing. Indeed, it depends on the sign of the coefficients $(-\sum_{h\in H} b^{kr}_{hi}\beta_h)$ of the resource variables appearing in the last term defining the reduced cost of an arc (3.91). If these coefficients are all non negative, then the reduced cost extension functions are also non-decreasing. This condition is satisfied when no resource variables appear in the multiple path constraints (3.39), i.e., $b^{kr}_{hi} = 0$, $\forall\, k \in K$, $\forall\, i \in V^k$, $\forall\, r \in R^k$ and $\forall\, h \in H$.

When labelling algorithms are used to solve resource constrained problems, the labels require the reduced cost component and an additional component for each resource. Hence, the SPTW and the TSPTW use two-dimensional labels involving the reduced cost and the time, while the SPTWQ involves three-dimensional labels. Starting at the sink node, these algorithms iteratively construct paths in G^k which are stored as labels. Thus, each label corresponds to a partial path from the origin node $o(k)$ to the current node. It is represented by a vector containing a reduced cost component and $|R^k|$ components for the resources. Let $E^k_i = (\bar{\zeta}^k_i, \boldsymbol{\tau}^k_i)$ denote such a (reduced cost, resource vector)-label corresponding to a partial path ending at node i. Unlike classical labelling algorithms for unconstrained shortest path problems, the resource constrained algorithms require to keep at each node all multi-dimensional labels that are not dominated according to the following definition.

Definition. Label E_i dominates label E'_i, denoted $E_i \prec E'_i$, if and only if $E_i \leq E'_i$.

Under the three above conditions, the label dominance definition permits the elimination of those labels that cannot be present in any optimal shortest path solution.

Proposition. *Let E^k_i and E'^k_i be the labels associated with two partial paths from $o(k)$ up to node i in G^k. Let also E^k_j and E'^k_j be two feasible labels obtained from E^k_i and E'^k_i, respectively, by adding arc $(i,j) \in A^k$. If all extension functions f^{kr}_{ij} are non-decreasing and depend solely on the resource vector $\boldsymbol{T}^k_i$, and all coefficients $b^{kr}_{hi} = 0$, then $E^k_i \prec E'^k_i$ yields $E^k_j \prec E'^k_j$.*

Proof: On the one hand, the relation $\boldsymbol{\tau}^k_i \leq \boldsymbol{\tau}'^k_i$ implies that:

$$f^{kr}_{ij}(\boldsymbol{\tau}^k_i, \cdot) \leq f^{kr}_{ij}(\boldsymbol{\tau}'^k_i, \cdot) \tag{3.96}$$

and

$$\tau^{kr}_j = \max(a^{kr}_j, f^{kr}_{ij}(\boldsymbol{\tau}^k_i, \cdot)) \leq \max(a^{kr}_j, f^{kr}_{ij}(\boldsymbol{\tau}'^k_i, \cdot)) = \tau^{kr}_j,\ \forall\, r \in R^k. \tag{3.97}$$

Hence, $\tau_j^k \leq \tau_j'^k$. On the other hand, the relation $\bar{\zeta}_i^k \leq \bar{\zeta}_i'^k$ implies that:

$$\bar{\zeta}_j^k = \bar{z}_{ij}^k(\bar{\zeta}_i^k, \tau_i^k, \tau_j^k) \leq \bar{z}_{ij}^k(\bar{\zeta}_i'^k, \tau_i'^k, \tau_j'^k) = \bar{\zeta}_j'^k \,. \tag{3.98}$$

Relations (3.96) and (3.98) are direct consequences of the non-decreasing property of functions f_{ij}^{kr}, $\forall\, r \in R^k$ and $\bar{z}_{ij}^k$, respectively. The second relation (3.97) also follows from the fact that all of the reduced cost functions are non-decreasing: therefore, the lowest cost value is attained for the smallest feasible resource values, i.e., either a_j^{kr} or the value computed by using $f_{ij}^{kr}(\cdot)$. Hence, $(\bar{\zeta}_j^k, \tau_j^k) \leq (\bar{\zeta}_j'^k, \tau_j'^k)$ and the result follows from the label dominance definition. ■

For decreasing subproblem cost functions such as the ones encountered in applications involving soft time windows (Koskosidis, Powell and Solomon, 1992), flight departure synchronization (Ioachim *et al.*, 1994) or combined inventory and ship routing (Christiansen, 1996), the label elimination process is more complex. As opposed to the previous case where a label at node i is extended to at most one label at a successor node j (the new label may be infeasible), several non-comparable new labels may need to be created when the cost extension function is decreasing in terms of (increasing) resource values. Ioachim *et al.* (1998) provides an optimal algorithm when the $\bar{g}_i^k$, $i \in V^k$, $k \in K$ are linear functions. As an example of a relatively difficult objective function in (3.37), consider the minimization of the total duration (makespan) of all solution paths

$$\sum_{k \in K} (T_{d(k)}^k - T_{o(k)}^k) \,. \tag{3.99}$$

For each subproblem $k \in K$, the cost function involves a decreasing component in terms of variable $T_{o(k)}^k$ within the interval $[a_{o(k)}^k, b_{o(k)}^k]$. However, this objective function is increasing if the starting time $T_{o(k)}^k$ is restricted to be equal to a single time-tabled value $a_{o(k)}^k = b_{o(k)}^k$. Alternatively to the method presented in Ioachim *et al.*, (1998) for the case $a_{o(k)}^k \neq b_{o(k)}^k$, one can modify the initial network by discretizing the time window at node $o(k)$. Hence, the decreasing cost component is removed by creating several copies of the source node and by assigning the resulting fixed cost directly to the new nodes.

Commodity Aggregation: When several commodities are identical, the variables corresponding to these commodities can be aggregated to reduce the number of subproblems as well as the number of variables and constraints in the master problem. To illustrate this aggregation process, consider the linear relaxation (3.75)–(3.80) of the master problem (3.75)–(3.82) and assume that all commodities are identical (otherwise, the aggregation process can be performed separately on the various subsets of identical commodities). In this case, all networks G^k, $k \in K$, are identical and so are all sets of subproblem's extreme points Ω^k, $k \in K$. Denote this common set of extreme points by Ω and the column coefficients by c_p and a_{wp} (i.e., $\Omega = \Omega^k$, $c_p = c_p^k$ and $a_{wp} = a_{wp}^k$, $\forall\, k \in K$). Define also new variables as:

$$\theta_p = \sum_{k \in K} \theta_p^k, \;\; \forall\, p \in \Omega \,.$$

Since the variables θ_p^k are non-negative, the variables θ_p can only take non-negative values. Furthermore, summing the constraints in set (3.79), we have:

$$\sum_{k\in K}\sum_{p\in\Omega}\theta_p^k = \sum_{k\in K} 1 \Leftrightarrow \sum_{p\in\Omega}\sum_{k\in K}\theta_p^k = |K| \Leftrightarrow \sum_{p\in\Omega}\theta_p = |K|\,. \tag{3.100}$$

The linear relaxation (3.75)–(3.80) is therefore equivalent to:

$$\textit{Minimize} \sum_{p\in\Omega} c_p\theta_p + \sum_{s\in S} c_s Y_s \tag{3.101}$$

subject to:

$$\sum_{p\in\Omega} a_{wp}\theta_p + \sum_{s\in S} a_{ws}Y_s = n_w, \qquad \forall\, w \in N \tag{3.102}$$

$$\sum_{p\in\Omega} b_{hp}\theta_p + \sum_{s\in S} b_{hs}Y_s = b_h, \qquad \forall\, h \in H \tag{3.103}$$

$$a_s \le Y_s \le b_s, \qquad \forall s \in S \tag{3.104}$$

$$\theta_p \ge 0, \qquad \forall\, p \in \Omega \tag{3.105}$$

$$\theta_p = \sum_{k\in K}\theta_p^k, \qquad \forall\, p \in \Omega \tag{3.106}$$

$$\sum_{p\in\Omega}\theta_p^k = 1, \qquad \forall\, k \in K \tag{3.107}$$

$$\theta_p^k \ge 0, \qquad \forall\, k \in K,\ \forall\, p \in \Omega. \tag{3.108}$$

For a solution to (3.101)–(3.105) fractional in θ_p, there exists a solution in θ_p^k that satisfies (3.106)–(3.108). Indeed, assume that θ_p, $p \in \Omega$, is a solution to (3.101)–(3.105) and assign to θ_p^k, $k \in K$, $p \in \Omega$, the values:

$$\theta_p^k = \frac{\theta_p}{|K|}, \ \forall\, k \in K,\ \forall\, p \in \Omega.$$

By using (3.100), it is then easy to verify that this solution also satisfies (3.106)–(3.108). Since any solution consisting of the aggregated variables θ_p, $p \in \Omega_p$, can be converted into a solution in terms of the disaggregated variables θ_p^k, $k \in K$, $p \in \Omega$, the problem (3.101)–(3.105) can be used as the linear master problem. This aggregated linear master problem contains $|K|$ times less variables and has fewer constraints than the original one.

In the case where the solution of the aggregated linear master problem is integer, it is possible to convert it to a binary solution in terms of the variables θ_p^k. Indeed, assume that θ_p, $p \in \Omega$, is an integer solution to (3.101)–(3.105). By (3.100), at most $|K|$ variables θ_p are positive. Let $\theta_{p_1}, \theta_{p_2}, \ldots, \theta_{p_n}$ be these variables. Next, assign path p_1 to the first θ_{p_1} commodities, path p_2 to the next θ_{p_2}, ..., and path p_n to the last θ_{p_n} commodities. These assignments translate into the following mathematical expressions:

$$\theta_{p_1}^k = \begin{cases} 1 & \text{if } 1 \le k \le \theta_{p_1} \\ 0 & \text{otherwise,} \end{cases}$$

$$\theta_{p_2}^k = \begin{cases} 1 & \text{if } \theta_{p_1} + 1 \leq k \leq \theta_{p_1} + \theta_{p_2} \\ 0 & \text{otherwise,} \end{cases}$$

$$\vdots$$

$$\theta_{p_n}^k = \begin{cases} 1 & \text{if } \sum_{i=1}^{n-1} \theta_{p_i} + 1 \leq k \leq |K| \\ 0 & \text{otherwise,} \end{cases}$$

$$\theta_p^k = 0 \text{ if } p \notin \{p_1, p_2, \ldots, p_n\} \text{ and } 1 \leq k \leq |K|.$$

One can verify that this solution also satisfies (3.106)–(3.108). Finally, if the aggregated solution is mixed integer, both of the above conversion processes need to be applied accordingly.

3.5 CONCLUSION

This paper has developed a much needed integrating framework for deterministic time constrained vehicle routing and crew scheduling problems. It consists of a more general model than previously considered which encompasses all the important practical complexities examined to date in the literature. Each distinct type of problem explored so far in this environment can be cast as a special instance of this unified formulation. A number of prior research efforts have involved the direct application of different forms of this model to solve specific problem types in this class. Furthermore, approaches that have been based on a Set Partitioning/Covering formulation and additional constraints customizing each unique application represent indirect uses of the unified model. In addition to consolidating the most widely utilized time constrained vehicle routing and crew scheduling problem models, this framework extends beyond this environment. Any problem for which the solution can be expressed as a set of paths in a deterministic state-space graph can be formulated as a special case of the proposed unified model. As an example, this formulation can be used to model one of the earliest application of column generation, namely the Cutting-Stock Problem.

To solve the multi-commodity problems in this class, we have proposed a branch-and-bound framework. The lower bounds are obtained by using a decomposition approach. This exploits the special structure of these problems involving an objective function and many of the constraints separable by commodity. While the framework leads to a number of algorithmic choices for each module, we have concentrated on an extension of the Dantzig-Wolfe decomposition principle. We have proved that it remains valid even when nonlinear objective functions and constraints are present. We have also shown that it is more general than the prior column generation-based methods as is the compatible branching module used. Finally, we have scanned the dynamic programming algorithms designed for the different constrained shortest path subproblems or the more general single commodity problems and determined some needs for future research.

We believe that the framework proposed in this paper has unified much of the previous research directed at deterministic time constrained vehicle routing and crew scheduling problems. It is our hope that it will also constitute the basis for future approaches directed at problems of larger size and at new application areas.

Acknowledgments

This research was supported by the Quebec Government (Programme Synergie du Fonds de Développement Technologique) and by the Natural Sciences and Engineering Council of Canada. Marius M. Solomon was partially supported by the Patrick F. and Helen C. Walsh Research Professorship. We would also like to thank Andreas Nöu from the Royal Institute of Technology, Stockholm, for fruitful discussions regarding the paper.

References

Appelgren, L.H. (1969). A Column Generation Algorithm for a Ship Scheduling Problem. *Transportation Science*, 3:53–98.

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

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

Barnhart, C., C.A. Hane, E.L. Johnson and G. Sigismondi. (1991). An Alternative Formulation and Solution Strategy for Multi-Commodity Network Flow Problems. Report COC-9102, Georgia Institute of Technology, Atlanta, Georgia.

Barnhart, C., L. Hatay, and E.L. Johnson. (1995). Deadhead Selection for the Long-Haul Crew Pairing Problem. *Operations Research*, 43:491–499.

Barnhart, C., E.L. Johnson, R. Anbil and L. Hatay. (1994a). A Column Generation Technique for the Long-Haul Crew Assignment Problem. In T.A. Ciriani and R. Leachman, editors, *Mathematical Programming and Modeling Techniques in Practice*, pages 7–22, Optimization in Industry 2. John Wiley and Sons, New-York.

Barnhart, C., E.L. Johnson, G.L. Nemhauser and M.W.P. Savelsbergh. (1994b). Branch and Price: Column Generation for Solving Huge Integer Programs. Computational Optimization Center, COC–94–03, Georgia Institute of Technology, Atlanta.

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.

Bramel, J. and D. Simchi-Levi. (1997). On the Effectiveness of Set Coveringing Formulations for the Vehicle Routing Problem with Time Windows. *Operations Research*, 45:295–301.

Carraresi, P. and G. Gallo. (1984). Network Models for Vehicle and Crew Scheduling. *European Journal of Operations Research*, 16:139–151.

Christiansen, M. (1996). *Inventory and Time Constrained Ship Routing - A Mathematical Approach.* Ph.D. Dissertation, Norvegian University of Science and Technology, Trondheim, Norway.

Christofides, N., A. Mingozzi, and P. Toth. (1981). Space-State Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks*, 11:145–164.

Daduna, J.R., I. Branco and J. Paixão. (eds.) (1995). Computer-Aided Transit Scheduling. *Lecture Notes in Economics and Mathematical Systems 430.* Springer Verlag, Berlin.

Dantzig, G. and D. Fulkerson. (1954). Minimizing the Numbers of Tankers to Meet a Fixed Schedule. *Naval Research Logistics Quarterly*, 1:217–222.

Dantzig, G.B. and R.M. Van Slyke. (1967). Generalized Upper Bounding Techniques. *Journal of Computer and System Sciences*, 1:213–226.

Dantzig, G.B. and P. Wolfe. (1960). Decomposition Principle for Linear Programs. *Operations Research*, 8:101–111.

Desaulniers, G., J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon and F. Soumis. (1997a). Crew Pairing at Air France. *European Journal of Operational Research*, 97:245–259.

Desaulniers, G., J. Desrosiers, M.M. Solomon and F. Soumis. (1997b). Daily Aircraft Routing and Scheduling. *Management Science*, 43:841–855.

Desrochers, M. (1986). La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes. Université de Montréal, Centre de recherche sur les Transports, Publication #470. CP 6128, Succursale "Centre-ville", Montréal, Québec, H3C 3J7. (In French)

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 M. Desrochers and J.-M. Rousseau, editors, *Computer-Aided Transit Scheduling*, pages 395–406, Lecture Notes in Economics and Mathematical Systems 386. Springer Verlag, Berlin.

Desrochers, M., J.K. Lenstra and M.W.P. Savelsbergh. (1990). Classification of Vehicle Routing and Scheduling Problems. *European Journal of Operational Research*, 46:322–332.

Desrochers, M., J.K. Lenstra, M.W.P. Savelsbergh and F. Soumis. (1988). Vehicle Routing with Time Windows: Optimization and Approximation. In B. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 65–84, North-Holland, Amsterdam.

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

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.M. Solomon and F. Soumis. (1995). Time Constrained Routing and Scheduling. In M.O. Ball *et al.*, editors, *Network Routing*, Handbooks in Operations Research and Management Science 8, pages 35–139, Elsevier Science, Amsterdam.

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., P. Hansen, B. Jaumard, F. Soumis and D. Villeneuve. (1994). Dantzig-Wolfe Decomposition and Column Generation for Linear and Nonlinear Integer Programs. Working paper. GERAD, École des HEC, Montréal, Canada, H3T 2A7.

Desrosiers, J., P. Pelletier and F. Soumis. (1983). Plus court chemin avec contraintes d'horaires, *RAIRO*, 17:357–377, (in French).

Desrosiers, J., F. Soumis and M. Desrochers. (1984). Routing with Time Windows by Column Generation. *Networks*, 14:545–565.

Dror, M. (1994). Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42:977–978.

Dror, M. and P. Trudeau. (1989). Savings by Split Delivery Routing. *Transportation Science*, 23:141–145.

Dumas, Y., J. Desrosiers, E. Gélinas and M.M. Solomon. (1995). An Optimal Algorithm for the Traveling Salesman Problem with Time Windows. *Operations Research*, 43:367–371.

Dumas, Y., J. Desrosiers and F. Soumis. (1990). Optimizing the Schedule for a Fixed Vehicle Path with Convex Inconvenience Costs. *Transportation Science*, 24:145–151.

Dumas, Y., J. Desrosiers and F. Soumis. (1991). The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 54:7–22.

du Merle, O. (1995). Points intérieurs et plans coupants: Mise en oeuvre et développement d'une méthode pour l'optimisation convexe et la programmation linéaire structurée de grande taille. Ph.D. Dissertation, Université de Genève, Switzerland.

du Merle, O., D. Villeneuve, J. Desrosiers and P. Hansen. (1997). Stabilisation dans le cadre de la génération de colonnes. Les Cahiers du GERAD, G–97–08, École des HEC, Montréal, Canada, H3T 2A7.

Etschmaier, M.M. and D.F.X. Mathaisel. (1984). Aircraft Scheduling: The State of the Art. *AGIFORS Proceedings XXIV*, 181–225.

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. (1988). Aspects of Bus Crew Scheduling Using a Set Partitioning Model. In J.R. Daduna and A. Wren, editors, *Computer-Aided Transit Scheduling*, Lecture Notes in Economics and Mathematical Systems 308, pages 91–103, Springer-Verlag, Berlin.

Farley, A.A. (1990). A Note on Bounding a Class of Linear Programming Problems, includIncluding Cutting Stock Problems. *Operations Research*, 38:922–923.

Ferland, J.A. and L. Fortin. (1989). Vehicle Routing with Sliding Time-Windows. *European Journal of Operational Research*, 38:359–378.

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

Fisher, M.L. (1994). Optimal Solution of Vehicle Routing Problems Using Minimum k-Trees. *Operations Research*, 42:626–642.

Fisher, M.L. (1995). Vehicle Routing. In M.O. Ball *et al.*, editors, *Network Routing*, Handbooks in Operations Research and Management Science 8, pages 1–33, Elsevier Science, Amsterdam.

Fisher, M.L., K.O. Jörnsten and O.B.G. Madsen. (1997). Vehicle Routing with Time Windows – Two Optimization Algorithms. *Operations Research*, 45:488–498.

Ford, L. and D.R. Fulkerson. (1962). *Flows in Networks.* Princeton University Press, Princeton, N.J.

Fulkerson, D.R. (1972). Flow Networks and Combinatorial Operations Research. In A.M. Geoffrion, editor, *Perspectives on Optimization*, pages 139–171, Addison-Wesley, Reading, Mass.

Gamache, M. and F. Soumis. (1997). A Method for Optimally Solving the Rostering Problem. In Gang Yu, editor, *Operations Research in Airline Industry*, pages 124–157, Kluwer Academic Publishers, Boston.

Gamache, M., F. Soumis, G. Marquis and J. Desrosiers. (1998a). A Column Generation Approach for Large Scale Aircrew Rostering Problem. *Operations Research.* Forthcoming.

Gamache, M., F. Soumis, D. Villeneuve, J. Desrosiers and E. Gélinas. (1998b). The Preferential Bidding System at Air Canada. *Transportation Science.* Forthcoming.

Gélinas, S. (1990). Fabrication de tournées avec rechargement. Master Thesis, École Polytechnique de Montréal, Canada, (in French).

Gélinas, S. (1998). Problèmes d'ordonnancement. Ph.D. Dissertation, École Polytechnique de Montréal, Canada. In preparation.

Gélinas, S., M. Desrochers, J. Desrosiers and M.M. Solomon. (1995). A New Branching Strategy for Time Constrained Routing Problems with Application to Backhauling. *Annals of Operations Research*, 61:91–109.

Gélinas, S. and F. Soumis. (1998). A Dynamic Programming Algorithm for Single Machine Scheduling with Ready Times and Deadlines to Minimize Total Weighted Completion. *MIS Collection* in *The Annals of Operations Research.* Forthcoming.

Geoffrion, A.M. (1974). Lagrangian Relaxation and its Uses in Linear Programming. *Mathematical Programming Study*, 2:82–112.

Gilmore, P.C. and R.E. Gomory. (1961). A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, 9:849–859.

Goffin, J.-L., J. Gondzio, K. Sarkissian and J.-P. Vial. (1997). Solving Nonlinear Multicommodity Flow Problems by the Analytic Center Cutting Plane Method. *Mathematical Programming*, 76:131–154.

Goffin, J.-L. and J.-P. Vial. (1996). Shallow, Deep and Very Deep Cuts in the Analytic Center Cutting Plane Method. Les Cahiers de GERAD, G–96–20, École des HEC, Montréal, Canada, H3T 2A7.

Graves, G.W., R.D. McBride, I. Gershkoff, D. Anderson and D. Mahidhara. (1993). Flight Crew Scheduling. *Management Science*, 39:736–745.

Gu, Z., E.L. Johnson, G.L. Nemhauser and Y. Wang. (1994). Some Properties of the Fleet Assignment Problem. *Operations Research Letters*, 15:59–71.

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

Hane, C., C. Barnhart, E.L. Johnson, R. Marsten, G.L. Nemhauser and G. Sigismondi. (1995). The Fleet Assignment Problem: Solving a Large-Scale Integer Program. *Mathematical Programming*, 70:211–232.

Hansen, P., B. Jaumard, S. Krau and O. du Merle. (1998). An l_1-norm Bundle Method for the Multisource Weber Problem. Les Cahiers de GERAD, École des Hautes Études Commerciales, Montréal, Canada, H3T 2A7. (Forthcoming).

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. In French.

Hansen, P., B. Jaumard and M. Poggi de Arragão. (1992). Mixed Integer Column Generation and the Probabilistic Maximum Satisfiability Problem. *Proceedings of IPCO2*, Carnegie-Mellon University, Pittsburgh, 165–180.

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

Holm, S. and J. Tind. (1988). A Unified Approach for Price Directive Decomposition Procedures in Integer Programming. *Discrete Applied Mathematics*, 20:205–219.

Houck Jr., D.J., J-C. Picard, M. Queyranne and R.R. Vemuganti. (1980). The Traveling Salesman Problem as a Constrained Shortest Path Problem: Theory and Computational Experience. *Opsearch*, 17:93–109.

Ioachim, I., J. Desrosiers, F. Soumis and N. Bélanger. (1994). Fleet Routing with Schedule Synchronization Constraints. Les Cahiers du GERAD, G–94–48, École des HEC, Montréal, Canada, H3T 2A7.

Ioachim, I., S. Gélinas, J. Desrosiers and F. Soumis. (1998). A Dynamic Programming Algorithm for the Shortest Path Problem with Time Windows and Linear Node Costs. *Networks*. Forthcoming.

Kohl, N. and O.B.G. Madsen. (1997). An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangean Relaxation. *Operations Research*, 45:395–406.

Kohl, N., J. Desrosiers, O.B.G. Madsen, M.M. Solomon and F. Soumis. (1998). 2-Path Cuts for the Vehicle Routing Problem with Time Windows. *Transportation Science*. Forthcoming.

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.

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.

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.

Lasdon, L.S. (1970). *Optimization Theory for Large Systems*. Collier-MacMillan, New-York.

Lavoie, S., M. Minoux and E. Odier. A New Approach of Crew Pairing Problems by Column Generation and Application to Air Transport. *European Journal of Operational Research 35*, 45–58, 1988.

Lemaréchal, C. (1989). Non Differentiable Optimization. In G.L. Nemhauser, A.G.H. Rinnooy Kan and M.J. Todol, editors, *Optimization*, Handbooks in Operations Research and Management Science 1, pages 529–572, Elsevier Science, Amsterdam.

Levin, A. (1971). Scheduling and Fleet Routing Models for Transportations Systems. *Transportation Science*, 5:232–255.

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.

Madsen, O.B.G. (1976). Optimal Scheduling of Trucks — A Routing Problem with Tight Due Times for Delivery. In H. Strobel, R. Genser and M. Etschmaier, editors, *Optimization Applied to Transportation Systems*, pages 126–136, International Institute for Applied System Analysis (IIASA), CP–77–7, Laxenburgh, Austria.

Magnanti, T. (1981). Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects. *Networks*, 11:179–214.

Mehrorta, A. and M.A. Trick. (1993). A Column Generation Approach to Graph Coloring. Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh.

Min, H. (1989). The Multiple Vehicle Routing Problem with Simultaneous Delivery and Pickup Points. *Transportation Research*, 23A:377–386.

Mingozzi, A., L. Bianco and S. Ricciardelli. (1997). Dynamic Programming Strategies for the Traveling Salesman Problem with Time Window and Precedence Constraints. *Operations Research*, 45:365–377.

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.

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.

Ribeiro, C., M. Minoux and C. Penna. (1989). An Optimal Column-Generation-with-Ranking Algorithm for Very Large Scale Set Partitioning Problems in Traffic Assignment. *European Journal of Operational Research*, 41:232–239.

Ribeiro, C. and F. Soumis. (1994). A Column Generation Approach to the Multiple Depot Vehicle Scheduling Problem. *Operations Research*, 42:41–52.

Ryan, D.M. and B.A. Forster. (1981). An Integer Programming Approach to Scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pages 269–280, North-Holland, Amsterdam.

Ryan, D.M. (1992). The Solution of Massive Generalized Set Partitioning Problems in Air Crew Rostering. *Journal of the Operational Research Society*, 43:459–467.

Savelsbergh, M.W.P. (1993). A Branch-and-Price Algorithm for the Generalized Assignment Problem. Computational Optimization Center COC–93–03, Georgia Institute of Technology, Atlanta, Georgia.

Sexton, T. and L. Bodin. (1985). Optimizing Single Vehicle Many-to-Many Operations with Desired Delivery Times: I. Scheduling. *Transportation Science*, 19:378–410.

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.

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.

Stojković, M., F. Soumis and J. Desrosiers. (1998). The Operational Airline Crew Scheduling Problem. *Transportation Science*. Forthcoming.

Sweeney, D.J. and R.A. Murphy. (1979). A Method of Decomposition for Integer Programs. *Operations Research*, 27:1128–1141.

Vance, P.H., C. Barnhart, E.L. Johnson and G.L. Nemhauser. (1994). Solving Binary Cutting Stock Problems by Column Generation and Branch-and-Bound. *Computational Optimization and Applications*, 3:111–130.

Vanderbeck, F. and L.A. Wolsey. (1996). An Exact Algorithm for IP Column Generation. *Operations Research Letters*, 19:151–159.

Wedelin, D. (1995). An Algorithm for Large Scale 0-1 Integer Programming with Applications to Airline Crew Scheduling. *Annals of Operations Research*, 57:283–301.

Wolfe, P. (1963). A Technique for Resolving Degeneracy in Linear Programming. *Journal of the Society for Industrial and Applied Mathematics*, 11:205–211.

Ziarati, K., F. Soumis, J. Desrosiers, S. Gélinas and A. Saintonge. (1997). Locomotive Assignment with Heterogeneous Consists at CN North America. *European Journal of Operational Research*, 97:281–292.

4 THE INVENTORY ROUTING PROBLEM

Ann Campbell
Lloyd Clarke
Anton Kleywegt
Martin Savelsbergh

4.1 INTRODUCTION

The role of logistics management is changing. Many companies are realizing that value for a customer can, in part, be created through logistics management (Langley and Holcomb, 1996). Customer value can be created through product availability, timeliness and consistency of delivery, ease of placing orders, and other elements of logistics service. Consequently, logistics service is becoming recognized as an essential element of customer satisfaction in a growing number of product markets today.

Vendor managed resupply is an example of value creating logistics. Vendor managed resupply is an emerging trend in logistics and refers to a situation in which a supplier manages the inventory replenishment of its customers. Vendor managed resupply creates value for both suppliers and customers, i.e., a win-win situation. Vendors save on distribution cost by being able to better coordinate deliveries to different customers, and customers do not have to dedicate resources to inventory management.

Different industries are considering implementing vendor managed resupply. Traditionally, vendor managed resupply has been high on the wish list of logistics managers in the petrochemical and industrial gas industry. More recently, the automotive industry (parts distribution) and the soft drink industry (vending machines) have entered this arena.

One reason that vendor managed resupply is receiving a lot of attention is the rapidly decreasing cost of technology that allows monitoring customers' inventories. Vendor managed resupply requires accurate and timely information about the inventory status of customers.

If vendor managed resupply is a win-win situation for both suppliers and customers, and relatively cheap monitoring technology is available, then why is vendor managed resupply not applied on a larger scale? One reason is that it is a complex task to develop a distribution strategy that minimizes the number of stockouts and at the same time realizes the potential savings in distribution costs. The task of developing such a distribution strategy is called the inventory routing problem.

In this paper, we present and discuss the inventory routing problem (IRP) and approaches for its solution. The IRP is a challenging and intriguing problem that also provides a good starting point for studying the integration of different components of the logistics value chain, i.e., inventory management and transportation. Integration of production and transportation is another hot item on the wish list of logistics managers. Traditionally, production and transportation have been dealt with separately. However, it is expected that improvements may be obtained by coordinating production and transportation. It is less obvious how to do it.

The purpose of this paper is to introduce the IRP, to discuss its intrinsic complexity, to review some of the methods that have been proposed for its solution, and to present two new approaches that we are currently investigating.

The remainder of the paper is organized as follows. In Section 4.2, we formally define the IRP. In Sections 4.3 and 4.4, we take a closer look at single and two-customer problems. In Section 4.5, we review the literature. In Section 4.6, we propose two new solution approaches. In Section 4.7, we address some practical issues. Finally, in Section 4.8, we propose the creation of set of standard test problems.

4.2 THE INVENTORY ROUTING PROBLEM

The IRP is concerned with the repeated distribution of a single product, from a single facility, to a set of n customers over a given planning horizon of length T, possibly infinity. Customer i consumes the product at a given rate u_i (volume per day) and has the capability to maintain a local inventory of the product up to a maximum of C_i. The inventory at customer i is I_i at time 0. A fleet of m homogeneous vehicles, with capacity Q, is available for the distribution of the product. The objective is to minimize the average distribution costs during the planning period without causing stockouts at any of the customers.

Three decisions have to be made:

- When to serve a customer?
- How much to deliver to a customer when it is served?
- Which delivery routes to use?

The IRP differs from traditional vehicle routing problems because it is based on customers' usage rather than customers' orders.

The IRP defined above is deterministic and static due to our assumption that usage rates are known and constant. Obviously, in real-life, the problem is stochastic and dynamic. Therefore, an important variant of the IRP is the stochastic inventory routing problem (SIRP). The SIRP differs from the IRP in that the future demand of a customer is uncertain. In the SIRP, we are given the probability distribution of the demand u_{it} of customer i between decision points t and $t+1$ for $t = 1, ..., T-1$. Because future demand is uncertain, there is often a positive probability that a customer runs

out of stock, i.e, stockouts cannot always be prevented. Stockout costs can be modeled in various ways. We suggest a penalty function with a fixed as well as a variable component, i.e., $S_i + s_i d$, where S_i is a fixed stockout cost, s_i is a variable stockout cost (per unit shortage), and d is the shortage, i.e., the demand between the time of stockout and the replenishment delivery. The objective is to choose a delivery policy that minimizes the average cost per unit time, or the expected total discounted cost, over the planning horizon.

To gain a better understanding of the IRP and SIRP, as well as the difference between them, we analyze single and two-customer problems in the next two sections.

4.3 THE SINGLE CUSTOMER PROBLEM

The single customer analysis also applies when we have multiple customers but always visit only a single customer on a vehicle trip (direct delivery), and we have a sufficent number of vehicles to visit all customers that we want to visit in a day.

First, we consider the IRP. Let the usage rate of the customer be u, the tank capacity of the customer be C, the initial inventory level be I, the delivery cost to the customer be c, the vehicle capacity be Q, and the planning horizon be T.

It is easy to see that an optimal policy is to fill up the tank precisely at the time it becomes empty. Therefore the cost v_T for a planning period of length T is

$$v_T = \max\left(0, \left\lceil \frac{Tu - I}{\min(C, Q)} \right\rceil\right) c.$$

Next, we consider the SIRP in which we decide daily whether to make a delivery to the customer or not. The demand U between consecutive decision points, i.e., the demand per day, is a random variable with known probability distribution.

Jaillet *et al.* (1997) analyze the "d-day" policy that makes a delivery to the customer every d days and delivers as much as possible, unless a stockout occurs earlier. When a stockout occurs earlier, the truck is sent right away which incurs a cost S. It is assumed that deliveries are instantaneous, so no additional stockout penalties are incurred. Let p_j be the probability that a stockout first occurs on day j ($1 \leq j \leq d-1$). Then $p = p_1 + p_2 + \ldots + p_{d-1}$ is the probability that there is a stockout and $1 - p$ is the probability that there is no stockout in the period $[1, ..., d-1]$. Furthermore, let $v_T(d)$ be the expected total cost of this policy over a planning period of length T. We now have for $d > T$

$$v_T(d) = \sum_{1 \leq j \leq T} p_j (v_{T-j}(d) + S)$$

and for $d \leq T$

$$v_T(d) = \sum_{1 \leq j \leq d-1} p_j (v_{T-j}(d) + S) + (1 - p)(v_{T-d}(d) + c).$$

As a consequence, the expected total cost of filling up a customer's tank every d days over a T-day period ($T \geq d$) is given by

$$v_T(d) = \alpha(d) + \beta(d)T + f(T, d)$$

where $\alpha(d)$ is a constant depending only on d, $f(T,d)$ a function that goes to zero exponentially fast as T goes to ∞, and

$$\beta(d) = \frac{pS + (1-p)c}{\sum_{1 \le j \le d} jp_j},$$

with $p_d = 1 - p$. The value $\beta(d)$ is the long-run average cost per day. To find the best policy in this class, we need to minimize $v_T(d)$, which for large T means finding a d for which $\beta(d)$ is minimum.

The above d-day policies have the advantage that they can be used even if the inventory at the customer cannot be measured and we are informed only when a stockout occurs. The d-day policies have a number of disadvantages though. The first is that a d-day policy is not optimal in general if the inventory at the customer can be measured. Intuitively it is clear that policies that use information on the amount of inventory at the customer can do better than d-day policies. We give an example below in which the best d-day policy is compared with the optimal policy. The second disadvantage is that the stockout probabilities p_j used in the analysis of d-day policies are very hard to obtain, and may not be well defined, unless the inventory at the customer is always replenished up to the same level (for example, if the vehicle capacity is at least as large as the customer's storage capacity). The reason is that the probability p_j of a stockout exactly j days after the previous replenishment depends on the inventory level after replenishment.

To compare the best d-day policy with the optimal policy, consider a customer whose demand is uniformly distributed on the integers from 1 to 20. There is a fixed cost of 40 to replenish the customer, and an additional penalty of 50 each day that the customer experiences a shortage. Figure 4.1 shows the long-run average cost per day of the best d-day policy, as a function of the customer's storage capacity C, for different values of the vehicle capacity Q. Figure 4.2 shows the optimal long-run average cost per day, again as a function of the customer's storage capacity C, for different values of the vehicle capacity Q. If the vehicle capacity $Q = 10$, then the customer is visited almost every day under the optimal policy, and the best d-day policy has $d^* = 1$, i.e., the customer is visited every day. The long-run average cost per day is therefore almost the same for the optimal policy and the best d-day policy. However, if the vehicle capacity Q is larger than 10, then the optimal policy benefits from the greater flexibility of making the replenishment decision depend on the inventory at the customer. For example, if the vehicle capacity is 14 or 16, the customer is visited on average about twice every three days under the optimal policy, while a d-day policy visits the customer every day or every other day.

If the vehicle capacity Q is at least as large as the customer's storage capacity C, then it can be shown that there is an optimal policy π^* with a threshold l^*, such that if the inventory at the customer is less than l^*, then it is optimal to replenish the customer's inventory up to the customer's storage capacity C, and if the inventory at the customer is more than l^*, then it is optimal not to replenish. The proof of this result is similar to the proof of the optimality of (s,S) policies in classical inventory theory. We are currently working on extending these results to more general problem settings.

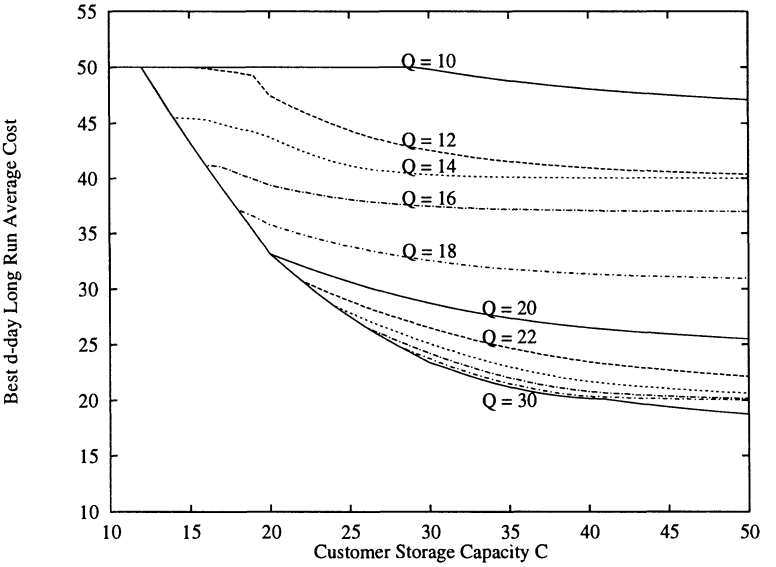


Figure 4.1 Long-run average cost per day of the best d-day policy

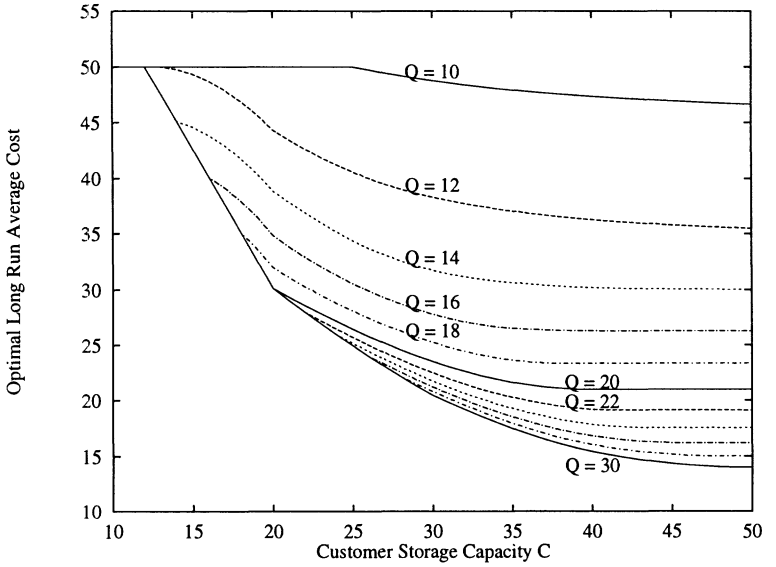


Figure 4.2 Long-run average cost per day of the optimal policy

4.4 THE TWO-CUSTOMER PROBLEM

When more than one customer is served, the problem becomes significantly harder. Not only do we have to decide which customers to visit next, but also how to combine

them into vehicle tours, and how much to deliver to each customer. Even if there are only two customers, these decisions may not be easy.

In a two-customer IRP, there are two extreme solutions: visit each customer by itself each time, and always visit both customers together. It is easy to express the cost associated with these solutions, where for simplicity we have assumed that $I_1 = I_2 = 0$:

$$v_T = \max\left(0, \left\lceil \frac{Tu_1}{\min(C_1, Q)} \right\rceil\right) c_1 + \max\left(0, \left\lceil \frac{Tu_2}{\min(C_2, Q)} \right\rceil\right) c_2,$$

where we have implicitly assumed that we can either implement the solution with one vehicle or that we have two vehicles, and

$$v_T = \left\lceil \frac{T}{\min(\frac{C_1}{u_1}, \frac{C_2}{u_2}, \frac{Q}{u_1+u_2})} \right\rceil c_{12},$$

where c_{12} denotes the cost of the tour through both customers.

It is easy to figure out which of these two extreme strategies is the best. However, there are other strategies possible: sometimes visit the customers together, and sometimes visit them by themselves. Intuitively, we expect that when one customer has a much higher usage rate or a much smaller tank size than the other, we would visit that customer by itself several times and occasionally visit the two of them together. However, what if this customer cannot take a full truckload? Or, what if the two customers are close together? And, if we visit them together how much do we deliver to each of them? We soon realize that the answer is not so obvious.

When the two customers are visited together, it is intuitively clear that given the amount delivered at the first customer, it is optimal to deliver as much as possible at the second customer (determined by the remaining amount in the vehicle, and the remaining capacity at the second customer). Thus the problem of deciding how much to deliver to each customer involves a single decision. However, making that decision may not be easy, as the following two-customer SIRP example shows.

The product is delivered and consumed in discrete units. Each customer has a storage capacity of 20 units. The daily demands of the customers are independent and identically distributed (across customers as well as across time), with $P[\text{demand} = 0] = 0.4$ and $P[\text{demand} = 10] = 0.6$. The shortage penalty is $s_1 = 1000$ per unit shortage at customer 1 and $s_2 = 1005$ per unit shortage at customer 2. The vehicle capacity is 10 units.

Every morning the inventory at the two customers is measured, and the decision maker decides how much to deliver to each customer. There are three possible vehicle tours, namely tours exclusively to customers 1 and 2, with costs of 120 each, and a tour to both customers 1 and 2, with a cost of 180. Only one vehicle tour can be completed per day.

This situation can be modeled as an infinite horizon Markov decision process, with objective to minimize the expected total discounted cost. Due to the small size of the state space, it is possible to compute the optimal expected value and the optimal policy.

Figure 4.3 shows the expected value (total discounted cost) as a function of the amount delivered at customer 1 (and therefore also at customer 2), when the inventory

at each customer is 7, and both customers are to be visited in the next vehicle tour (which is the optimal decision in the given state). It shows that the objective function is not unimodal, with a local minimum at 3, and a global minimum at 7. Consequently, just to decide how much to deliver to each customer may require solving a nonlinear optimization problem with a nonunimodal objective function. This is a hard problem, for which improving search methods are not guaranteed to lead to an optimal solution.

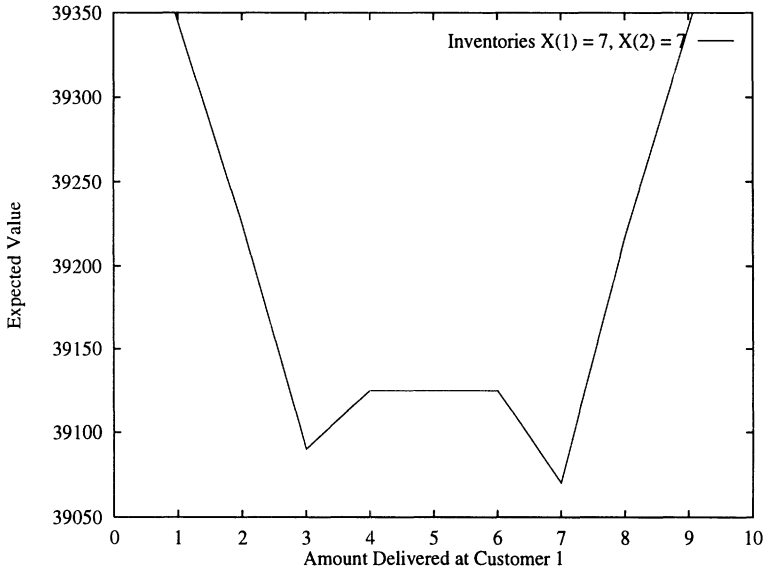


Figure 4.3 Nonunimodal objective function for determining the optimal delivery quantity

It is also interesting to observe that it is optimal to deliver more at customer 1 than at customer 2, although the shortage penalty at customer 2 is higher than the shortage penalty at customer 1, and all other data, including demand probabilities, costs, and current inventories, are the same for the two customers. However, this decision makes sense when we look ahead at possible future scenarios. If in the next time period, customer 1 uses 10 units and customer 2 uses 0 units (with probability 0.24), then at the next decision point the inventories will be 4 and 10 units respectively, and the vehicle will replenish 10 units at customer 1. In all other cases (with probability 0.76), the vehicle will replenish 10 units at customer 2 in the next time period. Thus, in all cases, the vehicle will visit only one customer in the next time period, and it is more than three times as likely to be customer 2. Also, in all cases customer 2 will have 10 or more units in inventory after the delivery in the next time period, whereas customer 1 will have only 4 units in inventory with probability 0.36. This illustrates the importance of looking ahead more than one time period when choosing the best action.

4.5 SOLUTION APPROACHES

Before discussing some of the solutions approaches that have been proposed and discussed in the literature, we present some general observations about the inventory routing problem and some common elements found in most solution approaches.

The inventory routing problem is a long-term dynamic control problem. This long-term control problem is already hard to formulate, it is almost impossible to solve. Therefore, almost all approaches that have been proposed and investigated up to now solve only a short-term planning problem. In early work, short-term was often just a single day, later short-term was expanded to a couple of days.

Two key issues need to be resolved with all of these approaches: how to model the long-term effect of short-term decisions, and which customers to include in the short-term planning period.

A short-term approach has the tendency to defer as many deliveries as possible to the next planning period, which may lead to an undesirable situation in the next planning period. Therefore, the proper projection of a long-term objective into a short-term planning problem is essential. The long-term effect of short-term decisions needs to capture the costs and benefits of delivering to a customer earlier than necessary. Delivering earlier than necessary, which usually means delivering less than a truck load, may lead to higher future distribution costs, but it reduces the risk of a stockout and may thus reduce future shortage costs.

We can distinguish two short-term approaches. In the first, it is assumed that all customers included in the short-term planning period have to be visited. In the second, it is assumed that customers included in the short-term planning period may be visited, but the decision whether or not to actually visit them still has to be made.

Decisions regarding who needs to be visited and how much should be delivered are usually guided by the following assumptions about what constitute good solutions:

- Always try to maximize the quantity delivered per visit.
- Always try to send out trucks with a full load.

When the short-term planning problem consists of a single day, the problem can be viewed as an extension of the vehicle routing problem (VRP) and solution techniques for the VRP can be adapted. Single day approaches usually base decisions on the latest inventory measurement and maybe on a predicted usage for that day. Therefore, they avoid the difficulty of forecasting long-term usage, which makes the problem much simpler.

When the short-term planning problem consists of several days, the problem becomes harder, but has the potential to yield much better solutions. Typically the resulting short-term problems are formulated as mathematical programs and solved using decomposition techniques, such as Lagrangean relaxation.

4.5.1 Literature review

It is not our intention to provide a comprehensive review of the literature, but rather to discuss papers that are representative of the solution approaches that have been proposed and investigated.

Federgruen and Zipkin (1984) approach the inventory routing problem as a single day problem and capitalize on many of the ideas from vehicle routing. Their

version of the problem has a plant with a limited amount of available inventory and the demands per day at a customer are assumed to be a random variable. For a given day, the problem is to allocate this inventory among the customers so as to minimize transportation costs plus inventory and shortage costs at the end of the day (after the day's usage and receipt of the day's delivery). Federgruen and Zipkin model the problem as a nonlinear integer program. Because of the inventory and shortage costs and the limited amount of inventory available, not every customer will be selected to be visited every day. This is handled in the model by the use of a dummy route that includes all the customers not receiving a delivery. The nonlinear integer program has the property that for any assignment of customers to routes, the problem decomposes into an inventory allocation problem which determines the inventory and shortage costs and a TSP for each vehicle which yields the transportation costs. This property is the key to the solution approach taken. The idea is to construct an initial feasible solution and iteratively improve the solution by exchanging customers between routes. Obviously, evaluating such exchanges is more computationally intensive than in standard vehicle routing algorithms. Each exchange defines a new customer to route assignment, which in turn defines a new inventory allocation problem and new TSPs.

Golden, Assad, and Dahl (1984) develop a heuristic that tries to minimize costs on a single day while maintaining an "adequate" inventory at all customers. The heuristic starts with computing the 'urgency' of each customer. The urgency is determined by the ratio of tank inventory level to tank size. All customers with an urgency smaller than a certain threshold are excluded. Next, customers are selected to receive a delivery one at a time according to the highest ratio of urgency to extra time required to visit this customer. A large TSP tour is iteratively constructed. Initially, a time limit for the total travel time of the tour, say TMAX, is set to the number of vehicles multiplied by the length of a day. Customers are added until this limit is reached or there are no more customers left. The final tour is partitioned into a set of feasible routes by enforcing that each customer must be filled up when it receives a delivery. If this turns out to be impossible, the heuristic can be re-run with a smaller value for TMAX.

