Comput. & Ops Res. Vol. 10, No. 2, pp. 63–211, 1983
Printed in Great Britain.

Pergamon Press Ltd.

CONTENTS

SPECIAL ISSUE

ROUTING AND SCHEDULING OF VEHICLES AND CREWS

The State of the Art

Preface 65
Abstract 67
Scope and Purpose 67

CHAPTER I. INTRODUCTION 69
1.1. Background
1.2. Classification of Routing and Scheduling Problems
1.3. Complexity and Computational Burden of Routing and Scheduling Problems
1.4. Set Partitioning and Set Covering Problems
1.5. Organization of This Report

CHAPTER II. ROUTING OF VEHICLES 79
2.1. The Traveling Salesman Problem (TSP)—Introduction
2.2. TSP—Optimal Approaches
2.3. TSP—Heuristic Approaches
2.4. Single Depot/Multiple Vehicle Routing
2.5. Selected Solution Techniques for Single Depot Vehicle Routing Problems
2.6. Multiple-Depot Vehicle Routing
2.7. The Fleet Size and Mix Problem
2.8. Stochastic Vehicle Routing
2.9. The Chinese Postman Problem
2.10 Capacitated Arc Routing Problems

CHAPTER III. VEHICLE AND CREW SCHEDULING PROBLEMS 117
3.1. Introduction
3.2. Vehicle Scheduling
3.3. Crew Scheduling
3.4. Vehicle Scheduling Algorithms
3.5. Crew Scheduling Algorithms

CHAPTER IV. COMBINED ROUTING AND SCHEDULING 149
4.1. Introduction
4.2. Description of Various Routing and Scheduling Problems
4.3. An Algorithm for the School Bus Routing and Scheduling Problem
4.4. Algorithms for Solving the Tractor Trailor Routing and Scheduling Problems with Full Loads
4.5. An Algorithm for the Tractor Trailer Routing and Scheduling Problem with Partial Loads
4.6. A Procedure for the Street Sweeper Routing and Scheduling Problem
4.7. An Algorithm for the Airplane Scheduling Problem
4.8. Static Dial-a-Ride Problems without Time Windows
4.9. Static Dial-a-Ride Routing and Scheduling Problems with Time Windows
4.10. Dial-a-Ride Demand Responsive Algorithms

CHAPTER V. IMPLEMENTATION ISSUES 181
5.1. Data Requirements
5.2. Interactive Computer Systems
5.3. Reliability Considerations
5.4. Databases for Routing and Scheduling
5.5. Organizational Acceptance of Results
5.6. Choosing the Right Model

CHAPTER VI. CONCLUSIONS AND FUTURE DIRECTIONS 189
6.1. Future Development and Use of Routing and Scheduling Systems
6.2. Future Research Directions in Modeling and Algorithmic Development
Acknowledgements

References 195

PREFACE

This is the first time in its ten year history that this journal has devoted an entire issue to a single paper. This does not represent a change in editorial policy, but rather an exercise of editorial flexibility to take advantage of a very unusual opportunity.

Several times in the past we have devoted issues to a single subject. These have been special issues with a guest editor who has selected, and sometimes even commissioned, the articles so that the assembled issue was coherent and complete. In this case, however, the issue is a single state-of-the-art paper prepared by four authors.

The subject of this issue is a very important component of the management of distribution systems. The authors note that distribution costs add about $400 billion per year to the cost of purchased goods in the U.S. alone. This is one measure of the importance of optimization in routing and scheduling. In addition, we must recognize the significance of this optimization in other areas, such as garbage collection and movement of people to work and to schools, which are not included in the $400 billion. Further, in an increasingly technological society which must take advantage of economies of scale, the relative importance of transportation, and hence of routing and scheduling, will continue to grow.

The authors have done a truly extraordinary job in terms of clarity of presentation as well as completeness of subject matter, and it is my opinion that this issue of CAOR will be of great and lasting value to our readers.

Lawrence D. Bodin, a member of the editorial advisory board of this journal, is a Professor of Management Science and Statistics at The University of Maryland. His primary research interests, reflected in his numerous research grants, include large-scale modeling, network analysis, applications of operations research to transportation systems, and routing and scheduling of vehicles and crews for mass transit and dial-a-ride systems. He has published numerous papers on operations research and transportation. He is also Associate Editor of *Operations Research.*

Bruce L. Golden is Chairman of the Department of Management Science and Statistics at The University of Maryland. His research interests include network optimization, mathematical programming, and applied statistics. He has published numerous articles in these fields. He is currently an Associate Editor of *Networks*, a member of ORSA's Long Range Planning Committee, Chairman of ORSA's Transportation Science Dissertation Prize Committee, and a Council member of both the Transportation Science and the Computer Science Sections of ORSA. Dr. Golden received his B.A. from the University of Pennsylvania in Mathematics. Later he earned his M.S. and Ph.D. in Operations Research from M.I.T.