Chien, Balakrishnan, and Wong (1989) also develop a single day approach, but theirs is distinctly different from that of Golden, Assad, and Dahl (1984), because it does not treat each day as a completely separate entity. By passing information from one day to the next, the system simulates a multiple day planning model. Assuming that the maximum usage per day for each customer is known, the daily profit can be defined in terms of a revenue per unit delivered and a penalty per unit of unsatisfied demand (lost revenue). Their heuristic tries to maximize the total profit on a single day. Once a solution for one day is found, the results are used to modify the revenues for the next day. Unsatisfied demand today is reflected by an increased revenue tomorrow. An integer program is created that handles the allocation of the limited inventory available at the plant to the customers, the customer to vehicle assignments, and the routing. A Lagrangean dual ascent method is used to solve the integer program.

Fisher *et al.* (1982) and Bell *et al.* (1983) study the inventory routing problem at Air Products, a producer of industrial gases. The objective considered is profit maximization from product distribution over several days. Rather than considering

demand to be a random variable or completely deterministic, demand is given by upper and lower bounds on the amount to be delivered to each customer for every period in the planning horizon. An integer program is formulated that captures delivery volumes, assignment of customers to routes, assignments of vehicles to routes, and assignment of start times for routes. This integer program is solved using a Lagrangean dual ascent approach.

In two companion papers, Dror, Ball and Golden (1985), Dror and Ball (1987) propose a way to take into consideration what happens after the short-term planning period. Using the probability that a customer will run out on a specific day in the planning period, the average cost to deliver to the customer, and the anticipated cost of a stockout, the optimal replenishment day t^* minimizing the expected total cost can be determined for each customer. If t^* falls within the short-term planning period, the customer will definitely be visited, and a value c_t is computed for each of the days in the planning period that reflects the expected increase in future cost if the delivery is made on day t instead of on t^*. If t^* falls outside the short-term planning period a future benefit g_t can be computed for making a delivery to the customer on day t of the short-term planning period. These computed values reflect the long term effects of short term decisions. An integer program is then solved that assigns customers to a vehicle and a day, or just day, that minimizes the sum of these costs plus the transportation costs. This leaves either TSP or VRP problems to solve in the second stage.

Some of the ideas of Dror and Ball are extended and improved in Trudeau and Dror (1992). Dror and Levy (1986) use a similar analysis to yield a weekly schedule, but then apply node and arc exchanges to reduce costs in the planning period.

Jaillet *et al.* (1997) and Bard *et al.* (1998a, 1998b) discuss an extension of this idea. They take a rolling horizon approach to the problem by determining a schedule for two weeks, but only implementing the first week. The scenario includes a central depot and customers that need replenishing to prevent stockout, but also included is the idea of satellite facilities. Satellite facilities are locations other than the depot where trucks can be refilled. An analysis similar to Dror and Ball's is done to determine an optimal replenishment day for each customer, which translates to a strategy for determining how often that customer should receive a delivery. A key difference is that only customers that have an optimal replenishment day within the next two weeks are included in the schedule. Incremental costs are computed that are the cost for changing the next visit to a customer to a different day but keeping the optimal schedule in the future. These costs are used in an assignment problem formulation that assigns each customer to a day in the two week planning horizon. This again yields a VRP for each day, but only the first week is actually routed. At the beginning of the next week, the problem will be solved again for the next two week horizon.

A slight variation on the inventory routing problem is the strategic inventory routing problem discussed by Webb and Larson (1995) and is related to Larson's earlier work on scheduling ocean vessels (1988). For many companies, the fleet of vehicles needs to be purchased or leased months or even years before actual deliveries to customers start taking place. The strategic inventory routing problem seeks to find the minimum fleet size to service the customers from a single depot. This determination is based on information currently known about customers' usage rates. Consequently, this minimum fleet size must be able to handle a reasonable amount of variation in

these usage rates. The fleet size estimate is determined by separating the customers into disjoint clusters and creating a route sequence for each cluster. A route sequence is a permanent set of repeating routes. Customers are allowed to be on more than one route in the sequence. The route sequences are created using a savings approach that maximizes vehicle utilization, which effectively minimizes the number of vehicles.

Anily and Federgruen (1990, 1991) look at minimizing long run average transportation and inventory costs by determining long term routing patterns. The routing patterns are determined using a modified circular partitioning scheme. After the customers are partitioned, customers within a partition are divided into regions so as to make the demand of each region roughly equal to a truckload. A customer may appear in more than one region, but then a certain percent of his demand is allocated to each region. When one customer in a region gets a visit, all customers in the region are visited. They also determine a lower bound for the long run average cost to be able to evaluate how good their routing patterns are.

Using ideas similar to those of Anily and Federgruen, Gallego and Simchi-Levi (1990) evaluate the long run effectiveness of direct shipping (separate loads to each customer). They conclude that direct shipping is at least 94% effective over all inventory routing strategies whenever minimal economic lot size is at least 71% of truck capacity. This shows that direct shipping becomes a bad policy when many customers require significantly less than a truckload, making more complicated routing policies the appropriate choice.

Another adaptation of these ideas can be found in Bramel and Simchi-Levi (1995). They consider the variant of the IRP in which customers can hold an unlimited amount of inventory. To obtain a solution, they transform the problem to a capacitated concentrator location problem (CCLP), solve the CCLP, and transform the solution back into a solution to the IRP. The solution to the CCLP will partition the customers into disjoint sets, which in the inventory routing problem, will become the fixed partitions. These partitions are then served similar to the regions of Anily and Federgruen. Chan, Federgruen, and Simchi-Levi (1998) analyze zero-inventory ordering policies for this problem setting and derive asymptotic worst-case bounds on their performance.

Minkoff (1993) formulated the stochastic inventory routing problem as a Markov decision process. He focused on the case with an unlimited number of vehicles. To overcome the computational difficulties caused by large state spaces, he proposed a decomposition heuristic. The heuristic solves a linear program to allocate joint transportation costs to individual customers, and then solves individual customer subproblems. The value functions of the subproblems are added to approximate the value function of the combined problem.

4.6 SOLUTION APPROACHES UNDER INVESTIGATION

In the next two subsections, we propose two new solution approaches that we are currently investigating.

4.6.1 An integer programming approach for the IRP

We have developed a two-phase algorithm for the IRP. In the first phase, we determine when and how much to deliver to each customer on each day of the planning period.

In the second phase, given that we know how much to deliver to each customer on each day of the planning period, we determine sets of delivery routes for each day.

At the heart of the first phase is an integer program. Define the following two quantities: $L_i^t = \max(0, tu_i - I_i^0)$, i.e., a lower bound on the total volume that has to be delivered to customer i by day t, and $U_i^t = tu_i + C_i - I_i^0$, i.e., an upper bound on the total volume that can be delivered to customer i up to day t. If d_i^t represents the delivery volume to customer i on day t, then to help ensure that no stockout occurs and that inventory capacity is not exceeded at customer i, we like to have that

$$L_i^t \leq \sum_{1 \leq s \leq t} d_i^s \leq U_i^t \qquad \forall i \, \forall t.$$

The total volume that can be delivered on a single day is limited by a combination of capacity and time constraints. Since vehicles are allowed to make multiple trips per day, we cannot simply limit the total volume delivered on a given day to the sum of the vehicle capacities.

The best way to model the resource constraints with some degree of accuracy and to specify a meaningful objective function is to explicitly use delivery routes. Therefore, let r represent a possible delivery route, T_r the duration of route r (as a fraction of a day), and c_r the cost of executing route r. Furthermore, let x_r^t be a 0-1 variable indicating whether route r is used on day t or not, and d_{ir}^t be a continuous variable representing the delivery volume to customer i on route r on day t. Then the resource constraints can be modeled as

$$\sum_{i:i \in r} d_{ir}^t \leq Q x_r^t \qquad \forall r \, \forall t,$$

$$\sum_r T_r x_r^t \leq m \qquad \forall t,$$

which ensures that we do not exceed the vehicle capacity on any of the selected routes and that the time required to execute the selected routes does not exceed the time available.

Consequently, the overall phase I model is given by

$$\min \sum_t \sum_r c_r x_r^t$$

$$L_i^t \leq \sum_{1 \leq s \leq t} \sum_{r:i \in r} d_{ir}^s \leq U_i^t \qquad \forall i \, \forall t,$$

$$\sum_{i:i \in r} d_{ir}^t \leq Q x_r^t \qquad \forall r \, \forall t,$$

$$\sum_r T_r x_r^t \leq m \qquad \forall t.$$

This model is not very practical for two reasons: the huge number of possible delivery routes and, although to a lesser extent, the length of the planning horizon. To make this integer program computationally tractable we only consider a small (but good) set of delivery routes and aggregate time periods towards the end of the planning horizon.

Aggregation is achieved by considering weeks rather than days towards the end of the planning horizon and using continuous variables rather than binary variables. To handle the approximate nature of the final part of the plan obtained this way, we embed the algorithm in a rolling horizon framework in which the algorithm is invoked every k days and only the first k days of the solution are actually implemented.

Our approach to selecting a small but good set of delivery routes is based on the concept of *clusters*. A cluster is a group of customers that can be served cost effectively by a single vehicle for a long period of time. Note that the cost of serving a cluster does not only depend on the geographic locations of the customers in the cluster, but also on whether the customers in the cluster have compatible inventory capacities and usage rates. After determining a set of disjoint clusters covering all customers, we consider only routes visiting customers in the same cluster.

The following approach is used to identify a good set of disjoint clusters covering all customers:

1. Generate a large set of possible clusters.
2. Estimate the cost of serving each cluster.
3. Solve a set partitioning problem to select clusters.

We use heuristic rules, mainly based on usage considerations, to limit the number of possible clusters. For example, five customers that all need a full truck load delivery per day will not be combined to form a cluster.

We estimate the cost of serving a cluster by solving an integer program. Let c_r denote the cost of an optimal route r through a subset of the customers in a cluster. Define the following variables. The total volume y_{ir} delivered to customer i on route r in the planning period and the route count z_r, and consider the following model

$$\min \sum_r c_r z_r$$

subject to

$$\sum_{i:i\in r} y_{ir} \le \min(Q, \sum_{i:i\in r} C_i) z_r \quad \forall r,$$

$$y_{ir} \le \min(Q, C_i) z_r \quad \forall r, \forall i \in r,$$

$$\sum_{r:i\in r} y_{ir} = T u_i \quad \forall i,$$

$$z_r \text{ integer}, y_{ir} \ge 0,$$

which ensures that the total volume delivered on route r in the planning period is less than or equal to the minimum of the vehicle capacity and the total storage capacity times the number of times route r was executed, that we do not deliver more to a customer than the minimum of the vehicle capacity and its tank capacity times the number of times route r was executed, and that the total volume delivered to a customer in the planning period is equal to its total usage during the planning period.

Note that the number of routes in a cluster is relatively small which makes this integer program relatively easy to solve. Furthermore, determining a set of disjoint

clusters has to be done only once as a preprocessing step before the actual planning starts.

The solution to the phase I model tells us how much to deliver to each customer for the next k days. This information is converted to a vehicle routing problem with time windows (VRPTW) for each day as follows. For each customer i, the inventory I_i^{t-1} at the start of day t can be computed as $I_i^0 + \sum_{1 \le s \le t-1} d_i^s - (t-1)u_i$. The time window $[a_i^t, b_i^t]$ for customer i on day t is set to guarantee that the delivery d_i^t can be made, i.e., $a_i^t = \max(0, (d_i^t - (C_i - I_i^{t-1}))/u_i)$ and $b_i^t = \min(24, I_i^{t-1}/u_i)$. We use a standard algorithm for the VRPTW to solve these instances.

For ease of exposition, we have ignored many of the important practical issues, such as dispense times at customers and refilling times at the facility. All these can be handled without complicating the model too much.

4.6.2 A dynamic programming approach for the SIRP

We model the SIRP as a discrete time Markov decision process (MDP). At the beginning of each day, the inventory at each customer is measured. Then a decision is made regarding which customers' inventories to replenish, how much to deliver to each customer, how to combine customers into vehicle tours, and which vehicle tours to assign to each of the vehicles. We call such a decision an itinerary. A vehicle can perform more than one tour per day, as long as all tours assigned to a vehicle together do not take more than a day to complete. Thus, all vehicles are available at the beginning of each day, when the tasks for that day are assigned. Although usage typically occurs throughout the day, and each customer's inventory therefore varies during the day, we assume that each customer's inventory is measured only at the beginning of the day, before decisions are made, and the state of the MDP is updated accordingly. The expected cost is computed taking into account the variation in inventory during the day, and the probability of stockout before the vehicle arrives at the customer's site.

We focus on the infinite horizon MDP; the finite horizon case can be treated in a similar way. The MDP has the following components:

1. The state x is the current inventory at each customer. Thus the state space $\mathcal{X}$ is $[0, C_1] \times [0, C_2] \times \cdots \times [0, C_n]$. Let $X_t \in \mathcal{X}$ denote the state at time t.

2. The action space $\mathcal{A}(x)$ for each state x is the set of all itineraries that satisfy the tour duration constraints, such that the vehicles' capacities are not exceeded, and the customers' storage capacities are not exceeded after deliveries. Let $\mathcal{A} \equiv \bigcup_{x \in \mathcal{X}} \mathcal{A}(x)$ denote the set of all itineraries. Let $A_t \in \mathcal{A}(X_t)$ denote the itinerary chosen at time t.

3. The known demand probability distribution gives a known Markov transition function Q, according to which transitions occur, i.e., for any state $x \in \mathcal{X}$, and any itinerary $a \in \mathcal{A}(x)$,

$$P[X_{t+1} \in B \mid X_t = x, A_t = a] = \int_B Q[dy \mid x, a]$$

4. Two costs are taken into account, namely transportation costs, which depend on the vehicle tours chosen, and a penalty when customers run out of inventory.

Let $c(x, a)$ denote the expected daily cost incurred if the process is in state x at the beginning of the day, and itinerary $a \in \mathcal{A}(x)$ is implemented.

5. The objective is to minimize the expected total discounted cost over an infinite horizon ($T = \infty$). Let $\alpha \in [0, 1)$ denote the discount factor. Let $V^*(x)$ denote the optimal expected cost given that the initial state is x, i.e.,

$$V^*(x) \equiv \inf_{\{A_t\}_{t=0}^{\infty}} E\left[\sum_{t=0}^{\infty} \alpha^t c(X_t, A_t) \middle| X_0 = x\right] \tag{4.1}$$

The actions A_t are restricted such that $A_t \in \mathcal{A}(X_t)$ for each t, and A_t has to depend only on the history $(X_0, A_0, X_1, \ldots, X_t)$ of the process up to time t, i.e., when we decide on an itinerary at time t, we do not know what is going to happen in the future.

Under certain conditions that are not very restrictive, the optimal expected cost in (4.1) is achieved by the class of stationary policies Π, which is the set of all functions that depend only on the current state and return an admissible itinerary for the current state. That is, a stationary policy $\pi \in \Pi$ is a function $\pi : \mathcal{X} \mapsto \mathcal{A}$, such that $\pi(x) \in \mathcal{A}(x)$ for all $x \in \mathcal{X}$. It follows that for any $x \in \mathcal{X}$,

$$\begin{aligned} V^*(x) &= \inf_{\pi \in \Pi} E\left[\sum_{t=0}^{\infty} \alpha^t c(X_t, \pi(X_t)) \middle| X_0 = x\right] \\ &= \inf_{a \in \mathcal{A}(x)} \left\{ c(x, a) + \alpha \int_{\mathcal{X}} V^*(y) Q[dy \mid x, a] \right\}. \end{aligned} \tag{4.2}$$

To determine an optimal policy, we need to solve the optimality equation (4.2). The three major computational requirements involved in solving (4.2) are the following.

1. Estimating the optimal cost function V^*.
2. Estimating the integral in (4.2).
3. Solving the minimization problem on the right hand side of (4.2) to determine the optimal itinerary for each state.

Rarely can these three computational tasks be completed sequentially. Usually an iterative procedure has to be used.

A number of algorithms has been developed to solve the optimality equation (to within a specified tolerance ε) if $\mathcal{X}$ is finite and the optimization problem on the right hand side can be solved in finite time (to within a specified tolerance δ). Examples are value iteration or successive approximation, policy iteration, and modified policy iteration. These algorithms are practical only if the state space $\mathcal{X}$ is small, and the optimization problem on the right hand side can be solved efficiently. None of these requirements are satisfied by practical instances of the SIRP, as the state space $\mathcal{X}$ is usually extremely large, even if customers' inventories are discretized, and the optimization problem on the right hand side has a vehicle routing problem as a special case, which is NP-hard. Our approach is therefore to develop approximation methods based on the MDP formulation above.

One approach is to approximate the optimal cost function $V^*(x)$ with a function $\hat{V}(x, \beta)$ that depends on a vector of parameters β. Some of the issues to be addressed when using this approximation method are the following.

1. The functional form of the approximating function $\hat{V}$. This may be the most important step in the approximation method, and also the one in which an intuitive understanding of the nature of the problem and the optimal value function plays the greatest role. A fair amount of experimentation is needed to develop and test different approximations. Functions $\hat{V}$ that are linear in β have the advantage that estimation algorithms for β with good theoretical properties have been developed, as discussed below.

2. Computational methods to estimate good values for the parameters β. Bertsekas and Tsitsiklis (1996) discuss a number of simulation based methods. They develop policy evaluation algorithms for which the parameter estimates β_t converge as $t \to \infty$, if $\hat{V}$ is linear in β, and the usual conditions for the convergence of many stochastic approximation methods hold. In addition, β_t converges to parameters β^π that give a best fit of the expected value function V^π under stationary policy π, if the errors are weighted by the invariant distribution under policy π. However, many of the algorithms exhibit undesirable behavior, and many theoretical properties of these approximation methods remain to be established.

3. The integral in (4.2) can be computed explicitly only for some simple demand distributions. If the number of customers is small ($n \leq 8$), numerical integration can be used. If the demand distributions are more complex, and the number of customers is larger, simulation is usually the most efficient method to evaluate the integral.

4. Methods have to be developed to solve the minimization problem on the right hand side of (4.2). This optimization problem probably requires significant computational effort to solve to optimality, because it involves determining delivery quantities as well as vehicle routes. Therefore, it seems that heuristic methods have to be developed to find good solutions. Such a heuristic has to provide a good trade-off between computational speed and solution quality, as the optimization problem has to be solved thousands of times while estimating the parameters β, and the quality of the eventual approximation $\hat{V}$ and associated policy $\hat{\pi}$ may depend to a large extent on the quality of the heuristic solutions to the minimization problem.

4.7 PRACTICAL ISSUES

A number of important issues that occur in practice, and that have not been discussed above, are addressed in this section.

Usage rates are assumed to be constant in the IRP and probability distributions of the demands between consecutive decision points are assumed to be known in the SIRP. In practice, the usage rates or the probability distributions of the demands are typically not known, but have to be estimated from inventory measurements. Often these data are not collected at regular intervals, and thus it may not be easy

to convert them to usage rates or probability distributions of demands. The data are also subject to other sources of noise, such as measurement errors, which cause several statistical problems. These estimation problems have to be resolved before an IRP or SIRP can be solved in practice. Furthermore, the models ignore the typical time varying characteristics of usage, such as weekly and seasonal cycles, and any dependence between the usage on successive days.

Currently the costs involved in making inventory measurements are not insignificant, and these measurements are usually made at most once per day. One should be able to obtain fairly accurate estimates of the inventory levels at times between measurements based on the most recent measurements and past data of usage rates. Exactly how to do this estimation has to be addressed. A related problem may be to determine an optimal policy for making these costly measurements. However, it is expected that the technology will soon be available to continuously track customers' inventories at very low cost. Therefore, in the SIRP the inventories are modeled as known at the times that decisions are made, and customers' future demands are modeled as random.

The models presented manage only a single resource, namely "vehicles", to perform distribution tasks. In practice, other resources are required as well, for example drivers. The work rules that apply to drivers are quite different from those that apply to vehicles; for example, a vehicle can work more hours per day than a driver. The assignment of customers to tours in such a way that these tours can be performed by the available drivers and make the best use of the drivers' time, is therefore likely to be at least as important a consideration as the utilization of vehicles. If a sufficient number of vehicles are available, then driver considerations are the only constraints, and the objective should be to develop optimal driver itineraries.

It is not only the availability of drivers that restricts the set of feasible routes. Often deliveries at customers can only take place during specific time periods of the day.

Many companies operate a heterogeneous fleet of vehicles instead of a homogeneous fleet of vehicles.

We have considered the distribution of a product from a single plant. Often a company operates several plants that produce the same product, and distribution to some customers can occur from a number of plants. It may be optimal to distribute to a customer from different plants on different days, depending on how well the customer can be combined in a vehicle tour with the other customers that are to be visited on the particular day.

Frequently, a company produces and distributes several products, using the same fleet of vehicles to transport the different products. Examples are the transportation of different grades of oil in compartmentalized vehicles, and the replenishment of beverages and snacks in vending machines and at restaurants. In this multi-product environment, besides deciding which customers to visit next and how to combine them into vehicle tours, we have to decide how much of each product to deliver to each visited customer.

We have assumed that a sufficient amount of the product is always available for distribution, and issues related to production capacity and scheduling are ignored. However, it is often necessary to coordinate production, storage, and transportation.

Inventory holding cost have not been addressed in the problem definition. In fact, this makes the problem more generic, because the treatment of inventory holding cost depends on the ownership and storage management of inventory at the plant and at the storage facilities of customers. For example, the distributor may be the same company that operates the production plant as well as the facilities at the next level of the distribution network (the "customers"), or the producer may distribute the product to and manage the inventory at independent customers (called vendor managed resupply), or an independent third party logistics provider may distribute the product from the producer to the customers, and manage their inventory. The treatment of inventory holding costs are different for the three cases above, but in all cases it can be incorporated relatively easily with the other costs.

System disruptions such as product shortages at the plant, vehicle breakdowns, work stoppages, and inventory measurement failures, are not incorporated. To address these issues, policies have to be developed to provide recourse actions when disruptions occur.

Travel times and costs are assumed to be known. A more realistic model may incorporate random travel times and costs. However, unless transportation occurs in heavily congested networks, a model assuming known travel times should give good results. If transportation networks are very congested, then the time of travel usually has a large impact on travel time besides the chosen route, and many other scheduling and routing issues have to be addressed. As the objective of the SIRP is to minimize the expected sum of the costs, only the expected travel costs need to be known, and not their distributions.

Many of the practical issues raised above can be easily incorporated in the models discussed and many of the solution approaches presented can be modified to handle them.

4.8 TEST PROBLEMS

We would like to provide researchers with challenging instances of difficult routing problems. A standard set of instances allows the comparison of the performance of algorithms and often it also provides an important stimulus for research. We have created a set of instances of the IRP that we hope will form such a test set. They have been derived from real data from a company we work with. They are available via the world wide web at http://tli.isye.gatech.edu/Testcases/irp.

References

Anily. S. and A. Federgruen. (1990). One Warehouse Multiple Retailer Systems with Vehicle Routing Costs. *Management Science*, 36(1):92–114.

Anily, S. and A. Federgruen. (1991). Rejoinder to 'One Warehouse Multiple Retailer Systems with Vehicle Routing Costs'. *Management Science*, 37(11):1497–1499.

Bard, J., L. Huang, M. Dror and P. Jaillet. (1998a). A Branch and Cut Algorithm for the VRP with Satellite Facilities. *IIE Transactions on Operations Engineering.* Forthcoming.

Bard, J., L. Huang, P. Jaillet and M. Dror. (1998b). A Decomposition Approach to the Inventory Routing Problem with Satellite Facilities. *Transportation Science.* Forthcoming.

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 On-Line Computerized Routing and Scheduling Optimizer. *Interfaces*, 13(6):4–23.

Bertsekas, D.P. and J.N. Tsitsiklis. (1996). *Neuro-Dynamic Programming.* Athena Scientific, New York, NY.

Bramel, J. and D. Simchi-Levi. (1995). A Location Based Heuristic for General Routing Problems. *Operations Research*, 43(4):649–660.

Chan, L.M.A., A. Federgruen and D. Simchi-Levi. (1998). Probabilistic Analysis and Practical Algorithms for Inventory-Routing Models. *Operations Research.* Forthcoming.

Chien, T., A. Balakrishnan and R. Wong. (1989). An Integrated Inventory Allocation and Vehicle Routing Problem. *Transportation Science*, 23(2):67–76.

Dror, M. and M. Ball. (1987). Inventory/Routing: Reduction from an Annual to a Short Period Problem. *Naval Research Logistics Quarterly*, 34(6):891–905.

Dror, M., M. Ball and B. Golden. (1985). Computational Comparison of Algorithms for the Inventory Routing Problem. *Annals of Operations Research*, 4(1-4):3–23.

Dror, M. and L. Levy. (1986). Vehicle Routing Improvement Algorithms: Comparison of a 'Greedy' and a Matching Implementation for Inventory Routing. *Computers and Operations Research*, 13(1):33–45.

Federgruen, A. and P. Zipkin. (1984). A Combined Vehicle Routing and Inventory Allocation Problem. *Operations Research*, 32(5):1019–1036.

Fisher, M., A. Greenfield, R. Jaikumar and P. Kedia. (1982). Real-Time Scheduling of a Bulk Delivery Fleet: Practical Application of Lagrangean Relaxation. Technical report, The Wharton School, University of Pennsylvania, Department of Decision Sciences.

Gallego, G. and D. Simchi-Levi. (1990). On the Effectiveness of Direct Shipping Strategy for the One-Warehouse Multi-Retailer r-Systems. *Management Science*, 36(2):240–243.

Golden, B., A. Assad and R. Dahl. (1984). Analysis of a Large Scale Vehicle Routing Problem with an Inventory Component. *Large Scale Systems*, 7(2-3):181–190.

Jaillet, P., L. Huang, J. Bard and M. Dror. (1997). A Rolling Horizon Framework for the Inventory Routing Problem. Working paper, Dept. of Management Science and Information Systems, College of Business Administration, University of Texas.

Langley Jr., C.J. and M.C. Holcomb. (1996). Creating Logistics Customer Value. *Journal of Business Logistics*, 13(2).

Larson, R. (1988). Transporting Sludge to the 106 Mile Site: An Inventory/Routing Model for Fleet Sizing and Logistics System Design. *Transportation Science*, 22(3): 186–198.

Minkoff, A.S. (1993). A Markov Decision Model and Decomposition Heuristic for Dynamic Vehicle Dispatching. *Operations Research*, 41(1):77–90.

Trudeau, P. and M. Dror. (1992). Stochastic Inventory Routing: Route Design with Stockouts and Route Failures. *Transportation Science*, 26(3):171–184.

Webb, R. and R. Larson. (1995). Period and Phase of Customer Replenishment: A New Approach to the Strategic Inventory/Routing Problem. *European Journal of Operations Research*, 85(1):132–148.

5 DYNAMIC VEHICLE ROUTING AND DISPATCHING

Michel Gendreau
Jean-Yves Potvin

5.1 INTRODUCTION

Real-time decision problems are playing an increasingly important role in the economy due to advances in communication and information technologies that now allow real-time information to be quickly obtained and processed (Séguin *et al.*, 1997). Among these, dynamic vehicle routing and dispatching problems have emerged as an intense area of research in the operations research community. Numerous examples may be found in Haines and Wolfe (1982), Powell, Jaillet and Odoni (1995) and Psaraftis (1995). In these problems, a set of vehicles is routed over a particular time horizon (typically, a day) while new service requests are occuring in real-time. With each new request, the current solution may be reconfigured to better service the new request, as well as those already assigned to a route.

Although dynamic vehicle routing and dispatching is a subclass of a broader domain called dynamic transportation, which includes fleet management, facility location, traffic assignment, air traffic control and many others (see Dror and Powell, 1993), it is itself a broad domain with a variety of problems. A first distinction can be made on the basis of the dispatching area, which could be either local (e.g., within an urban area), regional, national or international. Local area dispatching systems evolve within a narrower time horizon: they are thus highly dynamic and exhibit stringent response time requirements. Another distinction can be made with respect to the importance of the routing component, that is, the need to sequence requests within planned routes. Table 5.1 provides application examples that are classified along those lines.

Wide area dispatching problems are commonly found in truckload and less-than-truckload freight transportation systems (Bausch, Brown and Ronen, 1995; Bell *et*

Table 5.1 Dynamic vehicle dispatching

	local area	*wide area*
routing	courier services dial-a-ride	less-than-truckload trucking
no routing	emergency services	truckload trucking

al., 1983; Brown and Graves, 1981; Brown *et al.*, 1987; Powell, 1988). Less-than-truckload implies that multiple customer orders are consolidated onto the same vehicle, thus leading to important routing issues. This is to be contrasted with truckload applications where a vehicle is dispatched to a single customer. Local area dispatching systems includes "mobile server" applications, in particular emergency services (ambulances, police, immediate repair services) where each vehicle is dedicated to a single customer. As their wide area truckload trucking counterpart, one of the main challenges posed by these problems is to appropriately position a vehicle, once service is completed, to anticipate future demand. Theoretical studies on this topic may be found in Bertsimas and Van Ryzin (1991), Bertsimas and Van Ryzin (1993). Courier services, such as local express mail delivery, and dial-a-ride problems found in demand-responsive transportation systems, like transportation for the disabled, are examples of local area dispatching systems where many customer requests can be consolidated onto the same vehicle (Madsen, Ravn and Rygaard, 1995; Madsen, Tosti and Vœlds, 1995; Roy *et al.*, 1984; Shen *et al.*, 1995; Taillard *et al.*, 1997; Wilson and Colvin, 1977). As the paper's title indicates, the focus will be on this last category of problems, namely, highly dynamic local area dispatching problems involving routing issues. Such problems can be further divided along the following lines:

- *many-to-many* where each request includes both a pick-up and a delivery location versus *one-to-many (many-to-one)* where each request includes a single pick-up (delivery) location
- *capacitated* where each vehicle has a fixed capacity versus *uncapacitated.*

Problems of the many-to-many type with capacity constraints are the most difficult to address. Handling two-point requests raises specific difficulties as the pick-up and delivery locations must be assigned to the same route, and the pick-up point must necessarily precede the delivery point. The presence of capacity constraints (sometimes multi-dimensional, when specific seats are restricted to specific passenger types) further compounds the picture. In addition, realistic vehicle routing and dispatching problems involve many vehicles and include time windows to restrict the service time at customer locations. Typically, these time windows are soft and can be violated to some extent to account for early or late arrivals.

Table 5.2 shows specific application examples for each category of such local area vehicle routing and dispatching problems. In this table, a feeder system refers to a local dial-a-ride system aimed at "feeding" another, wider area, transportation system at a particular transfer location. For example, one could envision a minibus pick-up

service that drops passengers at a subway or train station. The courier service application refers to the local part of international shipping services (e.g., Federal Express) where the load is collected at different customer locations and brought back to a central depot for further processing and shipping. This is to be contrasted with a local express mail delivery service where both the pick-up and delivery locations are found in the same area and are serviced by the same vehicle. The repair service application mentioned in the table does not refer to classical immediate repair services, which are related to emergency services, but rather to an application mentioned in Madsen, Tosti and Vœlds (1995), where the repair service can be provided a long time after the customer request (even, a few days later). The real-time routing requirements come from the need to immediately tell the customer if his request can be accepted and, if it is the case, when the service crew will come at the customer's premise. Overall, the literature on these problems is rather scant, but the total research effort is certainly more important, as many results that are commercially exploited do not necessarily make their way into the "official" scientific literature.

Table 5.2 Dynamic local area vehicle routing and dispatching

	many-to-many	*one-to-many*
capacitated	dial-a-ride	feeder system
uncapacitated	express mail delivery	courier or repair services

Note that inventory-routing problems, like those reported in Dror, Ball and Golden (1986), Federgruen and Simchi-Levi (1995) and Minkoff (1993), where inventory considerations are integrated with routing issues to maintain appropriate inventory levels at customer locations, are not specifically addressed here. Demand forecasting in this context is often based on sophisticated models that can quite accurately predict inventory levels. Consequently, vehicle routes to replenish customers can be designed (for the most part) well before their execution, as undesirable real-time events like stock failures are unlikely to occur.

For the most part, the work reported in this paper is based on optimization methods that minimize or maximize an objective function, subject to a variety of constraints. Accordingly, different functions have been proposed in the literature to capture the gist of each particular problem. Such functions are approximations of what is desired in the real world and they typically differ from one application to another. However, a common structure is often observed: a weighted sum of a variety of components related to the operations cost of the dispatching system (total distance traveled, capacity utilization, throughput) and service quality to the customers (waiting time, ride time, lateness). An alternative approach is proposed in Shen *et al.* (1995), as the authors do not try to define any "realistic" objective function. Rather, they mimic the dispatcher's decision behavior, using a learning algorithm that exploits previous decisions taken by this dispatcher. Instead of a priori defining some objective function to be optimized, they thus try to reproduce the (unknown) function used by the dispatcher.

The remainder of the paper will focus on three out of the four categories of problems presented in Table 5.2, namely, dial-a-ride, repair, courier and express mail delivery services. We are not aware of any research results on feeder systems. The presentation order is not related to the complexity level of each problem, but rather to their "chronological" occurrence in the literature. Dynamic dial-a-ride problems, for example, are the most difficult problems to address but they have sparkled the interest of researchers for more than 20 years now, due to economically important real-world applications. Hence, the literature on this subject is somewhat richer and these problems are discussed first. Sections 5.3 and 5.4 then discuss repair, courier and express mail delivery services. Concluding remarks about future research directions are found in Section 5.5.

5.2 DIAL-A-RIDE PROBLEMS

Dial-a-ride problems are mostly found in demand-responsive transportation systems aimed at servicing small communities or passengers with specific requirements (elderly, disabled). These problems are of the many-to-many type and include capacity constraints (often, multidimensional) as well as soft time window constraints at the pick-up location, at the delivery location or both. The economic importance of these problems have spurred a number of research projects over the years. The following discussion divides the work reported in the literature on the basis of the adopted problem-solving approach, either simple insertion procedures or repetitive application of static algorithms at each input update.

5.2.1 Insertion methods

Research on dial-a-ride problems started in the early seventies with the pioneering work of Wilson *et al.* (1971), Wilson and Weissberg (1976), Wilson and Colvin (1977). This work was motivated by demand-responsive transportation systems found in Haddonfield, NJ and Rochester, NY. Here, a simple insertion rule is applied to incorporate each new request into the current solution. The insertion involves both the pick-up and delivery locations and takes care of the implicit precedence constraint between the pick-up and the delivery. The objective function to be minimized over all feasible insertion places includes the delay between the time of occurrence of the service request and its planned pick-up time, the ride time from the pick-up to the delivery locations and the deviation between the planned and promised pick-up times. An additional component is also aimed at spreading the total distance traveled among vehicles. The basic algorithm was later extended through deferred assignments to alleviate the myopic behavior of this simple insertion approach. In this case, the algorithm does not assign a customer immediately upon receiving the request but rather defers (when possible) the decision in the hope that knowledge of future requests will lead to a better assignment.

A similar insertion heuristic is proposed in Roy *et al.* (1984) for the transportation of the disabled. In this application, a fair amount of requests are known in advance. Initial routes are thus constructed for those requests. The algorithm first forms mini-clusters of customers that fit naturally together due to time-spatial proximity. A route is then constructed within each mini-cluster with an insertion procedure. Finally, the solution is improved by moving customers from one route to another.

Real-time requests are incorporated in the initial solution framework using the insertion procedure mentioned above. An important feature of this work is the exploitation of a rolling horizon: the earliest pick-up time of any given request must fall within the horizon to be considered for inclusion in the current solution. Thus, the focus is on the most important requests, namely, those with a time window that is closer to the current time. Decisions about requests with late time windows are deferred as much as possible, given that the situation is likely to change in the mean time.

Madsen, Ravn and Rygaard (1995) have recently developed another insertion approach for a demand responsive transportation system established by the Copenhagen Fire-Fighting service for the elderly and the disabled. As in the previous work, a fair amount of requests are known beforehand and are scheduled statically through an adaptation of the insertion algorithm of Jaw *et al.* (1986). Real-time requests are handled in a sequential fashion using an insertion rule that minimizes a weighted sum of components that measure the inconvenience to the new customer (waiting time, deviation from desired service time, excess ride time), the additional inconvenience to other customers already assigned to that route and various operations cost (like unused capacity).

5.2.2 Adaptation of static algorithms

In Psaraftis (1980), a dynamic programming algorithm initially developed for solving a single vehicle, static dial-a-ride problem without time windows is adapted to the dynamic case where new requests occur in real-time and are eligible for inclusion in the current route. The problem-solving approach applies the dynamic programming algorithm in a repetitive manner each time an input update occurs, using the current set of known, but yet unserviced, requests (either on-board or waiting to be picked-up). Due to the exponential time complexity of dynamic programming, only small problem instances with 10 requests at most are handled. Maximum shift constraints are imposed to avoid indefinite service deferment at a customer location due to unfavorable geographical characteristics (i.e., if a customer location is remote as compared with other customers, it can easily remain in last position in the current planned route). The maximum shift constraints state that the absolute difference between the customer's position in the route and its position in the initial "first-come-first-served" list of customer requests must be less than or equal to some prespecified threshold. Note that the presence of time windows is a natural way to preclude such indefinite deferment.

A military sealift emergency problem is addressed in Psaraftis (1988), where cargoes are moved by ship between a number of ports to optimize a prescribed measure of system performance under various operating constraints, in particular time window constraints on pick-up and delivery. The main contribution of this work is in the innovative exploitation of a rolling time horizon. At current time t_k, planned routes are developed only for known cargoes with an earliest pick-up time in the interval $[t_k,\ t_k + L]$ where L is the length of the rolling time horizon. However, a definitive assignment of cargoes to ships is made only for cargoes found in the interval $[t_k,\ t_k + \alpha L]$ where $0 < \alpha < 1$. In this way, later events within the rolling horizon are considered when firm decisions are taken for those that are closer to the current time. The clock is then moved up to the time of the next input update or to the lowest earliest pick-up time among unassigned cargoes, whichever of the two comes first. At each iteration,

a transportation problem is solved to assign known cargoes to ships, where the arc cost between a cargo and a ship measures the utility of the corresponding assignment. This utility is a complex function that measures the assignment's effect on the delivery time of all other loads assigned to the same ship, the ship's utilization and the port's congestion.

5.3 REPAIR AND COURIER SERVICES

As opposed to dial-a-ride systems, customer requests in the problems discussed here involve a single location (thus, facilitating their insertion in the current solution) and vehicle capacity is not an issue. Soft time windows are still present, however, to restrict the desired service time, while allowing early or late arrivals. It is worth noting that the service time at the customer's premise is an important issue in repair services, as these times are typically much larger than those observed in courier services. Furthermore, the service time is likely to be very different from one customer to another and may be subject to large variations (e.g., performing a given repair task may take longer than expected).

5.3.1 Repair services

Repair services usually involve a utility firm (electricity, gas, water and sewer, etc.) that responds to customer requests for maintenance or repair of its facilities at the customer's premise. An application for the repair of gas installations is described in Madsen, Tosti and Vœlds (1995). In this problem, the objective is to determine, for a fixed level of service and over a given planning horizon (usually, a week), an optimal strategy of route design and time window setting to minimize the total distance traveled. Here, the time windows are not user-defined, but are set by the company, and the routes are typically executed well after their determination. The real-time requirement of the routing algorithm comes from the need to quickly tell the customer if the request can be accepted and to specify the time window, chosen among a prespecified set of time slices, within which the service crew will arrive at the customer location. The algorithm handles new requests one by one. For each request, a subset of routes is selected based on proximity measures. Then, the insertion place that minimizes the detour over this subset of routes is identified. Finally, a time window is assigned to the customer with the objective of balancing the load among the time slices.

This application is to be contrasted with academic problems like the Dynamic Traveling Salesman Problem (DTSP) in Psaraftis (1988) and the Dynamic Traveling Repairman Problem (DTRP) in Bertsimas and Van Ryzin (1991) where the service requests must be serviced as soon as possible after their occurrence. In Bertsimas and Van Ryzin (1991), simplifying assumptions allow a tractable mathematical analysis of different routing policies aimed at minimizing the average time spent in the system by each customer (waiting time plus service time). That is, there are no planned routes as such, but rather prespecified policies for identifying the customer to be serviced next, among a queue of pending requests, when the vehicle has completed its service at the current customer location. This work is strongly inspired by queuing theory: a vehicle is viewed as a mobile server, with requests occurring in time according to a Poisson process and stochastic on-site service times. The authors also introduce

routing considerations through the analysis of a "traveling salesman policy" where service requests are accumulated to form sets of a fixed size. Since the vehicle is dispatched over those sets (and not over individual requests), the requests within each set must be sequenced. An extension of this policy for multiple vehicles with capacity constraints is analyzed in Bertsimas and Van Ryzin (1993). In this case, a vehicle can service at most q customers before returning to the depot. Thus, customers are clustered in groups of size q before a vehicle is dispatched to them. A similar analytic work has also been performed in the case of a single vehicle problem with pick-ups and deliveries, see Papastavrou and Swihart (1997).

Unfortunately, a thorough mathematical analysis is seldom possible in real-world settings. Ad hoc algorithmic techniques are thus applied, as in the dial-a-ride systems of Section 5.2 and in the repair application presented above, to tell the vehicle what its next destination(s) should be. This is done through the construction of planned routes that allow decisions to be taken with respect to all other known (or even, forecasted) requests. If the environment is not highly dynamic, the planned routes are not likely to change much over time and can also be used for later decisions involving the same vehicle or other vehicles.

5.3.2 Courier services

In this section, a courier service refers to the local portion of international express mail services (e.g., Federal Express) that must respond to customer requests in real-time. The load is collected at different customer locations and brought back to a central facility at the end for further processing and shipping. Little work has been published in the scientific literature on this subject. The only work motivated by this application (that we are aware of) is found in Gendreau *et al.* (1996). In this work, the authors exploit the power of new computing technologies, in particular parallel processing, to go beyond simple insertion methods to respond to new service requests in real-time. The solution procedure is based on an adaptive memory tabu search heuristic, initially designed for the static version of the problem where all requests are known in advance, see Taillard *et al.* (1997).

The tabu search exploits a neighborhood based on CROSS exchanges that extends previously developed exchange methods for problems with time windows. Using a pair of routes in the current solution, a sequence of consecutive customer locations, which can be of any arbitrary length, is first selected on each route. Then, the two sequences are swapped, that is, the sequence on one route replaces the corresponding sequence on the other route, and conversely. Given that the time windows define an implicit orientation on the routes (i.e., customers with early time windows are serviced first while those with late time windows are serviced last), the orientation of each sequence is maintained when it is moved to the other route.

The adaptive memory, originally proposed in the work of Rochat and Taillard (1995), is used to store the routes of the best solutions visited during the search. New starting solutions for the tabu search are then constructed by selecting and combining routes taken from different solutions in the adaptive memory. This way of creating new solutions is similar in spirit to the crossover operator found in genetic algorithms (see, Holland, 1975), although the number of "parent" solutions can be larger than two. The tabu search algorithm is further enhanced through a decomposition procedure where each problem (set of routes) is partitioned into a number of subproblems

(subsets of routes), using a sweep heuristic. A distinct tabu search is then applied to each subproblem. At the end, the resulting routes are merged back to form the final solution. Through this decomposition approach, the tabu search works on much smaller problems, thus allowing a more focused search (as previously observed in Taillard, 1993). The objective chosen by the authors consists of minimizing the total distance traveled plus a penalty for lateness at customer locations. By raising this penalty, the authors were able to improve previous solutions reported in the literature on some static problems with hard time windows proposed in Solomon (1987).

A parallel implementation of this algorithm on a network of SUN UltraSPARC workstations is reported in Badeau *et al.* (1997). The implementation follows a master-slave scheme where the master manages the adaptive memory and the slaves run the tabu search. A two-level parallelization scheme is used. First, many tabu search threads run in parallel and communicate through a common adaptive memory (since they all feed the memory with new improved solutions and request new starting solutions from it). Second, the decomposition procedure is performed in parallel within each search thread, as a different tabu search process is applied to each subproblem.

In the dynamic setting, the parallel tabu search runs in the background between events, trying to optimize the current set of planned routes based on known, but yet unserviced requests. As in the work of Psaraftis (1980), the algorithm developed for the static case is applied each time an input update occurs. Two different types of events are taken care of: the occurrence of a new service request and the arrival of a vehicle at a customer location. When a new service request is received, the tabu search processes are stopped and must send their best solution to the master process for possible inclusion in the adaptive memory. The new request is then incorporated within each solution in the adaptive memory through a classical insertion procedure. Once updated, the memory feeds the tabu search processes with new starting solutions that truly reflect the current state of the system for further reoptimization. A similar procedure is applied when a vehicle has finished its service at a given customer location. The tabu search processes are interrupted and must send their best solution to the master process for possible inclusion in the adaptive memory. The best solution in memory is then used to identify the vehicle's next destination, with the other solutions in memory being updated accordingly. The updated memory then feeds the tabu search processes with new starting solutions for further reoptimization.

A discrete-time simulator, using Solomon's data, was developed to test the algorithm under scenarios with up to 2-3 new requests per minute, on average. The computational results demonstrate the benefits of a sophisticated optimization procedure like tabu search to handle these new requests in real-time, as compared to other simpler methods, in particular classical insertion heuristics. Using tabu search, the total travel time, total lateness and the number of customer requests that are rejected (when the system is too heavily loaded) are all substantially reduced.

5.4 EXPRESS MAIL DELIVERY

In this application, express mail must be collected and delivered by the same vehicle from one point to another within a local (usually, urban) area. In Gendreau *et al.* (1998), the tabu search heuristic presented in Section 5.3 for the courier service is extended to address two-point requests. In this case, the neighborhood structure is

redefined, as a CROSS exchange can easily split the pick-up and delivery locations associated with a given request on two different routes. Instead, the authors use ejection chains where a (two-point) request on one route ejects a request from a second route, which in turn ejects another request from a third route, etc. The process ends with the insertion of the last ejected request in the route that started the chain or in any other route not yet visited. The ejection chain can be of any arbitrary length, although it must be elementary (i.e. it cannot come back to a previous route to eject another request from that route). Ideas along these lines have already been proposed in Glover (1996), Rego and Roucairol (1996), Thompson and Psaraftis (1993). Clearly, the neighborhood defined by such ejection chains can be quite large. A shortest path algorithm is thus proposed in Gendreau *et al.* (1998) to find the best ejection chain over a restricted portion of the entire neighborhood.

In Shen *et al.* (1995), a computer assistant for vehicle dispatching with a learning module is developed for the same type of application. This work departs from classical operations research approaches as the authors do not apply optimization techniques to minimize or maximize an objective function. Given that experienced professional dispatchers are very qualified and that the objective function to be optimized is rather fuzzy, the authors have followed an alternative problem-solving approach. A neural network model using the backpropagation learning algorithm of Rumelhart and McClelland (1986) is developed to automatically learn the decision process of a dispatcher from decision examples provided by that expert. More precisely, the network is fed with a set of examples, where each example includes a description of a dispatching situation that involves a new service request as well as the decision taken by the expert (i.e., the vehicle chosen to service the new request). Using this training set of examples, the network gradually adjusts the weights on its connections to reduce the error between its decisions and the dispatcher's decisions. Although the network has been trained in batch mode from previously collected data at a Montreal-based express mail delivery company, it would also be possible to train the network in real-time by incrementally adjusting the weights on its connections each time a new decision is taken by the dispatcher. Over time, the network's performance would improve, as new dispatching situations are encountered during the operations of the dispatching system. After a while, the network could perform autonomously and suggest appropriate dispatching decisions (to support a less experienced dispatcher, for example).

5.5 CONCLUDING REMARKS

The work performed to date on local area vehicle routing and dispatching problems leaves a number of unanswered questions and open research avenues. First, the issue of demand forecasting has not really been addressed yet, in contrast with dispatching systems found in emergency services and truckload trucking (see, for example, Powell, Jaillet and Odoni, 1995). In these applications, each vehicle is dedicated to a single customer and the issue of relocating vehicles to anticipate future demands, once the service is completed at the current customer location, is of paramount importance. In the application contexts examined in this paper, forecasting demands would positively impact the construction of planned routes, since those routes would now be constructed with some look-ahead. Forecasted demands can be dealt with deterministically (i.e., the planned routes include forecasted demands as well as known demands) or stochastically. The deterministic approach has some pitfalls as the construction of

a solution may well be driven by forecasted demands that will never occur, at the expense of known demands. With respect to the stochastic approach, some inspiration may come from the application of dynamic networks for truckload trucking problems, as reported in Powell, Jaillet and Odoni (1995), where the movements of vehicles are explicitly handled as random variables.

Although the current literature has mostly focused on a single source of uncertainty, (i.e., the time-space occurrence of new service requests), other sources of uncertainty, like cancellation of service requests, service delays due to congestion or vehicle breakdowns, etc., may be considered as well. These new sources of uncertainty raise interesting questions, for example, when should a problematic situation be responded to and what recourse actions would be appropriate in this situation.

The issue of diversion also deserves some attention. With the availability of real-time tracking devices, like global positioning systems, it is now possible to follow the trajectory of each vehicle. This information can be used, in particular, to divert a vehicle away from its current destination to service a new request that just occurred in the vicinity of its current position. Decisions are taken very quickly in this context, because vehicles are moving fast. This raises the issue of finding good trade-offs between computation time and solution quality. If the computations take too long, for example, diversion opportunities may be lost because the solution computed does not apply anymore to the current situation, which has evolved in the mean time.

More theoretical work may also be spent on the worst-case analysis of dynamic algorithms to evaluate how much is lost by not having the full information in advance. Finally, parallel implementation issues should get an increasing attention in the future, due to the very nature of these problems where prompt responses are needed.

Acknowledgments

Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Québec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR). This support is gratefully acknowledged.

References

Badeau P., M. Gendreau, F. Guertin, J.Y. Potvin and E.D. Taillard. (1997). A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research*, 5C:109–122.

Bausch D.O., G.G. Brown and D. Ronen. (1995). Consolidating and Dispatching Truck Shipments of Mobil Heavy Petroleum Products. *Interfaces*, 25:1–17.

Bell, W., L.M. Dalberto, M.L. Fisher, A.J. Greenfield, R. Jaikumar, P. Kedia, R.G. Mack and P.J. Prutzman. (1983). Improving the Distribution of Industrial Gases with an Online Computerized Routing and Scheduling Optimizer. *Interfaces*, 13: 4–23.

Bertsimas, D.J. and G. Van Ryzin. (1991). A Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plane. *Operations Research*, 39:601–615.

Bertsimas, D.J. and G. Van Ryzin. (1993). Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plane with Multiple Capacitated Vehicles. *Operations Research*, 41:60–76.

Brown, G.G. and G.W. Graves. (1981). Real-Time Dispatch of Petroleum Tank Trucks. *Management Science*, 31:764–772.

Brown, G.G., C.J. Ellis, G.W. Graves and D. Ronen. (1987). Real-Time Wide Area Dispatching of Mobil Tank Trucks. *Interfaces*, 17:107–120.

Dror, M., M. Ball and B.L. Golden. (1986). A Computational Comparison of Algorithms for the Inventory Routing Problem. *Annals of Operations Research*, 4:3–23.

Dror, M. and W.B. Powell, editors. (1993). Special Issue on Stochastic and Dynamic Models in Transportation. *Operations Research*, 41:1–235.

Federgruen, A. and D. Simchi-Levi. (1995). Analysis of Vehicle Routing and Inventory-Routing Problems. In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, editors, *Network Routing*, pages 297–373, North-Holland: Amsterdam.

Gendreau, M., F. Guertin, J.Y. Potvin and R. Séguin. (1998). Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-Ups and Deliveries. Report CRT–98–10, Centre de recherche sur les transports, Université de Montréal.

Gendreau, M., F. Guertin, J.Y. Potvin and E. Taillard. (1996). Tabu Search for Real-Time Vehicle Routing and Dispatching. Technical Report CRT-96-47, Centre de recherche sur les transports, Université de Montréal.

Glover, F. (1996). Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. *Discrete Applied Mathematics*, 65:223–253.

Haines, G.H. and R.N. Wolfe. (1982). Alternative Approaches to Demand Responsive Scheduling Algorithms. *Transportation Research*, 16A:43–54.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press: Ann Arbor.

Jaw, J.J., A.R. Odoni, H.N. Psaraftis and N.H.M. Wilson. (1986). A Heuristic Algorithm for the Multi-Vehicle Advance Request Dial-a-Ride Problem with Time Windows. *Transportation Research*, 20B:243–257.

Madsen, O.B.G., H.F. Ravn and J.M. Rygaard. (1995). A Heuristic Algorithm for a Dial-A-Ride Problem with Time Windows, Multiple Capacities and Multiple Objectives. *Annals of Operations Research*, 60:193–208.

Madsen, O.B.G., K. Tosti and J. Vœlds. (1995). A Heuristic Method for Dispatching Repair Men. *Annals of Operations Research*, 61:213–226.

Minkoff, A.S. (1993). A Markov Decision Model and Decomposition Heuristic for Dynamic Vehicle Dispatching. *Operations Research*, 41:77–90.

Papastavrou, J.D. and M.R. Swihart. (1997). A Stochastic and Dynamic Model for the Single-Vehicle Pick-up and Delivery Problem. Working Paper, School of Industrial Engineering, Purdue University.

Powell, W.B. (1988). A Comparative Review of Alternative Algorithms for the Dynamic Vehicle Allocation Problem. In B.L. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 297–373, North-Holland: Amsterdam.

Powell, W.B., P. Jaillet and A. Odoni. (1995). Stochastic and Dynamic Networks and Routing. In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, editors, *Network Routing*, pages 141–295, North-Holland: Amsterdam.

Psaraftis, H.N. (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.N. (1988). Dynamic Vehicle Routing Problems. In B.L. Golden and A.A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 223–248, North-Holland: Amsterdam.

Psaraftis, H.N. (1995). Dynamic Vehicle Routing: Status and Prospects. *Annals of Operations Research*, 61:143–164.

Rego, C. and C. Roucairol. (1996). A Parallel Tabu Search Algorithm using Ejection Chains for the Vehicle Routing Problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 223–248, Kluwer: Boston.

Rochat, Y. and E.D. Taillard. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1:147–167.

Roy, S., J.M. Rousseau, G. Lapalme and J. Ferland. (1984). Routing and Scheduling of Transportation Services for the Disabled: Summary Report. Technical Report CRT-473A, Centre de recherche sur les transports, Université de Montréal.

Rumelhart, D.E. and J.L. McClelland. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* The MIT Press: Cambridge, MA.

Séguin, R., J.Y. Potvin, M. Gendreau., T.G. Crainic and P. Marcotte. (1997). Real-Time Decision Problems: An Operational Research Perspective. *Journal of the Operational Research Society*, 48:162–174.

Shen, Y., J.Y. Potvin, J.M. Rousseau and S. Roy. (1995). A Computer Assistant for Vehicle Dispatching with Learning Capabilities. *Annals of Operations Research*, 61:189–211.

Solomon, M.M. (1987). Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research*, 35:254–265.

Taillard, E.D. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23:661–673.

Taillard, E.D., P. Badeau, M. Gendreau, F. Guertin and J.Y. Potvin. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31:170–186.

Thompson, P.M. and H.N. Psaraftis. (1993). Cyclic Transfer Algorithms for MultiVehicle Routing and Scheduling Problems. *Operations Research*, 41:935–946.

Wilson, N.H.M., J.M. Sussman, H.K. Wang and B.T. Higonnet. (1971). Scheduling Algorithms for Dial-a-Ride Systems. Urban Systems Laboratory Report USL TR-70-13, Massachusetts Institute of Technology.

Wilson, N.H.M. and H. Weissberg. (1976). Advanced Dial-A-Ride Algorithms Research Project: Final Report. Technical Report R-76-20, Department of Civil Engineering, Massachusetts Institute of Technology.

Wilson, N.H.M. and N.H. Colvin. (1977). Computer Control of the Rochester Dial-A-Ride System. Technical Report R-77-30, Department of Civil Engineering, Massachusetts Institute of Technology.

6 ON LANGUAGES FOR DYNAMIC RESOURCE SCHEDULING PROBLEMS

Warren B. Powell

"The elevator lift is being fixed for the next day. During that time we regret that you will be unbearable." - Sign in a Bucharest business.

6.1 INTRODUCTION

Consider the situation of an American businessman early in the century who is interested in making money by selling Arab oil in the United States. The first problem is designing the chemical process needed to convert the type of oil in Saudi Arabia to the kind of gasoline needed in the U.S., a problem that has been solved by a French chemist. The American businessman needs to determine if global energy prices will rise to the point to make the enterprise profitable, a problem that is well understood by an English economist. The economist needs to understand how this new source of oil will affect the other oil markets, which can be solved by a Russian mathematician.

It turns out that the easiest part of this problem is translating American English to British English to Russian to French to Arabic. The general problem of translating between spoken languages has been resolved centuries ago. Much more intractable is the problem of translating between the fields of business, economics, mathematics, chemistry and the oil industry. Particularly in today's relatively specialized economy, it is difficult to get people with different training to understand the subtle languages of different fields. The American businessman wants to make money in a way that will show up on a corporate balance sheet, reflecting local tax laws. The economist might consider only the costs of production and shipping. The Russian mathematician might formulate a spatial price equilibrium problem that uses simplistic game theoretic models to describe global oil trade. The chemist will focus on the subtle differences in the quality of the oil and the nature of the production process needed to produce appropriate types of gasoline. And the Arab oil sheik wants to be sure that he is selling more oil than Libya.

We face a similar problem when we want to formulate real-time logistics problems as mathematical models. The benefits seem apparent - we want to lower our costs and make more money for the company. This requires formulating our problem as a mathematical model (the equivalent of the economist) which can then be solved using a mathematical algorithm, designed by mathematicians. The model and algorithm must be implemented by a software engineer (analogous to our chemist above) who needs to design code that can be used by someone in operations, with an efficient implementation of the mathematician's algorithm, and which provides the necessary diagnostics to be useful to the modeler.

The challenge of building a production-quality real-time control system for complex operations, such as arise in transportation, requires the participation of different groups of people who contribute different skill sets. At a high level, we can identify four major skill groups that need to contribute to the design and implementation of a real-time control system:

- Industry
- Modeling
- Algorithms
- Software

Within each of these groups are numerous subgroups, speaking what we view as dialects of a major language. Within industry, the major subgroups are: management, operations, marketing and human resources. Management thinks of itself as setting company strategy, although its impact on day to day operations is often minimal. Operations is the group that really controls the company on a day-to-day basis, and the biggest challenge is developing a system that is accepted by operations. A real-time dispatch system, however, invariably has an impact on customer service, which requires the participation of marketing. Finally, a dispatch system will impact both the people using the tool (dispatchers, who may need additional training) and the drivers themselves, requiring the input of human resources.

The modeler is an important, emerging group which we believe will become a distinct specialty. While industrial participants will reflect a business background and algorithmic specialists will draw from mathematics, we believe that modelers will come from engineering and, as such, will evolve to become a separate population. A good modeler will combine a knowledge of business processes and the mathematical theory of algorithms, his/her training will be more familiar to those from science and engineering, with a strong emphasis on experimental skills. A good modeler will focus on developing efficient mathematical models which capture the important physical laws of a process, balancing the benefits of modeling accuracy with the cost of collecting the necessary data.

Specialists in algorithms are primarily mathematicians working on well-defined problems. As a result of the variety of different problem structures, subcommunities of specialists in areas such as linear, nonlinear, and integer programming have arisen. Other important groups include stochastic programming and dynamic programming. As a rule, these groups speak a reasonably common language, with strong dialectical differences among the subgroups.

Finally, the software engineer has emerged as a critical player. Here too, we find important subgroups, covering the model (problem representation), algorithm, user interface, database, and communication.

Our decision to separate modeling and algorithms represents a departure from standard practice. Historically, professionals within operations research focused on the design and implementation of efficient algorithms, with an appreciation of the importance of modeling, but relatively little science in the process. Increasingly, modeling is taking on the characteristics of a classical experimental science, and the design and testing of models is starting to look like other activities within engineering.

The role of languages should be fundamental to the field of modeling, but its importance has only really emerged in the area of real-time logistics as models have tried to step up to the role of actually running a company. We believe that the modeling of real-time systems, or more precisely, *operational planning*, is fundamentally different than classical strategic models operating on a static dataset. There are two reasons for this claim. First, the primary user of a strategic model is rarely someone from operations; most commonly, management is using the model to make decisions about infrastructure or markets. Such an individual does not understand operations at a level to criticize some of the finer assumptions of a strategic model. In fact, most strategic models do not produce highly detailed outputs, making it impossible to even criticize some of the assumptions of the model.

The second reason is that a strategic model is out of *context*. Real-time operational decisions are made against a rich fabric of information that is available only at the time that the decision is being made. This information includes not only the data that is in the computer, but other information obtained through visual inspection and telephone conversations (often referred to as *head knowledge*). The decision maker will know the weather, the current business climate (e.g. should he be emphasizing cost or service?), and recent historical events ("I just have to serve this customer on time today - I missed his last appointment!"). He may even know that the data in the computer was entered by someone who is notoriously sloppy and not reliable ("The driver is supposed to be available at 3pm, but he is never on time."). This background of contextual information makes it difficult for even a knowledgeable operations manager to critique decisions after the fact.

Our desire to study languages is motivated by a desire, ultimately, to build better operational planning tools than are available today. While classical research into mathematical models and algorithms will continue, improvements in operational models will, in our view, come from two directions. First, we need a broader set of skills to be brought to bear on the problem. These skill sets can be broadly organized under the four headings described above, realizing that each heading may include people from different fields. Second, we need better information about the problem we are optimizing, which requires the more active participation of different groups from within the company. Both of these directions involve the participation of people from fundamentally different backgrounds, who invariably speak different languages. If we want to solicit broader participation, we need to articulate a language to describe the problems that we are working on.

The goal of this paper is to focus attention on the language issue, and provide structure to the discussion. We make the argument that an understanding of the languages will do more than raise our sensitivity to the issue, but will actually change

the way we solve certain problems. Most importantly, we will seek to facilitate discussions between groups with complementary skill sets. Our focus is on large, complex problems, and the design and construction of production-caliber systems that will work in the real world. Thus, we take as a starting point that we will not be identifying problems with closed-form solutions, nor are we interested in relatively simpler problems where existing models and algorithms have proved satisfactory.

Our eventual goal is the development of an optimization environment that not only solves large, complex problems, but which is also transferable to other applications. We have seen the benefits of having general purpose solvers such as modern linear programming codes such as MINOS and Cplex; the value of these codes were significantly enhanced with the development of front-end interpreters such as GAMS and AMPL, which allow modelers with good math skills to "talk" to the solver through standard interfaces such as MPS-format files. This technology, however, is not up to the task of solving genuinely large, complex dynamic resource scheduling problems.

Our presentation begins in Section 6.2 with a discussion of specific applications, which defines the scope of problems that we are trying to solve. Here, we list not only specific examples, but propose a structure for what a problem instance is composed of. Next, Section 6.3 suggests a language for these problems. This language, which is a form of taxonomy, allows the modeler to step away from the minutiae of specific problem instances and focus on the fundamental structure of a problem. Then, Section 6.4 suggests a notational language for representing dynamic resource scheduling problems in an abstract way, that is the first step to identifying appropriate solution algorithms and coding the model on the computer. We make the argument that our mathematical formulation facilitates the communication between modelers, algorithmic specialists, software engineers, and industry. Section 6.5 addresses the representation of our mathematics in software. The software implementation also needs to address the interface with the mathematical algorithm, and the communication of the results back to the modeler and the user in the form of diagnostics. Finally, Section 6.6 offers a strategy for implementing models in the context of very complex problems.

"We will take your bags and send them in all directions." - Airline office in Copenhagen.

6.2 THE LANGUAGE OF APPLICATIONS

The first step in our process is to develop an understanding of the types of applications we wish to consider. Section 6.2.1 starts by giving a list of specific examples. Section 6.2.2 discusses in more general terms the structure of application–specific languages, since our eventual goal is to develop tools that cut across different application areas. Finally, Section 6.2.3 proposes a structure of an application, which forms the basis for our classification in Section 6.3.

6.2.1 Examples

- Inventory distribution - Classical inventory problems involve moving units of inventory to anticipate future demands, balancing the cost of distributing product to distribution points such as warehouses.

- Short-haul routing and scheduling - Drivers need to be scheduled through a sequence of loads with real-time updates, taking into consideration work rules, driver attributes, and load attributes.

- Load matching for long-haul trucking - Long-haul truckload motor carriers have the problem of matching drivers to loads, where the loads typically take one to three days to move.

- Driver/load management over a linehaul relay network - A tactical planning system manages over 6000 drivers and 10,000 loads per week over a national linehaul relay network.

- Tactical management of rail flatcars - A major railroad must optimize the repositioning of its fleet of 10,000 flatcars to move intermodal trailers and containers over its rail network. There are over 50 types of flatcars, each holding between one and eight of the over 40 types of trailers and containers.

- Routing and scheduling for chemical distribution - A system has been implemented for designing driver routes and schedules to deliver chemicals to customers which use the product at varying rates. Some customers have small tanks, which need to be clustered with other tanks to fully utilize the vehicle.

- Pipe cutting - A pipe manufacturer keeps stocks of pipes of different lengths in different bins. If an order comes in for a pipe of length, say, 12 feet, and there are no pipes in inventory of this length, the manufacturer needs to decide whether to take a 14 foot pipe out of inventory, leaving a 2 foot piece of scrap, or a 20 foot pipe, leaving an 8 foot section that may still be useful.

- Personnel training - A medical manufacturing company needs to send a group of experts around the country to train hospitals in the use of a new diagnostic equipment. After a hospital purchases the equipment, it calls to arrange for training, often with short notice.

- Machine scheduling - Machines, which might be drilling or stamping presses or similar equipment, needs to be scheduled to handle a sequence of demands.

- Ambulances - An ambulance is a type of machine that needs to respond to most requests very quickly.

- Personnel planning - Most companies face the problem of assigning people to projects that fit their skills, anticipating the needs of future projects, and realizing the effect of a project on a person (positive, as in training and experience, and negative, if the project exerts high demands on the individual, such as travel and time away from home).

6.2.2 Languages

Each of the examples above represents an application with a distinct language. A specific application is generally characterized by certain characteristics that describe the resources being managed. An obvious question arises: when are two applications really different, and when are they structurally the same with only cosmetic differences?

Is managing a fleet of taxis different than managing a fleet of trailers? Is truckload trucking really different than rail? Is making cars different than making oil? At one level, all of these problems are completely different. A truck would never be used to pick up a passenger at an airport. But we are modelers, and we are interested in how these problems differ in terms of our modeling approach.

Differences in vocabulary can even make two companies in the same industry look different. A long time ago, I had been working for a national less-than-truckload motor carrier, and started working for a direct competitor. One carrier moved freight out of satellites into "mother breaks," possibly incurring "linehaul variance" in order to make service. The competing carrier moved freight out of end-of-lines into "primary breaks," which possibly involved "moving air" in trucks in order to make service. The seemingly insignificant differences in vocabulary proved, initially, to be a major roadblock in my relationship with the new carrier.

Such contextual languages can magnify insignificant and meaningless differences. An alternative is to completely drop the language of the application and move to an entirely new vocabulary. For example, trucking, rail, ocean shipping and air freight are all industries that pose complex dynamic resource scheduling problems, where resources (trucks, boxcars, containers, aircraft) are resources that have to be scheduled to serve tasks (loads, shipments, boxes, customers). Such a vocabulary strips away the visual images that accentuate differences that contribute nothing to the actual structure of the problem. In the process, however, we discover that there are other differences that make boxcars different from truckload trailers, which are different from taxis.

These differences represent contextual data that is associated with each type of resource when placed in the setting of an application. One of the challenges of a language is making this contextual information explicit, so we can separate what is and is not important. At the same time, we need to appreciate the simplicity of saying "boxcar" instead of saying "a large reusable resource with relatively low value, which can serve a certain range of high volume products that generally have low service expectations, and which may enter and leave the system randomly."

6.2.3 Elements of an application

The first step in the development of a taxonomy is to describe the characteristics of a dynamic resource scheduling problem. A DRSP, in our view, is comprised of three fundamental elements:

- Resources - These are specific objects which need to be managed.
- Processes - Here we capture the physics of the problem, comprised of the laws that govern how the system evolves over time, and physical constraints that must be observed.
- Communication and control - Communication and control describes what decisions need to be made, who makes them, how information is provided to the decision maker, and how the decision maker separates good from bad decisions.

In the remainder of this section, we give examples of each of these. This structure becomes the basis of our taxonomy in Section 6.3.

Resources
Examples of different resources include:

- People - Drivers, crews (e.g. pilots), doctors (and other specialists), airline passengers, low income commuters.
- Equipment - Aircraft, locomotives, tractors, trailers.
- Conduits - Highways, pipes, conveyor belts.
- Goods - Food, computers, fertilizer, cigarettes.

It has been common in other fields to separate the *demands* placed on the system (referred to as customers, jobs, or loads, among others), from the systems that serve these customers. In our representation, the distinction between a customer and a server (in the language of queueing theory) or a job and a machine (in the language of machine scheduling) is somewhat artificial. We do view jobs and machines as distinct resources that need to be managed, with different attributes, but we have not found it useful to put them into fundamentally different categories. This does not mean that we will not explicitly recognize an entity called a customer (or job, or load); in fact, we have found it useful to refer to an object we call a *task*. However, in terms of our problem categorization, and our eventual mathematical model, the separation between resources such as machines, trucks, planes and people, representing resources managed by a company, and tasks, representing resources that serve an external customer, proved to be artificial, and as a result, we group them together.

Processes
Processes describe the physics of a problem, the laws that are not up to a manager or dispatcher, but rather are determined by the technology of the process in question.

- Physical constraints - These constraints always apply to a process at a point in time.
- Dynamics - These are the laws that govern the evolution of the system over time.

Examples of physical constraints include:

- Cannot have two objects in the same place at the same time.
- A single object cannot be in two places at the same time.
- Rate of process transformation - This describes the speed with which a resource may be modified over time.

Communication and control
Elements of the communications and control structure include:

- Controls - What decisions are made.
- Control structure - Who makes each decision.
- Information structure - Who has access to what information, and when.

- Measurement and evaluation - How do we determine when one solution is better than another, and who is performing the evaluation?

Controls can be discrete (assign a driver to a load) or continuous (restrict the flow of a liquid from one container to another). The control structure captures the hierarchy of decision makers, and the information structure represents who has access to what information, and when.

The issue of measurement and evaluation is a particularly challenging one. Most companies want to maximize profits, but the process of doing this requires managing a range of individual elements that are more easily measurable. Examples of goals that commonly arise in this problem class might include:

- Operating revenue, costs and profits.
- Speed and reliability of service to the customer.
- Utilization of equipment and other physical assets.
- Employee happiness and productivity.
- Return on financial investment.

In any setting, the issue is not which one of these to worry about, but what weight to put on each measurement, since all of them are important.

6.2.4 Sources of information

An important aspect of an application is the *sources* of information. We have identified the elements of an application as consisting of resources, processes and controls. A fourth class of data might be called *context*, capturing the rich tapestry of information that is not explicitly tied to resources, processes or controls, but which does nonetheless affect the decisions that are made. We view contextual data as a type of miscellaneous category, representing information that is not (at the moment) in the computer but which affects, usually in an ill-defined way, the decisions that are being made.

It is useful to briefly summarize the sources of data for each of our three major data classes:

- Resources - Most of the time, information on resources comes from computer databases, although it may be supplemented by other information (the computer tells you the aircraft is a Boeing 727, but a separate document tells you the speed of the aircraft). The process of implementing a model often forces the addition of data to a database that is important but which was never captured properly in the computer. In some cases, it is unavoidable that the knowledge of a resource exists only in someone's head, who probably got the information by looking out the window or talking on the phone (two forms of communication that will be outside the scope of a computer for the forseeable future).
- Processes - Information on processes generally comes from verbal descriptions from experienced operations professionals. Some information may come from the computer in terms of historical databases which capture past performance,

such as travel times, frequency of machine breakdowns, and so on. Typically, information regarding processes must be accumulated over time, since it is virtually impossible for any group of people to fully describe all the physical operations of a complex process.

- Controls - One of the most difficult sources of information to collect is the nature of the control process. The easiest information to collect is who makes what decision, although these lines of responsibility are often blurred. The most difficult piece of information is the process by which we determine the quality of a solution. It is relatively easy to compare apples and apples, particularly when the apples represent hard, measurable dollars. Real decisions, however, have to consider other issues, such as:

  - How to trade off noncomparable quantities, such as cost, service to the customer, handling of employees, and the productivity of the equipment?
  - How to trade off the quality of the information being used to make a decision?

  It is, of course, both useful and important to simply ask decision makers how they trade off different dimensions of the problem. While informative, these discussions are rarely accurate or complete, highlighting the simple fact that most people do not know how they make these trade offs. In particular, they do not always like to admit some of the trade offs they have to make when faced with difficult choices. For this reason, the theory of revealed preference can be especially valuable. Historical datasets can be used to record what decisions were actually made (and under what circumstances). In the process, it may be possible to learn about biases and attitudes toward risk.

It is the diversity of sources of information that creates the emphasis in this paper on languages. The participation of different groups of people, who together contribute the definition of the problem we are solving, requires the development of languages that facilitate this communication.

"You are invited to take advantage of the chambermaid." - A Japanese hotel.

6.3 A REPRESENTATION LANGUAGE

The goal of a representation system is to provide a means of communication between specific problem instances, and the structure of the problem in a way that is meaningful to a modeler. A successful architecture should allow a modeler to capture all the important elements of a problem in an elegant, compact manner. Our goal in developing a representational system is to provide the basis for a flexible classification system which will allow us to organize problems into fundamental classes with different structures.

An important side benefit of a taxonomy is to identify problems which may be superficially very different, but which share a common structure. Over time, a modeler will learn which problem structures lend themselves to specific solution approaches, as well as to recognize when a problem falls in a class for which there is no good solution. In this sense, the modeler is playing the role of the physician looking at

the symptoms of a patient, first to determine what disease the patient may have, and then to decide on a proper course of treatment.

Our goal is not to present a complete taxonomy; the complexity of problems that we consider in this paper make a full classification beyond the scope of this paper. However, we do feel it is useful to indicate the structure of a classification system, and indicate how this system can be used to facilitate the communication of results between application areas.

Classification systems have been used successfully in the past. The "$M/G/k$" paradigm for queueing systems, introduced by Kendall (see Gross and Harris, 1985, for a summary and references), served for decades to define the field of queueing theory. This framework actually takes the form "$A/B/X/Y/Z$" where A and B capture the arrival and service processes, while X, Y and Z represent the number of services, system capacity, and queue discipline. The machine scheduling literature uses "$\alpha|\beta|\gamma$" to represent, respectively, the machine environment, processing characteristics, and the number of machines (see Pinedo, 1995). Eiselt, Laporte and Thisse (1993) propose a "I / II / III / IV / V" taxonomy for classifying location problems, and use this to organize the extensive literature in location theory.

Following the structure of Section 6.2, we organize our representation system along three primary dimensions: resources, processes, and controls. We use the general style of queueing theory and machine scheduling, and propose that dynamic resource allocation problems be identified using the notation:

$$Resource||Process||Control$$

A complete specification of a taxonomy for this large problem class is beyond the scope of this paper, where our interest is more on highlighting the need for a classification system. In the remainder of the section, we highlight some of the key characteristics of these systems, followed by an illustration of how the general vocabulary of a classification system allows us to identify common properties of different problems.

6.3.1 Resources

The fundamental object that we are managing is called a resource, which can be machines, trucks, people, plants, or highways. We propose that some of the core characteristics of a resource can be divided along the following lines:

- Resource layering - The management of resources often requires the coupling of multiple types of resources in order to perform work, as in a driver and tractor pulling a trailer. In our view, a customer is a type of resource to be managed. Assigning a "job" to a "machine" represents a coupling of two resource layers to do "work."

- Resource attributes - Some resources are simple (a molding machine, a taxi, a trailer), some are more complex (aircraft, locomotives, industrial robot) while others are very complex (people). Some resources have static attributes (the molding machine) which do not change over time, while others have highly dynamic attributes that are always changing (people).

- Density of attribute space - We may have lots of resources with the same attributes (inventories of boxcars for a railroad) or lots of resources with very

different attributes, so that there are rarely more than one or two with the same attributes.

- Resource availability - Resources may enter and leave the system according to different physical processes. Customers often enter the system randomly and leave when we are finished, while other resources leave of their own accord (machine breakdowns).
- Value of time - Different types of resources may have highly different time values, suggesting that the scheduling of one or more resources dominates the scheduling of others.

6.3.2 Processes

Processes describe the physical laws that govern the management of the resources. A driver can get in a tractor and drive it, but a trailer cannot move by itself. It is here that we specify the constraints and structure of the system.

- Resource bundling - It might be that more than one resource is needed at the same time (two pilots in an aircraft, three locomotives pull a train). In the same category, we might say that one truck can hold several shipments, or one taxi might carry several customers at the same time.
- Reward structure - Often, more than one resource is required to accomplish a specific goal (satisfying a customer, or performing maintenance on a machine). The cost structure describes whether rewards (or costs) are additive or are more complex functions.
- Arrivals (A) and departures (D) of resources to and from the system - Here we capture the rules governing the arrival of resources to the system, or their departure. These processes may be fixed or stochastic, exogenous or controlled.
- Transformation controls - We may have operational goals that restrict the rate at which we are willing to transform resources.
- Physical constraints - Finally, we have physical constraints that restrict processes, such as conservation of mass, or technological limits like the speed of a machine or size of a buffer.

6.3.3 Communication and control

The last dimension of our taxonomy captures the communication and control structure, the process by which we make and evaluate decisions. Elements of this aspect of a system include:

- Actions - Actions may be discrete or continuous. For discrete actions, we may characterize decisions at different levels. The lowest level might represent simple steps such as coupling, uncoupling and transforming. Higher levels capture tightly coordinated decisions or broader strategies.
- Information profile - This dimension captures the difference between when we know about random events, and when we have to make decisions involving these

events. Systems where decisions are being made only as we learn about random events are highly dynamic and real-time, while other systems require decisions to be made well in advance, producing scheduled, predictable systems.

- Control structure - Complex systems are often controlled by different people, sometimes working at the same level of authority, or working at multiple levels.
- Evaluation - We have to represent the process of determining when one solution is better than another. Typically, we like to use a single-dimensional utility function such as costs or profits (or equivalent costs), but sometimes we need to more explicitly recognize multicriteria aspects, and possibly the presence of nonquantifiable qualities.

6.3.4 An illustration

Part of the value of a classification language is to identify similarities that are not immediately apparent on the surface. One example is a cutting stock problem. Consider a situation where you have bins of pipes of different lengths, ranging, say, from 1 to 20 feet. Orders come in randomly for lengths of pipe, and it is often necessary to take a longer length of pipe and cut it down. So, if we need a 12 foot length of pipe but do not have one, we can look at the different bins and choose something longer to cut down. We can choose a 14 foot piece, which will satisfy the demand and leave a two foot section as a leftover. Or, we can choose a 20 foot piece, leaving an eight foot section. It might be that there is a greater need for the eight foot section than the two foot, which might easily be discarded as waste.

While this problem sounds very different from a fleet management problem, in fact, the two are very similar when stated using similar terms. Both trucks and pipes are resources with different attributes. A truck in Chicago may be like the 14 foot section of pipe, while the truck in Cleveland is the 20 foot section of pipe. Assigning the 14 foot resource to the customer demand produces a two foot resource, just as assigning a Chicago truck to a load going from Chicago to Atlanta produces a resource in Atlanta. Thus, in both cases, we are choosing a resource to handle a task, which changes the attributes of the resource. The choice of the best resource to use depends on the value of the resource that is produced after the task is completed. The vocabulary of pipes and trucks serves only the disguise the similar qualities of the two problems.

" I don't do 'x' " - Logistics manager.

6.4 A MATHEMATICAL LANGUAGE

Mathematics is the language through which we communicate with mathematicians who develop algorithms, and communicate to software engineers who represent our data and implement the algorithms. As a rule, modelers adopt the language of a particular algorithmic approach when representing a problem. When we wish to optimize a problem, we usually find ourselves working within the framework of:

$$\min c^T x$$

subject to:

$$Ax = b \qquad x \geq 0$$

This paradigm requires us to model decisions in the context of a vector x, goals and objectives must be captured in the cost vector c, and the physics of our underlying process must be represented in the system of linear equations $Ax = b$. The success of linear programming is an indication that there is a wide variety of problems for which this language is sufficient. For more complex problems, it is too hard to use.

Linear programming, of course, is not the only way to solve a problem. In fact, there are a series of modeling paradigms that have been developed, each of which has evolved its own distinct language. Examples include:

- Constrained optimization - the mathematics of coupled decisions.
- Control theory - constrained optimization over time.
- Simulation - the mathematics of working with realizations.
- Markov decision processes - the mathematics of optimizing with expectations.

Constrained optimization is an especially powerful tool, but it does not deal well with uncertainty and random outcomes, and tends to be focused on making a decision at a single point in time. Classical control theory addresses the problem of optimization over time, but normally in a deterministic setting with relatively simple physics governing the evolution of the process over time (e.g. linear equations). Simulation and stochastic processes handle random outcomes extremely well, but normally work with simple rules and policies to handle the problem of making decisions within the process, hence their success in manufacturing processes and queueing networks. Markov decision processes embeds decision-making within a stochastic process, but depends on the language of state spaces and expectations, which generally cannot be used for large, complex problems.

A good analogy of mathematicians solving complex operational problems is the parable of the blind men and the elephant, represented in the poem by John Godfrey Saxe and reprinted in Figure 6.1. Specialists in a particular modeling technique invariably see a problem in the language of their training. The phrase most often used to communicate this pattern is a "hammer looking for a nail." Anyone who has ever modeled a complex problem has been guilty of this; in short, all of us are limited by the languages we speak. Speaking a language, and interpreting the world through this language, is not in and of itself a problem, as long as we are aware of the potential limitations. If we are successful at solving a problem using a particular language, then we have nothing to be concerned about. On the other hand, if we are finding ourselves unsuccessful at solving a problem, we should be aware that the solution may involve a restatement of the problem in a different language.

It is common in the field of modeling to start with an idea of the solution algorithm we want to use, and then formulate a problem in the language of this algorithm. The reason for this practice is simple: it is easy to formulate a problem in a way that can not be solved with current technology. At the same time, the practice can severely distort our view of the problem, because we have formulated it in the language of a particular technology. It is the premise of this paper that a more fundamental mathematical representation is needed, one which is not tied to a particular algorithmic approach, but which offers the flexibility of using a variety of different approaches. Most important, we feel that we need a mathematical representation that can handle

It was six men of Indostan
To learning much inclined,
Who went to see the elephant,
(Though all of them were blind).
That each by observation
Might satisfy his mind.

The First approached the elephant,
And, happening to fall,
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the elephant
Is nothing but a wall!"

The Second, feeling of the tusk,
Cried: "Ho! what have we here
So very round and smooth and sharp?
To me 'tis mighty clear
This wonder of an elephant
Is very like a spear!"

The third approached the animal,
And happening to take
The squirming trunk within his hands,
Thus boldly up and spake:
"I see," quoth he, "the elephant
Is very like a snake!"

The Fourth reached out his eager hand,
And felt about the knee:
"What most this wondrous beast is like
Is mighty plain," quoth he
" 'Tis clear enough the elephant
Is very like a tree."

The fifth, who chanced to touch the ear,
Said "E'en the blindest man
Can tell what this resembles most;
Deny the fact who can,
This marvel of an elephant
Is very like a fan!"

The Sixth no sooner had begun
About the beast to grope,
Than, seizing on the swinging tail
That fell within his scope,
"I see," quoth he, "the elephant
Is very like a rope!"

So, oft in theologic wars
The disputants, I ween,
Rail on in utter ignorance
Of what each other mean,
And prate about an elephant
Not one of them has seen!

John Godfrey Saxe

Figure 6.1 The Blind Men and the Elephant

the complexity of real problems, without creating an overly complex mathematical language that disguises the fundamental structure of the problem.

In this section, we highlight some of the issues that arise when translating our problem into mathematical notation. In keeping with the theme of this paper, we need to focus on *why* we are doing this; more specifically, with whom do we propose to communicate with this abstract language? There are three sets of users of a mathematical representation of a problem. First are the modelers, who have to translate the original problem into mathematics. Second are the algorithmic specialists, who can identify search algorithms when the problem is recognized to have a specific algebraic structure. Third are the software engineers, who need to explicitly represent data internal to the computer in the form of variables. Interestingly, software engineers have identified weaknesses in standard mathematical conventions and have, out of

necessity, invented new conventions that make software cleaner and more elegant. We propose to adopt some of these conventions in our own mathematical representation, in part to improve the clarity of our presentation, and in part to communicate more clearly with the software engineers who need to implement our expressions.

The goal of this section is to indicate that it is possible to develop a mathematical vocabulary that provides a more natural interface between the modeler, who needs to work with operations people from industry, and the more classical mathematical formulations around which algorithms are designed.

6.4.1 Resources

The foundation of our notation begins with a representation of the physical objects in the system, which we call resources. Our first step is to define the set of resources in the system at a point in time. For this, we define:

$\mathbf{a}_t$ = The vector of attributes of a resource at time t.
$\mathcal{A}$ = The space of possible values of $\mathbf{a}$

We need to provide for the presence of resource layering. This can be handled mathematically by defining subvectors and subspaces:

$\mathbf{a}_t^{(\ell)}$ = The vector of attributes of a resource in layer ℓ at time t.
$\mathcal{A}^{(\ell)}$ = The subspace of possible values of $\mathbf{a}^{(\ell)} \subseteq \mathcal{A}^{(\ell)} \subseteq \mathcal{A}$.

Hence we may write:

$$\mathcal{A} = \mathcal{A}^{(1)} \times \mathcal{A}^{(2)} \times \cdots \times \mathcal{A}^{(L)}$$

We use a superscript on $\mathbf{a}$ to indicate which subspace it belongs in. Thus, $a^{(1)} \in \mathcal{A}^{(1)}$, $a^{(2)} \in \mathcal{A}^{(2)}$ and $a^{(12)} \in \mathcal{A}^{(1)} \times \mathcal{A}^{(2)}$. All actions take place at some time t, so we sometimes drop the subscript t when there is no ambiguity.

We count the number of resources with a particular attribute using:

R_{at} = The number of resources in the system at time t with attribute $\mathbf{a}$
$R_t = \{\ldots, R_{at}, \ldots\}$

The vector R_t, then, combined with the space $\mathcal{A}$, captures the state of all the resources in our system at a particular time t. To solve a problem, we require as input R_0. This style, however, does not capture the availability of resources in the future in a natural way. For this reason, we generally find it convenient to let R_t^0 represent the vector of resources that first become available at time t. If we allow R_t^0 to be negative, then it represents exogenous changes in the availability of resources in the future.

This notation interfaces easily with standard databases, which store the status of each asset or resource and its attributes. It is the job of the modeler to determine which attributes are important, balancing realism with model complexity.

6.4.2 Controls

In classical optimization, decision variables are represented as x. In real applications, decisions are formulated in considerably different terms. For this reason, we suggest

an approach that begins with a formalism that is much closer to a real application, and then demonstrate the relationship between this more natural formalism and the standard mathematical representation of a problem.

We begin by characterizing three mechanisms for making decisions:

- Elementary decisions, or *actions* - This is the most basic step required to change the state of the system.
- Compositions - These represent small groups of elementary decisions that are often coupled together (such as, "every time we do X, we also do Y").
- Strategies - A strategy represents a larger pattern of coordinated activities that influence, but do not directly control, elementary decisions.

Optimization models often represent only elementary decisions, and depend on the mathematics of the problem to devise composite decisions and strategies (of course, a decision variable may represent a composition of several elementary decisions). Thus, these are *outputs* rather than inputs. In our approach, we believe the user should be able to specify the presence of certain types of compositions and strategies, and let the mathematics work *with* them.

The first step is to formalize the representation of elementary decisions. We propose that any action (in the class of DRSP's) can be represented as consisting of one of three fundamental transformations to the system:

- Couple - Two resources (in the same or different layers) are combined into a single, more complex resource.
- Uncouple - A complex resource is broken down into its elementary parts.
- Transform - A single elementary resource is modified to a resource with different attributes (but in the same layer).

In any system, resources are typically transformed using a small set of decisions available to the user. These are normally expressed using words like *"move driver empty to ..."*, *"set up the machine for ..."*, *"train a person to do ..."*, or *"deliver product to the customer ..."*. Expressed in these terms, such sets of instructions are normally very small. Applied to specific situations, on the other hand, they grow dramatically. For this reason, we use the concept of a *context dependent* set, borrowing a concept from object-oriented programming called function overloading. For example, we might define:

$\mathcal{D}^e$ = Set of elementary decisions available to the decision maker, containing basic steps that would modify a process.

$\mathcal{D}^e(\mathbf{a})$ = The set of decisions associated with transforming a single resource with attribute $\mathbf{a}$.

$\mathcal{D}^e(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ = The set of decisions associated with coupling two resources with attributes $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$.

We may need to capture the presence of multiple agents, at different levels in a hierarchy. Let $\mathcal{N}$ represent the set of decision makers. If we wish to capture a

hierarchical ordering of decision makers, define the subset $\mathcal{N}_k \in \mathcal{N}$ as the set of decision makers at the k^{th} level. Then, we write:

$\mathcal{D}^{(n)}$ = The set of decisions that may be made by decision maker $n \in \mathcal{N}_k$, where the level k is predefined for a particular model.

To avoid ambiguity, we will normally model the sets $\mathcal{D}^{(n)}$ as being mutually exclusive and, of course, collectively exhaustive. In some complex operations, this may not be the case, but we would propose that it would still be best to model each decision as being "owned" by a particular position or individual.

We need to capture the impact of a decision d on the state of the system. For this we define:

$$\delta_{at'}(d_t) = \begin{cases} 1 & \text{if decision } d_t \text{ made at time } t \text{ increases the number of resources with attribute } a \text{ at time } t' \geq t \text{ by 1} \\ -1 & \text{if decision } d_t \text{ made at time } t \text{ decreases the number of resources with attribute } a \text{ at time } t' \geq t \text{ by 1} \\ 0 & \text{otherwise} \end{cases}$$

We also need to *count* how often a decision is made, which we do using:

$x_t(d)$ = the number of times decision d is executed at time t

Any optimization specialist will easily recognize $x_t(d)$ as a standard decision vector in classical math programming, while the elements $\delta_{at}(d)$ are the elements of a constraint matrix. There is an important difference, however, in the modeling approaches. In the normal methodology of math programming, a modeler has to explicitly define the constraint coefficients $\delta_{at}(d)$ and the decision vector d, a process that may be simplified using interfaces such as AMPL or GAMS. In our approach, we ask the user to provide *elementary* decisions, such as are contained in the set $\mathcal{D}$, and then *derive* the resulting decision variables. The elements $\delta_{at}(d)$ are *endogenous* to the modeling process, not an input.

We have, at this point, represented elementary decisions using a new, highly flexible notation. It is important to realize, however, that while notation is an extremely important method of representing a problem, it is not the only, nor the best, method. Networks emerged in the 1970's as a popular modeling tool in part because they offered much faster solution times, and for certain problems, yielded integer solutions. But without question, one of the most widely appreciated dimensions of networks was that they were visual and, for many, much more accessible than algebraic representations. Of particular value was that a graphical depiction of a network was unambiguous, and represented a precise method of communicating the structure of a problem. The only limitation was that network diagrams could only be illustrative, given that it is impossible to actually draw networks with 50,000 arcs.

Our problem class also lends itself to a graphical depiction, even though our problems may not be pure networks. When we consider elementary decisions, it is clear that a resource can be represented as a node, and transformations as arcs. Coupling and uncoupling, on the other hand, requires a slightly richer visual vocabulary. In Figure 6.2, we suggest a simple way of visually communicating the coupling of flows. We believe that a richer visual vocabulary can be developed to extend the scope of problems that can be covered in this way.

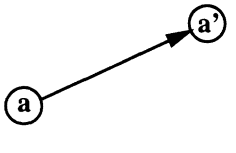


Figure a
Transformation

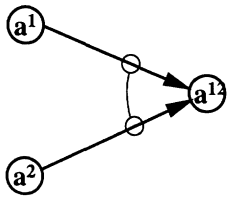


Figure b
Two–way coupling

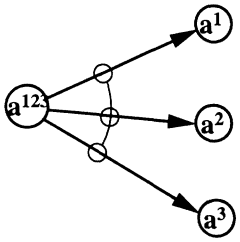


Figure c
Three–way uncoupling

Figure 6.2 Visual depictions of decisions.

6.4.3 Processes

We present a mathematical model of two aspects of processes. The first is the representation of the transformation of resources as they are acted on. Then, we discuss the modeling of physical constraints.

Transformations. For a given decision $d \in \mathcal{D}^e(a)$ or $\mathcal{D}^e(a^{(1)}, a^{(2)})$ we identify an action that changes the system. Consider a resource transformation decision $d \in \mathcal{D}^e(a)$. As a result of this decision, we obtain a new resource with attributes a'. The process of making this transformation generates a cost or contribution to the system, which we represent by:

$c(d)$ = the cost or contribution to the system resulting from a decision d, where $c \in \mathcal{C}$

Furthermore, this transformation takes a certain amount of time, represented using:

$\tau(d)$ = the amount of time required to complete decision d, where $\tau \in \mathcal{T}$

We capture all of these changes in a single mapping function:

$$\mathcal{M} : \mathcal{A} \times \mathcal{D} \rightarrow \mathcal{A} \times \mathcal{C} \times \mathcal{T}$$

For the case of a resource transformation, $\mathcal{M}$ produces the ordered triplet:

$$\mathcal{M}(a, d) \rightarrow (\mathbf{a}', c, \tau) \tag{6.1}$$

Using the concept of variable overloading, we can use the mapping $\mathcal{M}$ to handle other settings. In its most general form, it defines a mapping on a subset of resources with attributes $a \in \hat{\mathcal{A}}$:

$$\mathcal{M}(\hat{\mathcal{A}}, d) \rightarrow (\hat{\mathcal{A}}', c, \tau) \tag{6.2}$$

The strength of this representation is that it is closer to the form with which real problems are expressed. Given a resource vector R_{at}, and attribute space $\mathcal{A}$ and a transfer mapping $\mathcal{M}$, we can derive the constraints of a linear program.

Physical constraints. The standard math programming paradigm uses a set of constraints, often expressed as linear inequalities, that limit the set of feasible outcomes of a process. These constraints eliminate solutions that are both physically impossible, as well as those that are simply undesirable. In our view, we use the term physical constraint to eliminate the consideration of outcomes that are physically impossible, and use other means (in particular, the cost function c(d)), to limit the consideration of undesirable solutions.

We divide physical constraints into two broad groups:

- Physics - This captures basic laws governing conservation of mass, such as: a discrete resource cannot be in two places at the same time (conservation of resources); mass cannot be created or destroyed.
- Technology - These laws describe what can be accomplished within the limits of a particular technology, including the rate at which resources can be transformed (equation (6.1)), and what can feasibly be coupled or uncoupled. It is in this category that we would capture, for example, the size of a truck.

Conservation of mass may be represented using:

$$R_{at} - \sum_{d \in \mathcal{D}} x_t(d)\delta_{at}(d) \quad = \quad 0 \tag{6.3}$$

Equation (6.3) places a requirement that we make at least some decision about everything. We may equivalently use:

$$R_{at} - \sum_{d \in \mathcal{D}} x_t(d)\delta_{at}(d) \quad \geq \quad 0 \tag{6.4}$$

which allows us to "do nothing" to a particular resource. However, "doing nothing" is implicitly a decision, for which there must exist an appropriate mapping.

Technological restrictions come in two forms. First, they limit what resources can be coupled (and uncoupled). For example, you can couple a pilot with a jet, but you cannot couple two jets. These restrictions are best handled using a set of feasible couplings. The second restriction is in the rate at which resources are transformed. Consider a resource with attribute a; if we act on it with decision $d \in \mathcal{D}^e(a)$, then we produce a resource with attribute a'. We can represent the "flow" of resources from a to a' using:

$x_{a,a',t}$ = the number of resources with attribute a transformed to a' starting at time t

One way to capture technological limitations is to restrict this flow, using:

$$x_{a,a',t} \leq u_{a,a',t}$$

where:

$u_{a,a',t}$ = the maximum number of transformations of time $a \rightarrow a'$ that can occur at one point in time

6.4.4 *Controls*

We cover two topics under the category of controls. The first is the important issue of evaluation – How do we determine when one answer is better than another? Then, we present a metastrategy for solving large-scale, complex DRSP's.

Evaluation. One of the most difficult problems is evaluating a solution, or comparing two solutions to determine which one is better. For small problems, we may have well defined objective functions that reflect cost or service. As the problems become larger and more complex, we need to explicitly recognize that the real objective function is complex and poorly specified, and that any objective function we choose is at best a surrogate for the real problem. For example, most companies cannot clearly specify the proper tradeoff between cost and service. Instead, they are constantly guessing at the right tradeoff, and then looking for indications that they are making the right guess.

We can capture this process by proposing that we are trying to solve:

$$\min_{d \in \mathcal{D}} F(d)$$

where $F(d)$ is an unknown function, representing, for example, expected corporate profits over some time horizon. Since we cannot specify $F(d)$, we propose instead to solve its surrogate:

$$\min_{d \in \mathcal{D}} G(d)$$

where $G(d)$ is a measurable, possibly multidimensional, function. A central part of the language issue is realizing that companies really want to minimize $F(d)$, but the best we can do is solve $G(d)$. This means that part of our optimization problem is finding the right function $G(d)$.

If we wish to capture the presence of multiple decision makers, we would write:

$$\min_{d \in \mathcal{D}^{(n)}} G(d)$$

to represent decisions made by decision maker $n \in \mathcal{N}$. When this happens, we open the realm of interactions between and competition among different individuals. It is easy to find situations where individuals cooperate, and others where they compete. Thus, game theory may play a central role in these problems.

The question now is, what is $G(d)$? In most situations, a decision maker works with a set of functions $G_i(x)$ that represent specific statistics that they calculate to manage their job. The difficulty is that most of the time, we ignore these "user

defined" statistics. Instead, we impose our own cost function, which, most of the time, looks something like:

$$g_t(x_t) = \sum_{d \in \mathcal{D}} c_t(d) x_t(d) \tag{6.5}$$

We suggest that (6.5) should represent one of the elements that are considered, but that the other pet statistics of a decision maker should also be included. If we let $\mathcal{I}$ represent a set of such statistics, calculated using functions $g_i(x), i \in \mathcal{I}$, then we propose that these can be combined into a very general utility function. For example, we might define:

$$U(\mathbf{p}, g) = p_1(g - g_0) + p_2 e^{p_3(g-g_0)} \tag{6.6}$$

where $\mathbf{p} = (g_0, p_1, p_2, p_3)$ are parameters that control the shape of the function. Note that we use a special parameter g_0 that performs a scaling of g; we suggest that g_0 represents a target value for a particular quantity being measured. Now we may write:

$$g_t(\mathbf{w}, x_t) = \sum_{i \in \mathcal{I}} w_i U\left(g_{it}(x_t)\right) \tag{6.7}$$

where w_i is the weight given to dimension i, and $\mathbf{w}$ is the vector of weights. (In a multiagent environment, we would have a utility function $U^{(n)}$, weights $\mathbf{w}^{(n)}$ and a function $g_t^{(n)}(\mathbf{w}^{(n)})$ for each agent.) There is the modeling challenge of optimizing $G(\mathbf{w}, d)$ for a given $(\mathbf{w}, \mathbf{p})$, and the larger problem of determining $(\mathbf{w}, \mathbf{p})$, or more generally, determining the functional form of $g(\mathbf{w}, x)$ in (6.6) and (6.7).