Arjang A. Assad is an Assistant Professor in the Department of Management Science and Statistics at The University of Maryland. His interests include operations management, mathematical programming, and network modeling of transportation/distribution systems and he has conducted applied research projects with a number of firms having major distribution activities. Dr. Assad has also done consulting work for the U.S. Departments of Transportation and Energy. He received his Ph.D. in Management Science from M.I.T.

Michael O. Ball is an Assistant Professor of Management Science and Statistics at The University of Maryland. He is interested in urban mass transit systems, distribution systems, and distributed information systems. His publications focus on network reliability analysis and network optimization. Dr. Ball has been principal investigator on a U.S. Department of Transportation grant involving mass transit crew scheduling and has consulted with corporations on the analysis of distribution systems.

I take this opportunity to thank the authors on behalf of the readers of *C&OR* who will surely benefit greatly from this outstanding work. Although it is not our normal practice to formally thank our reviewers in print, the magnitude of the task involved in this case warrants an acknowledgement. Accordingly, I thank the following people for their valuable comments, suggestions, and judgements: Jatinder Gupta of The U.S. Dept. of Energy, Marshall Fisher of The Wharton School, E. Beltrami of Stonybrook, Edward Zielinski of Edward Don & Co., Dan King of Coca-Cola Ltd., Toronto, Victor Richard of Texas A&M, Gilbert Laporte of L'ecole

Des Hautes Etudes Commerciales, Montreal, James H. Bookbinder of The Toronto Transit Commission, Bruce W. Lamar of Arthur D. Little, Inc., Hoon-Liong Ong of The University of Waterloo, Ontario, Alexander Rinnooy Kan of Erasmus University, Rotterdam, Edward Baker of the University of Miami, Jacques Ferland and Jean-Marc Rousseau of the University of Montreal, Nicos Christofides of the Imperial College, London, and Harilaos Psaraftis of M.I.T.

SAMUEL RAFF
August 1982

ABSTRACT

In this paper, we describe the state-of-the art in the routing and scheduling of vehicles and crews. We discuss the applied setting in which these routing and scheduling problems arise, present mathematical formulations for them, and we survey representative solution algorithms. We also discuss issues relating to real-world computer implementation of these solution algorithms.

SCOPE AND PURPOSE

Private firms that undertake the distribution of their goods to customer locations, and public transportation authorities responsible for the provision of transportation services to users both rely upon a fleet of vehicles and associated crews. The effective management of these vehicles and crews gives rise to a variety of problems generally subsumed under the heading of "routing and scheduling problems". Given demands for service at various points in a transportation network over which the vehicles may travel, decisions concerning the spatial configuration of vehicle movements are classified as "routing problems". These problems usually involve the specification of a sequence of locations that a vehicle must visit. The celebrated traveling salesman and vehicle routing problems are two examples of routing problems. If explicit consideration is given to the times at which various locations are visited, one is faced with a "scheduling problem". In many cases, the spatial and temporal decisions interact heavily and result in "combined routing and scheduling problems".

The area of routing and scheduling for vehicles and crews has been the focus of intensive recent research activity. Major advances in this area have been made along both theoretical and applied dimensions. New formulations and efficient solution techniques have been proposed while, simultaneously, an extensive list of applications has been identified by practitioners. School-bus routing and scheduling, design of dial-a-ride systems, mass transit scheduling of vehicles and crews, pick-up and delivery distribution systems, etc. are only a few of the many applications arising in practice. The burst of research and applications-oriented activities in this area make this an opportune moment for a state-of-the-art survey to appear.

This report aims to provide a comprehensive survey of routing and scheduling problems. Its objectives include a classification and categorization of routing and scheduling problems, a review of algorithmic techniques and solution methodologies, and an overview of the major applications and known implementations. A special emphasis is placed on the interplay between methodological advances in the area and the practical implications of such advances. A comprehensive bibliography of 700 references is provided to facilitate further investigation for the interested reader.