Interestingly, for most companies, the functions g_i are quite well defined. It is only because our models require a cost function that operations research professionals tend to ignore these statistics, and focus instead on "our own" cost functions, which are often nothing more than different statistics. While we may argue that these are better in some way, they generally suffer from two limitations: first, they are new, so we do not have any sense what the numbers should look like (quick: what is the latest value of the S&P 500?) and second, we often cannot readily take a cost and translate it to an action I should be taking.

Finally, we have to reflect our desire to minimize costs over some horizon. Using a discount factor α to reflect the time value of money, we may write:

$$G(\mathbf{x}) = \sum_{t=0}^{T} U(\mathbf{p}, g_t(x_t))\alpha^t \tag{6.8}$$

More often, we need to explicitly recognize the presence of random elements, so we wish to solve:

$$G(\mathbf{x}) = E\left\{\sum_{t=0}^{T} U(\mathbf{p}, g_t(x_t))\alpha^t\right\}$$

Solution approach. The criticism of general "formulations" is that they are ultimately unsolvable, reflected in the common practice of starting with a solution technique and then trying to fit the problem into that approach. This process ensures

that if the modeler can formulate a problem, there will be an algorithm available to solve it. This is especially true in the context of large, dynamic systems.

We propose an alternative, *metastrategy* for solving large, complex, dynamic systems. Our strategy is based in part on two observations:

- Greedy strategies generally work well in dynamic settings.
- Any large problem can be decomposed into problems that are sufficiently small that they can always be quickly and easily solved.

These observations are best applied to problems that are dynamic (which is the class of problems we are considering here), stochastic (virtually all dynamic problems are fundamentally stochastic) and with imperfect data (which applies to almost all real systems). In short, as the problem becomes messier, the solution strategies become simpler.

A measure of the success of this approach is given in a series of recent papers (Powell and Carvalho, 1996a, 1996b). In (1996b), a large multicommodity network flow problem, with GUB constraints on certain arcs, was solved within three percent of an optimal linear programming relaxation, by solving nothing more than thousands of simple sorts. In Powell and Shapiro (1995), an ultra-large scale dynamic crew scheduling problem was solved by iteratively solving 20,000 very small transportation problems.

We call this solution approach *massive dynamic decomposition.* It is fundamentally different than classical decomposition methods, which attempt to solve a single, large-scale problem to optimality. In our method, we solve sequences of very small subproblems, that are linked together in a special way. The concept is illustrated in Figure 6.3. The large box represents the original optimization problem (6.8) that solves the problem over the entire attribute space $\mathcal{A}$, and over the entire planning horizon T. Each small box represents a specific optimization problem that is easily solved; in particular, we design these so that integer solutions are easy to obtain. Examples would be a simple sort, an assignment problem, a transportation problem or a small transshipment problem. They might also include small machine scheduling problems, or small set partitioning problems.

To develop our method, we begin by writing (6.8) recursively:

$$G_t(R_t) = \max_{x_t} g_t(x_t) + G_{t+1}(R_{t+1}) \tag{6.9}$$

subject to flow conservation, technology limitations and system dynamics, where we assume that R_t captures the state of the system. Because we do not know G_{t+1}, we replace it by an approximation that is adaptively estimated. Our approximation, which we denote $\hat{G}_t^k$ at iteration k, is defined recursively by:

$$\tilde{G}_t^k = \max_{x_t} g_t(x_t) + \hat{G}_{t+1}^k(R_{t+1}) \tag{6.10}$$

We further breakdown the problem by solving sequences over subsets of the attribute space. Let $\mathcal{A} = \mathcal{A}(1) \cup \mathcal{A}(2) \cup \ldots \cup \mathcal{A}(M)$ represent a partitioning of the attribute space. An appropriate partitioning might represent spatial boundaries or divisions of labor (different decision makers working at the same level of authority). For a given

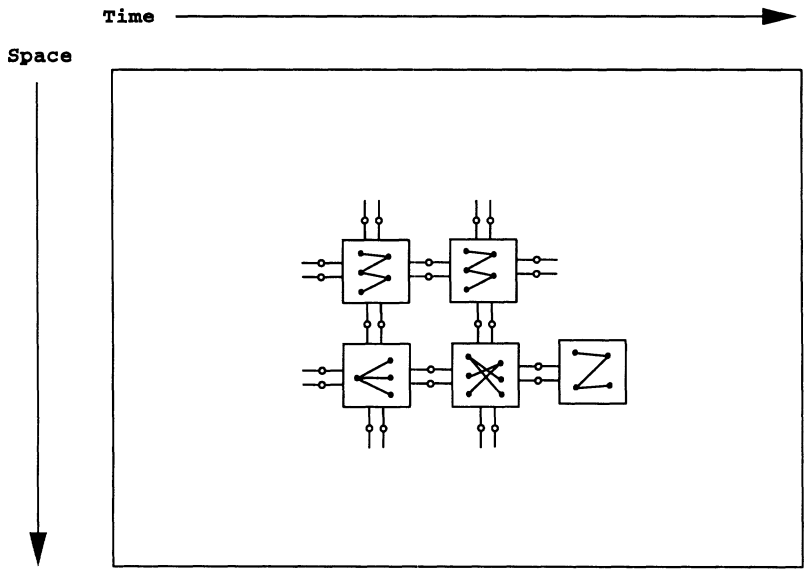


Figure 6.3 An illustrative decomposition.

subspace $\mathcal{A}(m)$, we define the subvector $R_t(m)$ which counts the number of resources with each attribute in that subspace. We can now write:

$$\tilde{G}_t^k(m) = \max_{x_t} g_t(x_t, m) + \hat{G}_{t+1}^k(R_{t+1}) \tag{6.11}$$

subject to appropriately restricted versions of the constraints for flow conservation, technology and dynamics. Let $\pi_t^k(m)$ be the dual variable associated with the flow conservation constraint (6.3) for subproblem m. We may now use this dual to update our approximation $\hat{G}_t^k(R_{t+1})$ (note that this function is defined over the entire vector R_{t+1}, and not just over $R_{t+1}(m)$). Several methods can be used to accomplish this updating, depending on the specific form of approximation chosen for $\hat{G}$, hence we represent the updating abstractly using the mapping:

$$\mathcal{Q}(\hat{G}_t^k, \pi_t^k) \rightarrow \hat{G}_t^{k+1}$$

It is important to choose a functional form for $\hat{G}$ so that (6.11) is easy to solve (and yields integer solutions). Particularly nice functional forms are linear, which tends to be unstable, and piecewise linear, convex and separable. Both forms may result in (6.11) reducing to a pure network. If this is not the case, it may be necessary to solve an integer programming problem, but if the decomposition is chosen well, the problem is likely to be extremely small and easy to solve.

"Place orders early because in a big rush we will execute customers in strict rotation."
- A sign in a tailor shop

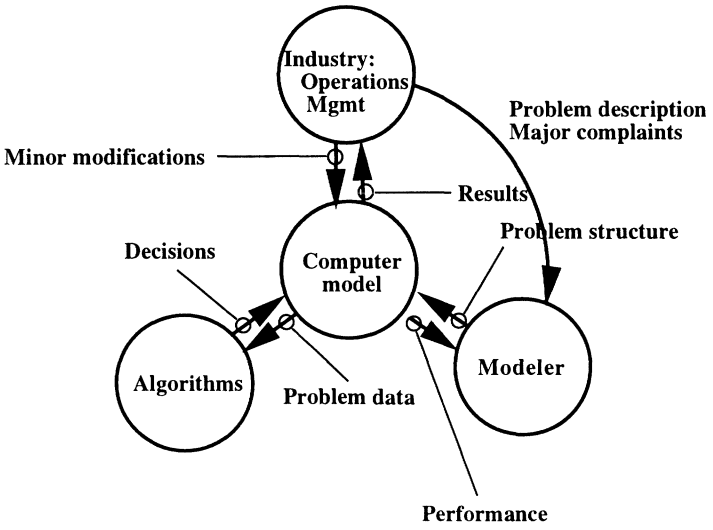


Figure 6.4 Lines of communication

6.5 A SOFTWARE LANGUAGE

Ultimately, the entire modeling process is designed to transform a problem into a form that can be captured on the computer, a process that requires the translation of the problem to software. For this, we must speak the language of software engineers.

A software implementation of a problem must accomplish three core functions:

- Problem representation - It must capture the elements of any problem, as represented by the resources, processes and controls.
- Problem solution - It must be able to "solve" the problem, typically by interfacing with one or more optimization solvers.
- Communication and feedback - It must be able to communicate the results of the solution procedure in languages that make sense to the different users of the system.

The best phrase describing the computer representation of a problem is a "computer model." This tired and well-worn phrase is, somewhat unfortunately, appropriate; unfortunately, because it is often misapplied. For many, the computer model and optimization solver are one and the same. In our view, the optimization solver (or solvers) are a part of a model, but are generally a small part. The more important part of a computer model is the representation of the problem itself.

Figure 6.4 provides a simplified depiction of the flow of information between different groups, covering "industry", modelers, software engineers, and mathematicians (algorithms). The challenge of software is to speak the languages of modelers, algo-

- Class: resources
  - Data:
    - Elements of the attribute space $\mathcal{A}^{(\ell)}$ for each layer ℓ.
    - Number of resources R_{at} of with attribute a at time t.
  - Methods:
    - Read and write routines.
    - Elementary assign, access and allocate routines.
- Class: processes
  - Data:
    - Various parameters required to describe physical and technological processes.
  - Methods:
    - The transfer function $\mathcal{M}$
    - The cost function $c(d)$
    - The time function τ
- Class: controls
  - Data:
    - Structure of the control process
    - Parameters governing the tradeoff of different dimensions of performance.
  - Methods:
    - Functions to rank two solutions - $G_t(x_t)$
    - Functions to provide the gradient of a solution with respect to specific parameters - $\nabla G(x)$.

Figure 6.5 Illustrative set of high-level object classes

rithms and industry. The major oversimplification of this simple illustration is the notion that industry can be represented using a single circle.

6.5.1 Problem representation

Problem representation, or, in the parlance of computer science, *domain knowledge encapsulation*, is the task of capturing the problem within software. Using our taxonomy, this means representing resources, processes and controls within software. This information should be captured in the form of software *objects*, each of which is comprised of certain data, and *methods* that act on that data. We can view resources, processes and controls as *classes*, which we may further subdivide in some cases into subclasses.

The notion of subclasses is especially valuable in the context of describing resources. If a resource is a class, resources divided into different layers (such as people, airplanes, tractors and trailers) are subclasses. They share basic attributes such as being discrete objects in time and space, but otherwise are substantially different. Complex resources such as people and airplanes are probably conveniently divided into further subclasses.

The representation of resources as software objects is, by now, widespread. Schrijver (1993), for example, describes a class library for the load matching problem of truckload trucking. We imagine that there may easily be hundreds (if not thousands) of programming shops that are solving transportation and logistics problems using an object-oriented methodology. The designs of these class libraries are rarely published, so it is hard to assess the state of the field. Most importantly, this elegant and powerful way of writing software is now an industry standard, and represents the "language" that software engineers speak. It is this reason that we have adopted an object style in our classification language, and in our mathematical notation.

6.5.2 Problem solution

The actual solution of our model is comprised of three steps:

- Design a suitable decomposition of the problem, considering the structure of the control system, the quality of the data, and the mathematical structure of the resulting subproblem.
- Identify the mathematical structure of the subproblem and an appropriate solver.
- Design and implement a computational strategy for solving the subproblems.

The first step is the most difficult, since it forces us to balance the capabilities of our library of solvers for subproblems with the organizational structure of the company, and the quality of the data. The designer needs to consider issues of diagnosability and data quality, which generally push for smaller decompositions, against solution quality, which generally encourages larger decompositions.

Given a particular decomposition, we must draw from our library of subproblem solvers. A simple library might include:

- Sort routines, hash tables
- Network solvers
  - Assignment problem
  - Transportation problem
  - Transshipment problem
- Specialized machine scheduling solvers
- Crew scheduling systems
- General linear programming
- General integer programming

Ferland, Hertz and Lavoie (1996) propose a class library of solvers for discrete optimization problems. The goal is to develop a class library of solvers that can be easily incorporated into specific applications. We support this concept, and suggest that the design of a general class library for representing DRSP's may enhance our ability to design optimization objects for solving them.

The last step is to design a computational strategy. The entire system might be implemented on a single processor, or spread among multiple processors on the same machine, different machines in the same location, or even spread around a series of machines located in different parts of the country. We propose using the strategy of massive dynamic decomposition to break large problems into large numbers of small problems. These small problems can be solved using our solver library, and if they are sufficiently small, there should be no significant computational problems. Furthermore, this solution technique will scale linearly with problem size, suggesting that we will not have difficulty solving large problems.

6.5.3 Communication

The last step is to communicate our results for evaluation and implementation. The choice of information to present depends, of course, on who we are presenting to. In our simple paradigm, the software needs to present results that are meaningful to three groups:

- Software engineer - Are there any bugs? Is the model being executed properly? Is it efficiently coded?
- Modeler - Is the system producing high quality solutions in a reasonable amount of time? Was the problem modeled correctly, in terms of making the right approximations, and using suitable decompositions? Are there errors in the data or the representation of the problem?
- Industry-operations - Is the system doing what we expect? Is the solution "high quality"? Is the real problem being represented correctly? Am I getting useful information? Can I explain why it is doing something?

Considerable expertise exists in the area of instrumentation to help software engineers. The bigger challenge for the modeling community is to address the combined questions of the modeler and industry representative. These questions can be broadly divided into four groups:

- Action - What should I do?
- Explanation - Why should I do it? The most frequently asked question is why *didn't* the system recommend a specific action?
- Information - What additional information can the system provide to help me make a decision on my own, over and above information that I already have?
- Evaluation - Are the recommendations valid? Is the quality of the solution high?

In systems today, the communication of a recommended action is usually only the starting point of a larger interrogation. If the recommendation is something that

I find inherently reasonable (and probably what I was going to do anyway) then I will probably implement it and move on. The interesting recommendations are those which run counter to my normal patterns, and where perhaps I should be changing what I would do. It is in these recommendations that potential benefits lie, but there are also many pitfalls, since the recommendation may be wrong. Since *all* recommendations must be considered in the context of poor data and a misspecified model, counterintuitive recommendations must be viewed skeptically, and primarily as an invitation to more questions.

With an appreciation of *what* people want to see from a software system, we have the challenge of *how* to display it. We propose that information from a model can be displayed graphically and textually. In so doing we are excluding other modes of communication such as sound, smell, taste, touch and ESP, although we acknowledge the power of intuition.

It has been our experience that modelers, with a modest understanding of the problem, are most comfortable with graphical displays for communicating activities, and selected high level performance statistics such as the objective function, CPU time, and other aggregate measures capturing cost and service. Others, especially those in operations, benefit from the ability of graphics to communicate the big picture and identify problem areas for further analysis. Then, they depend on a variety of selected statistics that they have used over the years to guide certain decisions. For example, just as many of us have become accustomed to looking at the Dow Jones Industrial Average as a measure of stock market performance, people in operations have their own set of pet statistics that they are comfortable with. It is our experience that experienced people in operations become extremely proficient using measures of performance that, on the surface, can seem confusing and uninformative to us.

Rather than criticize these measures because they are less meaningful (or worse) to modelers, we need to acknowledge the value of the experience operations professionals have gained in using these statistics. This is particularly true when we consider the different modes with which people in operations will evaluate a solution. Typically, users look at the results of a model dynamically in real-time or statically in a test setting. We can label these two modes as *out of context* and *in context*. Out–of–context evaluations are the easiest because we control the information available to the user. Much more realistic are in–context evaluations, where the user is looking at a recommendation at the same point in time that all the other information about the problem is available (including data not in the computer).

The most important guideline that emerges from this discussion is the importance of communicating results in a familiar format. Graphics is a powerful language because we all immediately respond to pictures. The only guidelines here concern the consistent use of shapes (wide lines mean more flow), motion (to indicate activity) and color (green is good, red is bad). Even more important is to consistently display certain activities in a certain way; experienced users will become accustomed to almost anything, as long as it is consistent. Other powerful visual cues involve the use of filters to highlight change (show me what is different). A human is exceptionally good at identifying changes if he is allowed to see the evolution from one solution to the next. At the same time, he is very poor at identifying changes when he is shown one solution, and then another, without any sense of what changed.

"Please do not feed the animals. If you have any suitable food, give it to the guard on duty." - Sign at a Budapest zoo.

6.6 A MODELING PROCESS

This paper proposes an approach for modeling complex problems, that takes as a starting point that the problem is too complex to describe with any accuracy, and where collecting (accurately) *all* of the data that *might* be useful is simply impractical. In particular, we propose the following starting axioms in the modeling process:

- Data is not correct, because humans are inherently tolerant of bad data.
- Operations people cannot describe what they are doing or why.
- Management does not understand their business, and cannot articulate with any accuracy how the operations people are solving the problem, or how they should solve the problem (but they believe they can).
- No single group of people can agree on what is best.
- Almost everyone can contribute an important piece of some part of the problem.

These problems demand an iterative process that involves guessing at many elements of the problem, and then using the performance of the system to refine our understanding of what the issues are, and what data is needed to support the system.

The basic steps in our paradigm are roughly as follows:

- Start with an industrial application.
- Use the taxonomy to identify the problem class.
- Use the mathematical notation to represent the problem abstractly.
- Design an appropriate decomposition that captures the important tradeoffs and which yields a solvable subproblem.
- Use software objects to implement the math.
- Apply an optimization engine to solve the problem class.
- Use graphical and textual outputs to communicate results back to the users.
- Continuously modify the process to improve the results.

This process is then imbedded within a larger process that starts with an initial formulation of the problem, and then iteratively improves our model of the problem. This process may sound obvious to most modelers, but we would claim that most modelers will try, as an initial design, to formulate the most accurate, precise model that seems practical given available data and the capabilities of our solvers. In our view, we should start with a much simpler model with the goal of getting *something* up and running and in front of the user. We propose to use the *simplest* model that gives meaningful results, even though we may be painfully aware of the limitations of our initial model.

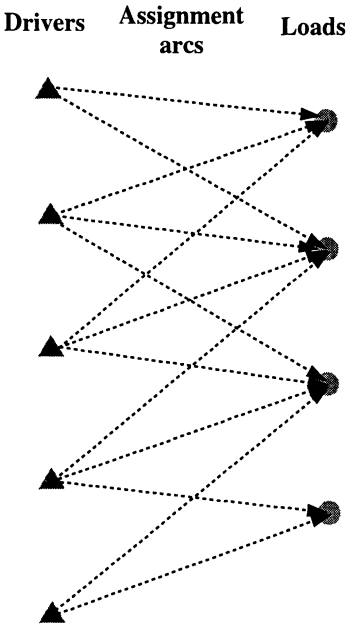


Figure 6.6 Global assignment network for assigning drivers to loads.

A simple example illustrates this process. Consider the problem of assigning a resource (such as a driver or locomotive) to a task (a load or a train). It would be very reasonable to formulate this problem initially as a myopic assignment model, as illustrated in Figure 6.6. This model seems simple and appealing, and easy to solve using standard network algorithms. However, when viewed in the context of a problem with bad data, our ability to solve this problem globally to an optimal solution is actually an impediment to the process of diagnosing the model and identifying the sources of bad data. A better approach is to implement a procedure where every driver is greedily assigned to the "best" load. Such an approach, which seems to have major weaknesses, is much easier to design and implement, and easier to diagnose. Only when the data improves should we use an algorithm that integrates more of the data using mechanisms such as dual variables.

Once a simple model is up and running, it is important to focus on basics such as the accuracy of the data coming to the system and the accuracy of our modeling of the physics of the problem, and ignore in the first iterations the actual quality of the solution. In most cases, the solution will be poor, but not because of the algorithm. After iterating on the data requirements and the physical process, the time will come when the data appears to be right, but the answer is not very good, and a user can find better solutions than the computer using only the information available to the computer. This is the signal that an advancement in the search procedure is needed.

This process then continues indefinitely (we assume that the problem is sufficiently complex that it can *always* be improved).

This strategy suggests the following process of implementing and improving a model:

- Start with the simplest possible model that represents a reasonable starting point to begin identifying data problems.
- Develop the model quickly to solicit user inputs as fast as possible.
- Design the software so that it is easy to change.
- Allow users to criticize specific actions in context.
- Respond to instance-specific criticisms.
- Adjust the model so that it fits within a well-defined, manageable work process.

In closing, we would claim that the paradigm suggested in this paper is evolutionary in nature, and integrates well-understood themes and concepts in the modeling community, but integrates them in a way that has not been proposed formally before.

"Ladies, leave your clothes here and spend the afternoon having a good time." A laundry in Rome.

References

Eiselt, H.A., G. Laporte and J.-F. Thisse. (1993). Competitive Location Models: A Framework and Bibliography. *Transportation Science*, 27:44–54.

Ferland, J.A., A. Hertz and A. Lavoie. (1996). An Object-Oriented Methodology for Solving Assignment-Type Problems with Neighborhood Search Techniques. *Operations Research*, 44:347–359.

Gross, D. and C. Harris. (1985). *Fundamentals of Queueing Theory.* John Wiley and Sons, New York.

Pinedo, M. (1995). *Scheduling: Theory, Algorithms and Systems.* Prentice Hall, New York.

Powell, W.B. and T. Carvalho. (1996a). Dynamic Control of Logistics Queueing Network for Large-scale Fleet Management. SOR Report 96-01, Department of Civil Engineering and Operations Research, Princeton University.

Powell, W.B. and T. Carvalho. (1996b). Multicommodity Logistics Queueing Networks. *European Journal of Operations Research.* Forthcoming.

Powell, W.B. and J. A. Shapiro. (1996). A Dynamic Programming Approximation for Ultra Large-Scale Dynamic Resource Allocation Problems. SOR Report 96-06, Department of Civil Engineering and Operations Research, Princeton University.

Schrijver, P.R. (1993). *Supporting Fleet Management by Mobile Communications.* P.R. Schrijver, The Netherlands.

7 SOLVING STOCHASTIC ROUTING PROBLEMS WITH THE INTEGER L-SHAPED METHOD

Gilbert Laporte
François V. Louveaux

7.1 INTRODUCTION

The deterministic *Vehicle Routing Problem* (VRP) is defined on a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a vertex set, and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is an edge set. Vertex v_1 represents a depot at which are based m identical vehicles of capacity D, while the remaining vertices are customers. A travel cost matrix $C = (c_{ij})$ and a travel time matrix $T = (t_{ij})$ are defined on E. Since the problem is defined on an edge structure, C and T are symmetrical. There exist, however, asymmetric versions of the VRP defined on an arc structure arising, for example in urban contexts with several one-way streets. The VRP consists of designing a set of m least cost vehicle routes starting and ending at the depot, such that each customer is visited exactly once, and satisfying some side constraints. We consider the following constraints.

i) **Capacity constraints.** With each customer v_i is associated a demand d_i. All demands are either collected or delivered, but not both. Then, the total demand of each vehicle route may not exceed D.

ii) **Time constraints.** There is a service time s_i associated with each customer. Then the total duration of each route, including service and travel times, may not exceed a given bound B.

For a recent survey and a bibliography on the deterministic VRP, see Fisher (1995), and Laporte (1997).

In several practical contexts, one or several components of the problem are random. This gives rise to a *Stochastic Vehicle Routing Problem* (SVRP). Here are three common examples:

i) **Stochastic customers.** Each customer v_i is present with probability p_i and absent with probability $1 - p_i$.

ii) **Stochastic demands.** The demand d_i of each customer is a random variable.

iii) **Stochastic times.** Service times s_i and travel times t_{ij} are random variables.

When some data are random, it is no longer possible to require that all constraints be satisfied for all realizations of the random variables. It follows that the decision maker may either require the satisfaction of some constraints with a given probability, or the incorporation into the model of corrective actions to be taken when a constraint is violated. The first case is referred to as *Chance Constrained Programming* (CCP). In the VRP with stochastic demands or times, the chance constrained model is relatively simple to solve as it can often be modeled as a deterministic VRP (Stewart and Golden, 1983; Laporte, Louveaux and Mercure 1989, 1992). In the second case, the SVRP can be cast as a *Stochastic Program with Recourse* (SPR). Here, a *first stage solution* is determined before knowing the realizations of the random variables. In a second stage, a recourse or corrective action can then be taken. For example, in the capacity constrained SVRP where demands are collected, the vehicle may return to the depot to unload as soon as it is full; in the time constrained SVRP, the duration of a route may exceed B, in which case a penalty may have to be paid. Such situations do not arise in deterministic contexts. In an SPR, the aim is to determine a solution of least expected total cost. For classical references on SVRPs, see Bertsimas (1992), Bertsimas, Jaillet and Odoni (1990), Golden and Stewart (1983), Jaillet (1988). For a recent survey, see Gendreau, Laporte and Séguin (1996). For a general reference on stochastic programming, see the recent book by Birge and Louveaux (1997).

In practice, SVRPs are much more difficult to solve than their deterministic counterparts, but in recent years, interesting advances have been made towards their solution by exact and approximate methods. In particular, the development of the *Integer L-Shaped Method* (Laporte and Louveaux 1993) has enabled the solution of several classes of SVRPs with recourse. The objective of this paper is to report on these developments. The remainder of this paper is organized as follows. In Section 7.2, we provide a general formulation of SVRPs with recourse. The Integer L-Shaped Method is summarized in Section 7.3. In Section 7.4, we report on implementations and results obtained on four different SVRPs. The conclusion follows in Section 7.5.

7.2 FORMULATION

The following generic two-index formulation can be applied to all SVRPs introduced in Section 7.1. Let x_{ij} be an integer variable equal to the number of times edge (v_i, v_j) appears in the first stage solution. If $i, j > 1$, then x_{ij} can only take the values 0 or 1; if $i = 1$, x_{ij} can also be equal to 2 if a vehicle makes a return trip between the depot and v_j. The problem is then:

$$\text{(SVRP)} \qquad \text{Minimize} \sum_{i<j} c_{ij} x_{ij} + Q(x) \qquad (7.1)$$

subject to

$$\sum_{j=2}^{n} x_{1j} = 2m \tag{7.2}$$

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 \qquad (k = 2, \ldots, n) \tag{7.3}$$

$$\sum_{v_i, v_j \in S} x_{ij} \leq |S| - 1 \qquad (S \subset V \backslash \{v_1\}; 2 \leq |S| \leq n-2) \tag{7.4}$$

$$0 \leq x_{ij} \leq 1 \qquad (2 \leq i < j \leq n) \tag{7.5}$$

$$0 \leq x_{1j} \leq 2 \qquad (j = 2, \ldots, n) \tag{7.6}$$

$$x = (x_{ij}) \text{ integer}, \tag{7.7}$$

where $Q(x)$ is the expected second stage recourse function.

Apart from the objective function, this model is that of the *m-Traveling Salesman Problem* (m-TSP) which generalizes the classical 1-TSP formulation (Dantzig, Fulkerson and Johnson, 1954). Constraints (7.2) and (7.3) are called degree constraints; subtour elimination constraints (7.4) prevent the formation of subtours disconnected from the depot. Note that demands and vehicle capacity are not taken into account in this model, except implicitly in $Q(x)$. Contrary to what is done in the deterministic case (Laporte, Nobert and Desrochers, 1985), there are no constraints to prevent the formation of routes including the depot, but violating capacity or duration constraints. However, it makes sense to impose, for example, constraints specifying that the expected demand of a route does not exceed D or that its expected duration does not exceed B. Such constraints have exactly the same structure as those used in the deterministic case.

The expected recourse function $Q(x)$ is problem dependent and is also related to the particular choice of possible recourse actions. For example, in the capacity constrained SVRP with collections, a possible recourse action is to return to the depot when the vehicle is full in order to unload, and then resume collections as planned. Alternatively, the remaining part of the planned route could be reoptimized. Another policy could be to plan a preventive return to the depot even if the vehicle is not full. In such a case, this decision could depend on the amount already collected and on the distance separating the vehicle from the depot. Such recourse policies are of course quite sophisticated and rather involved. The choice of a recourse action also depends on the time at which information becomes available. Always taking the same example, a vehicle not yet full may return to the depot if it is known that going to the next customer would cause its capacity to be exceeded.

The choice of an appropriate recourse policy depends of course on the situation at hand, but also on what is computationally feasible. In general, $Q(x)$ cannot be written as a linear function of x. Moreover, in most cases it will also be impractical to derive an expression for $Q(x)$ as a mathematical integer program in terms of the x_{ij} variables. Consider for example the SVRP with stochastic times, where the recourse consists of penalties proportional to overtime on routes whose duration exceed B. If a two-index formulation is used, then there is no way of knowing to which vehicle route belongs an edge (v_i, v_j), with $x_{ij} = 1$. Hence, the penalty cannot be expressed analytically in terms of the $x_{ij}s$. This difficulty does not arise in a three-index model (with the

third index specifying the vehicle associated with an edge), but only relatively small problems can be solved using such formulations (Laporte, Louveaux and Mercure, 1992). If penalties for overtime are not proportional to excess duration, then even in the case of a three-index formulation, $Q(x)$ cannot be simply formulated in terms of the x variables. In any case, $Q(x)$ typically does not have nice analytical properties such as continuity and convexity as it is the solution of a second-stage stochastic integer program (Birge and Louveaux, 1997). Note, however, that in many SVRPs, it is relatively easy to compute $Q(x)$, for a number of recourse policies, once a solution in terms of x is known. The Integer L-Shaped Method presented in the next section exploits this property.

7.3 THE INTEGER L-SHAPED METHOD

The Integer L-Shaped Method belongs to the same class as Benders' decomposition (1962) and the L-Shaped Method for continuous stochastic programming (Van Slyke and Wets, 1969). In the context of the SVRP, it operates on a so-called "current problem" (CP) obtained by relaxing (SVRP) in three ways: *i)* integrality constraints (7.7) are relaxed; *ii)* subtour elimination constraints (7.4) are relaxed; *iii)* the recourse function $Q(x)$ is replaced by a lower bound θ on its value. As in standard branch-and-cut methods, integrality is gradually reached through a branching process, and violated constraints (7.4) are introduced as they are found to be violated. In addition, a lower bounding constraint on θ, called "optimality cut", is introduced into (CP) at integer solutions. In other words, contrary to what is done in deterministic branch-and-cut, a cut may be introduced at a feasible integer solution as θ may be strictly less than $Q(x)$ and therefore, the objective value is not yet known and better solutions may exist down the current branch.

We now outline the Integer L-Shaped Method for the SVRP. This method assumes that given a feasible solution x, then $Q(x)$ can be computed. Moreover, a finite lower bound L on $Q(x)$ is assumed to be available.

Step 0. Set the iteration count $\nu := 0$ and introduce the bounding constraint $\theta \geq L$ to (CP). Set the value $\overline{z}$ of the best known solution equal to ∞. The only pendent node corresponds to the initial current problem.

Step 1. Select a pendent node from the list. If none exists stop.

Step 2. Set $\nu := \nu + 1$ and solve (CP). Let (x^ν, θ^ν) be an optimal solution.

Step 3. Check for any subtour elimination constraint violation using, for example, one of the heuristics described in Padberg and Grötschel (1985). If at least one violation of constraints (7.4) can be identified, introduce a suitable number of these constraints into (CP), and return to Step 2. Otherwise, if $cx^\nu + \theta^\nu \geq \overline{z}$, fathom the current node and return to Step 1.

Step 4. If the solution is not integer, branch on a fractional variable. Append the corresponding subproblems to the list of pendent nodes and return to Step 1.

Step 5. Compute $Q(x^\nu)$ and set $z^\nu := cx^\nu + Q(x^\nu)$. If $z^\nu < \overline{z}$, set $\overline{z} := z^\nu$.

Step 6. If $\theta^\nu \geq Q(x^\nu)$, then fathom the current node and return to Step 1. Otherwise, impose the cut

$$\theta \geq L + \left(Q(x^\nu) - L\right)\left(\sum_{\substack{1<i<j \\ x^\nu_{ij}=1}} x_{ij} - \sum_{1<i<j} x^\nu_{ij} + 1\right) \tag{7.8}$$

into (CP) and return to Step 2.

Most steps of this algorithm have been explained, except for constraints (7.8). These cuts use the fact that a feasible SVRP solution is fully characterized by the variables x_{ij} for which $i > 1$. They state that either the current solution must be maintained, in which case (7.8) becomes $\theta \geq Q(x^\nu)$, or a new solution must be identified, in which case (7.8) becomes $\theta \geq L$ or less, and thus redundant. Alternatively, one could impose

$$\sum_{\substack{1<i<j \\ x^\nu_{ij}=1}} x_{ij} \leq \sum_{1<i<j} x^\nu_{ij} - 1. \tag{7.9}$$

The advantage of (7.9) is that all its variables have a coefficient of 1 and are less likely to cause numerical problems (see, Séguin, 1994). On the other hand, constraint (7.8) can more easily be lifted as shown by Laporte and Louveaux (1993) and by Hjorring and Holt (1996). Finally note that similar to what is commonly done in branch-and-cut, valid inequalities can be introduced to reinforce the current relaxed problems. In particular valid inequalities involving θ, also known as "lower bounding functionals", can sometimes be generated. An example is provided in the following section.

7.4 IMPLEMENTATION AND RESULTS

We now illustrate the computation of $Q(x)$, of L, and of a lower bounding functional on two versions of the SVRP with stochastic demands. For the first two computations, we work on the standard m-vehicle SVRP. In the third example, only one vehicle is used.

7.4.1 Computation of $Q(x)$

In this example, there are m vehicles of capacity D, and each customer v_i has a stochastic demand d_i. A standard policy is for vehicles to follow their planned route and collect demand as long as their capacity is not exceeded. If at some point the vehicle capacity is exceeded, "failure" is then said to occur. The vehicle then returns to the depot to unload and resumes collections starting at the point of failure. The simplest case occurs when the following two conditions hold: *i)* the cumulative load of the vehicle at any point is exactly equal to D with probability 0; *ii)* the probability of having two failures on the same vehicle route is negligeable. Condition *i)* holds, for example, whenever d_i is a continuous random variable; condition *ii)* is satisfied provided that routes are planned in such a way that their total expected demand is not excessive with respect to the total vehicle capacity. Under these assumptions, Dror and Trudeau (1986) show how to compute the expected cost of recourse $Q^{k,\lambda}(x)$ for a given orientation $\lambda = 1$ or 2 of a route k, corresponding to $(v_{i_1} = v_1, v_{i_2}, \ldots, v_{i_{t+1}} = v_1)$. For $\lambda = 1$, this expected cost is

$$Q^{k,1}(x) = 2\sum_{j=3}^{t} \left[F^{j-1}(D) - F^{j}(D)\right] c_{1i_j}, \tag{7.10}$$

with $F^j(D) = P\left(\sum_{\ell=2}^{j} d_{i_\ell} \leq D\right)$. A similar expression can be computed for $\lambda = 2$, *mutatis mutandis*. The value of $Q(x)$ is then

$$Q(x) = \sum_{k=1}^{m} \min\{Q^{k,1}(x), Q^{k,2}(x)\}.$$

7.4.2 Computation of L

Consider the same problem as in Section 7.4.1 under the same two conditions. A valid lower bound L on $Q(x)$ is simply computed as follows. Relabel all customers so that $c_{1,i+1} \geq c_{1,i} (i = 2, \ldots, n-1)$ and let q_j be a lower bound on the probability of having at least j failures in the solution, with $j \leq m$, i.e.,

$$q_j = P\left(\sum_{i=2}^{n} d_i > (m + j - 1)D\right) \; (j = 1, \ldots, m).$$

Then a valid lower bound on $Q(x)$ is given by

$$L = 2\sum_{j=1}^{m} c_{1,j+1}\, q_j. \tag{7.11}$$

The validity of (7.11) is established as follows. Let $f_i(x)$ be the probability of having i failures in solution x. Then

$$\begin{aligned} Q(x) &\geq 2\sum_{i=1}^{m} f_i(x) \sum_{j=1}^{i} c_{1,j+1} \\ &= 2\sum_{j=1}^{m} c_{1,j+1} \sum_{i=j}^{m} f_i(x) \\ &\geq 2\sum_{j=1}^{m} c_{1,j+1}\, q_j = L. \end{aligned}$$

7.4.3 Computation of a lower bounding functional on $Q(x)$

Finally consider the same problem with $m = 1$. We now present in a simplified form a lower bounding functional developed by Hjorring and Holt (1996).

Consider a partial route $h = (v_{i_1} = v_1, v_{i_2}, \ldots, v_{i_r})$, with $r \leq n-1$, and let $H = \{v_{i_1}, \ldots, v_{i_r}\}$. Create an artificial customer $v_{i_{r+1}}$ with $d_{i_{r+1}} = \sum_{v_i \in V \setminus H} d_i$ and $c_{1,i_{r+1}} = \min_{v_i \in V \setminus H} \{c_{1i}\}$. Then construct route $k = (v_{i_1}, v_{i_2}, \ldots, v_{i_r}, v_{i_{r+1}}, v_1)$, compute $Q^{k,\lambda}(x)$ for $\lambda = 1$ and 2, as in (7.10), and observe that $\theta_h = \min\{Q^{k,1}(x), Q^{k,2}(x)\}$ is a lower bound on $Q(x)$ for any solution including h as a partial route. Therefore

$$\theta \geq (\theta_h - L)\left(\sum_{\ell=1}^{r-1} x_{i_\ell i_{\ell+1}} - (r-2)\right) + L$$

is a valid lower bounding functional on $Q(x)$.

7.4.4 *Available results*

To our knowledge, the Integer L-Shaped Method has been applied to four different SVRPs. We summarize the available results in Table 1. These results indicate that stochastic problems can only be solved to optimality for relatively small sizes, as compared to their deterministic counterpart. The last column suggests that problem difficulty increases when the first-stage part of the objective function provides little information on the exact expected solution cost.

Table 1: Summary of computational results

Problem	*Reference*	*Computation of $Q(x)$*	*Computation of L*	*Extra cuts*	*Largest size solved*	*Main source of difficulty*
Stochastic travel times, m vehicles	Laporte, Louveaux and Mercure (1992)	Direct probabilistic computation	0	Lower bounding functional on aggregate travel time	$n = 20$	Value of penalty for excessive duration
Stochastic customers, 1 vehicle	Laporte, Louveaux and Mercure (1994)	Dynamic programming	Linear relaxation of an integer linear program	Lower bounding functional on the expected route length	$n = 50$	Number of stochastic customers
Stochastic customers and demands, m vehicles	Séguin (1994), Gendreau, Laporte and Séguin (1995)	Direct probabilistic computation on a transformed objective	0 for the transformed objective	None	$n = 46$ $n = 70$ (stochastic demands only)	Number of stochastic customers
Stochastic demands, 1 vehicle	Hjorring and Holt (1996)	Direct probabilistic computation	Partial routes	Lower bounding functional on partial routes	$n = 90$	Demand/capacity ratio

7.5 CONCLUSION

Most of the developments witnessed in recent years in the area of exact algorithms for the SVRP have been centered on the application of the Integer L-Shaped Method proposed by Laporte and Louveaux in 1993. As the optimality cut (7.7) central to this method can be sometimes quite weak, computational enhancements are necessary to make this method more effective. As suggested by Laporte and Louveaux (1993), the cut can be lifted by examining neighbors of the current solutions. In addition, lower bounding functionals on the value of the expected recourse $Q(x)$ can sometimes be derived, as exemplified by the work of Hjorring and Holt (1996). It is encouraging to note that several medium size SVRPs can now be solved to optimality while this was impossible only a few years ago.

Acknowledgments

This research was in part supported by the Canadian Natural Sciences and Engineering Research Council under grant OGP0039682, and by the "Fonds National de la Recherche Scientifique" of Belgium. This support is gratefully acknowledged. Thanks are due to René Séguin for his valuable comments.

References

Benders, J.F. (1962). Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numerische Mathematik*, 4:238–252.

Bertsimas, D.J. (1992). A Vehicle Routing Problem with Stochastic Demand. *Operations Research*, 40:574–585.

Bertsimas, D.J., P. Jaillet and A.R. Odoni. (1990). A Priori Optimization. *Operations Research*, 38:1019–1033.

Birge, J.R. and F.V. Louveaux. (1997). *An Introduction to Stochastic Programming.* Springer-Verlag, New York.

Dantzig, G.B., D.R. Fulkerson and S.M. Johnson. (1954). Solution of a Large-Scale Traveling-Salesman Problem. *Operations Research*, 2:392–410.

Dror, M. and P. Trudeau. (1986). Stochastic Vehicle Routing with Modified Savings Algorithm. *European Journal of Operational Research*, 23:228–235.

Fisher, M.L. (1995). Vehicle Routing. In M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks on Operations Research and Management Science*, pages 1–33. North-Holland, Amsterdam.

Gendreau, M., G. Laporte and R. Séguin. (1995). An Exact Algorithm for the Vehicle Routing Problem with Stochastic Customers and Demands. *Transportation Science*, 29:143–155.

Gendreau,M., G. Laporte and R. Séguin. (1996). Stochastic Vehicle Routing. *European Journal of Operational Research*, 88:3–12.

Hjorring, C. and J. Holt. (1996). *New Optimality Cuts for a Single-Vehicle Stochastic Routing Problem.* Unpublished manuscript.

Jaillet, P. (1988). A Priori Solution of a Traveling Salesman Problem in which a Random Subset of Customers are Visited. *Operations Research*, 36:929–936.

Laporte, G. (1997). Vehicle Routing. In M. Dell' Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 223–240, Wiley, Chichester.

Laporte, G. and F.V. Louveaux. (1993). The Integer L-Shaped Method for Stochastic Integer Problems with Complete Recourse. *Operations Research Letters*, 13: 133–142.

Laporte, G., F.V. Louveaux and H. Mercure. (1989). Models and Exact Solutions for a Class of Stochastic Location-Routing Problems. *European Journal of Operational Research*, 39:71–78.

Laporte, G., F.V. Louveaux and H. Mercure. (1992). The Vehicle Routing Problem with Stochastic Travel Times. *Transportation Science*, 26:161–170.

Laporte, G., F.V. Louveaux and H. Mercure. (1994). A Priori Optimization of the Probabilistic Traveling Salesman Problem. *Operations Research*, 42:543–549.

Laporte, G., Y. Nobert and M. Desrochers. (1985). Optimal Routing under Capacity and Distance Restrictions. *Operations Research*, 33:1050–1073.

Padberg, M.W. and M. Grötschel. (1985). Polyhedral Computations. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*, pages 307–360, Wiley, Chichester.

Séguin, R. (1994). Problèmes stochastiques de tournées de véhicules. Ph.D. Thesis, Département d'informatique et de recherche opérationnelle, Publication CRT–979, Centre de recherche sur les transports, Université de Montréal.

Stewart Jr., W.R. and B.L. Golden. (1983). Stochastic vehicle routing: A comprehensive approach. *European Journal of Operational Research* 14:371–385.

Van Slyke, R. and R.J.-B. Wets. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics* 17:638–663.

8 CREW SCHEDULING IN AIR TRANSPORTATION

Guy Desaulniers, Jacques Desrosiers
Michel Gamache, François Soumis

8.1 INTRODUCTION

This paper addresses some crew scheduling problems faced by airlines during both the planning and the operational phases. Given a flight schedule and a fleet assignment, the planning process consists first in solving a crew pairing problem for each fleet of aircraft and then in constructing a monthly work schedule for each individual crew member. At the operational level, some crew pairings and schedules must be modified in order to compensate for certain disruptions that occur during the operations. The operational crew scheduling problem involves making these modifications.

Recent work has focused on the development of efficient optimization software for the solution of these air crew scheduling problems. Almost all this software relies on the same general solution methodology, namely a column generation approach embedded in a branch-and-bound algorithm. During the last decade, this approach has become very popular for solving various vehicle routing and crew scheduling problems. For airline transportation problems, it started with the paper by Lavoie, Minoux and Odier (1988) which was the result of work done at Air France during the 1984–85 period. In that paper feasible schedules are generated by solving classical shortest path problems on an acyclic graph. The first application for urban mass transit systems consisted in constructing schedules for bus drivers (Desrochers and Soumis, 1988). In that case, feasible schedules are generated by solving resource constrained shortest path problems using results obtained by Desrochers (1986) in his Ph.D. dissertation. Adaptations of this methodology to the airline context were developed at the beginning of the nineties. The evolution of the research done in urban mass transit can be found in the Proceedings of the Workshops on Computer-Aided Transit Scheduling (see for example, Desrochers and Rousseau, 1992). In the airline area, the development of the research can be found in the Proceedings of the AGIFORS meetings and in many papers cited in this survey.

In this paper we highlight the common structure of several crew scheduling problems. We give the general solution methodology used to derive near-optimal solutions, and even optimal solutions in some cases. We also present some of the results recently obtained for various real-world airline crew scheduling applications.

8.2 A VARIETY OF CREW SCHEDULING PROBLEMS

This section defines the crew scheduling problems encountered in the airline industry and briefly reviews the recent literature on each problem.

Pairing Construction: The crew pairing problem consists in constructing a set of pairings that cover at minimal cost a given set of flight legs. Usually all the flight legs pertain to the same fleet of aircraft. A pairing is a sequence of duties – a duty is a combination of flight legs, connections and ground waiting times forming a legal work day – separated by rest periods. It is worked by a crew leaving and returning to the same city; this city is called the crew base. Each pairing must be constructed in accordance with the collective agreement and the airline regulations. Individual crew members are not considered in this problem.

In the last two decades, over twenty papers have been published on the crew pairing problem. It can be modeled as either a Set Covering or a Set Partitioning problem, depending whether or not flight leg over-covering is permitted. Since the huge number of legal pairings make both models very difficult to solve, recent approaches proposed in the literature are composed of two modules: a pairing generator and a Set Covering/Partitioning problem optimizer. The main differences between these approaches lie in how and when the pairings are generated. Anbil *et al.* (1991), Graves *et al.* (1993), Hoffman and Padberg (1993), and Chu, Gelman and Johnson (1997) have proposed to generate *a priori* legal pairings using heuristic rules and to derive a good integer solution from this restricted set of pairings. Heuristic pairing generators have also been used by Wedelin (1995) in a feedback process relying on an approximation algorithm for integer programs, and by Crainic and Rousseau (1987), Lavoie, Minoux and Odier (1988), and Barnhart *et al.* (1994) in various column generation approaches. Finally, Desaulniers *et al.* (1997) have developed an optimal column generation approach in which pairings are generated by solving shortest path problems with resource variables. Their approach permits the integration of almost all collective agreement rules encountered in the airline industry. The reader is referred to the survey paper of Desrosiers *et al.* (1995) for a more detailed literature review.

Monthly Schedule Construction: The construction of monthly work schedules consists in assigning pairings and rest periods to each employee while considering preassigned activities such as training periods, holidays, medical appointments, etc. Each of these activities must be included at a specific time in the monthly schedule of a given employee. Several rules of the collective agreement govern the assignment of pairings and rest periods to the employees. Furthermore, the composition of the crew assigned to each pairing must comply with requirements on both individual qualifications and the distribution of on-board functions. The monthly work schedule construction process differs from one airline to another, but can usually be classified as one of the following three approaches: bidline, rostering or preferential bidding.

Bidline: In a bidline approach, schedules covering all pairings are constructed independently of the employees. Then, each employee bids for the available schedules. Finally, schedules are assigned according to a priority system unique to each airline. When necessary, some schedules are rebuilt to include preassignments that were not considered during the schedule construction phase; this usually causes several pairings to be left uncovered. The bidline approach is the easiest schedule construction approach and will not be addressed in this paper. Those interested in it may consult Glanert (1984) and Jones (1989).

Rostering: In a rostering approach, a personalized schedule is constructed for each employee while taking into account the employee's qualifications and preassigned activities. Several heuristics have been developed to solve this problem and they are summarized in Gamache and Soumis (1998). In recent works, the rostering problem is often modeled as a Set Partitioning problem. The methodology proposed by Ryan (1992) consists in first generating *a priori* a set of schedules using a heuristic generator and then solving the associated Set Partitioning problem by a conventional integer programming approach. Both Gamache and Soumis (1998) and Gamache *et al.* (1998a) have solved the problem by a column generation approach that relies on shortest path problems with resource variables to generate the feasible schedules.

Preferential Bidding: In a preferential bidding approach, personalized schedules are constructed as in the rostering approach while also considering a set of weighted items specified by the employees to reflect their preferences. These weights are used to evaluate the satisfaction score of feasible schedules and are chosen so as to differentiate the schedules of an employee as well as to reflect the priority of senior over junior employees. The aim of this approach is to assign to each crew member a schedule that maximizes the satisfaction of his or her preferences under strict seniority restrictions, i.e., this maximization must never be done at the expense of a more senior employee. Few researchers have tackled the preferential bidding problem due to its high complexity and to the fact that until recently few airlines were using this approach. Moore, Evans and Ngo (1978) and Byrne (1988) have proposed similar greedy heuristics to solve it. More recently, Gamache *et al.* (1998b) have formulated the problem as a Set Partitioning problem and used a column generation method embedded in a specialized branch-and-cut algorithm to solve it.

Operational Modifications: During day-to-day operations, airline carriers often have to deal with unexpected situations such as flight schedule changes, flight delays, employee absences, etc. The operational crew scheduling problem consists in quickly modifying parts of individual monthly work schedules in order to minimize the deviation from the planned schedules and the inconvenience experienced by the passengers and staff. This topic represents a new area of research and, so far, the paper of Stojković, Soumis and Desrosiers (1998) is the only one to address it. The authors present a column generation approach similar to the one proposed by Desaulniers *et al.* (1997) for the crew pairing problem.