CHAPTER 1
INTRODUCTION

1.1. BACKGROUND

Logistics may be defined as "the provision of goods and services from a supply point to a demand point"[201]. A complete logistics system covers the entire process of moving raw materials and input requirements from suppliers to plants, the conversion of the inputs into products at certain plants, the movement of the products to various warehouses or depots, and the eventual delivery of these products to the final customers. The distribution activities of a firm comprise all movements and storage of goods "downstream" from the plants. The last step in these movements (from distribution centers to customers), which may be called local transportation or delivery, is the most costly link of the distribution chain[135]. For this step to be carried out effectively, the firm must approach the planning and execution of its transportation activities in a rational manner in full view of the economies involved.

Effective distribution management presents a variety of decision-making problems at all three levels of strategic, tactical, and operational planning. Decisions relating to the location of facilities (plants, warehouses, or depots) may be viewed as strategic, while the problems of fleet size and mix determination could be termed tactical. Finally, on the operational level, various decisions concerning the routing and scheduling of vehicles and the staffing of such vehicles with crews require ongoing attention on a day-to-day basis. Clearly, the distinction between strategic, tactical, and operational planning should not be interpreted too rigidly, especially in view of the close interaction between the decisions involved. For example, the location of plants, warehouses, and fleet depots has a major impact on the distribution activities of a firm. Generally, the locations of all facilities are required as input data for planning the local transportation activities. Conversely, such siting decisions rely upon distribution or transportation costs between various geographic locations. The resulting combined production/inventory/distribution problem is shared by many firms in the private sector. The work by Geoffrion and Graves[268] is a good instance of a modeling effort that integrates the various aspects of the combined problem just described. For private firms, a detailed study of the distribution problem involves an explicit consideration of vehicle routing and fleet size and mix issues. These issues, in turn, are analyzed in conjunction with a plan for routing and scheduling the vehicles assigned to the distribution function. Thus, whereas highly aggregate location models traditionally employ rough estimates of the point-to-point transportation costs, location decisions for a firm with an in-house fleet must incorporate the level of detail associated with the economies of customer deliveries or collections. At this level the need for effective and flexible planning tools for routing and scheduling activities becomes quite evident.

In addition to the location of depots, effective planning of deliveries generally requires inputs concerning a variety of other "exogenous" decisions which include:

—districting the size of the area served out of each depot
—fleet size and mix available at each depot
—allocation of delivery activities between the in-house fleet and an outside common carrier
—customer service levels: frequency of deliveries to each customer.

Given the decisions listed above, the firm may then route and schedule its vehicles to perform the assigned functions at minimal cost. This step requires an optimum-seeking algorithm to identify the best configuration of routes and schedules which brings us to the main focus of this report. The main goal of the preceding discussion is to emphasize the impact of various "higher level" decisions on the final design of a routing and scheduling system. Furthermore, it should be remarked that recent advances in routing and scheduling procedures

now hold the promise of integrating the planning of outbound transportation with some of the higher-level decisions mentioned before. For example, Christofides[135] describes a successful integration of delivery decisions with issues relating to customer service and fleet size determination. Fisher *et al.*[221] discuss another successful implemention where the outbound transportation and inventory management functions are integrated.

The issues raised in the preceding discussion are in no way limited to the private sector. For example, the location/distribution problem arises in various public services. In solid waste collection, communities must decide on the number and locations of disposal facilities and determine how the refuse ought to be transported to such locations. Similarly, in mass transit systems, one must determine the locations of garages to house buses so as to allow for a cost-effective servicing of existing bus lines by the fleet of vehicles. Similar issues arise in the location of emergency units, e.g. fire or police stations in a city (see Larson and Odoni[421] and Beltrami[62]).

The importance of distribution problems is evident from the magnitude of the associated distribution costs. Surveys by Kearny[379] show that physical distribution costs account for about 16% of the sales value of an item. Of this, about a fourth is due to downstream distribution of the final product from distribution centers to customers. The same study estimates the annual distribution costs at approximately $400 billion in the United States and some £15 billion in the United Kingdom. The following examples should impart some flavor of the distribution and transportation-related costs in both private and public sectors.

—Fisher and Jaikumar[220] estimate deliveries by trucks during 1975 to account for 69 billion miles of travel and some $5.7 billion worth of fuel.

—In 1979, the transportation costs for a major U.S. pharmaceutical company totalled over $1.2 million a month.

—The annual distribution costs of one of the twenty largest distributors of propane gas in the United States exceeded $6 million in 1979.

—The monthly operating cost for the aircraft and crews of a major freight airline was over $2.5 million in 1979. Large passenger-carrying airlines in the U.S. have comparable operating costs.

—In the mid-1970's, the annual budget of New York City's Department of Sanitation was about $200 million of which roughly 80% covered the wages and benefits of some 11,000 sanitation workers. The major activity of this department was refuse collection.