8.3 A UNIFIED FORMULATION FOR AIR CREW SCHEDULING PROBLEMS

This section presents a unified formulation for all the crew scheduling problems described in the preceding section. Before stating it, we discuss the common structure of these problems that forms the basis of the formulation and introduce the appropriate notation.

Common Structure: All airline crew scheduling problems consist in finding feasible schedules for a set of commodities in order to cover a set of tasks. The commodities represent crews at specified home base stations in crew pairing problems and individual crew members in all the other crew scheduling problems. The tasks correspond to the flight legs in crew pairing; the specific on-board functions on all flight legs in operational crew scheduling; and the pairings in rostering and preferential bidding.

All these problems can be defined on a time-space network structure where each node represents a given location at a given time. Each arc (i, j) represents a given activity that begins at the location and time of node i, and ends at the location and time of node j. Depending on the problem, the activities that can be considered are flight legs, duties, pairings, briefings, debriefings, ground waiting time, night rests, days off, training periods, etc. A network is defined for each commodity based on its characteristics (home base station, availability, qualifications, ...). All feasible schedules for a commodity correspond to paths in the network associated with the commodity. All constraints restricting path feasibility are taken into account either by the network design (minimal connection times between flight legs, consecutive night rests and rest periods, juxtaposition of activities, preassigned activities, crew member qualifications, initial and final conditions) or during path construction via the use of resource variables (maximum time away from base in a pairing, maximum number of landings in a duty and in a pairing, minimum number of days off per period, minimum and maximum numbers of flight hours per period, ...).

The set of selected schedules is also subject to a set of global constraints (task covering constraints, pairing distribution across the different crew bases, minimum crew qualification requirements, requirements for each on-board function, ...). These constraints link together schedules of several commodities and, therefore, destroy the network structure of the problem. Note that, in rostering and preferential bidding problems, tasks must often be covered more than once since each of them corresponds to a pairing that requires several crew members for the same on-board function.

The objective function consists in minimizing total crew costs in crew pairing; minimizing a weighted sum of crew costs and inconvenience experienced by passengers and crew members in operational crew scheduling; minimizing the number of uncovered pairings in rostering; and maximizing the satisfaction of each employee in preferential bidding. Although they differ from one problem to another, these objective functions are all separable by schedule and by commodity.

Generic Air Crew Scheduling Problem: The common structure of the crew pairing, rostering, preferential bidding and operational crew scheduling problems allows us to define the following generic air crew scheduling problem: given a set of commodities and a network for each commodity that permits the generator of all feasible schedules for the associated commodity, find a set of optimal schedules, one for each commodity, such that all the tasks are covered a sufficient number of times and all

other global constraints are satisfied. Without loss of generality, we assume that the generic crew scheduling problem is a cost minimization problem.

Notation: Let W be the set of tasks to perform over a given horizon and K the set of commodities available to perform them. Denote by a_w the number of commodities required to accomplish task w and G^k the network associated with commodity $k \in K$. Next, let Ω^k be the set of feasible paths (schedules) in G^k, $k \in K$, and H the set of global constraints other than the task covering constraints. Associate with each path $p \in \Omega^k$ the following three parameters: a^k_{wp} which takes value 1 if p covers task $w \in W$, and 0 otherwise; b^k_{hp} the contribution of p to the global constraint $h \in H$; and c^k_p the cost of p. Finally, denote by b_h the right-hand side of global constraint $h \in H$.

A binary path variable θ^k_p is associated with each path $p \in \Omega^k$, $k \in K$. It takes value 1 if commodity k is assigned to path p, and 0 otherwise. The formulation also involves supplementary variables Y_s, $s \in S$, that are restricted to corresponding sets of feasible values I_s, $s \in S$. Such a variable may be a slack or a surplus variable allowing the violation of a task covering constraint or a global constraint, or it may simply be part of a global constraint to count the number of commodities used in a solution. The unit cost of Y_s, $s \in S$, is denoted by c_s, while its contributions to the covering constraint for task $w \in W$ and the global constraint $h \in H$ are given by a_{sw} and b_{sh}, respectively.

Formulation: Using this notation, the generic air crew scheduling problem can be formulated as the following generalized Set Partitioning model:

$$\text{Minimize} \quad \sum_{k \in K} \sum_{p \in \Omega^k} c^k_p \theta^k_p + \sum_{s \in S} c_s Y_s \tag{8.1}$$

subject to:

$$\sum_{k \in K} \sum_{p \in \Omega^k} a^k_{wp} \theta^k_p + \sum_{s \in S} a_{ws} Y_s = a_w, \qquad \forall\, w \in W \tag{8.2}$$

$$\sum_{k \in K} \sum_{p \in \Omega^k} b^k_{hp} \theta^k_p + \sum_{s \in S} b_{hs} Y_s = b_h, \qquad \forall\, h \in H \tag{8.3}$$

$$\sum_{p \in \Omega^k} \theta^k_p = 1, \qquad \forall\, k \in K \tag{8.4}$$

$$Y_s \in I_s, \qquad \forall\, s \in S \tag{8.5}$$

$$\theta^k_p \in \{0,1\}, \qquad \forall\, k \in K, \forall\, p \in \Omega^k. \tag{8.6}$$

The objective function (8.1) seeks to minimize total cost. The task covering constraints are expressed by (8.2). These constraints specify that every task $w \in W$ must be accomplished by exactly a_w commodities, unless a slack and/or a surplus variable permit the violation of this requirement. The set of global constraints is given by (8.3). Provided the path variable binary requirements (8.6) are satisfied, the convexity constraints (8.4) indicate that exactly one schedule must be assigned to each commodity. Note that these constraints are usually not present in the crew

pairing problem since $a_w = 1$ for each flight w to cover. Finally, the values that can be taken by the supplementary variables are restricted by constraint set (8.5).

8.4 SOLUTION METHODOLOGY

This section presents the solution methodology proposed for the generic air crew scheduling problem, namely, a column generation method embedded in a branch-and-bound scheme. We briefly describe the column generation approach applied to solve the linear relaxation of the problem and then discuss the various strategies used to obtain integer solutions. Additional details can be found in Desrosiers *et al.* (1995) and Desaulniers *et al.* (1997).

8.4.1 Linear Relaxation

The linear relaxation of the generic air crew scheduling problem is obtained from (8.1)-(8.6) by replacing the binary requirements (8.6) with the non-negativity constraints

$$\theta_p^k \geq 0, \forall\, k \in K, \forall\, p \in \Omega^k. \tag{8.7}$$

Since it usually contains a huge number of variables, this linear relaxation, as well as all the others encountered during the branch-and-bound process, is solved by a column generation approach (see Lasdon, 1970) that permits us to avoid the explicit enumeration of all the path variables. In this context, the linear relaxation (8.1)–(8.5) and (8.7) is referred to as the master problem.

Column Generation Process: The column generation process consists in solving alternately a restricted master problem and several subproblems, one for each commodity. The restricted master problem is derived from the master problem by considering a subset of its path variables which is updated at each iteration. Its role consists in finding the best solution with the available path variables and it is solved using the primal and/or dual simplex algorithms. Each subproblem is a shortest path problem with resource variables that permits the generation of all the feasible paths for the corresponding commodity. Using the dual information associated with the current restricted master problem solution, the role of the subproblems consists in generating feasible paths of negative marginal cost that, barring degeneracy, will help to improve the current solution. The subproblems are solved by dynamic programming. Starting with just supplementary variables (if no initial solution is provided), the column generation process ends when no paths of negative marginal cost can be generated by the subproblems.

Note that the constraints linking multiple paths are considered at the master problem level while those concerning a single path at a time are managed at the subproblem level via the network structure or the use of resource variables.

Network Structures: All air crew scheduling problems can be defined on time-space networks, one for each commodity. However, depending on the problem considered, the elementary component that defines the structure of these networks may differ from one problem to another. For crew pairing and operational crew scheduling problems, either flight-leg-based or duty-based networks may be used. The first structure seems more appropriate than the second one when the number of feasible duties is very

large (for example, for regional carrier problems in which a duty may contain a large number of flight legs; see Desaulniers *et al.*, 1998b) since it avoids the explicit enumeration of all the duties. However, it requires additional resource variables to model the rules restricting the feasibility of a duty. When the second structure is chosen, these rules can easily be considered during the complete enumeration of the duties. For rostering and preferential bidding problems, the structure of the networks is always based on the pairings derived from the crew pairing problem. Therefore, only monthly schedule feasibility rules need to be taken into account in these problems since pairing feasibility has been ensured during crew pairing construction.

Subproblem Notation: Let N^k, A^k, $o(k)$ and $d(k)$ be the node set, arc set, source node and sink node of G^k, respectively. Associate with each arc $(i,j) \in A^k$, $k \in K$, a binary variable X_{ij}^k to represent the flow on arc (i,j). Associate also with each node $i \in N^k$, $k \in K$, a vector of resource variables $\boldsymbol{T}_i^k = \{T_i^{kr} \mid r \in R^k\}$ where R^k represents the set of resources used to restrict path feasibility in G^k. Furthermore, for commodity $k \in K$, denote by $g_{ij}^k(\boldsymbol{T}_i^k)$ the cost of using arc $(i,j) \in A^k$ as a function of the resource variable values at node i; $g_j^k(\boldsymbol{T}_j^k)$ the cost of visiting node $j \in N^k \backslash \{o(k)\}$ as a function of the resource variable values at that node; $f_{ij}^{kr}(\boldsymbol{T}_i)$ the extension function of resource $r \in R^k$ on arc $(i,j) \in A^k$ that provides a lower bound on the value of the resource variable T_j^{kr} as a function of the resource variable values at node i; and $[l_i^{kr}, u_i^{kr}]$ the resource interval that restricts the values that can be taken by the resource variable T_i^{kr}, $i \in N^k$, $r \in R^k$. Finally, let $\boldsymbol{\alpha} = \{\alpha_w \mid w \in W\}$, $\boldsymbol{\beta} = \{\beta_h \mid h \in H\}$ and $\boldsymbol{\gamma} = \{\gamma^k \mid k \in K\}$ be the vectors of the dual variables associated with the constraint sets (8.2), (8.3) and (8.4), respectively.

Subproblem Formulation: Using this notation, the subproblem used to generate negative marginal cost paths for commodity $k \in K$ can be formulated as the following shortest path problem with resource variables:

$$\begin{aligned} \text{Minimize} \quad & \sum_{(i,j)\in A^k} \Big(g_{ij}^k(\boldsymbol{T}_i^k) - a_{w,ij}^k \alpha_w - b_{h,ij}^k \beta_h \Big) X_{ij}^k \\ & + \sum_{j \in N^k \backslash \{o(k)\}} g_j^k(\boldsymbol{T}_j^k) \Big(\sum_{i:(i,j)\in A^k} X_{ij}^k \Big) - \gamma^k \end{aligned} \tag{8.8}$$

subject to:

$$\sum_{j:(o(k),j)\in A^k} X_{o(k),j}^k = \sum_{i:(i,d(k))\in A^k} X_{i,d(k)}^k = 1, \tag{8.9}$$

$$\sum_{i:(i,j)\in A^k} X_{ij}^k - \sum_{i:(j,i)\in A^k} X_{ji}^k = 0, \qquad \forall\, j \in N^k \setminus \{o(k), d(k)\} \tag{8.10}$$

$$X_{ij}^k \Big(f_{ij}^{kr}(\boldsymbol{T}_i^k) - T_j^{kr} \Big) \le 0, \qquad \forall\, r \in R^k,\ \forall\, (i,j) \in A^k \tag{8.11}$$

$$l_i^{kr} \le T_i^{kr} \le u_i^{kr}, \qquad \forall\, r \in R^k, \forall\, i \in N^k \tag{8.12}$$

$$X_{ij}^k \text{ binary}, \qquad \forall\, (i,j) \in A^k. \tag{8.13}$$

The objective function (8.8) seeks to minimize the total marginal cost. Constraint sets (8.9)–(8.10) define the path structure of the subproblem in which one unit of flow must be sent from the source node $o(k)$ to the sink node $d(k)$. Compatibility between flow and resource variables is established in relations (8.11) which indicate that if arc (i,j) is part of the solution (i.e., $X_{ij}^k = 1$) then lower bounds on the values of the resource variables at node j are provided by the extension functions on arc (i,j). Resource variables are restricted to the intervals (8.12) while flow variables must take binary values (8.13). This shortest path problem with resource constraints can be solved by dynamic programming (Desrochers, 1986).

Cost and Extension Functions: The arc cost functions $g_{ij}^k(\boldsymbol{T}_i^k)$, $(i,j) \in A^k$, the node cost functions $g_j^k(\boldsymbol{T}_j^k)$, $j \in N^k \backslash \{o(k)\}$, and the resource extension functions $f_{ij}^{kr}(\boldsymbol{T}_i^k)$, $(i,j) \in A^k$, may all be nonlinear, non convex or discontinuous, as long as they are non-decreasing. However, in the applications for which results are reported in Section 8.5, the arc cost functions are constant (for instance, equal to the duration of the arc activity or to its associated satisfaction score). Moreover the node cost functions are null except in crew pairing, where the cost function at the sink node is a nonlinear, non-decreasing function of several resource variables, namely, total time away from base, total flying time and total number of landings. As for the resource extension functions, depending on the problem and the arc, they are either constant or linear of the form:

$$f_{ij}^{kr}(\boldsymbol{T}_i^k) = T_i^{kr} + t_{ij}^{kr}, \tag{8.14}$$

where t_{ij}^{kr} represents the amount of resource $r \in R^k$ consumed on arc $(i,j) \in A^k$. The constant functions are typically used to reset the value of the associated resource variable (for instance, the number of consecutive working hours is reset to zero at the beginning of a rest period) while the linear ones cumulate the overall amount of resource consumed during a schedule (for instance, the total number of landings per pairing or the total working time per monthly schedule).

8.4.2 Integer Solutions

A branch-and-bound scheme is used to derive integer solutions. At each node of the search tree, a lower bound is evaluated using solving a linear relaxation using column generation as described in the previous section. Several branching strategies can be used depending on the problem considered and they can even be combined during the same solution process.

Optimal Strategies: The optimal inter-task fixing strategy consists in requiring on one branch that two selected tasks be performed consecutively by the same commodity without specifying which commodity must perform them; and forbidding this on the other branch. Such decisions are taken into account at the subproblem level either by modifying the network structure or by restricting the state solution space in the dynamic programming algorithm.

The optimal special-ordered-sets fixing strategy selects a task and divides the set of commodities into two groups according to this task. These decisions are also dealt with at the subproblem level. On one branch, the task is forced to be performed by a commodity of the first group by removing it from the second group of networks.

On the other branch, the task is removed from the first group of networks and thus assigned to the second group.

These two optimal strategies can be applied only if the right-hand side a_w of all task covering constraint (8.2) is equal to one. When this is not the case, one can always divide the tasks that must be covered more than once into sub-tasks that must be covered exactly once.

Heuristic Strategies: Heuristic branching strategies may also be used when the problems considered are very large. For instance, the two optimal strategies mentioned above can be transformed into heuristic ones by creating a single branch at a time. Another heuristic strategy often used to reduce the solution time is called the path fixing strategy. It simply consists in raising to one the value of a selected path variable taking a fractional-value in the current linear relaxation. Such a decision is implemented by adjusting the right-hand-side values of the master problem and by removing from all networks the tasks covered by the selected path.

Heuristic branching strategies are often used together with other acceleration techniques. One such technique consists in taking more than one branching decision at once. For instance, all path variables taking a value greater than a predetermined threshold can be set to one at the same branching node. Another acceleration strategy aims to reduce the well-known tailing off effect of column generation by stopping the linear relaxation solution process when the improvement of the objective value over a given number of iterations is not sufficient. Finally, solving subproblems can be accelerated by limiting the set of states explored by the dynamic programming algorithm. In this way, a larger number of labels is eliminated at each node of the networks, resulting in a smaller number of paths being generated at each iteration and, possibly, in the elimination of optimal paths.

Cutting Planes: In order to tighten the linear relaxations, cutting planes can be integrated into the branch-and-bound scheme. Usually, cuts are added at the master problem level. However, in certain cases, they can also be inserted at the subproblem level. For instance, in preferential bidding, when the computed optimal solution of a linear relaxation contains a subset of fractional-valued path variables for the most senior employee and these variables have different scores, a cut restricting the score of a schedule for that employee can be added to its corresponding subproblem in the form of a bounded resource variable (Gamache *et al.*, 1998b). Indeed, since the highest possible score schedule has to be assigned to the most senior employee, all schedules whose score is greater than the weighted sum of these schedules' scores (the weights are given by the path variable values) cannot be part of an optimal solution.

Applications: In the applications for which results are reported, optimal inter-task fixing was used for crew pairing and operational scheduling, while path fixing was used for rostering and preferential bidding along with the acceleration techniques discussed earlier. Cutting planes were introduced when necessary during the solution process for preferential bidding problems.

8.5 COMPUTATIONAL RESULTS

This section reports recent results obtained by the GENCOL optimizer on real-world data for the four air crew scheduling problems. Developed at GERAD since the begining of the 1980s, GENCOL is based on the solution methodology described in the previous section. Except for the results for the preferential bidding problem, all the results are presented in tables with the same format. Thus, in Table 8.1 (crew pairing), Table 8.2 (rostering), and Table 8.4 (operational crew scheduling), the information is divided into six blocks. The first block presents the input size of each instance solved. The second block indicates the solution time required to solve the linear relaxation of each instance. The third block gives the details of the search for an integer solution: the number of branching nodes explored, the solution time needed to obtain an integer solution after the linear relaxation solution, and the optimality or integrality gap. The fourth block provides global information about the solution process: the total number of column generation iterations, the total number of columns generated, the global solution time and its split between the subproblems and the master problem. The fifth block presents the savings obtained by GENCOL for each instance as compared to the solutions produced by the existing method. Finally, the computer and optimizers used to solve the instances are specified in the last block. Table 8.3 which presents the preferential bidding problem results will be described in Section 8.5.3.

In general for the four problems, we can see that the number of columns generated is reasonable considering the size of the problems treated, i.e., millions of feasible columns. The branching strategy developed for each problem is efficient since only a few (and sometimes no) branching decisions are needed to produce an integer solution. Moreover, for crew pairing, rostering and operational crew scheduling problems, optimality/integrality gaps are small. For preferential bidding problems, huge gaps were completely closed by introducing the innovative cutting planes discussed in Section 8.4.2.

Finally we observe that, for the three problems where savings can be calculated, important economies can be achieved using the proposed solution approach. In a large company such as Air France, for example, a 1% saving on the total number of credited hours may represent a yearly reduction of ten million dollars in operating costs.

8.5.1 Crew Pairing

Table 8.1 presents the results produced by GENCOL on eleven medium-haul Air France instances (see Desaulniers *et al.*, 1997). These instances consider a one-week horizon for flight attendants (FAP) or pilots (PP) assigned to a specific type of aircraft (A300, A310, B737, etc.). Their size is characterized by the number of flight legs and the number of bases. The flight legs are the tasks to cover and therefore indicate the number of constraints in the master problem. The number of subproblems is seven times the number of bases: one for each departure day. As the results of Table 8.1 show, these two elements directly influence solution time.

All optimality gaps are relatively small (below 0.5%) for all eleven instances and even null for three of them. Moreover, the solution to the linear relaxation of these three instances was found to be integer. In the other cases, the search for an integer

Table 8.1 Results for crew pairing problems

PAIRING	9204 FAP A300	9204 FAP A310	9204 FAP A320	9204 FAP B727	9204 FAP B737	9201 FAP A320	9201 FAP B737	9209 FAP A320	9209 PP A320	9209 FAP B737	9209 PP B737
Input											
Flight Legs	280	154	392	342	477	701	566	739	743	1157	570
Bases	2	2	2	2	2	1	2	1	1	2	2
Stations	40	20	26	69	63	30	45	33	33	63	41
Nodes	2868	675	5399	1184	3829	983	1246	2253	1860	8829	2051
Arcs	40074	4390	115089	7258	41238	3601	6356	13052	10421	95504	16148
Resources	4	4	4	4	4	4	4	4	5	4	5
Linear Relaxation											
CPU (sec.)	769	19	1692	78	1033	44	390	369	215	8608	854
Integer Solution											
B. & B. Nodes	16	0	0	14	20	0	26	1	2	5	25
CPU (sec.)	1861	12	808	130	2757	42	944	376	249	13084	2140
Optimality Gap (%)	0.26	0	0	0.49	0.43	0	0.11	0.03	0.09	0.05	0.34
Global Information											
Columns Generated	3667	960	4402	2344	7365	1844	5950	5163	6848	20127	20150
CPU_{SP} (sec.)	2390	6	926	123	2583	7	361	268	63	11089	804
CPU_{MP} (sec.)	43	1	24	12	133	14	106	67	83	1818	192
CPU (sec.)	3004	19	1692	227	3589	44	1001	419	263	13983	2185
Saving (%)	8.29	4.43	5.30	1.89	7.15	4.28	7.34	8.91	3.48	8.49	0.49
Computer			SUN Sparc Station 2						HP 9000/730		
Optimizer			GENCOL 2.0/XMP						GENCOL 3.0/CPLEX		

solution was not difficult since few branching nodes were explored. Finally, note that the number of columns (pairings) generated varied between 960 and 20,150.

The savings of the GENCOL solutions compared to the Air France solutions range from 0.49% to 8.91%. In crew pairing problems, the expected savings usually differ for long-haul and short-haul fleets. For short-haul instances, the time worked by the employees on flight segments represents between 90% and 95% of the total cost. Thus, a maximum saving of 5% to 10% on the total cost can be expected by using an optimizer on such problems. For long-haul instances, this time represents less than 70% of the total cost, giving rise to higher potential savings. In other experiments, GENCOL has yielded savings ranging from 10% to 25% for Air France long-haul instances.

8.5.2 Rostering

Results for Air France rostering instances are presented in Table 8.2 (see Gamache *et al.*, 1998a). These instances consider the construction of monthly schedules for cabin chiefs (CC) and hostesses and stewards (HS) at a specific base (Orly or Roissy). Rostering problems are characterized by the number of mother pairings (pairings that must be assigned to an entire crew) and the number of employees. The former number corresponds to the number of tasks to be covered and the latter one indicates the number of schedules to construct. The number of constraints in the master problem is given by the sum of these two numbers, while the number of subproblems

to solve at each iteration of the column generation process is equal to the number of employees. However, note that for the HS problems, the right hand side of the task covering constraints is equal to the number of daughter pairings (pairings that must be assigned to a crew member) since more than one employee must be assigned to a pairing.

Initially, the CC problem and HS problem for Roissy contained 2798 constraints and 284 subproblems, and 4070 constraints and 1151 subproblems, respectively. Given their huge size, these two problems were divided respectively into two and three similar problems having approximately the same size.

Table 8.2 Results for rostering problems

ROSTERING	CC			HS			
	Orly	Roissy1	Roissy2	Orly	Roissy1	Roissy2	Roissy3
Input							
Mother Pairings	422	1248	1266	454	977	969	973
Daughter Pairings	422	1248	1266	3226	3256	3270	3284
Crew Members	56	145	139	237	382	383	386
Nodes	814	1931	1969	1286	2388	2389	2383
Arcs	29755	51901	56348	147486	130147	129668	133451
Subproblems	56	145	139	237	382	383	386
Resources	6	6	6	6	6	6	6
Linear Relaxation							
CPU (sec.)	247.4	1511.5	1281.6	823.1	2069.4	2127.5	1407.6
Integer Solution							
B.& B. Nodes	36	57	89	215	31	39	12
CPU (sec.)	253.5	6433.0	11620.7	1105.5	2672.2	2951.8	1432.6
Optimality Gap (%)	0.358	n.a	n.a	0.627	n.a.	n.a.	n.a.
Global Information							
MP Iterations	197	606	811	290	528	584	345
Columns Generated	4652	12998	17808	9204	13508	13508	8374
CPU_{SP} (sec.)	231.6	813.0	1180.5	859.0	1877.5	2258.5	1362.8
CPU_{MP} (sec.)	416.7	7131.5	11721.8	1069.6	2864.1	2820.8	1477.4
CPU (sec.)	648.3	7944.5	12902.3	1928.6	4741.6	5079.3	2840.2
Saving (%)	3.9	0.8		7.4		7.6	
Computer	HP 9000/735						
Optimizers	GENCOL 3.0/CPLEX 2.0						

A major difficulty encountered in rostering problems is the presence of a large number of subproblems to solve at each iteration of the column generation process, that is, one for each employee. A strategy consisting in solving only a subset of these subproblems at each iteration was introduced in the solution process. By adjusting the dual variables more rapidly, this strategy – which can be compared to partial pricing at the subproblem level – has considerably decreased solution time.

This strategy and others have permitted us to solve large scale rostering problems in reasonable times. Moreover, when the integrality gap could be computed, it was always found to be less than 0.6%. The results of Table 8.2 also show that the savings obtained by GENCOL on Air France instances were up to 7.6% compared with the solutions produced by the existing method at Air France.

8.5.3 *Preferential Bidding*

To deal with strict seniority constraints and the variety of bids that each employee can make, a sequential approach must be used to solve the preferential bidding problem. At each iteration of this sequential approach, a Set Partitioning problem (denoted $\mathcal{IP}^k$) similar to those encountered in crew pairing or rostering problems has to be solved. In fact, there are as many Set Partitioning problems to solve as there are schedules to construct.

Table 8.3 presents results for some instances of the Air Canada preferential bidding problem (see Gamache *et al.*, 1998b). These instances are characterized by: the number of employees which indicates the number of Set Partitioning problems as well as the number of subproblems to solve; the number of tasks which is proportional to the size of the master problem; and the variety of bids that are offered and made which influences the number of identical-score schedules.

Table 8.3 Results for preferential bidding problems

PREFERENTIAL BIDDING	YYZ 320	YYZ DC9	YUL 320	YUL DC9
Input				
Number of Pilots	108	82	46	62
Number of Pairings	568	602	267	329
Number of Nodes	77 802	69 837	16 610	28 497
Number of Arcs	128 201	118 347	28 146	45 194
Global Information				
Total CPU (hours:minutes)	5:42	8:01	1:04	1:10
Master Problem (%)	28	58	10	21
Subproblems (%)	72	42	90	79
General Information				
Cutting Planes				
$\mathcal{IP}^k$-problems with Gaps	4	4	5	1
Total Number of Cuts	8	5	9	1
Internal Branching				
$\mathcal{IP}^k$-problems with Fractional Solutions	32	22	13	10
Total Number of Branching Nodes	151	82	46	67
Saving (%)		5		
Computer		HP 9000/715		
Optimizers		GENCOL 4.0/CPLEX 3.1		

The instances consider the construction of monthly work schedules for pilots at a specific base (YYZ–Pearson or YUL–Dorval) assigned to a specific aircraft type (A320 or DC9). The first block of the table describes the size of the instances treated: the numbers of pilots, pairings, nodes and arcs. The second block provides global information. The third block indicates general information: the number of $\mathcal{IP}^k$ problems for which cutting planes were added and the total number of cutting planes added, the number of $\mathcal{IP}^k$ problems needing branch-and-bound decisions and the total number of branching nodes explored. The fourth block indicates the savings achieved by the GENCOL solutions. Finally, the computer and optimizers used to solve the four instances are specified in the fifth block.

Compared with the solution times presented in the other tables, the solution times required for the preferential bidding problems are huge. Recall that this is due to the large number of Set Partitioning problems to solve for each instance, one for each employee. Note that cutting planes were used in only a few $\mathcal{IP}^k$ problems, but were in each case very effective in closing the gaps that sometimes reached up to 99%.

At Air Canada, managers and union leaders have negotiated a special agreement: employees will agree to increase their workload by 5% under the condition that the company provides monthly work schedules that respect their bids and seniority constraints. After using for three months the schedules computed by GENCOL, 95% of Air Canada's employees voted in favor of the new schedules which, we observe, have resulted in savings of 5% for each problem.

8.5.4 Operational Crew Scheduling

The solution method developed to solve the operational crew scheduling problem must be considered as a prototype. The main goal was to estimate the potential of column generation techniques for solving this problem. The model that has been used is closely related to the one developed to solve the crew pairing problem.

Table 8.4 Results for operational crew scheduling problems

OPERATIONAL CREW SCHEDULING	PROBLEM 1 TEST 1.1	PROBLEM 1 TEST 1.2	PROBLEM 2 TEST 2.1	PROBLEM 2 TEST 2.2
Input				
Active Tasks	15	58	28	114
Frozen Tasks	91	48	182	96
Nodes	526	1369	966	2179
Arcs	840	3589	1567	6031
Subproblems	14	23	21	30
Linear Relaxation				
CPU (sec.)	5.7	202.4	23.1	810.6
Integer Solution				
B.& B. Nodes	0	0	0	5
CPU (sec.)	0	0	0	396.6
Integrality Gap (%)	0	0	0	0
Global Information				
MP Iterations	14	82	17	228
Columns Generated	231	978	430	6115
CPU_{SP} (sec.)	5.9	233.4	28.2	1118.0
CPU_{MP} (sec.)	0.1	7.2	0.4	83.7
CPU (sec.)	7.2	242.1	30.4	1207.2
Saving (%)	n.a.		n.a.	
Computer	HP 9000/715			
Optimizers	GENCOL 3.0/CPLEX 3.0			

The prototype has been tested on instances from a US carrier and the results are presented in Table 8.4 (see Stojković, Soumis and Desrosiers, 1998). The operational crew scheduling problem is characterized by the number of active tasks and the number of subproblems. Active tasks are flight segments that are completely within the

operational period that has to be modified. All other flight segments are in contrast called frozen tasks, i.e., they will be considered as preassigned. For small instances, the presence of such frozen tasks among the generated columns helps to obtain a naturally integer solution to the linear relaxation.

A major difficulty with the solution of operational crew scheduling problems is the use of many resources (those used for both crew pairing and rostering) in the shortest path problems. This increases both the number of labels to be compared and the solution time. This is confirmed by the results of Table 8.4 which indicate that over 90% of the solution time is spent solving the subproblems. Finally, note that the solution time increases rapidly with the number of active tasks. Indeed, for the two relatively small problems considered, a factor of four in the number of active tasks has multiplied the solution time by approximately 35 and 40, respectively.

8.6 CONCLUSION

This paper has presented recent computational results on real-world instances of four air crew scheduling problems. These results were produced by an optimizer based on a column generation approach. The paper has shown that this general solution methodology is efficient for solving the air crew scheduling problems considered. It is flexible enough to integrate all real-world constraints that must be taken into account in the different problems. Moreover, specific strategies can easily be introduced into this general approach to improve the solution time for particularly difficult problems. These solutions have yielded appreciable savings for most of the problems solved.

In the coming years, the following research directions could be considered. During the past ten years, the column generation approach has increased in popularity, as is illustrated by its application to airline crew scheduling problems. However the size of the problems that have to be solved is one of the major difficulties to overcome; indeed, large companies already offer from 5,000 to 10,000 flights per week. Since schedules are usually irregular over a month, the problem sizes become astonishing. Many acceleration strategies will have to be developed.

A second challenge for researchers is the use of operations research in day-to-day operations. The difficulty consists in developing tools that will be able to adequately inform the decision makers and resolve unplanned situations in real time. In fact, these problems are generally smaller than those encountered in the planning phase but usually they are more complex because of the simultaneous interaction of many aspects. Routing of aircraft together with pairing construction as well as simultaneous pairing and roster construction are good examples of this kind of problem. The first steps in this direction are encouraging.

Finally, the solution of the interaction problems described in the preceding paragraph are important for small companies. For these companies, the interaction between the different phases of the schedule construction is quite important. Intregation attemps have already begun in other areas of transportation. For example, in urban mass transit the cost of bus driver schedules is more important than the cost of the vehicles. The sequential approach consisting of constructing bus trips prior to driver schedules is not natural; the two problems should be solved simultaneously. Knowledge of and expertise in interaction problems needs to be transfered from one transportation area to the other. In fact, much research and development remains to be done.

Acknowledgments

This research was supported by the Quebec Government (Programme Synergie du Fonds de Développement Technologique) and by the Natural Sciences and Engineering Council of Canada. The authors would also like to thank Julie Falkner for her precious help in revising the final version of this paper.

References

Anbil, R., E. Gelman, B. Patty and R. Tanga. (1991). Recent Advances in Crew Pairing Optimization at American Airlines. *Interfaces*, 21:62–74.

Barnhart, C., E.L. Johnson, R. Anbil and L. Hatay. (1994). A Column Generation Technique for the Long-Haul Crew Assignment Problem. In T.A. Ciriani and R. Leachman, editors, *Mathematical Programming and Modeling Techniques in Practice*, Optimization in Industry 2, pages 7–22, John Wiley and Sons, New-York.

Byrne, J. (1988). A Preferential Bidding System for Technical Aircrew. *1988 AGIFORS Symposium Proceedings*, 28:87–99.

Chu, H. D., E. Gelman and E. Johnson. (1997). Solving Large Scale Crew Scheduling Problems. *European Journal of Operational Research*, 97:260–268.

Crainic, T.G. and J.M. Rousseau. (1987). The Column Generation Principle and the Airline Crew Scheduling Problem. *INFOR*, 25:136–151.

Desaulniers, G., J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis and D. Villeneuve. (1998). A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93, Kluwer, Norwell, MA.

Desaulniers, G., J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon and F. Soumis. (1997). Crew Pairing at Air France. *European Journal of Operational Research*, 97:245–259.

Desaulniers, G., J. Desrosiers, A. Lasry and M.M. Solomon. (1998b). Crew Pairing for a Regional Carrier. Forthcoming.

Desrochers, M. (1988). La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes. Centre de Recherche sur les Transports, Publication #470. Université de Montréal, Montréal, Canada, H3C 3J7.

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

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.M. Solomon and F. Soumis. (1995). Time Constrained Routing and Scheduling. In M.O. Ball *et al.*, editors, *Network Routing*, Handbooks in Operations Research and Management Science 8, pages 35–139, Elsevier Science, Amsterdam.

Jones, R.D. (1989). Development of an Automated Airline Crew Bid Generation System. *Interfaces*, 19:44–51.

Gamache, M. and F. Soumis. (1998). A Method for Optimally Solving the Rostering Problem. *Operations Research in Airline Industry*. (Forthcoming).

Gamache, M., F. Soumis, G. Marquis and J. Desrosiers. (1998a). A Column Generation Approach for Large Scale Aircrew Rostering Problem. *Operations Research.* Forthcoming.

Gamache, M., F. Soumis, D. Villeneuve, J. Desrosiers and E. Gélinas. (1998b). The Preferential Bidding System at Air Canada. *Transportation Science.* Forthcoming.

Glanert, W. (1984). A Timetable Approach to the Assignment of Pilots to Rotations. *1984 AGIFORS Symposium Proceedings*, 24:369–391.

Graves, G.W., R.D. McBride, I.Gershkoff, D.Anderson and D. Mahidhara. (1993). Flight Crew Scheduling. *Management Science*, 39:736–745.

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

Lasdon, L.S. (1970). *Optimization Theory for Large Systems.* Collier-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.

Moore, R., J. Evans and H. Ngo. (1978). Computerized Tailored Blocking. *1978 AGIFORS Symposium Proceedings*, 18:343–361.

Ryan, D.M. (1992). The Solution of Massive Generalized Set Partitioning Problems in Air Crew Rostering. *Journal of the Operational Research Society*, 43:459-467.

Stojković, M., F. Soumis and J. Desrosiers. (1998). The Operational Airline Crew Scheduling Problem. *Transportation Science.* Forthcoming.

Wedelin, D. (1995). An Algorithm for Large Scale 0-1 Integer Programming with Applications to Airline Crew Scheduling. *Annals of Operations Research*, 57:283–301.

9 PATH, TREE AND CYCLE LOCATION

Martine Labbé
Gilbert Laporte
and Inmaculada Rodríguez-Martín

9.1 INTRODUCTION

Traditional network location theory is concerned with the optimal location of facilities which can be considered as single points (emergency medical service stations, switching centers in communication networks, bus stops, mail boxes, etc.) However, in many real problems the facility to be located is too large to be modeled as a point. Examples of such problems include the location of pipelines and high speed train lines, the design of emergency routes, newspaper delivery routes, subway lines, etc . We will refer to this kind of facilities as *extensive facilities* or *structures*, and they may have the shape of a path, a tree, a cycle or a more general subgraph.

Applications of extensive facilities location models can be found in location analysis, routing, and network design. As an example, consider the problem of locating a path connecting two given nodes of a network with the objectives of minimizing the total path length and minimizing the total travel distance between some given demand points and the path (see Current *et al.*, 1987). The necessary information for solving this problem includes the distances given by the lengths of the shortest paths from each demand point to the closest node on the facility. That is, once the path is located it is assumed that the clients will take the shortest routes to reach the facility. So, this problem can be viewed as a location problem, a network design problem, or a routing problem.

Many location problems involving structures have been formulated as single objective problems (Traveling Salesman Problem, Shortest Path Problem, Minimum Spanning Tree Problem, etc.). The minimization of the total distance or of the total cost are the most frequent criteria. However, in the last twenty years there has been a growing interest in multiobjective formulations, to make the mathematical models

closer to reality. One possible multiobjective approach is to consider the tradeoff between the facility cost and the accessibility of demand nodes, two objectives that are generally in conflict.

The most usual ways of measuring accessibility are the *total travel distance* (or *sum of distances*) that clients must travel to reach the nearest node in the facility (an extension of the median criterion) and the *coverage* of the facility. In the context of network design, routing and location of extensive facilities, two types of coverage appear: *direct* and *indirect*. Under direct coverage a node is said to be covered only if it is in the facility. Under indirect coverage a node is said to be covered if it is within a covering distance r of any node in the facility. Of course, direct coverage is a special case of indirect coverage ($r = 0$).

In many recent papers, biobjective location problems of extensive facilities involving cost minimization and accessibility maximization (minimizing the sum of distances or maximizing coverage) have been formulated as integer programs. Generally, in multicriteria problems an optimal solution does not exist because the objectives are in conflict. The concept of optimal solution is then replaced by that of *efficient solution* (also *noninferior* or *Pareto optimal*). An efficient solution is one for which an improvement in one of the objectives will necessarily result in a loss in at least one of the other objectives.

The efficient set of multiobjective integer programs may be very large and may even grow exponentially with problem size, making it vary hard to compute. It is because of this computational burden that approximated techniques are so important. Two main solution techniques for multiobjective problems formulated as integer programs are the *constraint method* and the *utility function method* (see Steuer, 1986). The constraint method converts all but one of the objectives into constraints so as to optimize with respect to only a single objective. For a bicriterion problem involving min cost and max accessibility the two formulations would be minimizing cost subject to an accessibility constraint, or maximizing accessibility subject to a cost constraint. The utility function method consists of combining the different objectives into a unique objective function to be optimized. A very often used technique is the *weighting method*, in which every objective is multiplied by a non-negative weight. Then, the weighted objectives are summed to form a weighted-sum objective function. This methodology provides only efficient solutions that are on the convex hull of the solution set.

In this paper, we study biobjective location problems of one extensive facility on a network, under min cost and max accessibility criteria. We review the existing literature, propose integer programming formulations for the problems on undirected networks, and try to establish the complexity in all cases. We consider only those problems involving desirable facilities. In fact, we focus on three well defined objectives:

- *(o1)* min-cost: minimize the total cost of the facility;
- *(o2)* max-cover: maximize the total demand covered by the facility;
- *(o3)* min-distsum: minimize the total distance from the facility to the nodes of the network;

This leads to three possible biobjective formulations: *(min-cost, max-cover)*, *(min-cost, min-distsum)* and *(max-cover, min-distsum)*. Moreover, for $i < j$, each biobjective problem *(oi, oj)* may be approached in three differents ways:

- optimize a linear combination of the objectives: $\lambda_1(oi) + \lambda_2(oj)$,
- optimize *(oi)* subject to a restriction on *(oj)*, or
- optimize *(oj)* subject to a restriction on *(oi)*.

We do not consider in this paper the other classical criterion in location analysis: the *min-max* criterion. The reason is that we think that max-cover and min-distsum are better suited than the min-max criterion to the generic goal of maximizing accessibility. Indeed, the latter intends to minimize the damage caused to the furthest demand point and is more appropriate to emergency service location problems such as that of fire stations.

As to the facilities to be located, we restrict ourselves to the five following types of extensive facilities:

- Paths with fixed end points (P_{o-d}).
- Paths without fixed end points (P)
- Trees (T).
- Cycles with a fixed origin (C_o).
- Cycles without a fixed origin (C).

We consider them as the most representative. Path and cycle structures are common in routing applications, and the tree is commonly associated with network design problems since it is the least costly connected structure. Our work is complementary to that of Mesa and Boffey (1996) in that it is more centered on formulation and complexity issues.

The remainder of this paper is organized as follows. Section 1.2 is devoted to notations and integer programming formulations of the problems we consider.

In Sections 1.3 to 1.7 we review the literature on biobjective location of extensive facilities, involving min-cost, max-cover and min-distsum. We dedicate an entire section to each kind of facility, and begin each one by discussing the complexity of the problems. The conclusion is presented in Section 1.8.

9.2 NOTATIONS AND FORMULATIONS

An undirected *network* $G = (V, E)$ is defined by a set V of *nodes* and a set E of *edges*. The nodes are the extremities of the edges defining G. We will denote by n the number of nodes and by m the number of edges of the network. A *tree* network is a network without cycles. Let c_e denote the cost of an edge $e = (i, j) \in E$. The *distance* $d(x_1, x_2)$ between two points x_1, $x_2 \in G$ is equal to the length of a shortest path joining x_1 and x_2. For a given set $S \subseteq V$, let $\delta(S) = \{e = (i, j) \in E : i \in S$ and $j \notin S\}$ be the set of edges with an end node in S and the other one outside S.

Table 9.1 Integer programming notations

Decision variables:
$x_e = \begin{cases} 1 & \text{if edge } e \text{ is in the solution,} \\ 0 & \text{otherwise;} \end{cases}$
$y_i = \begin{cases} 1 & \text{if node } i \text{ is in the solution,} \\ 0 & \text{otherwise;} \end{cases}$
$z_i = \begin{cases} 1 & \text{if node } i \text{ is indirectly covered,} \\ 0 & \text{otherwise;} \end{cases}$
$z_{ij} = \begin{cases} 1 & \text{if } i \text{ is assigned to } j, \\ 0 & \text{otherwise.} \end{cases}$

If $S = \{i\}$, $i \in V$, then $\delta(i) = \{(i,j) : (i,j) \in E\}$ represents the set of edges that are incident to node i.

As to the demand structure, we will consider that demand arises only at the nodes. Let $w_i \geq 0$ be the demand associated with $i \in V$. Let $W = \sum_{i \in V} w_i$ be the total demand in the network. If $w_i = 1$ for all $i \in V$ we say the network is *unweighted*; otherwise the network is *weighted*.

When considering covering, for a given value $r \geq 0$ we define $S_i = \{j \in V : d(i,j) \leq r\}$ as the set of nodes that can indirectly cover node i. If $r = 0$ then we talk of direct covering, a special case of indirect covering.

To derive integer programming formulations we need the additional notations given in Table 9.1.

Given a nonnegative constant K, we can define cost, coverage and distsum either as objective functions or as constraints in the following way. When considered as objective function, the cost may be written as

$$\text{minimize} \sum_{e \in E} c_e x_e. \tag{9.1}$$

When considered as a constraint, it is expressed as

$$\sum_{e \in E} c_e x_e \leq K. \tag{9.2}$$

Similarly, for the coverage, the objective function is

$$\text{maximize} \sum_{i \in V} w_i z_i, \tag{9.3}$$

and the constraint is

$$\sum_{i \in V} w_i z_i \geq K. \tag{9.4}$$

If $K = W$, that is, if all the demand of the network must be covered, we will talk of *total* covering of the demand, while if $K < W$ we will talk of *partial* covering of the

demand. Note that in an unweighted network, a constraint on the direct covering of the facility is a constraint on the number of nodes it contains. So the most general form of covering is indirect partial weighted covering. In both cases concerning covering, the following constraint must be added to the model:

$$z_i \leq \sum_{j \in S_i} y_j, \forall i \in V. \tag{9.5}$$

It ensures that a node i is covered only if some node $j \in S_i$ is in the solution. Finally, distsum yields the objective function

$$\text{minimize} \sum_i \sum_j w_j d(j,i) z_{ji}, \tag{9.6}$$

and the constraint

$$\sum_i \sum_j w_j d(j,i) z_{ji} \leq K. \tag{9.7}$$

Here again, the following additional constraints must be included into the model:

$$\sum_{j \neq i} z_{ij} + y_i = 1, \forall i \in V, \tag{9.8}$$

$$z_{ij} \leq y_j, \forall i,j \in V. \tag{9.9}$$

Constraints (9.8) force each node i to be either in the solution or assigned to another node j. Constraints (9.9) mean that a node i can be assigned to a node j only if j is in the solution.

We now present constraints defining the five different location structures we consider.

- Paths with fixed end points o and d (P_{o-d}).

$$\sum_{e \in \delta(i)} x_e = \begin{cases} 2y_i, & i \neq o,d \\ 1, & i = o \text{ or } d \end{cases} \tag{9.10}$$

$$\sum_{e \in \delta(S)} x_e \geq y_k, \quad \forall S \subset V,\ o \text{ or } d \notin S, k \in S, \quad 2 \leq \mid S \mid \leq n-1 \tag{9.11}$$

$$y_o = 1 \tag{9.12}$$

$$y_d = 1. \tag{9.13}$$

- Paths without fixed end points (P).

$$\sum_{e \in E} x_e = \sum_{i \in V} y_i - 1 \tag{9.14}$$

$$\sum_{e\in\delta(S)} x_e \geq y_k + y_h - 1, \quad \forall S \subset V, k \in S,\ h \notin S, \qquad (9.15)$$
$$2 \leq |\ S\ | \leq n-1$$

$$\sum_{e\in\delta(i)} x_e \leq 2y_i, \forall i \in V. \qquad (9.16)$$

- Trees (T).

$$\sum_{e\in E} x_e = \sum_{i\in V} y_i - 1$$
$$\sum_{e\in\delta(S)} x_e \geq y_k + y_h - 1, \quad \forall S \subset V, k \in S,\ h \notin S,$$
$$2 \leq |\ S\ | \leq n-1.$$

- Cycles with a fixed origin (C_o).

$$\sum_{e\in\delta(i)} x_e = 2y_i, \forall i \in V \qquad (9.17)$$

$$\sum_{e\in\delta(S)} x_e \geq 2y_k, \quad \forall S \subset V,\ o \notin S, k \in S, \qquad (9.18)$$
$$2 \leq |\ S\ | \leq n-1$$

$$y_o = 1.$$

- Cycles without a fixed origin (C).

$$\sum_{e\in\delta(i)} x_e = 2y_i, \quad \forall i \in V$$
$$\sum_{e\in\delta(S)} x_e \geq 2(y_k + y_h - 1), \quad \forall S \subset V, k \in S,\ h \notin S, \qquad (9.19)$$
$$2 \leq |\ S\ | \leq n-1.$$

Constraints (9.10) imply that each internal node of the path has degree two and each extremity has degree one. Constraints (9.11) are subtour elimination constraints. Constraints (9.14) and (9.15) define a tree facility, since constraint (9.14) requests a number of edges equal to the number of nodes minus one in the solution, and constraints (9.15) imply connectivity. If we add (9.16) we define a path, since (9.16) forces the degree of each node in the facility to be at most equal to two. Constraints (9.17) impose that each node of the solution has degree two. Subtour elimination constraints (9.18) and (9.19) differ according to whether the cycle has a fixed origin or not.

Conveniently combining all these constraints, those used to define cost, covering and distsum, and those used to model the facilities, we arrive at generic integer programming formulations for all the locations problems on undirected networks considered in this paper. Note that partial edges are not allowed and that the most general type of covering, partial, weighted and indirect, is always considered. We now give some examples.

Example 1: Find in a general network a minimum cost path from a node o to a node d that indirectly covers a total demand of value at least K. This problem is formulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{e\in E} c_e x_e \\
\text{s.t.} \quad & \\
& \sum_{i\in V} w_i z_i \geq K \\
& z_i \leq \sum_{j\in S_i} y_j, && \forall i \in V \\
& \sum_{e\in\delta(i)} x_e = \begin{cases} 1, & i = o \text{ or } d \\ 2y_i, & \text{otherwise} \end{cases} \\
& \sum_{e\in\delta(S)} x_e \geq y_k, && \forall S \subset V, k \in S,\ o \text{ or } d \notin S, \\
& && 2 \leq |\, S \,| \leq n-1 \\
& y_o = 1 \\
& y_d = 1 \\
& x_e, y_i, z_i \in \{0,1\} && \forall i \in V, \forall e \in E.
\end{aligned}$$

Example 2: Find in a network a tree with cost less than or equal to K that minimizes the sum of distances to the nodes. The corresponding formulation is:

$$\begin{aligned}
\text{minimize} \quad & \sum_i \sum_j w_j d(j,i) z_{ji} \\
\text{s.t.} \quad & \\
& \sum_{e\in E} c_e x_e \leq K \\
& \sum_{j\neq i} z_{ij} + y_i = 1, && \forall i \in V \\
& z_{ij} \leq y_j, && \forall i,j \in V \\
& \sum_{e\in E} x_e = \sum_{i\in V} y_i - 1 \\
& \sum_{e\in\delta(S)} x_e \geq y_k + y_h - 1, && \forall S \subset V, k \in S,\ h \notin S, \\
& && 2 \leq |\, S \,| \leq n-1 \\
& x_e, y_i, z_{ij} \in \{0,1\}, && \forall i,j \in V, \forall e \in E.
\end{aligned}$$

Example 3: Locate a tour on a general network , with start and end node o, minimizing the sum of distances to the nodes and imposing a constraint on the facility cost. This can be formulated as:

$$\text{minimize} \quad \sum_i \sum_j w_j d(j,i) z_{ji}$$

s.t.

$$\sum_{e \in E} c_e x_e \leq K$$

$$\sum_{j \neq i} z_{ij} + y_i = 1, \qquad \forall\, i \in V$$

$$z_{ij} \leq y_j, \qquad \forall\, i, j \in V$$

$$\sum_{e \in \delta(i)} x_e = 2y_i, \qquad \forall\, i \in V$$

$$\sum_{e \in \delta(S)} x_e \geq 2y_k, \qquad \forall\, S \subset V, k \in S,\ o \notin S,\ 2 \leq |\, S \,| \leq n-1$$

$$y_o = 1$$

$$x_e, y_i, z_{ij} \in \{0,1\}, \qquad \forall\, i, j \in V, \forall\, e \in E.$$

9.3 PATHS WITH FIXED END POINTS (P_{O-D})

The problem of finding an optimal path connecting two given points of a tree is trivial since there is only one such a path. On the other hand, in general networks the nine biobjective problems considered are NP-hard, since each of them can be reduced either to the problem of finding a Hamiltonian path or a shortest Hamiltonian path from o to d (see Hakimi *et al.*, 1993) for the *min-cost s.t. min-distsum* and *min-distsum s.t. min-cost* cases). We now review the existing literature on biobjective location of path-shaped facilities on general networks, considering min-cost, max-cover and min-distsum.

9.3.1 λ_1(min-cost) + λ_2(max-cover)

Current *et al.* (1985) introduce the biobjective problem of locating a path from a predetermined starting node to a predefined terminus node in order to minimize its length and maximize the covered demand. They name the problem the Maximum Covering Shortest Path problem (MCSP) when indirect covering of the demand is allowed, and the Maximum Population Shortest Path problem (MPSP) when the demand has to be directly covered, and present integer programming formulation for both cases. They use the weighted method to generate efficient solutions, and solve the problem by relaxing the integrability and subtour elimination constraints. If noninteger solutions appear, then branch and bound is applied. When subtours occur, the necessary subtour breaking constraints are added and the problem is solved again. A sample MPSP problem with 15 nodes and 34 edges is given.

9.3.2 Min-cost s.t. max-cover

The total weighted indirected problem has been studied in Current *et al.* (1984) and Current *et al.* (1994). It was first introduced by Current *et al.* (1984) with the name of Shortest Covering Path Problem (SCPP) and presented as a synthesis of the Location Set Covering Problem (LSCP) and the Shortest Path Problem (SPP). In Current *et al.* (1984), an integer programming formulation of the problem is given and two particular instances, with 20 nodes and 190 edges and 15 nodes and 33 edges respectively, are

solved relaxing the integrality and subtour elimination constraints. Branch and bound is applied when noninteger solutions appear and subtour elimination constraints are added when necessary.

Current *et al.* (1994) show that this problem is NP-hard since, if the instance where each node is only covered by itself is considered, then it reduces to the problem of finding the shortest Hamiltonian Path from o to d. The integer programming formulation they use, in a directed network, is as follows:

$$\text{minimize} \quad \sum_i \sum_j c_{ij} x_{ij}$$

s.t.

$$\sum_i x_{ij} - \sum_k x_{jk} = \begin{cases} -1, & j = o \\ 0, & j \in V, j \neq o, d \\ 1, & j = d \end{cases} \tag{9.20}$$

$$\sum_i \sum_{j \in S_k} x_{ij} \geq 1, \qquad \forall k \in V \tag{9.21}$$

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq \mid Q \mid -1 \qquad \begin{array}{l} \forall Q \subset V \\ 2 \leq \mid Q \mid \leq n-2 \end{array} \tag{9.22}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V. \tag{9.23}$$

A heuristic and an exact method are given. The heuristic is based upon a Lagrangian relaxation of the problem in which constraint set (9.21) concerning covering restrictions are dualized. This Lagrangian relaxation has the structure of a shortest path problem with additional subtour elimination constraints. A subgradient optimization method is used to find multipliers, discarding those that lead to negative cycles in the shortest path subproblems. The exact method is based upon a branch and bound procedure which uses the bounds generated by the heuristic.

Both the heuristic and the branch and bound method are tested on 160 randomly generated networks with up to 80 nodes and 8000 edges. The heuristic solves optimally up to the 84% of all those problems. The average gap between the heuristic solution and the Lagrangian relaxation solution is smaller than 1%, with a maximum of 17%. Moreover, most problems are solved in less than 1000 seconds, although the largest ones require almost one hour.

9.3.3 (Min-cost, min-distsum)

Current *et al.* (1987) provide an algorithm (MONET) that generates efficient solutions to the bicriterion problem called the Median Shortest Path Problem (MSPP). MONET is an enumeration algorithm based on the k-shortest path problem. Its complexity is $O(kn^2+n^3+k^2)$ on complete networks and $O(km \log n+mn \log_{2+m/n} n+k^2)$ on sparse networks. Compared with the weighted method, MONET seems to be a good and quick algorithm to generate efficient solutions for medium size instances of the MSPP. Moreover, it identifies a series of efficient solutions that the weighted method does not find because they are not on the convex hull of the solution set.

9.4 PATHS WITHOUT FIXED END POINTS (P)

The number of paths on a tree is $O(n^2)$ and therefore all problems consisting of locating a path on a tree are polynomial, unless there is a nonpolynomial objective function to be evaluated for each path. For example, the nine problems we consider can be solved in $O(n^3)$ operations. However, on general networks all these problems are NP-hard, since each of them can be reduced either to the Hamiltonian path or to the shortest Hamiltonian path problem (the proofs for *min-cost s.t. min_distsum* and *min-distsum s.t. min-cost* are provided in Hakimi *et al.*, 1993).

9.4.1 Min-distsum s.t. min-cost

On tree networks, Minieka and Patel (1983) consider the problem of finding the core or path median of length l, when partial edges are allowed in the solution. They did not succeed in fully characterizing the solution, and the problem of developing an efficient algorithm remained unsolved until Minieka (1985) proposed a solution method whose complexity depends on the number of leaves of the tree. In fact, the algorithm consists, for every pair of leaves i, j with $d(i,j) \geq l$, in moving a path of length l along the unique path from i to j until a certain condition becomes true, choosing then the best among all those candidates paths. Since the number of leaves of a tree is $O(n)$ time and the best path for a given pair of leaves can be found in linear time, Minieka's algorithm has a complexity of $O(n^3)$.

On general networks, Richey (1990) studies the problem of finding the path which minimizes the sum of distances to the vertices, provided its length is not larger than a given value. The solution may contain partial arcs. Richey shows that the problem is NP-hard in general, since it can be reduced to the Hamiltonian Path problem, but when the constraint on the length is an inequality it can be solved for a class of networks (series-parallel networks) using pseudo-polynomial algorithms. If partial edges are not allowed, the problem is also NP-hard (see Hakimi *et al.*, 1993).

9.5 TREES

The constrained formulations of the biobjective problems *(min-cost, max-cover)* and *(min-cost, min-distsum)* are NP-hard even on tree networks. The proof for *(min-cost, min-distsum)* is given by Hakimi *et al.* (1993). Here is the proof for *(min-cost, max-cover)*.

Consider the decision version of the partial weighted direct *min-cost s.t. max-cover* and *max-cover s.t. min-cost* problems on a tree network: given a tree G and positive numbers K and l_o, is there a subtree T such that the cost of T is less than or equal to l_o and the demand covered by T is at least K? To prove that this problem is NP-complete we reduce the Partition Problem to it. Given a sequence $a_1, a_2, ..., a_n$ of positive integers with $\sum_{i=1}^{n} a_i = S$, the Partition Problem asks whether there is a subset $I \subset \{1, ..., n\}$ with $\sum\{a_i : i \in I\} = S/2$. We build a star tree as indicated in Figure 9.1, where the demand of node i, $i = 0, 1, ..., n$, is denoted by w_i, $w_0 = 0$, the length of each edge $(0, i)$ is $w_i > 0$ and $W = \sum_{i=1}^{n} w_i$. Let $l_o = W/2$ and $K = W/2$. Then it is easy to see that a solution to our problem exists for the network of Figure 9.1 if and only if there is a subset of nodes with total demand equal to $W/2$, that is, if and

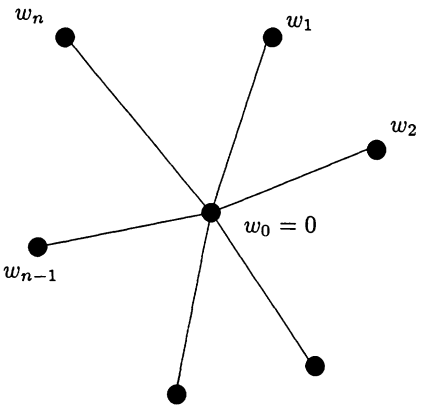


Figure 9.1 A star tree with n nodes

only if the Partition Problem has a solution. As the Partition Problem is NP-hard , so are our problems (*min-cost s.t. max-cover* and *max-cover s.t. min-cost*).

On the other hand, the three formulations involving *max-cover* and *min-distsum* are trivial on tree networks (they have the whole tree as solution), and polynomial on general networks (they reduce to the Minimum Spanning Tree (MST) problem).

9.5.1 λ_1*(min-cost)* + λ_2*(max-cover)*

The problem of minimizing a linear combination of the min-cost and max-cover objectives has been studied by several authors, but it has only been treated on tree networks. It seems that the problem was first introduced by Hutson and ReVelle (1989), who proposed an integer programming bicriterion formulation for the case with direct covering of the demand and use the weighted method to approximate the efficient set. The optimal integer solution to the relaxed weighted problem (found using branch and bound if necessary) is an efficient solution for the biobjective problem. They also considered the case where some edges are required to be in the solution, that is, the case of 'adding to an existing structure'. Hutson and ReVelle (1993) used the same approach to solve the problem with indirect covering of the demand.

Tamir (1992) shows that the weighted method used by Hutson and ReVelle (1989) can be implemented in $O(n^2)$ time using dynamic programming. Church and Current (1993) propose a new integer formulation and another $O(n^2)$ time algorithm called BUCS (Building a Covering Subtree), that finds the subtree minimizing a linear combination of min-cost and max-cover with given weights. BUCS is a constructive algorithm which builds the best subtree from a given edge. Then the best of those subtrees is chosen as the solution. BUCS can also be applied to indirect covering of

the demand when partial arcs are allowed, and to the problem of adding to an existing structure.

Finally, Kim *et al.* (1996) give $O(n)$ and $O(n \log^2 n)$ time dynamic programming algorithms for solving the linear combination formulation with direct and indirect covering of the demand, respectively. The algorithms can be seen as special cases of a method developed to solve a more general problem. The authors take the sum of the facility length and of the demand covered as the function to minimize, but the definitions and recursive equations in which their method is based can be easily modified to handle with the minimization of a linear combination instead of this simple sum.

9.5.2 Min-cost s.t. max-cover

On tree networks, the total direct covering problem is trivial since the solution is the whole tree. Hutson and ReVelle (1993) propose an integer programming formulation of the problem with total indirect covering. The solution method they give extends reduction techniques used in Toregas and ReVelle (1972) and Toregas and ReVelle (1973) to solve the Location Set Covering Problem on tree networks. Kim *et al.* (1990) consider a formulation with total indirect covering which differs from the one in Hutson and ReVelle (1993) since there are now different covering distances r_i for each $i \in V$, and partial edges are allowed in the solution. The authors present an $O(n)$ time optimal solution method based on the algorithm for the Set Covering Problem. Maffioli (1991) studies the problem with partial unweighted direct covering, that is, the problem of finding the minimum cost subtree with at least K nodes. He gives an $O(K^2 n)$ dynamic programming exact algorithm.

On general networks, Kim *et al.* (1989) formulate the Subtree r-Cover Problem, which consists of finding the minimal length subtree of a network that indirectly covers all the nodes (total weighted indirected covering). A node i is said to be covered by the facility if it is within a given distance r_i from it, and partial arcs are allowed in the solution. The authors show that this problem is NP-hard, and give a pseudo-polynomial algorithm that exploits the structure of the networks and is polynomial on some of them (e.g., in cactus networks, i.e., in networks in which each block or maximum nonseparable subgraph is a single edge or a cycle). As a particular case of the partial unweighted direct formulation, Ehrgott *et al.* (1995) study the problem of finding the minimum length subtree with exactly K nodes in a general unweighted network. They show this problem is NP-hard and present different heuristic approaches based on spanning tree and shortest path methods and on the exact algorithm for solving the problem in polynomial time on tree networks.

9.5.3 Max-cover s.t. min-cost

The problem of maximizing the demand covered subject to the constraint that the cost of the tree is less than or equal to a given value l has only been treated on tree networks. It is trivial when l is at least equal to the total cost of the tree. Tamir (1993) shows that the $O(n^2/\epsilon)$ $(1+\epsilon)$-approximation method he proposes to solve the min-distsum s.t. min-cost problem can be easily modified to solve the problem with indirect covering of the demand. Another related article is that of Church and Current (1993), who apply branch and bound to solve the problem of minimizing a

linear combination of objectives *min-cost* and *max-cover*, with direct covering of the demand, subject to a budget (cost) constraint.

9.5.4 λ_1(min-cost) + λ_2(min-distsum)

On tree networks, Kim *et al.* (1993) and Kim *et al.* (1991) present two different methods with $O(n)$ complexity to solve the Median Subtree Location Problem , consisting of minimizing the sum of the facility cost and sum of distances. The method proposed in Kim *et al.* (1991) is in some way similar to Goldman's algorithm for the median of a tree (see Goldman, 1971). On the other hand, the dynamic programming method proposed in Kim *et al.* (1993) can be easily adapted to solve the problem of minimizing a convex combination of the total length and the sum of distances.

On general networks, Kim *et al.* (1991) optimize the facility cost plus the sum of distances, and name the problem the Median Subtree Problem. They show the problem is NP-hard and give a solution method that uses the concept of block of a network. The method reduces the original network N to a network N' such that N' contains a solution subtree T having a nonempty intersection with each block of N'. It is then shown that the problem can be decomposed by solving, in linear time, a Median Subtree problem on each spanning tree within each block of N'. This leads to a pseudo-polynomial algorithm, although it is polynomial when the underlying network is a cactus.

9.5.5 Min-distsum s.t. min-cost

On tree networks, Tamir (1993) gives an algorithm that generates a $(1+\epsilon)$-approximation in $O(n^2/\epsilon)$ time. The algorithm is based on a standard application of the interval partitioning method suggested in Sahni (1977) and makes use of dynamic programming techniques. Minieka (1985) formulates the problem with an equality constraint on the cost of the facility. Note that in this case the solution may contain partial arcs. He gives a polynomial time solution method which, starting from the set M of medians of the tree, builds up a solution subtree T. This method is based on the fact that, for any feasible length l, there is a minimum distsum subtree of size l, T_l, satisfying $T_l \cap M \neq \emptyset$. On general networks, Richey (1990) gives a pseudo-polynomial algorithm that solves the problem when partial edges are allowed.

9.6 CYCLES WITH AN ORIGIN (C_O)

All nine problems we consider under this heading are NP-hard since they all reduce either to the Hamiltonian cycle problem or to the Traveling Salesman Problem (TSP).

9.6.1 Min-cost s.t. max-cover

Gendreau *et al.* (1995) study the problem of determining the minimum length Hamiltonian cycle in a subset of V, such that all nodes in a set W are within a specified distance r from the cycle. One of the vertices is fixed as depot or start and end point of the tour. This is, using our notation, a minimize cost s.t. maximize cover formulation with total weighted indirect covering of the demand. An integer linear formulation is given and exact and heuristic solution methods are proposed.

The heuristic method combines the GENIUS heuristic for the TSP (see Gendreau *et al.*, 1992) with the PRIMAL1 set covering heuristic of Balas and Ho (1980). The solution is used as an initial upper bound in an exact branch and cut algorithm.

Both the heuristic and the exact method are tested on randomly generated networks with up to 600 nodes, of which up to 75 must be in the tour. Computational results show that the average heuristic gap is smaller than 1% and the average gap in the root of the branch and cut is smaller than 0.5%.

9.7 CYCLES WITHOUT A FIXED ORIGIN (C)

The nine bicriteria problems we consider, involving locating a cycle without a fixed origin on a general network, are NP-hard, since each of them reduces to the Hamiltonian cycle problem or to the TSP.

Current and Schilling (1994b) introduce two bicriteria problems consisting of locating a tour in a graph visiting only n_o vertices. The first objective in both problems is to minimize the total tour length. The second objective is to maximize the total demand within some prespecified maximal travel distance from a the tour in one problem (the Maximal Covering Tour Problem (MCTP)), and to minimize the sum of distances to the vertices not in the facility in the other one (the Median Tour Problem (MTP)). So, in the MCTP the objectives are *(minimize cost, maximize cover)*, with indirect covering of the demand, and in the MTP are *(minimize cost, minimize distsum)*. The authors propose integer programming formulations for both problems, and show that the MCTP can be viewed as a particular case of the MTP. Moreover, it is proved that both problems are NP-hard, since they reduce to the TSP if $n_o = n$. A heuristic procedure for approximating the efficient frontier of the solution set of the MTP is presented.

9.7.1 Min-cost s.t. max-cover

Current and Schilling (1989) have studied the problem with total indirect covering of the demand. This problem which they call the Covering Salesman Problem (CSP) is NP-hard, since it is a generalization of the TSP. An integer programming formulation is given and a heuristic solution procedure is presented. The heuristic, named COVTOUR, is based upon solution procedures for the set covering problem (SCP) and the TSP. In fact, it consists in solving first a SCP and then solving a TSP on a complete graph with node set given by the solution of the SCP. The solution given by COVTOUR is always feasible, but it may not be optimal and it is not possible to measure its quality. Both the given integer programming formulation and the heuristic can be modified to force certain nodes to be in the solution, and can thus be used to solve the C_o problem.

Note that the CSP is in some way related with the Generalized Traveling Salesman Problem (GTSP) (Fischetti *et al.*, 1995 and 1995b), a variant of the TSP in which nodes are partitioned into clusters and the salesman has to visit at least one node of each cluster. The sets S_i of nodes that can cover node i in the CSP can be identified with the clusters of the GTSP.

Tables 9.2 to 9.6 summarize the literature review we present in Sections 1.3 to 1.7. Their aim is to show which problems have been treated in some way and which of them remain unstudied, thus highlighting the existing gaps.

Table 9.2 Paths with fixed end points on G

objective function	*(min-cost, max-cover)*	*(min-cost, min-distsum)*	*(max-cover min-distsum)*
$\lambda_1(oi) + \lambda_2(oj)$	Current *et al.* (1985)	Current *et al.* (1987)	
(oi) s.t. (oj)	Current *et al.* (1984), Current *et al.* (1994)		
(oj) s.t. (oi)			

Table 9.3 Paths without fixed end points

objective function	*(min-cost, max-cover)*	*(min-cost, min-distsum)*	*(max-cover min-distsum)*
$\lambda_1(oi) + \lambda_2(oj)$			
(oi) s.t. (oj)			
(oj) s.t. (oi)		Minieka (1985), Minieka and Patel (1983), Richey (1990)	

Table 9.4 Trees

objective function	*(min-cost, max-cover)*	*(min-cost, min-distsum)*	*(max-cover min-distsum)*
$\lambda_1(oi) + \lambda_2(oj)$	Church and Current (1993), Hutson and ReVelle (1989), Hutson and ReVelle (1993), Kim *et al.* (1993), Tamir (1992)	Kim *et al.* (1993) Kim *et al.* (1991)	
(oi) s.t. (oj)	Hutson and ReVelle (1993), Kim *et al.* (1990), Maffioli (1991), Kim *et al.* (1989), Ehrgott *et al.* (1995)		
(oj) s.t. (oi)	Tamir (1993)	Tamir (1993), Minieka (1985), Richey (1990)	

Table 9.5 Cycles with an origin

objective function	*(min-cost, max-cover)*	*(min-cost, min-distsum)*	*(max-cover min-distsum)*
$\lambda_1(oi) + \lambda_2(oj)$			
(oi) s.t. (oj)	Gendreau *et al.* (1997)		
(oj) s.t. (oi)			

Table 9.6 Cycles without a fixed origin

objective function	*(min-cost, max-cover)*	*(min-cost, min-distsum)*	*(max-cover min-distsum)*
$\lambda_1(oi) + \lambda_2(oj)$			
(oi) s.t. (oj)	Current and Schilling (1989)		
(oj) s.t. (oi)			

9.8 CONCLUSION

Problems involving the location of extensive facilities in networks arise in a variety of practical contexts. In comparison with point location problems, they have received rather limited attention in recent years. Some progress has been made towards their solution by heuristic or exact techniques, but by and large, the field remains relatively unstructured and unexplored. We hope our proposed classification will help organize the field and will identify areas of relevance where further research is warranted.

Acknowledgments

This work was partially funded by the EC Human Capital Mobility Program under contract # CHRX-CT-93-0087, and by the Canadian Natural Sciences and Engineering Research Council under grant OGP0039682. This support is gratefully acknowledged. Thanks are also due to an anonymous referee for his valuable comments.

References

Balas, E. and A. Ho. (1980). Set Covering Algorithms using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study. *Mathematical Programming*, 12:37–60.

Church, R. and J. Current. (1993). Maximal Covering Tree Problems. *Naval Research Logistics*, 40:129–142.

Current, J. (1988). The Design of a Hierarchical Transportation Network with Transshipment Facilities. *Transportation Science*, 22:270–277.

Current, J., J.L. Cohon and C.S. ReVelle. (1984). The Shortest Covering Path Problem: An Application of Locational Constraints to Network Design. *Journal of Regional Science*, 24:161–183.

Current, J., H. Pirkul and E. Rolland. (1994). Efficient Algorithms for Solving the Shortest Covering Path Problem. *Transportation Science*, 28:317–327.

Current, J., C.S. ReVelle and J.L. Cohon. (1985). The Maximum Covering/Shortest Path Problem: A Multiobjective Network Design and Routing Formulation. *European Journal of Operational Research*, 21:189–199.

Current, J., C.S. ReVelle and J.L. Cohon. (1987). The Median Shortest Path Problem: A Multiobjective Approach to Analyze Cost vs. Accessibility in the Design of Transportation Networks. *Transportation Science*, 21:188–197.

Current, J. and D.A. Schilling. (1989). The Covering Salesman Problem. *Transportation Science*, 23:208–213.

Current, J. and D.A. Schilling. (1994b). The Median Tour and Maximal Covering Tour Problems: Formulation and Heuristics. *European Journal of Operational Research*, 73:114–126.

Ehrgott, M., J. Fretag, H.W. Hamacher and F. Maffioli. (1995). Heuristics for the K-Cardinality Tree Subgraph Problems. Working paper.

Fischetti, M., J.J. Salazar Gonzalez and P. Toth. (1995). The Symmetric Generalized Travelling Salesman Polytope. *Networks*, 26:113–123.

Fischetti, M., J.J. Salazar Gonzalez and P. Toth. (1997). A Branch-and-Cut Algorithm for the Symmetric Generalized Travelling Salesman Problem. *Operations Research*, 45:378–394.

Gendreau, M., A. Hertz and G. Laporte. (1992). New Insertion and Postoptimization Procedures for the Traveling Salesman Problem. *Operations Research*, 40:1086–1094.

Gendreau, M., G. Laporte and F. Semet. (1997). The Covering Tour Problem. *Operations Research*, 45:568–576.

Goldman, A.J. (1971). Optimal Center Location in Simple Networks. *Transportation Science*, 5:406–409.

Hakimi, S.L., E.F. Schmeichel and M. Labbe. (1993). On Locating Path- or Tree-Shaped Facilities on Networks. *Networks*, 23:543–555.

Hutson, V.A. and C.S. ReVelle. (1989). Maximal Direct Covering Tree Problems. *Transportation Science*, 23:188–299.

Hutson, V.A. and C. ReVelle. (1993). Indirect Covering Tree Problems on Spanning Tree Networks. *European Journal of Operational Research*, 65:20–32.

Kim, T.U., T.J. Lowe, A. Tamir and J.E. Ward. (1996). On the Location of a Tree-Shaped Facility. *Networks*, 28:167–175.

Kim, T.U., T.J. Lowe and J.E. Ward. (1991). Locating a Median Subtree on a Network. *INFOR*, 29:153–166.

Kim, T.U., T.J. Lowe, J.E. Ward and R.L. Francis. (1989). A Minimum Length Covering Subgraph of a Network. *Annals of Operations Research*, 18:245–260.

Kim, T.U., T.J. Lowe, J.E. Ward and R.L. Francis. (1990). A minimum-Length covering Subtree of a Tree. *Naval Research Logistics*, 37:309–326.

Maffioli, F. (1991). Finding the Best Subtree of a Tree. Politecnico di Milano. Technical report 91.041.

Mesa, J.A. and T.B. Boffey. (1996). A Review of Extensive Facility Location in Networks. *European Journal of Operational Research*, 95:592–603.

Minieka, E. (1985). The Optimal Location of a Path or Tree in a Tree Network. *Networks*, 15:309–321.

Minieka, E. and N.H. Patel. (1983). On Finding the Core of a Tree with a Specified Length. *Journal of Algorithms*, 4:345–352.

Richey, M.B. (1990). Optimal Location of a Path or Tree on a Network with Cycles. *Networks*, 20:391-407.

Sahni, S. (1977). General Techniques for Combinatorial Approximations. *Operations Research*, 25:920–936.

Steuer, R.E. (1986). *Multiple Criteria Optimization: Theory, Computation, and Application.* Wiley series in probability and mathematical statistics-applied. Wiley.

Tamir, A. (1992). On the Complexity of Some Classes of Location Problems. *Transportation Science*, 26:352–354.

Tamir, A. (1993). Fully Polynomial Approximation Schemes for Locating a Tree-Shaped Facility: A Generalization of the Knapsack Problem. Working paper. Tel-Aviv University.

Toregas, C. and C. ReVelle. (1972). Optimal Location under Time or Distance Constraints. *Papers of the Regional Science Association*, 28:133–143.

Toregas, C. and C. ReVelle. (1973). Binary Logic Solutions to a Class of Location Problems. *Geographical Analysis*, 5:145-155.

10 PARALLEL METAHEURISTICS

Teodor Gabriel Crainic
Michel Toulouse

10.1 INTRODUCTION

Heuristics have been, and continue to be, an essential component of the methodological approaches used to address combinatorial optimization formulations, in general, and transportation applications, in particular. In the last ten to fifteen years, metaheuristics have profoundly changed the way we solve these problems and have significantly contributed to efficiently address complex, hard problem settings (see, for example, Crainic and Laporte, 1997, or Golden *et al.*, 1998).

Parallel versions of metaheuristic methods are proposed with increasing frequency. The usual goals for parallel computing are also invoked here: reasonable computing times with more realistically formulated and sized problem instances. A second benefit is increasingly being acknowledged: in appropriate settings (e.g., multithread strategies), parallel metaheuristics may be much more robust than sequential versions relative to differences in problem types and characteristics and the corresponding parameter calibration issues.

In fact, the number of researchers and studies dedicated to parallel metaheuristics has reached the level where surveys, taxonomies and syntheses are proposed: Jog, Suh and Gucht (1991), Lin, Punch and Goodman (1994), Gordon and Whitley (1993), Gordon (1994), Cantù-Paz (1995), Hoffmeister (1991), Sargent (1988), Greening (1989, 1990), Azencott (1992a), Lee and Lee (1992a), Verhoeven and Aarts (1995), Voß(1993), Pardalos *et al.* (1995), Laursen (1996), Crainic, Toulouse and Gendreau (1997), and Holmqvist, Migdalas and Pardalos (1997), among others. However, most of these works study parallel metaheuristics that belong to one methodology only. Even when several metaheuristic approaches are considered in the same paper, surveys are presented along exclusive methodological lines without the benefit of a global view.

The goal of this paper is to start to bridge this gap. Our approach is based on the observation that despite a large number of efficient implementations for particular problems, relatively few fundamental ideas have been used to design parallel strategies for metaheuristics. It is thus our belief that by examining the commonalities among parallel implementations across the field of metaheuristics, insights may be gained, trends may be discovered, and research challenges may be identified.

The paper is organized as follows. Section 10.2 presents a few statistics illustrating the increasing interest in parallel metaheuristics and presents the criteria we use to survey the literature. Sections 10.3, 10.4 and 10.5 are dedicated to the survey and discussion of issues related to the parallelization of metaheuristic search methods: tabu search, simulated annealing, and genetic methods, in particular. Section 10.6 summarizes the conclusions derived from the survey and points to research directions and challenges. Throughout the paper, particular attention is paid to applications of parallel metaheuristics to transportation problems.

10.2 PARALLEL METAHEURISTICS

Compared to exact search methods, such as branch-and-bound, metaheuristic methods do not aim to systematically explore the whole solution space. Instead, they attempt to examine only parts thereof where, according to certain criteria, one believes good solutions can be found. Avoiding local optima traps and cycling, as well as making reasonably sure that the search has not overlooked promising regions, are the usual objectives of designing good metaheuristics.

Metaheuristics for optimization problems may be described summarily as a "walk through neighbourhoods", a search trajectory through the solution domain of the problem at hand. These are iterative procedures that *move* from a given solution to another solution in its *neighbourhood*. Thus, at each iteration, one evaluates moves towards solutions in the neighbourhood of the current solution, or in a suitably selected subset. According to various criteria (objective value, feasibility, probability, etc.), a number of good (though not necessarily improving) moves are selected and implemented. Tabu search and simulated annealing methods usually implement one move at each iteration, while evolutionary methods may generate several new individuals at each generation-iteration. Moves that are evaluated at each iteration may belong to only one type (e.g., add an element to the solution) or may belong to several quite different types (e.g., evaluate both add and drop moves). Moves may marginally modify the solution or drastically inflect the search trajectory. The first case is often referred to as the *local search* phase. The diversification phase of tabu search or the application of mutation operators in an evolutionary process are typical examples of the second alternative; this last case may also be described as a change in the "active" neighbourhood.

The metaheuristics most oftenly used and parallelized are either *evolutionary approaches – genetic algorithms* (**GA**: Holland, 1975; Goldberg, 1989; Whitley, 1994; Fogel, 1994; Michalewicz, 1992; etc.), *simulating annealing* methods (**SA**: Metropolis *et al.*, 1953; Kirkpatrick, Gelatt and Vecchi, 1983; Laarhoven and Aarts, 1987; Aarts and Korst, 1989; etc.), or *tabu search* procedures (**TS**: Glover 1986, 1989, 1990, 1996; Glover and Laguna, 1993; Osman and Kelly, 1996, etc.). Other methods, such as GRASP (Feo and Resende, 1995) and *ant colony systems* (Dorigo, 1992; Colorni,

GA		SA		TS	
< 1990	≥ 1990	< 1990	≥ 1990	< 1990	≥ 1990
22	88	33	42	1	35

Table 10.1 Number of Publications of Parallel Metaheuristics

Dorigo and Mannienzzo, 1991, 1992) have also been proposed recently and we include them in the paper as well.

The continuous and rapid increase in the number of parallel metaheuristic developments and applications is illustrated in Table 10.1. The table synthesizes the reports, theses, and articles referenced in this paper according to the type of approach. The contributions are further separated according to the year of publication: before and after 1990. These two columns represent approximately the same time span, since most contributions before 1990 have been reported in the latter half of the 80's. The list is not exhaustive, but it yields a number of observations that will be more thoroughly addressed in this paper:

- Genetic-type methods have received the most attention. This is not surprising given the fundamental "parallel" nature of evolutionary methods.
- Many of the applications of parallel metaheuristics belong to various computer science fields: VLSI design – especially circuit placement and routing – image processing, artificial intelligence, etc. Regarding combinatorial optimization, most applications address the *travelling salesman problem (TSP)* and the *quadratic assignment problem (QAP).*
- Parallel tabu search methods strongly emerged after 1990. Furthermore, while addressing the same general application areas previously indicated, it is parallel TS procedures that have been more often dedicated (20 out of the 36) to problems relevant to transportation science: TSP, *vehicle routing problems (VRP)*, and *network design.*
- A number of other techniques also emerged after 1990, such as parallel GRASP (Feo, Resende and Smith, 1994; Pardalos *et al.*, 1995), *ant colony systems* (Colorni, Dorigo and Mannienzzo, 1991, 1992; Dorigo, Maniezzo, and Colorni, 1996), and hybrid methods that combine elements of several methodological approaches (e.g., GA and SA).

In order to bring some order to our survey of parallel metaheuristics, independent of particular methodological characteristics, we classify them according to the level of impact the parallelization strategy has on the algorithmic design of the metaheuristic. This classification, which has been used to study exact search methods (see, for

example, Gendron and Crainic, 1994), allows us to bring together the efforts deployed according to different metaheuristic methodologies. It classifies parallel methods according to the following three strategies:

Type 1: parallelization of operations within an iteration of the solution method. This strategy, also called low-level parallelism, is rather straightforward and aims solely to speed up computations, without any attempt at a better exploration (except when the same total wall clock required by the sequential method is allowed in the parallel process) or higher quality solutions. It yields the same solution method as the sequential one, only faster.

Type 2: decomposition of the problem domain or search space. This strategy is based on the principle that computing power may be dedicated to solve a host of smaller problems, out of which an improved overall solution may be extracted or constructed. The search trajectory of the resulting parallel approach is different, however, from that of the corresponding sequential method.

Type 3: multisearch threads with various degrees of synchronization and cooperation. This strategy attempts a more thorough exploration of the solution space by initiating several search threads that simultaneously proceed through the domain.

When describing parallel metaheuristics using this taxonomy, we often refer to the performance of the methods. It is beyond the scope of this paper to address the issue of how to measure the performances of metaheuristics. The interested reader may consult Barr and Hickman (1993), Barr *et al.* (1995), and Toulouse, Crainic and Gendreau (1996a). It may be useful, however, to recall that the most widely used performance measures attempt to compute some kind of acceleration. This *speed-up* is defined (relative to the definition of time) as the ratio of sequential time to parallel time to solve a particular problem on a particular machine. In notation form, let T_s be the total solution time of a sequential procedure (normally, the best one available) for a given problem and computer, and T_p the execution time of a parallel implementation with p processors. The speed-up is then $S(p) = \frac{T_s}{T_p}$. For various specifications of the serial and parallel procedures, one obtains the measures presented in Barr and Hickman (1993).

Two other performance measures are often used. *Work* (W), $W = T_p \times p$, computes the total effort required by the parallel procedure. This measure is particularly relevant when the optimum is not necessarily reached (or proven), and the quality of the solution attained by two procedures will be compared. *Scalability* measures the capability of a parallel procedure to display the same performance level when the problem size and the number of available processors grow proportionally.

10.3 TYPE 1: LOW-LEVEL PARALLELIZATION

Type 1 parallelizations correspond to the concurrent computation of some or all operations making up an iteration of the corresponding sequential method. Evaluation of individuals and moves are typical operations subject to Type 1 parallelization.

As mentioned by several authors (Crainic, Toulouse and Gendreau, 1997; Rousset-Ragot and Dreyfus, 1992; Trienekens de Bruin, 1992; Gendron and Crainic, 1994),

Type 1 parallelization strategies aim directly to reduce the execution time of a given solution method. In fact, when the same number of iterations are allowed to both sequential and parallel versions of the method and the same operations are performed at each iteration (e.g., the same set of candidate moves is evaluated and the same selection criterion is used), the parallel implementation follows the same exploration path through the problem domain as the serial method and yields the same solution. As a result, standard parallel performance measures apply straightforwardly.

Some implementations modify the parallel version to take advantage of the extra computing power available without altering the basic search method. For example, one may evaluate all the moves in the neighbourhood of the current solution instead of only those in a given subset. The resulting search patterns of the serial and parallel implementations are then different in most cases. Yet, because the fundamental algorithmic design is not altered, these approaches still qualify as low-level parallelism. Speed-up measures may thus be applied directly, even though the notion of *solution quality* (i.e., does the method find a better solution?) may qualify the acceleration measures.

10.3.1 Genetic Algorithms

In the literature, low-level parallelization of genetic algorithms is called *global* parallelization (Gordon, 1994; Cantú-Paz, 1995). Historically, this strategy was presented in what probably constitutes the first attempt to construct a parallel metaheuristic: the parallel genetic algorithms of Grefenstette (1981). In this approach, only one population is considered, and there are generally no restrictions on the selection and crossover operators used. The focus is on the parallelization of the fitness evaluation of individuals and, to a lower extent, of the crossover and mutation operators one applies to obtain a new population generation.

Parallel fitness evaluation is obtained via a synchronous master-slave approach, where the master allocates individuals to slave processors to be evaluated. There is no communication among slave processes; only the master communicates with the slaves at the beginning and end of the evaluation process. To obtain a new generation, the population is divided among processors and the usual genetic operators (selection, crossover, mutation, etc.) are applied to each partition in parallel. Note that on shared-memory computers, there are generally no explicit masters. To begin the evolution, the population is divided and each subpopulation is assigned to a specific process. Then, a synchronization phase between generations ensures that the relevant information is available to all.

Parallel genetic algorithms of this type are described by Fogarty and Huang (1990; see also Cui and Fogarty, 1992). They used a transputer network for a pole-balancing application and reported that 1) transputer network topology is not important when computing time is more significant than communication time, and 2) communication overhead increases very fast with the number of processors.

Abramson and Abela (1992) and Abramson, Mills and Perkins (1993) implemented similar strategies for school time-tabling problems on a shared-memory computer (an Encore Multimax with 16 processors) and for train timetable construction on a distributed-memory machine (a Fujitsu AP1000 with 128 processors), respectively. Almost linear speed-ups were reported for both applications using up to 16 processors,

while performances deteriorated rapidly with additional processors on the distributed-memory computer.

Hauser and Männer (1994) used three different shared-memory computers – a 7-processor NERV, a 16-processor SparcServer and a 20-processor KSR1 – to implement a global parallel genetic algorithm (PGA) for a cell placement problem. Relatively good performances were reported only on the NERV multiprocessor (speed-up of 5 using 6 processors, compared to a speed-up of 2 with 8 processors on the SparcServer and 3 with 20 threads on the KSR1) due to the very low communication overhead of the machine. Recently, Chen, Nakao and Fang (1996) applied a master-slave strategy to image restoration problems using a cluster of IBM RISC6000 (from 1 to 20 machines). The best speed-up (10) was reported when 15 processors were used.

A different, asynchronous perspective is mentioned by Hoffmeister (1991) when only the fitness evaluation is to be performed in parallel. Slave processes are no longer synchronized, which may help avoid processor idle time. It does not seem, however, that this research direction has been explored any further.

It thus appears that global parallelization strategies are limited. Indeed, while fitness evaluation is a time-consuming operation and is a likely candidate for efficient parallelization, the case is less obvious concerning the genetic operators: their application is generally simpler and the extra cost of partitioning the population among processes and the resulting communications may offset the gains of parallel treatment, especially on distributed-memory architectures.

10.3.2 Simulated Annealing

A major issue in designing parallel simulated annealing (PSA) methods is whether the parallelization strategy affects the statistical convergence properties of the simulated annealing method. The issue has been central to all development efforts in the field and a serious debate on whether specific parallel strategies do indeed affect the convergence properties, or whether this impact is really significant has been animating the PSA community. In the present review, we consider that PSA methods use low-level parallelism if they do not, or are assumed not to, affect the convergence of the method and do not modify the search trajectory of the sequential SA.

An iteration of the sequential simulated annealing algorithm consists of four main steps: select a move, evaluate the cost function, accept/reject the move and replace the current solution if the move is accepted. Two main approaches are used to design Type 1 parallel simulated annealing methods: *single-trial* parallelism where only one move is examined at a time, and *multiple-trial* strategies where several moves are evaluated simultaneously (Aarts and Korst, 1989; Roussel-Ragot and Dreyfus, 1992).

In single-trial methods, functional parallelism is applied to some of the computations that make up an SA iteration: one process implements the sequential SA and decomposes computing intensive tasks into smaller problems which are assigned to the other processors. Kravitz and Rutenbar (1986), who studied the cell placement problem (an important component of VLSI layout design) on multiprocessor computers, applied such a strategy and decomposed the move-evaluation step. They reported moderate speed-ups (2 on 3 processors), which they did not expect to improve with more processors. See also Bannister and Gerla (1989) for a single-trial implementation.

The single-trial strategy clearly does not alter the algorithmic design or the convergence properties of the sequential SA, which explains its initial appeal. It is, however, strongly dependent on the application and implementation. Furthermore, it tends not to speed up computations significantly. Hence, research has moved toward strategies, such as multiple-trial approaches, where a higher degree of parallelism may be attained.

In multiple-trial parallelizations, each processor executes the four steps of a simulated annealing iteration. This does not raise particular problems for the first three steps since these tasks are essentially independent for different potential moves. The replacement step is, however, a fundamentally sequential operation: only one modification operation may be executed at any given moment on the current solution. Consequently, executing this step in parallel may cause significant problems. Consider, for example, two processes that start from the same solution. Each selects a move, evaluates it and determines that it may be accepted. Each evaluation, performed concurrently with the other, assumes that the initial configuration is not changed. Then the two accepted moves are implemented, sequentially. But, as soon as one of the two moves is implemented, the evaluation of the other one is no longer valid and its implementation may lead to a configuration and cost function evaluation significantly different from what was computed by the individual process.

The concurrent execution of the replacement steps may thus yield erroneous evaluations of the cost function, resulting in a search trajectory different from the sequential one or even in violation of the statistical convergence hypotheses of the SA method. How to avoid this problem or to control the amplitude of the error introduced by the multiple-trial parallelization strategies constitutes a central issue in the literature on parallel simulated annealing procedures.

According to our classification, multiple-trial implementations that belong to Type 1 parallelism correspond to methods where the parallel trials start from the same solution, have access to moves in the entire neighbourhood, and always result in an *error-free* cost function evaluation. The error-free evaluation is achieved because the replacement of the current solution is either restricted to a single accepted move (the others being discarded) or to moves that do not interact with each other. The latter approach is called the *serializable subset* concept, where a *serializable subset* corresponds to a set of moves that always produces the same result when applied to the current state of the system, independent of the order in which they are applied (a trivial serializable subset contains only rejected moves).

Witte, Chamberlain and Franklin (1990, 1991) studied assignment problems and proposed a move-evaluation strategy that may be compared to the probing approach in tabu search methods. Three processors were used for each move evaluation. One performs the evaluation. The others two proceed as if the evaluation result is already known: one starts from the current configuration (move rejected), the other from the configuration resulting from an eventual move acceptance. Thus, when the decision of the first processor is known, one process is killed, while for the other one some work has already been accomplished. The number of levels in the tree can grow until the pool of processors has been exhausted. While imaginative, there is much unnecessary work done (in addition to the usual communication and dispatch overhead) and performances have not been very impressive. In fact, the authors reported a speed-up of 2.6 when 8 processors were used. Nabhan and Zomaya (1995) improved

somewhat the performances of this approach by modifying the information that is exchanged among processes. Only the moves actually performed to reach the new solutions are communicated, rather than the whole solutions as in the original method by Witte, Chamberlain and Franklin. Nabhan and Zomaya reported modest performance improvements for small problems (20-city TSP). The speed-up improved when the problem size was increased (almost 3.5 for 6 processors, for a 100-city TSP).

Roussel-Ragot and Dreyfus (1990, 1992; see also Roussel-Ragot, Kouicem and Dreyfus, 1990) studied the cell placement problem and experimented (for 25, 49 and 81 cells) on networks of transputers. Two network sizes were used: 3 and 6 "slaves", plus one master that monitored the annealing schedule, chose the accepted move, and updated the memories of each processor. In the high temperature mode of the annealing schedule, all processors were synchronized after the accept/reject step and one accepted move was chosen randomly to replace the current solution. In low temperature mode, processors computed asynchronously until one processor accepted a move. Then, all processors were synchronized and updated with the new current solution. Thus, there was no error in the evaluation of the cost function. Performances were not very good. Borut and Silc (1994) also proposed an error-free PSA based on synchronization and random selection of the "new" global solution, but did not report on their implementation.

Kravitz and Rutenbar (1987; see also Rutenbar and Kravitz, 1986) studied the same problem, and implemented an error-free multiple-trial strategy where only non-interacting moves were executed according to a simple serializable subset idea. This, however, resulted in very poor performance in high temperature mode because of the considerable amount of move cancellations.

We include these error-free multiple-trial implementations in the Type 1 parallelism category based on the assumption that these strategies, where at each iteration all processes have access to all potential moves, do not alter the search trajectory or the convergence properties of the sequential SA (multiple-trial strategies where evaluation errors occur, such as in Casotto, Romeo and Sangiovanni-Vincentelli, 1987, are classified as Type 2). The issue is not, however, altogether settled. Thus, for example, in Banerjee, Jones and Sargent (1990), the authors state that the error-free implementation of Kravitz and Rutenbar (1987) has the same convergence properties as the sequential version, while in Roussel-Ragot and Dreyfus (1990) it is stated that this same implementation "is quite different from the sequential one at high temperature since the acceptance ratio is not the same". The debate is not, however, of great practical significance since Type 1 error-free multiple-trial strategies appear to display rather poor efficiency performances. In fact, to accept only one move implies frequent synchronizations and a lot of wasted work. Even if the number of lost moves decreases when the low temperature phase is reached, performances are significantly affected. As for the serializable subset concept, identifying a good serialization for a given subset of moves may prove to be as complex and computing-intensive as the resolution of the initial problem.

10.3.3 Tabu Search

Low-level parallelism has also been applied to tabu search methods (Crainic, Toulouse and Gendreau, 1997). These implementations correspond to a master thread which executes a sequential tabu procedure, but the possible moves in the neighbourhood of

the current solution are evaluated in parallel by slave processors at each iteration. The slave processes may evaluate only the moves in the set they receive from the master process, or may probe beyond each move in the set. The master process receives and processes the information resulting from the slave operations, and then selects and implements the next move. The master also gathers all the information generated during the tabu exploration, updates the memories, decides when to activate different search strategies – diversification, for example – and when to stop the search.

Chakrapani and Skorin-Kapov (1992, 1993, 1995) studied quadratic assignment problems and proposed a Type 1 strategy which performs the evaluation of moves in the neighbourhood in parallel. The implementation was developed for the Connection Machine CM-2, a massively parallel SIMD machine, and was designed to take advantage of the special features of the computer. The authors reported that for problems already described in the literature, they either attained or improved the best known solutions, in a significantly smaller number of iterations. Furthermore, they were also able to determine good suboptimal solutions to larger problems in reasonable time. A similar strategy has also been proposed for the TSP (Chakrapani and Skorin-Kapov, 1993a).

Taillard (1991; see also, Taillard, 1993a) also addressed QAP problems and reported experimental results on a ring of 10 transputers. The set of possible moves was partitioned and each set was assigned to a different processor. Each processor then evaluated the pairwise interchange moves and identified the best one. There was no specific master processor. Instead, once a processor identified its best move, it broadcasted it to all other processors, which then performed all the normal tasks of the master: selecting and implementing the move, making the necessary adjustments and updates, partitioning the neighbourhood, etc. No implementation details were given. Load balancing through partition of the neighbourhood was acknowledged as critical, but no indication was given on how it was performed. On several problem sets proposed in the literature (essentially, the same set used by Chakrapani and Skorin-Kapov, 1993) with problem instances up to size 100, Taillard reported very good solutions, improving the best known values of many of the problems tested and obtaining suboptimal solutions (conjectured but not proven to be optimal) for problems up to size 64.

Crainic, Toulouse and Gendreau (1995) studied and compared several synchronous parallelization strategies for a tabu search procedure for the location-allocation problem with balancing requirements. Experiments were conducted on a heterogeneous network of SUNSparc workstations. With respect to Type 1 parallelization approaches, two variants were implemented: 1) slaves evaluated candidate moves only; 2) *probing*: slaves also performed a few local search iterations (best results obtained with 2). The second variant performed marginally better. However, both variants were outperformed by multithread (Type 3) implementations. Another Type 1, master-slave parallelization scheme is presented by Garcia, Potvin and Rousseau (1994) for the VRP with time-window constraints.

The success of Type 1 parallelizations for tabu search procedures appears more significant than for genetic approaches or simulated annealing methods. Yet, even for TS it depends heavily upon the problem characteristics. Thus, for problems where the neighbourhood is very large but the time to evaluate and perform a given move is relatively small, quadratic assignment and vehicle routing problems being

classical examples, Type 1 parallelizations are very effective and near-linear speed-ups have been reported (Chakrapani and Skorin-Kapov, 1993). Performances are less interesting when the time required by one serial iteration is relatively important compared to the total solution time, resulting in executions with only a few hundred moves compared to the tens of thousands required by a typical VRP tabu search, for example. Parallel methods which attempt a more thorough exploration of the solution space than the sequential version appear to be superior (Crainic, Toulouse and Gendreau, 1995, 1995a).

10.4 TYPE 2: PARALLELIZATION BY DOMAIN DECOMPOSITION

Type 2 parallelization methods for metaheuristics are generally based on the decomposition of the decision variable vector into disjoint subsets. The heuristic procedure is then applied to each subset, the variables outside the subset being considered fixed. Such strategies are generally implemented in some sort of a master-slave framework:

- A *master* process serially devises the initial partition. During the search, it regularly modifies the partition. Modifications may be performed at intervals that are either prefixed or determined during the execution, or when restarting the method.
- *Slave* processes concurrently and independently explore their assigned partitions. The search may proceed exclusively within the partition, the other variables being considered fixed and unaffected by the moves which are performed, or they may have access to the entire solution vector.
- When slaves have access to the entire neighbourhood, the master must combine the partial solutions obtained from each subset into a complete solution to the problem.

Note that a decomposition based on the partition of the decision variable vector may leave large portions of the solution space unexplored. Therefore, in many applications, the partition is repeated to create different segments of the decision variables vector and the search is restarted.

No genetic implementations and only one GRASP procedure have been found to follow domain decomposition ideas. Therefore, this section addresses mainly simulated annealing and tabu search methods.

10.4.1 Genetic Algorithms

Consider an individual binary representation of length n (binary representations are used for simplicity of exposition considerations only). By fixing any set of $n - p$ bits (genes, in GA vocabulary) and stating that the remaining p bits are *free* and may take any value (this corresponds to replacing their current values with a *don't care* symbol in GA terms), one creates a *schema* (Holland, 1975; Goldberg, 1989) which represents all the possible individuals obtained by assigning specific values to the p genes. The set of all such schemata represents an implicit partition of the solution space. A parallel GA implementation based on Type 2 principles could then be built based on these schemata, according to two mechanisms. The first corresponds to sequential GA operators applied exclusively to the p free genes of each schema. The

second iteratively adapts the partition rules and selects the individuals that will yield the schemata.

We have found no parallel GA implementation that follows these mechanisms, or any other idea of individual representation partition. All the PGA we encountered in the literature that incorporate partition principles take advantage of the "natural" parallelism inherent to evolutionary techniques and work on population partitions.

We consider, however, that parallel genetic algorithms based on several subpopulations belong to Type 3 strategies. We justify this classification with the observation that even when processes are initiated with different sets of individuals, which may form a partition of some initial population, genetic operators tend to homogenize the subpopulations. In fact, because the genetic operators are not restricted to a subset of genes, one may potentially obtain any individual of the population, starting from any of the individuals of any particular subpopulation. Hence, the individual GA process is not restricted to given subpopulations; it has access to the whole population. Therefore, these methods do not work on true partitions of the variable vector or of the solution space.

10.4.2 GRASP

We found one Type 2 parallel implementation of the GRASP method in Feo, Resende and Smith (1994). Here, large instances of the maximum independent set problem were solved by decomposing them into many smaller problems distributed among processors and each solved by a sequential GRASP. Almost linear speed-ups were reported on an Alliant FX/80 computer with 8 processors.

10.4.3 Simulated Annealing

Most multiple-trial parallelization strategies for simulated annealing methods are based on the partition of the data structure. Two main approaches may be identified. The first, which corresponds to the so-called *error-free algorithms*, attempts to avoid evaluation errors and the loss of convergence properties via a strict partition of the move set and the restoration of the global state (evaluation of the objective function) following each synchronization. In the second approach, errors are admitted and the emphasis is on controlling the amount of erroneous computation.

Note that for PSA algorithms executed on shared-memory systems, it is not costly to regularly update the global state of the current solution such that errors do not accumulate during the computation. In distributed systems, however, each processor has its own copy of the data, including the "current" solution, and such global updates are costly in communication time. Thus, significant errors may accumulate locally. Trade-offs must be achieved between the frequency of the global state update and the level of error in the PSA computation, and many studies have addressed the issue of how much error may be tolerated. (e.g., Banerjee, Jones and Sargent, 1990; Durand, 1989; Hong and McMillin, 1992; etc.).

Felten, Karlin and Otto (1985) studied the TSP and experimented with 64 cities using up to 64 processors on a hypercube computer. An initial tour configuration was randomly generated and partitioned into P subsets of adjacent cities. Each subset was then assigned to one processor. Each processor performed local swaps on adjacent cities for a number of iterations, followed by a synchronization phase where cities

were rotated among processors. Parallel moves did not interact with each other due to the spatial decomposition of the decision variables, and each synchronization phase ensured the integrity of the global state. Hence, there was no error. Almost linear speed-ups were reported. A similar approach was used for the TSP by Allwright and Carpenter (1989), and by Devadas and Newton (1986) for the topological optimization of the multiple level array logic problem (on a Sequent Balance 8000 computer). Both used a similar spatial decomposition scheme and reported similar behaviours and speed-ups.

Bongiovani, Crescenzi and Guerra (1995) studied the shape detection problem and proposed an error-free PSA, strongly inspired by the particular characteristics of the problem. In this problem, given a shape represented in terms of parameters and an image, the occurrences of the shape in the image must be identified by estimating good values for the parameters. The authors formulated the problem as an optimization model aimed at minimizing the difference between the given image and the image obtained by the set of parameters, and experimented on a Encore Multimax computer with 16 processors. Their PSA used functional decomposition (single-trial, Type 1 parallelism) at high temperatures, and a multiple-trial strategy at low temperatures. An error-free computation was ensured by restricting the moves to the same sub-image.

Many of the studies proposing Type 2 parallelization approaches have been directed towards the VLSI design - cell placement problem, including Casotto, Romeo and Sangiovanni-Vincentelli (1986, 1987) who experimented (for 4, 6, 30, 101 and 122 cells) on up to 8 processors of the Sequent Balance 8000 shared-memory parallel computer. In this approach, the set of cells was partitioned into P subsets and each subset was assigned to a different processor. Moves (swaping the position of two cells) were generally restricted among cells in the same subset and were performed asynchronously. When a move involved cells on two different subsets, the processor that initiated the move synchronized with the other processor for the cell exchange. However, although the set of cells was partitioned and moves were generally restricted to the same subset, the cost function involved variables other than the cells themselves. Therefore, parallel executions of the last step of the annealing iteration interacted with each other and created errors in the cost evaluation. The authors addressed the issue of identifying partitions that reduced the errors and discussed the impact of clustering and dynamic partition modification schemes on the number of erroneous moves (between cells in different subsets). The authors also noted that, since the computation was asynchronous, the parallel search trajectory deviated from the sequential one, and that the error went to zero as the temperature decreased. This second result, often mentioned in PSA literature, is easily explained by the fact that as temperatures decrease, fewer and fewer moves are accepted. Finally, the authors indicated that the errors introduced at high temperatures did not seem to affect the performances of the parallel method and report a speed-up close to 1. Similar observations were made (Casotto, Romeo and Sangiovanni-Vincentelli, 1987a) when massive parallelism was applied to the same cell placement problem using the Connection Machine. The authors noted, however, that the error increased with the number of moves executed in parallel.

One of the reasons for partitioning variables among processors is to prevent the same variable from being involved simultaneously in more than one move. This same

goal can be achieved, however, by *locking* the variables involved in a move. In the PSA context, this mechanism allows, at any given time, only one processor, which owns the lock, to update a given variable. Any processor that attempts to execute a move involving a locked variable can either wait for it to become available or attempt a different move.

Locks have been used by Jayaraman and Rutenbar (1987) for the floorplanning problem on a distributed-memory hypercube. Processors were clustered and each cluster was assumed to implement an error-free parallel move evaluation. Locks reduced the movement of data among clusters. The access rights were modified during computation such that each cluster could have access to the whole solution space. Each cluster performed several moves before the global state was restored through synchronization. Such a "lazy" restoration strategy resulted in a significant amount of evaluation error. A maximum speed-up of 7.5 on 16 processors was reported. A similar mechanism was used in Darema, Kirkpatrick and Norton (1987) for the cell placement problem. There was an overhead for imposing locks, however. The authors reported an overhead of 1% and 11% respectively for the two parallelization methods tested, and indicated that the overhead increased with the number of processors for both methods.

Banerjee, Jones and Sargent (1990) studied the cell placement problem, and experimented (for 32, 64, 183, 286, 469 and 800 cells) with 4 to 16 processors of a hypercube distributed-memory system. Cells were partitioned into subsets and moves interacted with each other. The issue of tolerance to error was explicitly addressed and an "integrated error control" scheme was proposed. Similar work is described in Jones and Banerjee (1987), Sargent (1988), and Rose *et al.* (1986) for the cell placement problem, as well as in Brouwer and Banerjee (1988) for the channel routing problem. Jayewardena (1990) applied the same decomposition ideas to the image reconstruction problem on a network of transputers.

Jayaraman and Darema (1988) specifically addressed the issue of error tolerance for parallel simulated annealing methods. The set of cells of a placement problem was partitioned into subsets such that two processors would not try to move the same cell. Each processor had a local view of the global state. The authors studied the impact of the frequency of synchronization and the number of processors on error tolerance. As expected, they found that the error increased as the frequency of synchronizations decreased and as the number of processors increased. The combined error created by synchronization and parallelism affected the convergence of the simulated annealing algorithm, of which parallelism emerged as the most important factor.

A similar study can be found in Durand (1989). Vertices of the graph partitioning problem were affected to different processors of a shared-memory system. The author tested different levels of synchronization to measure the impact on the errors generated. The tests showed that PSA was tolerant to temporary error (frequent synchronizations) but at the cost of computation efficiency. On the other hand, when synchronization was relaxed, the error increased with the number of processors. An even more detailed analysis of the PSA error tolerance can be found in Hong and McMillin (1992). As for Greening and Darema (1989), they studied the partition shape and showed that it affects the cell mobility and the cost function errors. Different partition shapes can increase or reduce the frequency of synchronizations required to keep the same quality of convergence.

Greening (1990, 1990a) also studied the impact of evaluation errors on the convergence properties of the parallel SA method and analyzed the respective effects of *instantaneous* and *accumulated* errors. The author also addressed the issue of the error-free multiple-trial implementations based on spatial decomposition (e.g., Felten, Karlin and Otto, 1985; Allwright and Carpenter, 1989; Devadas and Newton, 1986). Indeed, although several authors (Banerjee, Jones and Sargent, 1990; Casotto, Romeo and Sangiovanni-Vincentelli, 1987; Darema, Kirkpatrick and Norton, 1987) considered these implementations to behave like the sequential SA, for Greening these procedures were unable to mimic the statistical properties of the sequential SA method. According to this author, the spatial decomposition "changes the pattern of the state space exploration, and thus changes the expected solution quality and execution time". The authors of the present paper are not aware if this debate about the statistical behaviour of some multiple-trial implementations is still open in the PSA community. However, since there is no doubt that the search trajectory is indeed modified by a Type 2 parallelization, we tend to sit with Greening on this issue.

A somewhat different type of PSA implementation was reported by Banerjee and Jones (1986). The authors addressed the cell placement problem on a hypercube with the same number of processors as cells to place. Moves were evaluated in parallel by having processors on the same dimension interact. Accepted moves were then broadcasted. This method was very communication intensive: some 58% of the time processors communicated or waited for data. Furthermore, the approach appears difficult to scale. For the same problem, Rose, Snelgrove and Vranesic (1990) proposed a hybrid where the high temperature mode of the SA method was replaced with an heuristic that assigned cells to the chip. The low temperature phase was then executed in parallel.

Boissin and Lutton (1993) developed a parallel SA that can be implemented on a massively parallel computer. Experiments were performed using two combinatorial optimization problems: the QAP, and the unconstrained minimization of a 0-1 quadratic function. Interesting performances were reported on a 16K Connection Machine. Wong and Fiebrich (1987) reported another massive parallel SA implementation.

10.4.4 Tabu Search

Typical tabu search implementations of Type 2 parallel strategies partition the vector of decision variables directly and perform a search on each subset. This approach was part of the preliminary experimentation in the study of synchronous parallel methods for tabu search undertaken by Crainic, Toulouse and Gendreau (1995). It performed poorly, mainly because the nature of the class of problems used for testing – multicommodity location with balancing requirements – which requires a significant computation effort to evaluate and implement moves, and thus results in a limited number of moves for the entire search.

More success was been achieved by Type 2 parallel methods proposed for problems for which numerous iterations may be performed in a relatively short time, and thus restarting the method with several different partitions does not require unreasonable computational efforts. TSP and VRP formulations belong to this class of applications.

Fiechter (1994) studied the TSP. For the intensification phase of the method he proposed, each process optimized a specific slice of the tour. At the end of the

intensification phase, processes synchronized to recombine the tour and modify (shift part of the tour to a predetermined neighbouring process) the partition. To diversify, each process determined among its subset of cities a candidate list of most promising moves. The processes then synchronized to exchange these lists, so that all would build the same final candidate list and apply the moves. The algorithm was implemented on a network of transputers arranged in a ring structure. The author reported near-optimal solutions to large problems (500, 3000 and 10000 vertices), and almost linear speed-ups (less so for the 10000-vertex problems).

Taillard (1993) studied parallel TS methods for vehicle routing problems. His parallelization strategies were simulated using four processors on a Silicon Graphics 4D/35 workstation for a classical set of problems ranging from 50 to 199 cities (Christofides, Mingozzi and Toth, 1979), and on a non-Euclidean 385-city problem based on actual distances and population figures from a Swiss region. The first strategy applied to Euclidean problems with uniformly distributed cities. It decomposed the domain into polar regions, to which vehicles were allocated. Each subproblem was then solved by an independent tabu search. All processors were synchronized after a certain number of iterations (according to the total number of iterations already performed), and the partition was modified: tours, undelivered cities and empty vehicles were exchanged between adjacent processors. Load balancing problems seemed to impair this approach. The second strategy was aimed at non-Euclidean problems, or problems where cities were not uniformly distributed. The main difference between the two strategies appeared in the partitioning method (the space was partitioned based on the arborescence build by the shortest paths from the depot to all cities), and in the information that is exchanged (the best solution only).

Porto and Ribeiro (1995, 1996; see also Porto, Kitajima and Ribeiro, 1996) studied the task scheduling problem for heterogeneous systems and proposed several synchronous PTS procedures where a master process determined and modified partitions, synchronized slaves and communicated the best solutions. Several communication schemes were evaluated on an IBM 9076 Scalable POWER 1, using PVM, for varying number of processors and problem sizes. Interesting results were reported, even for strategies involving a high level of communications. Almost linear speed-ups were observed, better performances being notted for larger problem instances.

It is worth noting that currently, one of the most successful sequential metaheuristics for the VRP is a tabu search method called *adaptive memory* (Rochat and Taillard, 1995; Glover, 1996). In this approach, cities are initially separated into several subsets, and routes are built using a construction heuristic. These initial routes are then stored in a structure called an *adaptive memory*. A combination procedure then builds a complete solution using the routes in the memory, and the solution is further improved using a tabu search method. The routes of "good" solutions are then deposited into the same memory, which thus adapts to reflect the current state of knowledge of the procedure. The process then starts again with a new solution built from the routes stored in the adaptive memory. The method stops when a prespecified number of calls to the adaptive memory have been performed. This approach clearly implements the principles of Type 2 decomposition using a serial procedure – see also the interesting developments in the vocabulary building strategies for tabu search proposed by Glover (1996). Adaptive memory principles are now successfully applied to other problem classes and are opening interesting research perspectives (Glover,

1996). However, interestingly enough, most parallel applications of this approach are now found in multithread strategies (Type 3).

10.5 TYPE 3: MULTIPLE SEARCH STRATEGIES

Parallel methods, which consist of several concurrent searches of the solution space, are classified as Type 3 parallelization strategies. These methods may or may not use the same metaheuristic approach. They may start from the same or different initial solution and may communicate during the search or only at the end to identify the best overall solution. Communications may be performed synchronously or asynchronously, and may be event-driven or executed at predetermined or dynamically decided moments. These strategies belong to the *p-control* class according to the taxonomy proposed by Crainic, Toulouse and Gendreau (1997), and are identified as *multiple-walk* by Verhoeven and Aarts (1995).

Multiple search strategies may be further divided into two classes: *independent* and *cooperative* approaches. In the former, several searches are initiated and the best result is selected at the end from among the results of the individual processes. Independent search approaches are thus equivalent to an accelerated restarting strategy. To define a cooperative multiple search, a number of important parameters must be further decided upon (Toulouse, Crainic and Gendreau, 1996):

- the connection topology defining how processes are linked;
- the method of communication (broadcast, propagation, using a central memory, etc.)
- the processes between which information exchanges are to be performed;
- the type of communication – synchronous or asynchronous;
- the time when to exchange information;
- the information to exchange.

The setting of these parameters may have a significant impact on the behaviour of the parallel procedure and the search performance. This is opening up new research avenues and we will return to this issue in the conclusion of this paper.

10.5.1 Genetic Algorithms

We find in the multiple search category the most widely used form of parallel genetic algorithms. Two main parallelization strategies are found: coarse and fine-grained parallelism.

For *coarse-grained* parallel genetic algorithms, parallelism is obtained by replicating the approach used in global PGA on several processors across subpopulations. Usually, the same genetic algorithm is used for all populations, although some researchers (e.g., Schlierkamp-Voosen and Mühlenbein, 1994; Herdy, 1992) have pondered the possibility of using different strategies for different populations. A *migration* operator is added to the list of genetic operators. It allows the exchange of information among subpopulations. The *emigration policy* determines how individuals are selected: best-fit, randomly, randomly among better than average individuals, etc. The *migration*

rate specifies if migration involves only one individual at a time, or a pool of selected individuals. The *migration interval* determines when migration may take place, and it is usually defined in terms of a number of generations. The *immigration policy* indicates how individuals are replaced in the receiving subpopulation: worst ones dropped, random selection, random selection among the less fit individuals, etc. The information exchanges are further determined by the neighbourhood structure. In the *island* model individuals may migrate towards any other subpopulation, while in the *stepping-stone* model only direct neighbours are reachable. (In this latter approach, neighbourhoods overlap to allow for propagation of individuals). In general, the connection structure of the processors of the parallel machine on which experiments are carried out determines the connection topology defining how subpopulations are linked. Migration may be performed either synchronously or asynchronously.

Tanase (1987) proposed a stepping-stone model on 4-D hypercube topology applied to a class of Walsh polynomials (the "Tanase functions"). Migration (fixed rate, random selection from the best) occurred at regular, 5-generation intervals along one dimension of the hypercube; each subsequent migration took place along a different dimension. Two approaches were studied: same search strategy applied to all subpopulations and different strategies. Results were reported for 2, 4, 8, 16, 32 and 64 processors; they indicated that: 1) parallel methods with different parameter values seem to perform well without knowing the best parameter settings; 2) parallel versions find results of similar quality to serial GA with near linear speed-ups.

Tanase (1989) also performed an extensive study of migration parameters. Two main conclusions emerged. First, extreme policies, such as migrating many individuals too often or few individual very infrequently, degrade performances. This result confirmed the pioneering findings of Grosso (1985). Second, even without migration, the parallel GA outperforms the serial procedure. The migration operator is further studied by Seredynski (1994) in the context of dynamic mapping and load balancing problems. Parameters such as the frequency of migrations, the number and selection strategy of the individuals for each migration, and the strategy for incorporating individuals in a new population were analyzed and tested.

Belding (1995) extended Tanase's work using the Royal Road functions (Holland, 1993) with KSR1 and KSR2 parallel computer systems. Migration destinations were chosen randomly. The author reported that, in general, the global optimum was found more often when migration was used than otherwise. Surprisingly, faster convergence was observed when the migration rate was high. The results of the study conducted by Munetomo, Takai, and Sato (1993) further emphasized the impact on the performances of distributed PGA of the frequency of exchanges among subpopulations. Similar results were reported by Kommu and Pomeranz (1992) who showed that with proper communication the size of each subpopulation could be substantially reduced, thus improving the computational efficiency, without decreasing the quality of the final solution.

Pettey, Leuze and Grefenstette (1987) implemented a model where the best individual found at each generation in each subpopulation is broadcasted to all other subpopulations. The model was applied to functions F1 to F5 from De Jong's Test Suite (De Jong, 1975) using 1, 2, 4, 8 and 16 Intel iPSC/2 processors, for population sizes of 50, 100, 200, 400 and 800, respectively. Pettey and Leuze (1989) further explored this implementation.

Cohoon *et al.* (1987) developed an island model to study the role of migration on the level of evolutionary change and evolution. The authors observed that new solutions were often found shortly after new individuals were mixed into the population. They also observed that the exchange topology was not important for the performance of the parallel GA, as long as communication lines were dense and short. They experimented with a placement problem and the parallel GA with migration outperformed the corresponding parallel GA without migration and the serial GA. Later, Cohoon, Martin and Richards (1991a,b) and Cohoon *et al.* (1991) extended the results to various problems in VLSI and floorplan design.

Mühlenbein, Schomisch and Born (1991a,b) described a stepping-stone model applied to difficult function-optimization problems. Experimentations were reported for 4 and 8 transputers. The basic strategy is quite classic: the same search strategy for all processes, a variable migration interval and the exchange of the best individuals. The authors introduced, however, a very important addition: when a subpopulation did not improve for a certain number of generations, a local hill-climbing heuristic was applied. This approach obtained very good results. It was noted, however, that including the local optimizing procedure made it difficult to exactly determine the relative importance for the quality of the method of the heuristic and the parallel GA. The success of the approach has convinced, however, most of the PGA community and it is now widely used.

Schnecke and Vornberger (1996) proposed a PGA for the VLSI placement and routing problem using 12 transputers of the Parsytec parallel computer. Each processor executed a different GA strategy. There was no migration among the subpopulations; rather the paper emphasized the self-adaptation of the search strategies. Thus, at fixed intervals, the different GAs were ranked and the search strategies were adjusted according to the "best" one by importing some of its characteristics (mutation rate, crossover rate, etc). The paper refers to several other papers where self-adaptation strategies were developed and tested. In particular, Lis (1996) applied self-adaptation to the mutation rate. The author implemented a farming model where a master processor managed the overall population and sent the same set of best individuals to slave processors, each of which had a different mutation probability. Periodically, according to the mutation rate of the process that obtained the best results, the mutation rates of all slave processors were shifted one level up or down and populations were recreated by the master processor using the best individuals of the slave processors. Starkweather, Whitley and Mathias (1991; see also Whitley and Starkweather, 1990a,b) also suggested that an adaptive mutation rate might help achieve better results for PGA. The same authors also noted that if partial solutions could be combined to form better solutions, then a parallel GA would probably outperform a serial GA. If on the other hand, the recombination generally yielded a less-fit individual, the serial GA would outperform the PGA.

Davis, Liu and Elias (1994) applied a parallel GA to the VLSI circuit synthesis problem using 20 SPARC workstations. Although the computer architecture offered distributed memory, the implementation made use of the Linda language, which creates a virtual-shared memory programming environment for the cluster of workstations. Three parallelizations were compared: global (low-level), coarse-grained interacting and non-interacting PGA. Best solutions were obtained by the coarse-grained interacting implementation, followed by the global approach, with the non-interacting

PGA obtaining solutions that were worse than those obtained with the sequential version. The three parallel versions used approximately the same computation time for a fixed number of generations.

Shonkwiler (1993) made use of independent searches. The same genetic algorithm was run on each processor (using different seeds for the random number generator), and there was no migration among the populations other than gathering the end results. The paper reports on the experimentations conducted on a cluster of SunSparc stations (2, 4 and 6 stations) on the password, the Sandia Mountain and the Inverse Fractal problems. (See also Ghannadian, Shonkwiler, and Alford, 1993 and Miller and Shonkwiler, 1992). Superlinear speed-ups were claimed and a probabilistic model was proposed to explain the performance of the independent search method. Similar models have also been proposed for independent searches using different metaheuristics, namely by Taillard (1993) and Battiti and Tecchiolli (1992).

An interesting hierarchical coarse-grained PGA was reported by Branke, Kohlmorgen, and Schmeck (1995): each subpopulation was assigned to a cluster of processors, classical Type 1 parallelism being implemented on each cluster. Migration was permitted once the average quality of the individuals in subpopulations reached a certain level. Hence, since not all subpopulations evolved at the same speed, the ones that had attained the specified level lent processors to the other subpopulations.

Levine (1996) proposed a stepping-stone PGA for the set partitioning problem, a difficult combinatorial formulation often encountered in routing and scheduling applications, most predominantly in air crew scheduling. Each subpopulation comprised 100 individuals and migration was initiated every 1000 iterations. The best individual in a subpopulation was selected to migrate and it replaced an individual selected in the receiving population, by a probabilistic tournament. Each subpopulation exchanged individuals with its four neighbours (subpopulations were arranged in a two-dimensional toroidal mesh), alternating among them each migration. Experiments were conducted using from 1 to 128 processors (by steps of powers of 2) on an IMP SP with 128 nodes, each of which was an IBM RS/6000 Model 370. Several medium problems (few rows, less than 3000 columns) and a few medium to large problems (8000 to 45000 columns and 400 to 823 rows) were used for testing. Optimal or near optimal solutions were obtained for the medium problems. No integer solution was found on problem instances with many constraints. It was observed that increasing the number of subpopulations was beneficial to the quality fo the solution.

Fine-grained parallelizations are asynchronous methods that divide the population into a large number of subsets. Ideally, subsets are of cardinality one, each individual being assigned to a processor. Each subset is then connected to several others in its neighbourhood (the *deme*) and the genetic operators are applied through asynchronous exchanges between individuals in the same deme only. The deme may be of fixed topology (consisting of individuals residing in particular processors) obtained by a random walk or by applying a given Hamming distance, etc. An individual is then selected from the deme and a crossover operation is performed with the original individual. Selection may be based on various criteria: local fitness distribution, tournament, local ranking, etc. Neighbourhoods generally overlap to allow propagation of individuals or individual characteristics and mutations. Such methods are also identified as *massively parallel*, because ideally a processor is assigned to each individual, and as *cellular models*, because a parallel GA with a fixed topology deme and a rel-

ative fitness policy may be shown to be a finite cellular automata with probabilistic rewrite rules and an alphabet equal to the set of strings in the search space (Whitley, 1993).

Manderick and Spiessens (1989) experimented with a fine-grained GA on functions F1 to F5, using a Symbolics Lisp Machine. The individuals of the population were placed on a planar grid and a small, fixed-size neighbourhood was assigned to each individual. This neighbourhood usually consisted of an individual's neighbours on the grid. The application of selection and crossover operators was restricted to the neighbourhoods, contrary to a sequential implementation. In the implementation of the algorithm on the DAP parallel computer (1024 processors) described in Spiessens and Manderick (1990, 1991), all the individuals were updated in parallel, if the population was not larger than the number of processors.

Mühlenbein, Gorges-Schleuter and Krämer (1987, 1988) applied a similar approach to the TSP using an Encore computer, which is a shared-memory system. The size of the neighbourhood was fixed to 4 neighbours, a local selection strategy was used, a hill-climbing heuristic was applied to individuals, and GA operators were applied to the resulting local optimum. Mühlenbein (1989) applied the same basic strategy to solve the QAP, but divided the individuals into two equal subsets placed on two rings. Each individual thus had two neighbours on each ring. A 64-processor transputer network was used. Later, Mühlenbein (1991a) applied the same fine-grained PGA approach to the graph partitioning problem. In the latter paper, the author suggested that different hill-climbing strategies could be applied concurrently, but did not implement this strategy. See also von Laszewski and Mühlenbein (1991). Interesting overviews of Mühlenbein's views on parallel genetic algorithms, their design, the role of hill-climbing heuristics and their applications may be found in Mühlenbein (1991, 1992 and 1992a).

Gorges-Schleuter (1989, 1991, 1991a) studied fine-grained PGA and applied the strategy to the TSP using 64 processors of a transputer network. According to the author, similar to complex physical systems, the global behaviour of genetic algorithms might be better understood when viewed as an emergent phenomena resulting from the self-organization of simple and locally interacting rules. Similar to Mühlenbein (1989), individuals were divided into two equal subsets and placed on two rings. Demes were of size 8, and composed of neighbouring nodes on the two rings – the demes were thus overlapping. The selection was simple: one of the offspring replaced the parent. The mate for the crossover operation was chosen either by proportional or linear ranking selection (Goldberg and Deb, 1991). Several survival strategies (offspring replacing their parents) were tested and the strategy that accepted only offspring fitter than the local parent yielded the best solution, the best median solution and the lowest deviation.

Gorges-Schleuter also compared the similarity, with respect to the genotype, of demes that were at a physically different distance from one another. It was reported that the difference in the average Hamming distance between demes increased as the demes were more physically distant. This proves the existence of "niches" in fine-grained PGA. The author then described the impact of linear versus planar population structures on the gene pool. He reported that the fast propagation of the planar structure allowed advantageous genes to quickly overtake the whole population, but the very best solutions were missed. In Gorges-Schleuter (1992), the author continued

his study of the fine-grained PGA, and examined the impact of different local mating strategies on the formation of niches. A one-gene, two-allele model of individuals was used, and there were no selection or mutation operators. Results showed that mate selection with one fixed parent was more capable of producing stable local islands.

Maruyama, Hirose and Konagaya (1993) introduced an asynchronous fine-grained PGA applied to the graph partitioning problem using a cluster of workstations and a Sequent Symmetry computer. The authors attempted to adapt fine-grained PGA to coarse-grained parallel computer architectures. Each processor had an active individual and a buffer of several suspended individuals. Active individuals were sent to all other processors. Processors then randomly selected one individual among those received from the other processors, the new individual replacing one of the suspended individuals according to the fitness function. Crossover consisted of replacing part of the active individual by a part of one of the suspended individuals. In an unusual approach, only one offspring was produced, the other parts of the active and suspended individuals being rejected. Mutation was then applied and the modified active individual was compared to the suspended individuals. If it could not survive, it was then replaced by one of the suspended individuals according to the fitness function. Tests were performed with 15 processors for the Sequent Symmetry and 6 processors for the cluster of workstations. The authors reported near linear speed-ups for the same quality of solution on both types of coarse-grained architectures. A similar approach was proposed in Maruyama, Konagaya and Konishi (1992), with the difference that the buffer for each processor contained only individuals received from the other processors.

A similar "logical" fine-grained algorithm for coarse-grained computers was proposed by Tamaki and Nishikawa (1992), for the job shop scheduling problem. It is a logical parallel method because it was first implemented on a sequential computer, then implemented on 6 and 12 transputers to speed up the computation. The demes were composed of individuals at a Hamming distance of 1. Selection was based on the local fitness distribution, crossover chose a mate randomly in the neighbourhood, and mutation was performed using one bit selected randomly. Implementing fine-grained PGA on coarse-grained computers was also one of the objectives of Voigt, Santibáñez-Koref and Born (1992; see also Voigt and Born, 1990). The algorithm was applied to the F6 function using transputers. There was an initial level of selection and reproduction operators in the local environment; these local environments being linked according to the interconnection structure of the coarse-grained computer. Then, there were selection and reproduction policies for the interactions among local environments. The authors experimented with 1 to 7 processors and different fixed topologies to define the demes: ring, torus, lattice, hypercube, etc. Hill-climbing was applied when an individual did not improve for a given number of generations. Selection used a ranking scheme, and the offspring replaced the worst neighbour if its fitness was better. The best results were obtained for ring local environments with 3 individual demes. Additional fine-grained, cellular, PGA were proposed by Sannier and Goodman (1987), Talbi and Bessière (1991, 1991a), Muntean and Talbi (1991), etc.

Collins and Jefferson (1991) developed a fine-grained PGA for the multilevel graph partitioning problem using the massively parallel Connection Machine. The main goal of the paper was to characterize the difference between panmictic (i.e., at the

level of the entire population) and local selection/crossover schemes. It is known that panmictic selection/crossover converge on a single peak of multimodal functions, even when several solutions of equal quality exist. Tests were performed on a function with two optimal solutions, and the panmictic approach never found both solutions. The authors noted that modifications to the panmictic selection and crossover operators that changed this behaviour made use of global information which was not suited for parallel implementation. On the other hand, local selection and crossover displayed an obvious parallelism. In these implementations the demes were made of a random walk on a 1- or 2-dimensional grid, the deme size being a function of the length of the random walk. Selection was local from the random walk and used local fitness distribution and local linear ranking. The experimentation used 16000 processors, for a population size of 2^{13} to 2^{19}. The method using local selection and crossover consistently found both optimal solutions, was resistant to premature convergence because each deme could have explored different peaks, found optimal solutions faster, and was more robust. Gordon (1994) further compared the natural parallelism available in global, coarse-grained and fine-grained PGA. See Maniezzo (1993) for another example of fine-grained parallel implementation on the CM2 Connection Machine.

Schwehm (1992) implemented a fine-grained PGA on a massively parallel computer, the MasPar MP-1 using 1024 of the 16384 processors. The author investigated which network topology was best-suited to fine-grained PGA. The diameter of the network determined how long it took for good solutions to propagate over the entire environment. Long diameters isolated phenotypes with little chance of combining good alleles. Short diameters prevented genotypes to evolve since good solutions soon dominated, which lead to premature convergence. The author tested the following 5 topologies: ring of 1024 elements; torus of 32×32 elements; 3-cube of $16 \times 8 \times 8$ elements; 5-cube of 4^5 elements, 10-cube of 2^{10} elements. The torus showed the best results: higher and lower dimensionality resulting in slower convergence. A similar study on the interconnection topologies of fine-grained PGA may be found in the papers of Baluja (1992, 1993). According to Baluja, the overlapping populations of the cellular PGA corrects the drawback of coarse-grained PGA where convergence of a subpopulation to an equilibrium state may prevent the incorporation of new material because of its incompatibility with existing information. However, subpopulations of cellular PGA can also be dominated by the genotype of strong individuals. Three different topologies were studied regarding their capability to prevent this problem. Numerical results suggested that 2D array topologies obtain the better results. The author also suggested that a more rigorous model of the interactions in PGA is needed since we still do not fully understand the complexity added by these interactions to the behaviour of this particular metaheuristic.

Few authors compare the respective performances of coarse and fine-grained parallel genetic algorithms, and conclusions are generally not clear-cut (Cantù-Paz, 1995). For example, Baluja (1993) compared one coarse-grained and three fine-grained algorithms and found the latter to perform better, while the tests conducted by Gordon and Whitley (1993) indicated opposite conclusions. Baluja argues that the difficulty in defining a performance measure equally adaptable to the two parallelization types plays a major role in our inability to conclude on their relative performance.

10.5.2 Simulated Annealing

Numerous efforts to develop Type 3 parallel simulated annealing strategies, either independent or cooperative, are reported in the literature. A very interesting development in PSA consists in the systematic inclusion of principles and operators from genetic algorithms in simulated annealing multithread procedures. These hybrids tend to perform very well.

Aarts *et al.* (1986) proposed the first Type 3 parallelization scheme for simulated annealing, called *division strategy* by Aarts and Korst (1989). Rather than having several processors execute moves from the same current solution or from a same subset of decision variables, processors worked independently on different short Markov chains. Let L be the length of the Markov chain (number of iterations) executed by an SA program before reaching equilibrium at temperature t. The division strategy consisted of executing L/P SA iterations on P processors at temperature t. At the L/P-th iteration, the processors synchronized and one solution was chosen to be the initial configuration for the next temperature. Hence, this was a synchronous cooperative scheme with global exchange of information at each interaction. Unfortunately, the length of the chain could not be reduced arbitrarily without significantly affecting the convergence properties. This was particularly true at low temperatures where a large number of steps was required to preserve equilibrium. Consequently, at low temperatures, the processors were clustered and in each cluster a parallel move strategy (Type 2) was applied. Monien, Ramme and Salmen (1995) presented another example of this last strategy.

This strategy has been applied by Diekmann, Lüling and Simon (1993) to the TSP, the graph partitioning problem and the link assignment problem, using a network of transputers. The selection of the initial configurations was based on the Boltzmann distribution. A speed-up of 85 over 121 processors was reported. According to the authors, the convergence behaviour and the quality of solution were similar to the sequential algorithm and independent of the number of processors. In their study of this type of parallelization strategy, Aarts *et al.* (1986) had predicted an upper bound of 30 useful processors after which, the speed-up and the quality of solution were supposed to decrease. Apparently, the results of Diekmann, Lüling and Simon prove that the prediction was too pessimistic.

Interactions among processors are not limited to being executed only at the end of each Markov chain. In fact, there is a wide variety of possibilities here. The simplest approach consists of only one interaction at the end. This corresponds to the independent search model studied by Azencott (1992). Lee and Lee (1992a,b) also examined this approach, as well as variants where the SA threads interact periodically (the periods may be fixed or dynamic) in synchronous or asynchronous modes. They used 2, 4, 8 and 16 IntelPSC/2 processors, and a 32-processor BBN Butterfly GP1000 computer. For the graph partitioning problem, dynamic interval exchange strategies generally performed best. Also, asynchronous strategies where communications proceeded through a central memory (blackboard architecture) obtained solutions of equal or better quality compared to synchronous parallel schemes.

According to Laursen (1994), how to schedule these interactions is still an open problem. The author proposed to use the selection and migration operators of PGA to help address this issue. On each processor there were k simulated annealing procedures and all performed concurrently for l iterations. After the l iterations, processors

were paired and each migrated (copied) a number of states to its pair. After migration, each processor had $2k$ states, and this number was reduced to k by selection. These new k states were the initial configurations from which the k concurrent simulated annealing processes on each processor restarted the search. Experiments have been conducted for the QAP, the graph partitioning problem and the weighted vertex cover problem using a network of 16 transputers. The author tested three migration strategies: no migration, global and local (stepping-stone). Global migration consisted of bringing the best states to a given processor, which then chose the best among the best and broadcasted them to all processors. This strategy suffered a 10% to 25% overhead in communication cost and produced very bad solutions. The no communication (migration) approach was the fastest strategy but produced lower quality solutions compared to the local migration strategy, which incurred a 2% overhead.

Another PSA division strategy with coordination inspired from the GA operators is presented in Mahfoud and Goldberg (1995). In their paper, n Markov chains were evaluated concurrently. The SA-GA hybrid algorithm proceeded as follows: after $n/2$ iterations, two parents were selected from the population of the n current solutions. Two children were generated using a GA crossover operator, followed by a mutation operator. Probabilistic trial competitions (i.e., the winning probability depends on the current temperature and the relative difference between the fitness evaluation of the individuals that are competing with each other; several such competitions may be defined according to the exact winning probability formula, as well as the exact competition rules: one on one, the parents combined versus one of the offspring, etc.) were held between the children and the parents and the replacement step was performed according to the outcome of the competition. The temperature was lowered when the population reached equilibrium. The parallelization of this algorithm is straightforward. It divided the population of n Markov chains into P subpopulations of n/P Markov chains. Crossover, mutation and probability trials were applied to individuals of each local subpopulation. Two prototypes were implemented on the CM-5 Connection Machine, one synchronous and one asynchronous. In synchronous mode, the migration operator periodically sent the subpopulations (distributed memories) to a master processor that randomly redistributed the individuals to different processors. In asynchronous mode, processors were periodically paired and $(n/P)/2$ individuals were exchanged.

A different approach to building genetic-annealing parallel methods was proposed by Ram, Sreenivas and Subramaniam (1996). The authors described two PSA methods and applied them to the job shop and travelling salesman problems, using a network of 18 SUN workstations. Both methods were composed of two phases. The second phase of each method consisted of an independent search process, which was the same for both methods. In the first phase of the first approach, the search threads started from different initial solutions, performed a number of iterations and exchanged their respective best solutions; all threads then restarted from the overall best solution. In the first phase of the second method, a genetic method was used to obtain a population of good solutions that was then used to initiate the independent search phase of the method. The results tended to show that given a fixed time for the first phase, the quality of the solution determined by the GA decreased as the problem size increased. When more time was allocated to the GA phase, the overhead of the parallel method became too high. Moreover, when the number of processors was

increased, the number of good solutions in the population was less than the number of processors, hence some were initialized with "bad" solutions. Other GA-SA hybrids have been proposed by Sirag and Weisser (1987), Brown, Huntley and Spillane (1989), Lin, Kao and Hsu (1991), Boseniuk and Ebeling (1988, 1991), Boseniuk, Ebeling and Engel (1987), Goldberg (1990) and Varanelli and Cohoon (1995).

An innovative approach to developing PSA methods has been proposed by Fleischer (1996; see also Fleischer and Jacobson, 1996). The author made use of control theory to devise a negative feedback mechanism to control the modifications to the temperature parameter and thus to the transition probabilities. The mechanism is based on probabilistic information regarding the quality of the current solutions as evaluated by two or more SA procedures that proceed concurrently and synchronously. These ideas were tested on the independent set and vertex cover problems with encouraging results. It is noteworthy that this approach does not adversely impact the convergence properties of the simulating annealing method.

10.5.3 GRASP

A Type 3 parallelization for GRASP is proposed by Pardalos, Pitsoulis and Resende (1995) for the QAP. It is an independent thread method implemented on a Kendal Square KRS-1 with 128 processors, 64 of which were used in the experiment. An excellent average speed-up (62) was reported.

10.5.4 Tabu Search

Type 3 parallelizations for tabu search methods follow the same basic pattern: p threads search through the same solution space, starting from possibly different initial solutions and using possibly different tabu search strategies. In their taxonomy, Crainic, Toulouse and Gendreau (1997) identify these various possibilities as Single (initial) Point Different Strategies (SPDS), Multiple Point Single Strategy (MPSS), and Multiple Point Different Strategies (MPDS). The search threads may proceed in a totally independent fashion, the best solution being identified at the end. These strategies are identified as *independent multithread* methods. When information is exchanged among tabu threads, the so-called *cooperative multithread* methods, synchronous communications have been mainly implemented. Asynchronous procedures are, however, increasingly being developed. Consequently, one observes an increased awareness of the issues related to the definition and modelling of cooperation.

Battiti and Tecchiolli (1992) used the QAP to present a tabu search with a hashing procedure used to cause the search to react to the detection of cycles by suitably modifying the length of the tabu lists. The authors then analyzed a parallelization scheme where several independent search processes started the exploration of the domain from different, randomly generated, initial configurations. The authors then proceeded to derive probability formulas for the success of the global search that tended to show that the independent search parallelization scheme was efficient – the probability of success increased, while the average success time decreased with the number of processors – provided the tabu procedure did not cycle.

Taillard also studied parallelization strategies that perform many independent searches, each starting with different initial solutions. The main study is found in his paper on parallel tabu methods for job shop scheduling problems (Taillard, 1994).

For this type of problem, Taillard showed that a tabu search approach, which includes a diversification phase, is very competitive. It is simpler to implement and generally more efficient than either the simulated annealing or the shifting bottleneck procedures (the two best heuristics proposed at the time). It helped establish new best known solutions for every problem in two sets of benchmark problems, while optimally solving random problems with m machines $\ll n$ jobs (e.g., $m = 5$, $n = 2000$) in polynomial mean time. Several parallelization ideas focussing on speeding up computations related to the neighbourhood evaluation (Type 1 parallelism) did not yield good results, either because the available computing platforms (a ring of transputers and a 2-processor Cray computer) were not suitable for the implementations, or because the communication times were much higher than the computation ones. Taillard then proceeded to examine the theoretical bases of the independent multithread parallelization approach for "random" iterative algorithms (tabu search, simulated annealing, etc.). His results showed that the conditions needed for the parallel approach to be "better" than the sequential one are rather strong, where "better" was defined as the probability of the parallel algorithm achieving success with respect to some condition (in terms of optimality or near-optimality) by time t being higher than the corresponding probability of the sequential algorithm by time pt. However, the author also mentioned that, in many cases, the empirical probability function of iterative algorithms was not very far from an exponential one and that the independent multithread parallelization approach is very efficient. The results for the job shop problem seem to justify this claim.

Rego and Roucairol (1996) proposed a tabu search approach for the VRP based on ejection chains, and implemented an independent multithread parallel version where each thread used a different set of parameter settings but started from the same solution. The method was implemented in a master-slave setting, where each slave executed a complete sequential tabu search. The master gathered the solutions found by the threads, selected the overall best, and reinitialized the threads for a new search. Low-level (Type 1) parallelism was used to accelerate the move evaluations of the individual searches, as well as a post-optimization phase. Experiments were conducted on a network of four SUNSparc workstations and showed the method to be competitive on a set of standard VRP problems (Christofides, Mingozzi, and Toth, 1979).

Malek *et al.* (1989) implemented and compared serial and parallel simulated annealing and tabu search algorithms for the travelling salesman problem. The authors reported that the parallel tabu search implementation outperformed the serial one, and consistently produced comparable or better results than sequential or parallel simulated annealing. The parallel experiments were performed on a 10-processor Sequent Balance 8000 computer, for the 25, 33, 42, 57 and 100-city problems with proven optimum solution, and for the 50 and 75-city problems where only best-know solutions are available. The implementation proceeded with one main process and four child processes. Each child process ran a serial tabu search algorithm with different tabu conditions and parameters. The child processes were stopped after a specified time interval, the solutions were compared, and bad areas of solution space were eliminated. The child processes were then restarted with a good solution and an empty tabu list. Note that, in order to strictly implement this strategy, the diversification long-term memory function was disabled.

De Falco *et al.* (1994) and De Falco, Del Balio and Tarantino (1995) studied the QAP and the mapping problem, respectively, and experimented on transputer networks (with 16, 32 and 64 processors), a MAsPAr MPP-1 SIMD computer, and a Convex Meta Series MIMD machine. They implemented a multithread strategy, where the best solutions from individual processes were exchanged at each iteration among their neighbours. At each iteration, each process performed a local search from its best solution. Then, processes synchronized and the neighbours exchanged their respective best solutions; local best solutions were replaced with imported ones only if the latter solutions were better. The authors indicated that they obtained better solutions when cooperation was included compared to an independent thread strategy. Superlinear speed-ups were reported.

A continuously increasing number of asynchronous cooperative multithread search methods is being proposed. All such developments we have identified, use a *central memory* for inter-thread communications. Each individual search thread starts from a different initial solution and generally follows a different search strategy. Exchanges are asynchronously performed and proceed through the central memory.

As far as we can tell, Crainic, Toulouse and Gendreau (1997) proposed the first such strategy for tabu search as part of their taxonomy of parallel TS methods. The authors also presented a thorough comparison of various parallelization strategies based on this taxonomy (Crainic, Toulouse and Gendreau, 1995 and 1995a). The authors implemented several Type 1 and 2 strategies, one independent multithread approach, and a number of synchronous and asynchronous cooperative multithread methods. The multicommodity location problem with balancing requirements was used for the experimentation conducted on a 16-transputer network. The authors report that the parallel versions achieved better quality solutions than the sequential TS and that, in general, asynchronous methods outperformed synchronous ones. The independent threads and the asynchronous cooperative approaches offered the best performances.

An interesting approach to the development of Type 3 cooperative multithread parallel tabu search methods is based on *adaptive memory* ideas (see Section 10.4). It has been particularly used for real-time routing and vehicle dispatching problems (Gendreau *et al.*, 1996), as well as for VRP with time window restrictions (Taillard *et al.*, 1995; Badeau *et al.*, 1995). A general implementation framework would have each thread construct an initial solution and improve it through a tabu search or any other procedure. Each thread deposits the routes of its improved solution in the adaptive memory. It then constructs a new initial solution out of the routes in the adaptive memory, improves it, communicates its routes to the adaptive memory, and so on. A "central" process manages the adaptive memory and ensures communication among the independent threads. It also determines when the procedure stops based on the number of calls to the adaptive memory, the number of successive calls without improving the best solution, or a time limit. In an interesting development, Gendreau *et al.* (1996) also exploited parallelism within each search thread: the set of routes was decomposed along the lines proposed in Taillard's work (1993). Very interesting performances have been observed on a network of workstations, especially when the number of processors is increased.

Crainic and Gendreau (1997) proposed a cooperative multithread PTS for the fixed cost, capacitated, multicommodity network design problem. The individual

tabu search threads differ in their initial solution and parameter settings. Communications are performed asynchronously through a central memory device (blackboard implementation). The authors compared five strategies to retrieve a solution from the pool when requested by an individual TS thread. The strategy that always returns the overall best solution displayed the best performances when few (4) processors were used. When the number of processors increased, a probabilistic procedure, based on the rank of the solution in the pool, appeared to be best. The parallel procedure improved the quality of the solution and required less computing time compared to the sequential version, particularly for large problems with many commodities (results for problems with up to 700 design arcs and 400 commodities are reported). The experimental results have also emphasized the need for the individual threads to proceed unhindered for some time (e.g., the first diversification move) before initiating exchanges of solutions. This ensures that local search histories can be established and good solutions can be found to establish the central memory as an *elite candidate* set. On the other hand, early and frequent communications yielded a totally random search that proved to be quite ineffective. The authors finally reported that the cooperative multithread procedure also outperformed an independent search strategy that uses the same search parameters and starts from the same initial points. Other implementations of asynchronous cooperative multithread PTS methods were presented by Andreatta and Ribeiro (1994) and Aiex *et al.* (1996; see also Martins, Ribeiro, and Rodriguez, 1996) for the problem of partitioning integrated circuits for logical testing.

Crainic and Gendreau (1997a) report the development of a hybrid combining their cooperative multithread PTS method with a genetic engine. The GA initiates its population with the first elements of the central memory of the PTS. Asynchronous migration (rate = 1) subsequently transfers the best solution of the GA to the PTS, and other solutions of the pool towards the GA. The hybrid appears to perform well, especially on larger problems where the best solutions known are improved. It is noteworthy that the GA alone was not performing well and that it was the parallel TS procedure that identified the best results once the genetic method contributed to the quality of the central memory.

Multithread parallelization thus appears to offer very interesting perspectives for metaheuristics. However, several issues remain to be addressed.

Synchronous implementations, where information is exchanged at regular intervals have been reported for the three classes of metaheuristics examined in this paper. In general, these implementations outperform the serial methods in solution quality. For tabu search (Crainic, Toulouse and Gendreau, 1995) and simulated annealing (Graffigne, 1992) synchronous cooperative methods appear to be outperformed, however, by independent search procedures. Yet, the study by Lee and Lee (1992a) reversed this trend. Their results show the independent thread approach to be outperformed by two strategies of synchronous cooperating parallel threads. This points to interesting research issues which should be further investigated since Lee and Lee used a dynamically adjusted synchronization interval that modified the traditional synchronous parallelism paradigm. This is also the case for genetic algorithms: Cohoon *et al.* (1987) and Cohoon, Martin and Richards (1991a, 1991b) report that parallel search with migration operators applied at regular intervals outperform the same method without migration. Asynchronous cooperative multiple search strategies ap-

pear to have been less studied. Our group has undertaken the study of distributed multithread tabu search methods (Crainic, Toulouse and Gendreau, 1995a, 1997). The results which are already available seem to indicate that these approaches offer better results than synchronous and independent searches, but more theoretical and empirical work is still required.

An important issue that is beginning to be acknowledged in the parallel metaheuristic community, concerns the definition of cooperation schemes and their impact on the search behaviour and performance. A number of basic communication issues in designing multithread parallel metaheuristics are discussed by Toulouse, Crainic and Gendreau (1996). A more thorough analysis tends to show (Toulouse, Crainic and Sansó, 1997a,b) that cooperative parallel metaheuristics form dynamic systems and that the evolution of these systems may be more strongly determined by the cooperation scheme than by the optimization process. This points to the urgent need to better understand cooperative procedures and the impact of communication parameters and design on their behaviour.

10.6 CONCLUSIONS AND PERSPECTIVES

Metaheuristic parallel search methods, particularly tabu search, simulated annealing and genetic algorithms, were reviewed, classified and examined according not to particular methodological characteristics, but following the unifying approach of the level of parallelization. To our knowledge, it is the first time that such a review has been attempted and, although not exclusive, it examined and identified the commonalities among parallel implementations across the field of metaheuristics.

It thus appears clearly that beyond the often very clever implementations that take advantage of the particular characteristics of the problem and computer at hand, a relatively limited number of parallelization strategies have been proposed.

All types of parallelization approaches have their merits and will certainly continue to be applied when appropriate. One may identify, however, a rather clear trend in the development of such methodologies: from low-level parallelization of computing-intensive tasks, to domain decomposition, to multithread methods where several searches explore the problem space concurrently.

Multithread approaches are increasingly proving their worth. These methods allow to increase the variety of the search threads. Not only by a more efficient implementation of restarting strategies, but more particularly by the possibility of having different types of searches – same method with different parameter settings or even different metaheuristic and exact searches – proceeding concurrently. Thus, a more thorough exploration of the solution space of a given problem instance may be obtained. As an additional benefit, multithread methods appear more robust than their sequential counterparts relative to the differences in problem types and characteristics. Such approaches also offer a relatively easy way of harnessing the power of the networks of workstations present in most firms and organizations. It is our belief that the field of multithread parallel metaheuristics, and particularly, cooperative methods, will continue to grow and offer exciting research possibilities.

Of prime importance are the issues related to the definition, understanding and control of cooperation. We have barely begun to explore this field and we realize that cooperative parallelism sometimes obeys laws that we do not fully comprehend, and that the parallel search we build may evolve by following its own dynamics and

not according to our optimization goals. We need a better understanding of these dynamics and of the related global search behaviour of cooperative search, of the impact of the various design parameters on this behaviour, and of the means available to control the search trajectory. A few research efforts have been initiated, but the field is still largely unexplored.

An interesting research issue, which has been addressed by the GA and SA communities but has been largely neglected regarding tabu search, concerns the search properties of the parallel method, particularly cooperative ones, with respect to the basic hypotheses of the methodology. Thus, there has been little research dedicated to the study of parallel TS viewed as one "meta" tabu search method: How do we recreate global memories out of the individual recollections of the individual threads? How do we direct the parallel process into intensification and diversification phases? How do we make the most of the information being exchanged and thus turn the communication search overhead into an improving factor? etc. (With respect to this last question, it is worth remembering that the exchanged solutions form a type of elite "population".) By addressing these issues, one may not only improve the overall performance of the parallel TS procedure, but also achieve a better understanding of the cooperation process.

Hybrid methods appear to offer very interesting perspectives. They may dramatically improve the performance of some metaheuristics, even in their sequential version. We witness increasingly ideas of one methodology (e.g., GA) being used in another area (e.g., SA). We believe this holds a promising future. Similarly, the cooperation among search threads of different types of metaheuristics or exact searches (e.g., branch-and-bound and tabu search in parallel) opens up attractive perspectives.

Finally, here are a few other preoccupations and promising research questions related to parallel metaheuristics; with respect to cooperative search procedures, one does not always compare the parallel procedure to the appropriate sequential one, which results in sometimes inaccurate speed-up claims. (ii) More comparisons among different types of metaheuristics are needed. (iii) More attention should be paid when designing parallel procedures to the load balancing issue. This is especially relevant for Type 1 and 2 implementation where "slave" processors may easily become idle and work for very different spans. (iv) Software libraries of parallel procedures and test problems should be built. These already exist or are being undertaken for some problem classes and mainly for sequential methods. Their existence for parallel procedures would greatly enhance our ability to construct new procedures and to evaluate and compare existing ones more thoroughly. (v) Ant colony systems are "naturally" parallel procedures but true parallelizations have yet to be attempted.

Thus parallel computation appears to be a most-promising research and development area for metaheuristics. It already efficiently addresses large instances of complex problems and holds great promises for the future. Many questions and challenges still exist but, as this review shows, a more unified view of parallel metaheuristics may significantly help to address them.

Acknowledgments

We would like to thank the anonymous referees whose comments have helped us write a better paper. Funding for this project has been provided by the Natural Sciences and

Engineering Council of Canada, through its Research and Strategic Grant programs, and by the Fonds F.C.A.R. of the Province of Québec.

References

Aarts, E. and J. Korst. (1989). *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, New York, NY.

Aarts, E.H.L, F.M.J de Bont, J.H.A. Habers and P.J.M van Laarhoven. (1986). Parallel Implementations of Statistical Cooling Algorithms. *Integration, The VLSI Journal*, 3:209–238.

Abramson, D. and J. Abela. (1992). A Parallel Genetic Algorithm for Solving the School Timetabling Problem. In G. Gupta and C. Keen, editors, *15th Australian Computer Science Conference*, pages 1–11. Department of Computer Science, University of Tasmania.

Abramson, D., G. Mills and S. Perkins. (1993). Parallelization of a Genetic Algorithm for the Computation of Efficient Train Schedules. In D. Arnold, R. Christie, J. Day and P. Roe, editors, *Proceedings of the 1993 Parallel Computing and Transputers Conference*, pages 139–149. IOS Press.

Adamidis, P. and V. Petridis. (1996). Co-operating Populations with Different Evolution Behaviors. In *IEEE 1996 International Conference on Evolutionary Computation*, pages 188–191.

Aiex, R.M., S.L. Martins, C.C. Ribeiro and N.R. Rodriguez. (1996). Asynchronous Parallel Strategies for Tabu Search Applied to the Partitioning of VLSI Circuits. Monografias em ciência da computação, Pontifícia Universidade Católica de Rio de Janeiro.

Allwright, J.R.A. and D.B. Carpenter. (1989). A Distributed Implementation of Simulated Annealing for the Travelling Salesman Problem. *Parallel Computing*, 10:335–338.

Andreatta, A.A. and Ribeiro C.C. (1994). A Graph Partitioning Heuristic for the Parallel Pseudo-Exhaustive Logical Test of VLSI Combinational Circuits. *Annals of Operations Research*, 50:1–36.

Azencott, R. (1992). Parallel Simulated Annealing: An Overview of Basic Techniques. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 37–46. John Wiley & Sons, New York, NY.

Azencott, R. editor. (1992a). *Simulated Annealing Parallelization Techniques.* John Wiley & Sons, New York, NY.

Badeau, P., F. Guertin, M. Gendreau, J.-Y. Potvin and É.D. Taillard. (1997). A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research C*, 5(2):109–122.

Baiardi, F. and S. Orlando. (1989). Strategies for a Massively Parallel Implementation of Simulated Annealing. In *Parallel Architectures and Languages, PARLE'89*, pages 273–287.

Baluja, S. (1992). A Massively Distributed Parallel Genetic Algorithm. Technical Report CMU-CS-92-196R, Carnegie Mellon University.

Baluja, S. (1993). Structure and Performance of Fine-Grain Parallelism in Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 155–162. Morgan Kaufmann, San Mateo, CA.

Banerjee, P. and M. Jones. (1986). A Parallel Simulated Annealing Algorithm for Standard Cell Placement on Hypercube Computer. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 34–37. IEEE Computer Society Press, Washington, DC.

Banerjee, P., M. Jones and J. Sargent. (1990). Parallel Simulated Annealing Algorithm for Cell Placement on Hypercube Multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 1(1):91–106.

Bannister, J. and M. Gerla. (1989). Design of the Wavelength-Division Optical Network. Report CSD-890022, UCLA Computer Science Department.

Barbosa, V.C. and E. Gafni. (1989). A Distributed Implementation of Simulated Annealing. *Journal of Parallel and Distributed Computing*, 6(2):411–434.

Barr, R.S., B.L. Golden, J.P. Kelly, M.G.C. Resende and W.R. Stewart. (1995). Designing and Reporting on Computational Experiments with Heuristic Methods. *Journal of Heuristics*, 1(1):9–32.

Barr, R.S. and B.L. Hickman. (1993). Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts Opinions. *ORSA Journal on Computing*, 5(1):2–18.

Battiti, R. and G. Tecchiolli. (1992). Parallel Based Search for Combinatorial Optimization: Genetic Algorithms and TABU. *Microprocessors and Microsystems*, 16(7):351–367.

Beasley, D., D.R. Bull and R.R. Martin. (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, 15(2):58–69.

Belding, T. (1995). The Distributed Genetic Algorithm Revisited. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 114–121. Morgan Kaufmann, San Mateo, CA.

Bianchini, R. and C.M. Brown. (1992). Parallel Genetic Algorithms on Distributed-Memory Architectures. Technical Report TR 436, University of Rochester, Computer Science Department.

Boissin, N. and J.L. Lutton. (1993). A Parallel Simulated Annealing Algorithm. *Parallel Computing*, 19(8):859–872.

Bongiovanni, G., P. Crescenzi and C. Guerra. (1995). Parallel Simulated Annealing for Shape Detection. *Computer Vision and Image Understanding*, 61(1):60–69.

Borut, R. and J. Silc. (1994). Using Parallel Simulated Annealing in the Mapping Problem. *Lecture Notes in Computer Science 817*, pages 797–800.

Boseniuk, T. and W. Ebeling. (1988). Optimization of NP-Complete Problems by Boltzmann-Darwin Strategies Including Life Cycles. *Europhysics Letters*, 6(2): 107–112.

Boseniuk, T. and W. Ebeling. (1991). Boltzmann, Darwin and Haeckel Strategies in Optimization Problems. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 430–444. Springer-Verlag, Berlin.

Boseniuk, T., W. Ebeling and A. Engel. (1987). Boltzmann and Darwin Strategies in Complex Optimization. *Physics Letters A*, 125(6/7):307–310.

Branke, J., U. Kohlmorgen and H. Schmeck. (1995). A Distributed Genetic Algorithm Improving the Generalization Behavior of Neural Networks. *Lecture Notes in Computer Science 912*, pages 107–121.

Brouwer, R. and P. Banerjee. (1988). A Parallel Simulated Annealing Algorithm for Channel Routing on a Hypercube Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-88*, pages 4–7. IEEE Computer Society Press, Washington, DC.

Brown, D.E., C.L. Huntley and A.R. Spillane. (1989). A Parallel Genetic Heuristic for the Quadratic Assignment Problem. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 406–415. Morgan Kaufmann, San Mateo, CA.

Cantú-Paz, E. (1995). A Summary of Research on Parallel Genetic Algorithms. Report 95007, University of Illinois at Urbana-Champain.

Cantú-Paz, E. and D.E. Goldberg. (1996). Predicting Speedups of Ideal Bounding Cases of Parallel Genetic Algorithms. Report 96008, University of Illinois at Urbana-Champain.

Casotto, A., F. Romeo and A.L. Sangiovanni-Vincentelli. (1986). A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 30–33. IEEE Computer Society Press, Washington, DC.

Casotto, A., F. Romeo and A.L. Sangiovanni-Vincentelli. (1987). A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells. *IEEE Transactions on Computer-Aided Design*, 6(5):838–847.

Casotto, A., F. Romeo and A.L. Sangiovanni-Vincentelli. (1987a). Placement of Standard Cells Using Simulated Annealing on the Connection Machine. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-87*, pages 350–353. IEEE Computer Society Press, Washington, DC.

Chakrapani, J. and J. Skorin-Kapov. (1992). A Connectionist Approach to the Quadratic Assignment Problem. *Computers & Operations Research*, 19(3/4):287–295.

Chakrapani, J. and J. Skorin-Kapov. (1993). Massively Parallel Tabu Search for the Quadratic Assignment Problem. *Annals of Operations Research*, 41:327–341.

Chakrapani, J. and J. Skorin-Kapov. (1993a). Connection Machine Implementation of a Tabu Search Algorithm for the Traveling Salesman Problem. *Journal of Computing and Information Technology*, 1(1):29–36.

Chakrapani, J. and J. Skorin-Kapov. (1995). Mapping Tasks to Processors to Minimize Communication Time in a Multiprocessor System. In *The Impact of Emerging Technologies of Computer Science and Operations Research*, pages 45–64. Kluwer, Norwell, MA.

Chamberlain, R.D., M.N. Edelman, M.A. Franklin and E.E. Witte. (1988). Simulated Annealing on a Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-88*, pages 540–544. IEEE Computer Society Press, Washington, DC.

Chen, H. and N.S. Flann. (1994). Parallel Simulated Annealing and Genetic Algorithms: A Space of Hybrid Methods. In Y. Davidor, H.-P. Schwefel and R Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 428–438. Springer-Verlag, Berlin.

Chen, R.J., R. Meyer and J. Yeckel. (1993). A Genetic Algorithm for Diversity Minimization and its Parallel Implementation. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 163–170. Morgan Kaufmann, San Mateo, CA.

Chen, Y.-W., Z. Nakao and X. Fang. (1996). Parallelization of a Genetic Algorithm for Image Restoration and its Performance Analysis. In *IEEE International Conference on Evolutionary Computation*, pages 463–468.

Christofides, N., A., Mingozzi and P. Toth. (1979). The Vehicle Routing Problem. In N. Christofides, A. Mingozzi, P. Toth and C. Sandi, editors, *Combinatorial Optimization*, pages 315–338. John Wiley, New York.

Chung, M.J. and K.K. Rao. (1986). Parallel Simulated Annealing for Partitioning and Routing. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers; ICCD'86*, pages 238–242. IEEE Computer Society Press, Washington, DC.

Cohoon, J., S. Hedge, W. Martin and D. Richards. (1987). Punctuated Equilibria: A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 148–154. Lawrence Erlbaum Associates, Hillsdale, NJ.

Cohoon, J., S. Hedge, W. Martin and D. Richards. (1991). Distributed Genetic Algorithms for the Floorplan Design Problem. *IEEE Transactions on Computer-Aided Design*, 10:483–492.

Cohoon, J., W. Martin and D. Richards. (1991a). Genetic Algorithm and Punctuated Equilibria in VLSI. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 134–144. Springer-Verlag, Berlin.

Cohoon, J., W. Martin and D. Richards. (1991b). A Multi-Population Genetic Algorithm for Solving the k-Partition Problem on Hyper-Cubes. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 134–144. Morgan Kaufmann, San Mateo, CA.

Collins, R.J. and D.R. Jefferson. (1991). Selection in Massively Parallel Genetic Algorithms. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256. Morgan Kaufmann, San Mateo, CA.

Colorni, A., M. Dorigo and V. Manniezzo. (1991). Distributed Optimization by Ant Colonies. In *Proceedings European Conference on Artificial Life*, pages 134–142. North-Holland, Amsterdam.

Colorni, A., M. Dorigo and V. Manniezzo. (1992). An Investigation of Some Properties of an 'Ant Algorithm'. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 509–520. North-Holland, Amsterdam.

Crainic, T.G. and M. Gendreau. (1997). A Cooperative Parallel Tabu Search for Capacitated Network Design. Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Crainic, T.G. and M. Gendreau. (1997a). Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. Publication CRT-97-27, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Crainic, T.G. and G. Laporte. (1997). Planning Models for Freight Transportation. *European Journal of Operational Research*, pages 409–438.

Crainic, T.G., M. Toulouse and M. Gendreau. (1995). Synchronous Tabu Search Parallelization Strategies for Multicommodity Location-Allocation with Balancing Requirements. *OR Spektrum*, 17(2/3):113–123.

Crainic, T.G., M. Toulouse and M. Gendreau. (1995a). Parallel Asynchronous Tabu Search for Multicommodity Location-Allocation with Balancing Requirements. *Annals of Operations Research*, 63:277–299.

Crainic, T.G., M. Toulouse and M. Gendreau. (1997). Towards a Taxonomy of Parallel Tabu Search Algorithms. *INFORMS Journal on Computing*, 9(1):61–72.

Cui, J. and T.C. Fogarty. (1992). Optimization by Using a Parallel Genetic Algorithm on a Transputer Computing Surface. In M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 246–254. IOS Press, Amsterdam.

Darema, F., S. Kirkpatrick and V.A. Norton. (1987). Parallel Algorithms for Chip Placement by Simulated Annealing. *IBM Journal of Research and Development*, 31:391–402.

Darema, F. and G. F. Pfister. (1987a). Multipurpose Parallelism in VLSI Computer-Aided Design: A Case Study. Report Research Report RC12516, IBM T.J. Watson Research Center.

Davis, M., L. Liu and J.G. Elias. (1994). VLSI Circuit Synthesis Using a Parallel Genetic Algorithm. In *IEEE'94 Conference on Evolutionary Computing*, pages 104–109.

De Falco, I., R. Del Balio and E. Tarantino. (1995). Solving the Mapping Problem by Parallel Tabu Search. Report, Istituto per la Ricerca sui Sistemi Informatici Paralleli-CNR.

De Falco, I., R. Del Balio, E. Tarantino and R. Vaccaro. (1994). Improving Search by Incorporating Evolution Principles in Parallel Tabu Search. In *Proceedings of the International Conference on Machine Learning*, pages 823–828.

De Jong, K.A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan.

Devadas, S. and A.R. Newton. (1986). Topological Optimization of Multiple Level Array Logic: on Uni and Multi-Processors. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 38–41. IEEE Computer Society Press, Washington, DC.

Diekmann, R., R. Lüling and J. Simon. (1993). Problem Independent Distributed Simulated Annealing and its Applications. In R.V.V. Vidal, editor, *Lecture Notes in Economics and Mathematical Systems, volume 396*, pages 17–44. Springer Verlag, Berlin.

Dorigo, M. (1992). *Ottimizzazione, Apprendimiento Automatico, ed Algoritmi Basati su Metafora Naturale.* PhD thesis, Politecnico di Milano.

Dorigo, M., V. Manniezzo and A. Colorni. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41.

Durand, M.D. (1989). Parallel Simulated Annealing: Accuracy vs. Speed in Placement. *IEEE Design & Test of Computers*, 6(3):8–34.

Durand, M.D. (1989a). Cost Function Error in Asynchronous Parallel Simulated Annealing Algorithms. Technical Report CUCS-423-89, University of Columbia.

Elo, S. (1994). A Parallel Genetic Algorithm on the CM-2 for Multi-Modal Optimization. In *Proceedings of the International Conference on Machine Learning*, pages 818–822.

Felten, E., S. Karlin and S.W. Otto. (1985). The Traveling Salesman Problem on a Hypercube, MIMD Computer. In *Proceedings of the International Conference on Parallel Processing*, pages 6–10, New York, NY.

Feo, T.A. and M.G.C. Resende. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6(2):109–133.

Feo, T.A., M.G.C. Resende and S.H. Smith. (1994). A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research*, 42(5): 860–878.

Fiechter, C.-N. (1994). A Parallel Tabu Search Algorithm for Large Travelling Salesman Problems. *Discrete Applied Mathematics*, 51(3):243–267.

Fleischer, M.A. (1996). Cybernetic Optimization by Simulated Annealing: Accelerating Convergence by Parallel Processing and Probabilistic Feedback Control. *Journal of Heuristics*, 1(2):225–246.

Fleischer, M.A. and S.H. Jacobson. (1996). Optimization by Simulated Annealing: An Implementation of Parallel Processing Using Probabilistic Feedback Control. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 249–275. Kluwer, Norwell, MA.

Fogarty, T.C. and R. Huang. (1990). Implementing the Genetic Algorithm on Transputer Based Parallel Systems. In H.-P. Schwefel and R. Männer, editors, *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 145–149. Springer-Verlag, Berlin.

Fogel, D.B. (1994). Evolutionary Programming: An Introduction and Some Current Directions. *Statistics and Computing*, 4:113–130.

Forrest, S. (1991). Emergent Computation: Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Network. In S. Forrest, editor, *Emergent Computation*, pages 1–11. MIT/North-Holland.

Garcia, B.L., J.-Y. Potvin and J.M. Rousseau. (1994). A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints. *Computers & Operations Research*, 21(9):1025–1033.

Gendreau, M., P. Badeau, F. Guertin, J.-Y. Potvin and É.D. Taillard. (1996). A Solution Procedure for Real-Time Routing and Dispatching of Commercial Vehicles. Publication CRT-96-24, Centre de recherche sur les transports, Université de Montréal.

Gendron, B. and T.G. Crainic. (1994). Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):1042–1066.

Ghannadian, F., R. Shonkwiler and C.O. Alford. (1993). Parallel Simulated Annealing for the *n*-Queen Problem. In *IPPS: 7th International Parallel Processing Symposium*, pages 690–694. IEEE Computer Society Press.

Giordana, A., L. Saitta and F. Zini. (1994). Learning Disjunctive Concepts by Means of Genetic Algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 96–104.

Girodias, K.A., H.H. Barrett and R.L Shoemaker. (1991). Parallel Simulated Annealing Algorithm for Emission Tomography. *Phys. Med. Biol.*, 36(7):921–938.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 1(3):533–549.

Glover, F. (1989). Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu Search – Part II. *ORSA Journal on Computing*, 2(1):4–32.

Glover, F. (1996). Tabu Search and Adaptive Memory Programming – Advances, Applications and Challenges. In R.S. Barr, R.V. Helgason and J. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, Norwell, MA.

Glover, F. and M. Laguna. (1993). Tabu Search. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. Blackwell Scientific Publications, Oxford.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.

Goldberg, D.E. (1990). A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing. *Complex Systems*, 4: 445–460.

Goldberg, D.E. and K. Deb. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithm & Classifier Systems*, pages 69–93. Morgan Kaufman, San Mateo, CA.

Goldberg, D.E. and J. Richardson. (1987). Genetic Algorithms with Sharing for Multimodal Function Optimization. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 41–49. Lawrence Erlbaum Associates, Hillsdale, NJ.

Golden, B.L., E.A. Wasil, J.P. Kelly and I.-M. Chao. (1998). The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56, Kluwer, Norwell, MA.

Gordon, V.S. and D. Whitley. (1993). Serial and Parallel Genetic Algorithms as Function Optimizers. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kaufmann, San Mateo, CA.

Gordon, V.S., D. Whitley and A Bohm. (1992). Dataflow Parallelism in Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 533–542. North-Holland, Amsterdam.

Gordon, V.S. (1994). Locality in Genetic Algorithms. In *Proceedings International Conference on Machine Learning*, pages 428–432.

Gordon, V.S. and D. Whitley. (1994). A Machine-Independent Analysis of Parallel Genetic Algorithms. *Complex Systems*, 8:181–214.

Gorges-Schleuter, M. (1989). ASPARAGOS: An Asynchronous Parallel Genetic Optimization Strategy. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–427. Morgan Kaufmann, San Mateo, CA.

Gorges-Schleuter, M. (1991). ASPARAGOS: A Parallel Genetic Algorithm for Population Genetics. In J.D. Becker, I. Eisele and F.W. Mundemann, editors, *Parallelism, Learning, Evolution. Workshop on Evolutionary Models and Strategies - WOPPLOT 89*, pages 407–418. Springer-Verlag, Berlin.

Gorges-Schleuter, M. (1991a). Explicit Parallelism of Genetic Algorithm Through Population Structures. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 150–160. Springer-Verlag, Berlin.

Gorges-Schleuter, M. (1992). Comparison of Local Mating Strategies in Massively Parallel Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 553–562. North-Holland, Amsterdam.

Graffigne, C. (1992). Parallel Annealing by Periodically Interacting Multiple Searches: an Experimental Study. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 47–79. John Wiley & Sons, New York, NY.

Graham, P. and B. Nelson. (1995). A Hardware Genetic Algorithm for the Travelling Salesman Problem on SPLASH 2. In W. Moore and W. Luk, editors, *Field-Programmable Logic and Applications*, pages 352–361. Springer Verlag, Oxford, England.

Greening, D.R. (1989). A Taxonomy of Parallel Simulated Annealing Techniques. Technical Report No. RC 14884, IBM.

Greening, D.R. (1990). Asynchronous Parallel Simulated Annealing. *Lectures in Complex Systems*, 3:497–505.

Greening, D.R. (1990a). Parallel Simulated Annealing Techniques. *Physica D*, 42: 293–306.

Greening, D.R. and F. Darema. (1989). Rectangular Spatial Decomposition Methods for Parallel Simulated Annealing. In *Proceedings of the International Conference on Supercomputing*, pages 295–302.

Grefenstette, J. (1981). Parallel Adaptive Algorithms for Function Optimization. Technical Report CS-81-19, Vanderbilt University, Nashville.

Grosso, P. (1985). *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model.* PhD thesis, University of Michigan.

Hauser, R. and R. Männer. (1994). Implementation of Standard Genetic Algorithm on MIMD Machines. In Y. Davidor, H.-P. Schwefel and R Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 504–514. Springer-Verlag, Berlin.

Herdy, M. (1992). Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 207–217. North-Holland, Amsterdam.

Hoffmeister, F. (1991). Scalable Parallelism by Evolutionary Algorithms. In M. Grauer and D.B. Pressmar, editors, *Lecture Notes in Mathematical Systems and Economics.* Springer-Verlag, Berlin.

Hogg, T. and C. Williams. (1993). Solving the Really Hard Problems with Cooperative Search. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI93)*, pages 231–236. AAAI Press.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI.

Holland, J.H. (1993). Royal Road Functions. *Genetic Algorithm Digest*, 7(22).

Holmqvist, K., A. Migdalas and P.M. Pardalos. (1997). Parallelized Heuristics for Combinatorial Search. In A. Migdalas, P.M. Pardalos and S. Storoy, editors, *Parallel Computing in Optimization*, pages 269–294. Kluwer, Norwell, MA.

Hong, C-E. and B. M. McMillin. (1992). Relaxing Synchronization in Distributed Simulated Annealing. Report C.Sc. 92-06, University of Missouri at Rolla.

Huntley, C.L. and D.E. Brown. (1996). Parallel Genetic Algorithms with Local Search. *Computers & Operations Research*, 23(6):559–571.

Husbands, P. and F. Mill. (1991). Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, San Mateo, CA.

Jayaraman, R. and F. Darema. (1988). Error Tolerance in Parallel Simulated Techniques. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-88*, pages 545–548. IEEE Computer Society Press, Washington, DC.

Jayaraman, R. and Rutenbar R.A. (1987). Floorplanning by Annealing on Hypercube Multiprocessor. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-87*, pages 346–349. IEEE Computer Society Press, Washington, DC.

Jayewardena, C. (1990). Parallel Simulated Annealing: Image Reconstruction on a Network of Transputers. Technical Report 90-014, University of Bern.

Jeong, C.-S. and M.-H. Kim. (1990). Parallel Algorithm for the TSP on SIMD Machines Using Simulated Annealing. In *Proceedings of the International Conference on Application Specific Array Processors*, pages 712–721.

Jeong, C.-S. and M.-H. Kim. (1991). Fast Parallel Simulated Annealing Algorithm for TSP on SIMD Machines with Linear Interconnections. *Parallel Computing*, 17:221–228.

Jog, P. (1989). *Parallelization of Probabilistic Sequential Search Algorithms*. PhD thesis, Indiana University, Bloomington.

Jog, P. and D. Gucht. (1987). Parallelization of Probabilistic Sequential Search Algorithms. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 170–176. Lawrence Erlbaum Associates, Hillsdale, NJ.

Jog, P., J.Y. Suh and D.V. Gucht. (1991). Parallel Genetic Algorithms Applied to the Traveling Salesman Problem. *SIAM Journal of Optimization*, 1(4):515–529.

Jones, M. and P. Banerjee. (1987). Performance of a Parallel Algorithm for Standard Cell Placement on the Intel Hypercube. In *Proceedings of the 24th Design Automation and Machine Conference*, pages 768–778.

Kephart, J.O., T. Hogg and B.A. Huberman. (1989). Dynamics of Computational Ecosystems: Implication for DAI. *Distributed Artificial Intelligence*, 2.

Kephart, J.O., T. Hogg and B.A. Huberman. (1990). Collective Behavior of Predictive Agents. *Physica D*, 42:48–65.

Kim, Y. and M. Kim. (1990). A Step-Wise-Overlapped Parallel Annealing Algorithm on a Message-Passing Multiprocessor System. *Concurrency: Practice & Experience*, 2(2):123–148.

Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi. (1983). Optimization by Simulated Annealing. *Science*, 220:671–680.

Knight, R. L. and R.L. Wainwright. (1992). HYPERGEN - A Distributed Genetic Algorithm on a Hypercube. In *Proceedings of the 1992 IEEE Scalable High Performance Computing Conference*, pages 232–235. IEEE Computer Society Press, Los Alamitos, CA.

Kommu, V. and I. Pomeranz. (1992). Effect of Communication in a Parallel Genetic Algorithm. In *Proceedings of the 1992 International Conference on Parallel Processing*, volume III, Algorithms and Applications, pages 310–317. CRC Press.

Kosak, C., J. Marks and S. Shieber. (1991). A Parallel Genetic Algorithm for Network-Diagram Layout. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 458–465. Morgan Kaufmann, San Mateo, CA.

Kraft, P., M. Nölle, G., Schreiber, S. Marshall and H. Hans Burkhardt. (1995). A Parallel Genetic Algorithm for Optimizing Morphological Filters on Inhomogeneous Workstation Clusters. In S. Miguet and A. Montanvert, editors, *Proceedings of the Fourth International Workshop on Parallel Image Analysis (IWPIA'95)*, pages 171–182. LIP-ENS, Lyon, France.

Kravitz, R. and R. Rutenbar. (1986). Multiprocessor-Based Placement by Simulated Annealing. In *Proceedings of the 23th ACM/IEEE Design Automation Conference*, pages 567–573.

Kravitz, S.A. and R. Rutenbar. (1987). Placement by Simulated Annealing on a Multiprocessor. *IEEE Transactions on Computer-Aided Design*, 6:534–549.

Kröger, B., P. Schwenderling and O. Vomberger. (1991). Parallel Genetic Packing of Rectangles. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 160–164. Springer-Verlag, Berlin.

Laarhoven, P. and E. Aarts. (1987). *Simulated Annealing: Theory and Applications.* Reidel, Dordrecht.

von Laszewski, G. and H. Mühlenbein. (1991). Partitioning a Graph with a Parallel Genetic Algorithm. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, of Lecture Notes in Computer Science 496*, pages 165–169. Springer-Verlag, Berlin.

Laursen, P.S. (1994). Problem-Independent Parallel Simulated Annealing Using Selection and Migration. In Y. Davidor, H.-P. Schwefel and R Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 408–417. Springer-Verlag, Berlin.

Laursen, P.S. (1996). Parallel Heuristic Search – Introductions and a New Approach. In A. Ferreira and P.M. Pardalos, editors, *Solving Combinatorial Optimization Problems in Parallel, Lecture Notes in Computer Science 1054*, pages 248–274. Springer-Verlag, Berlin.

Lee, C.Y. and S.J. Kim. (1995). Parallel Genetic Algorithms for the Earliness-Tardiness Job Scheduling Problem with General Penalty Weights. *Computers & Industrial Engineering*, 28:231–243.

Lee, F.-H. A. (1995). *Parallel Simulated Annealing on a Message-Passing Multi-Computer.* PhD thesis, Utah State University.

Lee, K-G. and S-Y. Lee. (1992a). Efficient Parallelization of Simulated Annealing Using Multiple Markov Chains: An Application to Graph Partitioning. In T.N. Mudge, editor, *Proceedings of the International Conference on Parallel Processing*, pages 177–180. CRC Press.

Lee, S.-Y. and K.-G. Lee. (1992b). Asynchronous Communication of Multiple Markov Chains in Parallel Simulated Annealing. In T.N. Mudge, editor, *Proceedings of the International Conence on Parallel Processing*, volume III: Algorithms and Applications, pages 169–176. CRC Press, Boca Raton, FL.

Levine, D. (1996). A Parallel Genetic Algorithm for the Set Partitioning Problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 23–35. Kluwer, Norwell, MA.

Lin, F.-T., C.-Y. Kao and C.-C. Hsu. (1991). Incorporating Genetic Algorithms into Simulated Annealing. In *Proceedings of the Fourth International Symposium on AI*, pages 290–297.

Lin, S.-C., W. Punch and E. Goodman. (1994). Coarse-Grain Parallel Genetic Algorithms: Categorization and New Approach. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, pages 28–37. IEEE Computer Society Press.

Lis, J. (1996). Parallel Genetic Algorithm with the Dynamic Control Parameter. In *IEEE 1996 International Conference on Evolutionary Computation*, pages 324–328.

Mahfoud, S. W. and D. E. Goldberg. (1995). Parallel Recombinative Simulated Annealing: A Genetic Algorithm. *Parallel Computing*, 21:1–28.

Mahfoud, S.W. (1995). A Comparison of Parallel and Sequential Niching Methods. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143. Morgan Kaufmann, San Mateo, CA.

Malek, M., M Guruswamy, M. Pandya and H. Owens. (1989). Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 21:59–84.

Manderick, B. and P. Spiessens. (1989). Fine-Grained Parallel Genetic Algorithm. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–421. Morgan Kaufmann, San Mateo, CA.

Maniezzo, V. (1993). Genetic Evolution of the Topology and Weight Distribution of Neural Networks. *IEEE Transactions on Neural Networks*, 5(1):39–53.

Martins, S.L., C.C. Ribeiro and N.R. Rodriguez. (1996). Parallel Programming Tools for Distributed Memory Environments. Monografias em Ciência da Computação 01/96, Pontifícia Universidade Católica de Rio de Janeiro.

Maruyama, T., T. Hirose and A. Konagaya. (1993). A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel Systems. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 184–190. Morgan Kaufmann, San Mateo, CA.

Maruyama, T., A. Konagaya and K. Konishi. (1992). An Asynchronous Fine-Grained Parallel Genetic Algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 563–572. North-Holland, Amsterdam.

Mazza, C. (1992). Parallel Simulated Annealing. *RSA: Random Structures & Algorithms*, 3(2):139–148.

Merkle, L.D. and G.B. Lamont. (1993). Comparison of Parallel Messy Genetic Algorithms Data Distribution Strategies. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 191–198. Morgan Kaufmann, San Mateo, CA.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. (1953). Equation of State Calculation by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin.

Miller, K.R. and R. Shonkwiler. (1992). Genetic Algorithm/Neural Network Synergy for Nonlinearly Constrained Optimization Problems. In *Combinations of Genetic Algorithms and Neural Networks*, pages 248–257. IEEE Computer Society.

Monien, G., F. Ramme and H. Salmen. (1995). A Parallel Simulated Annealing Algorithm for Generating 3D Layouts of Undirected Graphs. *Lecture Notes in Computer Science 1027*, pages 396–408.

Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Publication Report 790, Caltech Concurrent Computation Program.

Moscato, P. and M.G. Norman. (1992). A 'Memetic' Approach for the Traveling Salesman Problem. Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 187–194. IOS Press, Amsterdam.

Mühlenbein, H. (1989). Parallel Genetic Algorithms, Population Genetics, and Combinatorial Optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 416–422. Morgan Kaufmann, San Mateo, CA.

Mühlenbein, H. (1991). Evolution in Time and Space - The Parallel Genetic Algorithm. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithm & Classifier Systems*, pages 316–338. Morgan Kaufman, San Mateo, CA.

Mühlenbein, H. (1991a). Asynchronous Parallel Search by the Parallel Genetic Algorithm. In V. Ramachandran, editor, *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 526–533. IEEE Computer Society Press, Los Alamitos, CA.

Mühlenbein, H. (1992). Parallel Genetic Algorithms in Combinatorial Optimization. In O. Balci, R. Sharda and S. Zenios, editors, *Computer Science and Operations Research*, pages 441–456. Pergamon Press, New York, NY.

Mühlenbein, H. (1992a). How Genetic Algorithms Really Work: Mutation and Hill-Climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 15–26. North-Holland, Amsterdam.

Mühlenbein, H., M. Gorges-Schleuter and O. Krämer. (1987). New Solutions to the Mapping Problem of Parallel Systems - the Evolution Approach. *Parallel Computing*, 6:269–279.

Mühlenbein, H., M Gorges-Schleuter and O Krämer. (1988). Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7(1):65–85.

Mühlenbein, H., M. Schomisch and J. Born. (1991a). The Parallel Genetic Algorithm as Function Optimizer. *Parallel Computing*, 17(6-7):619–632.

Mühlenbein, H., M. Schomisch and J. Born. (1991b). The Parallel Genetic Algorithm as Function Optimizer. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA.

Munetomo, M., Y. Takai and Y. Sato. (1993). An Efficient Migration Scheme for Subpopulation-Based Asynchronous Parallel Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 649–658. Morgan Kaufmann, San Mateo, CA.

Muntean, T. and E-G. Talbi. (1991). A Parallel Genetic Algorithm for Process-Processor Mapping. In M. Durand and F. El-Dabagh, editors, *2nd Symposium on High Performance Computing*, pages 71–82. North-Holland, Amsterdam.

Nabhan, T.M. and A.Y. Zomaya. (1995). A Parallel Simulated Annealing Algorithm with Low Communication Overhead. *IEEE Transactions on Parallel and Distributed Systems*, 6(12):1226–1233.

Natarajan, K.S. and S. Kirkpatrick. (1989). Evaluation of Parallel Placement by Simulated Annealing: Part I - the Decomposition Approach. Research Report RC15246, IBM T.J. Watson Research Center.

Natarajan, K.S. and S. Kirkpatrick. (1989). Evaluation of Parallel Placement by Simulated Annealing: Part II - the FLAT Approach. Research Report RC15247, IBM T.J. Watson Research Center.

Osman, I.H. and J.P. Kelly, editors. (1996). *Meta-Heuristics: Theory & Applications.* Kluwer, Norwell, MA.

Pardalos, P.M., L. Pitsoulis, T. Mavridou and M.G.C. Resende. (1995). Parallel Search for Combinatorial Optimization: Genetic Algorithms, Simulated Annealing, Tabu Search and GRASP. In A. Ferreira and J. Rolim, editors, *Proceedings of Workshop on Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Computer Science 980*, pages 317–331. Springer-Verlag, Berlin.

Pardalos, P.M., L. Pitsoulis and M.G.C. Resende. (1995). A Parallel GRASP Implementation for the Quadratic Assignment Problem. In A. Ferreira and J. Rolim, editors, *Solving Irregular Problems in Parallel: State of the Art.*

Pettey, C.B. and M.R. Leuze. (1989). A Theoretical Investigation of a Parallel Genetic Algorithm. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 398–405. Morgan Kaufmann, San Mateo, CA.

Pettey, C.B., M.R. Leuze and J.J. Grefenstette. (1987). A Parallel Genetic Algorithm. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 155–161. Lawrence Erlbaum Associates, Hillsdale, NJ.

Porto, S.C.S., J.P.F.W. Kitajima and C.C. Ribeiro. (1996). Performance Evaluation of a Parallel Tabu Search Task Scheduling Algorithm. Monografias em Ciência da Computação 35/96, Pontifícia Universidade Católica de Rio de Janeiro.

Porto, S.C.S. and C.C. Ribeiro. (1995). A Tabu Search Approach to Task Scheduling on Heteregenous Processors Under Precedence Constraints. *International Journal of High-Speed Computing*, 7:207–225.

Porto, S.C.S. and C.C. Ribeiro. (1996). Parallel Tabu Search Message-Passing Synchronous Strategies for Task Scheduling Under Precedence Constraints. *Journal of Heuristics*, 1(2):207–223.

Ram, D.J., T.H. Sreenivas and K.G. Subramaniam. (1996). Parallel Simulated Annealing Algorithms. *Journal of Parallel and Distributed Computing*, 37:207–212.

Rego, C. and C. Roucairol. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the VRP. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 253–295. Kluwer, Norwell, MA.

Rochat, Y. and É.D. Taillard. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1(1):147–167.

Rose, J.S., D.R. Blythe, W.M. Snelgrove and Z.G. Vranesic. (1986). Fast, High Quality VLSI Placement on an MIMD Multiprocessor. In *Proceedings of the IEEE Inter-*

national Conference on Computer-Aided Design: ICCAD-86, pages 42–45. IEEE Computer Society Press, Washington, DC.

Rose, J.S., W.M. Snelgrove and Z.G. Vranesic. (1990). Parallel Standard Cell Placement Algorithms with Quality Equivalent to Simulated Annealing. *IEEE Transactions on Computer-Aided Design*, 9:253–259.

Roussel-Ragot, P. and G. Dreyfus. (1990). A Problem-Independent Parallel Implementation of Simulated Annealing: Models and Experiments. *IEEE Transactions on Computer-Aided Design*, 9(8):827–835.

Roussel-Ragot, P. and G. Dreyfus. (1992). Parallel Annealing by Multiple Trials: An Experimental Study on a Transputer Network. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 91–108. John Wiley & Sons, New York, NY.

Roussel-Ragot, P., N. Kouicem and G. Dreyfus. (1990). Error-Free Parallel Implementation of Simulated Annealing. In H.-P. Schwefel and R. Männer, editors, *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, pages 231–241. Springer-Verlag, Berlin.

Rutenbar, R. and S. Kravitz. (1986). Layout by Annealing in a Parallel Environment. In *Proceedings of the IEEE International Conference on Computer-Aided Design: ICCAD-86*, pages 434–437. IEEE Computer Society Press, Washington, DC.

Sannier, A. and E. Goodman. (1987). Genetic Learning Procedures in Distributed Environments. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 162–169. Lawrence Erlbaum Associates, Hillsdale, NJ.

Sargent, J.S. (1988). *A Parallel Row-Based Algorithm with Error Control for Standard-Cell Placement on a Hypercube Multiprocessor.* PhD thesis, University of Illinois, Urbana-Illinois.

Schlierkamp-Voosen, D. and H. Mühlenbein. (1994). Strategy Adaptation by Competing Subpopulations. In Y. Davidor, H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 199–208. Springer-Verlag, Berlin.

Schnecke, V. and O. Vornberger. (1996). An Adaptive Parallel Genetic Algorithm for VLSI-Layout Optimization. In Y. Davidor, H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science 866*, pages 859–868. Springer-Verlag, Berlin.

Schwefel, H.-P. and R. Männer, editors. (1991). *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496.* Springer-Verlag, Berlin.

Schwehm, M. (1992). Implementation of Genetic Algorithms on Various Interconnection Networks. In M. Valero, E. Onate, M. Jane, J.L Larriba and B. Suarez, editors, *Parallel Computing and Transputers Applications*, pages 195–203. Amsterdam, IOS Press.

Seredynski, F. (1994). Dynamic Mapping and Load Balancing with Parallel Genetic Algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 834–839.

Shapiro, B. and J. Navetta. (1994). A Massively Parallel Genetic Algorithm for RNA Secondary Structure Prediction. *The Journal of Supercomputing*, 8(3):195–207.

Shonkwiler, R. (1993). Parallel Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 199–205. Morgan Kaufmann, San Mateo, CA.

Sirag, D.J. and P.T. Weisser. (1987). Toward a Unified Thermodynamic Genetic Operator. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 116–122. Lawrence Erlbaum Associates, Hillsdale, NJ.

Spiessens, P. and B. Manderick. (1990). A Genetic Algorithm for Massively Parallel Computers. In R. Eckmiller, G. Hartmann and G. Hauske, editors, *Parallel Processing in Neural Systems and Computers*, pages 31–36. North-Holland, Amsterdam.

Spiessens, P. and B. Manderick. (1991). A Massively Parallel Genetic Algorithm: Implementation and First Analysis. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 279–285. Morgan Kaufmann, San Mateo, CA.

Stanley, T.J. and T. Mudge. (1995). A Parallel Genetic Algorithm for Multi-objective Microprocessor Design. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 597–604. Morgan Kaufmann, San Mateo, CA.

Starkweather, T., D. Whitley and K. Mathias. (1991). Optimization Using Distributed Genetic Algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature, Lecture Notes in Computer Science 496*, pages 176–185. Springer-Verlag, Berlin.

Taillard, É.D. (1991). Robust Taboo Search for the Quadratic Assignment Problem. *Parallel Computing*, 17:443–455.

Taillard, É.D. (1993). Parallel Iterative Search Methods for Vehicle Routing Problems. *Networks*, 23:661–673.

Taillard, É.D. (1993a). *Recherches itératives dirigées parallèles.* PhD thesis, École Polytechnique Fédérale de Lausanne.

Taillard, É.D. (1994). Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing*, 6(2):108–117.

Taillard, É.D., P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31:170–186.

Talbi, E-G. and P. Bessière. (1991). Un algorithme génétique parallèle pour le problème de partitionnement de graphes. Technical Report RR-845-I, Institut IMAG.

Talbi, E-G. and P. Bessière. (1991a). A Parallel Genetic Algorithm for the Graph Partitioning Problem. In *Proceedings of the ACM International Conference on Supercomputing ICS91*, pages 312–320.

Tamaki, H. and Y. Nishikawa. (1992). A Parallel Genetic Algorithm Based on a Neighborhood Model and Its Application to Jobshop Scheduling. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 573–582. North-Holland, Amsterdam.

Tanase, R. (1987). Parallel Genetic Algorithm for a Hypercube. In J.J. Grefenstette, editor, *Proceeding of the Second International Conference on Genetic Algorithms and their Applications*, pages 177–183. Lawrence Erlbaum Associates, Hillsdale, NJ.

Tanase, R. (1989). Distributed Genetic Algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA.

Tanase, R. (1989). *Distributed Genetic Algorithms for Function Minimization.* PhD thesis, University of Michigan.

Toulouse, M., T.G. Crainic and M. Gendreau. (1996). Communication Issues in Designing Cooperative Multi Thread Parallel Searches. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory & Applications*, pages 501–522. Kluwer, Norwell, MA.

Toulouse, M., T.G. Crainic and M. Gendreau. (1996a). Issues in Designing Parallel and Distributed Search Algorithms for Discrete Optimization Problems. Publication CRT-96-36, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Toulouse, M., T.G. Crainic and B. Sansó. (1997a). An Experimental Study of Systemic Behavior of Cooperative Search Algorithms. Publication CRT-97-26, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Toulouse, M., T.G. Crainic and B. Sansó. (1997b). Systemic Behavior of Cooperative Search Algorithms. Publication CRT-97-55, Centre de recherche sur les transports, Université de Montréal, Montréal, QC, Canada.

Trienekens, H.W.J.M. and A. de Bruin. (1992). Towards a Taxonomy of Parallel Branch and Bound Algorithms. Report EUR-CS-92-01, Department of Computer Science, Erasmus University, Rotterdam.

Varanelli, J.M. and J.P. Cohoon. (1995). Population-Oriented Simulated Annealing: A Genetic/Thermodynamic Hybrid Approach to Optimization. In L.J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 174–181. Morgan Kaufmann, San Mateo, CA.

Verhoeven, M.G.A. and E.H.L Aarts. (1995). Parallel Local Search. *Journal of Heuristics*, 1(1):43–65.

Virot, B. (1992). Parallel Annealing by Multiple Trials: Experimental Study of a Chip Placement Problem Using a Sequent Machine. In R. Azencott, editor, *Simulated Annealing Parallelization Techniques*, pages 109–127. John Wiley & Sons, New York, NY.

Voigt, H.-M. and J. Born. (1990). A Structured Distributed Genetic Algorithm for Function Optimization. In M. Grauer and D.B. Pressmar, editors, *Parallel Computing and Mathematical Optimization*, pages 199–208. Springer-Verlag, Berlin.

Voigt, H.-M., I. Santibáñez Koref and J. Born. (1992). Hierarchical Structured Distributed Genetic Algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 155–164. North-Holland, Amsterdam.

Vose, M.D. and G.E. Liepins. (1991). Punctuated Equilibria in Genetic Search. *Complex Systems*, 5:31–44.

Voß, S. (1993). Tabu Search: Applications and Prospects. In D.-Z. Du and P.M. Pardalos, editors, *Network Optimization Problems*, pages 333–353. World Scientific Publishing Co., Singapore.

Whitley, D. (1993). Cellular Genetic Algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 658–658. Morgan Kaufmann, San Mateo, CA.

Whitley, D. (1995). Applications of Modern Heuristic Methods. In V.J. Raynard-Smith, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 55–67. Alfred Waller.

Whitley, D. and T. Starkweather. (1990a). Optimizing Small Neural Networks Using a Distributed Genetic Algorithm. In *Proceedings of the International Conference on Neural Networks*, pages 206–209. IEEE Press.

Whitley, D. and T. Starkweather. (1990b). GENITOR II: A Distributed Genetic Algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(3):189–214.

Whitley, L.D. (1994). A Genetic Algorithm Tutorial. *Statistics and Computing*, 4:65–85.

Witte, E.E., R.D. Chamberlain and M.A. Franklin. (1990). Parallel Simulated Annealing using Speculative Computation. In *Proceedings of the 19th International Conference on Parallel Processing*, pages 286–290.

Witte, E.E., R.D. Chamberlain and M.A. Franklin. (1991). Parallel Simulated Annealing using Speculative Computation. *IEEE Transactions on Parallel & Distributed Systems*, 2(4):483–494.

Wong, C.-P. and R.-D. Fiebrich. (1987). Simulated Annealing-Based Circuit Placement on the Connection Machine System. In *Proceedings of the IEEE International Conference on Computer Design; ICCD'87*, pages 78–82. IEEE Computer Society Press, Washington, DC.