—In 1974, New York State budgeted between 150 and 200 million dollars annually for school bus transportation. This budget was supplemented by funds for school transportation from local school districts. In 1974, the annual cost of leasing a school bus for 6 hours/day was roughly $11,000. This figure has increased to about $20,000 for 1981.

As these examples indicate, the costs associated with operating vehicles and crews for delivery purposes form an important component of total distribution costs. Consequently small percentage savings in these expenses could result in substantial total savings over a number of years. The significance of detecting these potential savings has become increasingly apparent due to escalating fuel costs, higher capital costs for the replacement of vehicles, growing salaries for crews, etc. These factors have caused a larger percentage of the total operating costs of an organization to be devoted to routing and scheduling activities. The use of analytic routing and scheduling models and techniques can be instrumental in realizing the savings alluded to before. When coupled with an effective management information system, the routing and scheduling methodology can assume a crucial role in the operational planning of distribution activities. Furthermore, these methods can be used as tactical planning tools, for instance, in the evaluation of how future demand patterns would impact the delivery system and the proposed fleet composition. This allows one to plan for potential capital savings by restructuring the availability of various resources.

Although cost minimization is the primary objective of most routing and scheduling problems, other objectives may assume primary importance especially in the context of service operations in the public sector. Safety and convenience are two other objectives that certain problems may focus on. For example, in school bus routing and scheduling, the objective is to minimize the total number of student-minutes on the bus since this measure is perceived to be highly correlated with safety. Consequently, school bus routes and schedules must be designed

to ensure that no student stays on the bus for an excessive length of time. Given this objective, it may be desirable to service the stops with the largest number of students near the end of a route (close to the school), even if the resulting routes turn out to be relatively long and hence costly in terms of crew and vehicle costs. Another example where safety is a major consideration concerns the routing and scheduling of street sweepers and vehicles for household refuse collection (see Section 4.4, Bodin and Kursh[88] and McBride[474]). Here, one attempts to minimize the number of u-turns or left-hand turns since such turns are dangerous to make on major arteries. This objective is usually achieved at the expense of increased "deadheading" (defined to be nonproductive travel time) resulting in additional costs. As a result, a set of routes and schedules that seem to be attractive when the safety criterion is invoked, may well deviate substantially from the minimal cost configuration of routes in the absence of safety considerations. This deviation from the minimum cost may be viewed as a premium paid for safety. In dial-a-ride services for the elderly or the handicapped, once again the primary objective is to provide convenient service to all users. Consequently an appropriate objective is to minimize the total inconvenience of all users. Measures of inconvenience must then be identified in a quantifiable form to allow the problem to be viewed as an optimization problem. A discussion of such measures may be found in Section 4.9.

In certain routing and scheduling problems, the choice of an appropriate objective function or measure of performance constitutes an important modeling question in its own right. For instance in school bus transportation for the State of New York, the aid formula according to which the state subsidizes school districts complicates the choice of an objective. The state subsidizes up to 90% of most transportation costs while the school district is responsible for developing the routes and schedules of school buses (see Bodin and Berman[82]). Consequently the school district may easily depart from an overall minimal cost solution in order to minimize its own costs possibly at the expense of the state.

To take an example from the private sector, the design of delivery routes and schedules for a commercial firm's distribution activities must take account of inventory costs and customer service levels as well as purely distribution-related costs (crews and vehicles). Generally, these are conflicting objectives as a decrease in distribution costs usually implies an increase in inventory costs and a lower customer service level (see Christofides[135]). An interesting example of using a composite objective function that incorporates both inventory and routing costs may be found in Fisher *et al.*[221].

1.2. CLASSIFICATION OF ROUTING AND SCHEDULING PROBLEMS

The basic output of all routing and scheduling systems is essentially the same: For each vehicle or driver, a route and a schedule is provided. Generally, the route specifies the sequence of locations to be visited and the schedule identifies the times at which the activities at these locations are to be carried out. An example of this output is displayed in Fig. 1.1 for a dial-a-ride subscriber service in Baltimore, Maryland (described more fully in Section 4.9). The information in Fig. 1.1 completely specifies the nature and order of the tasks that have to be performed.

One objective of this paper is to provide a classification of various routing and scheduling problems. These problems are first divided into the three groups: (1) routing, (2) scheduling, and (3) routing and scheduling, and later subdivided according to a more detailed classification scheme.

Since various routing and scheduling systems share the same type of output, one must distinguish between these problems based on other problem characteristics and the assumptions that surround a given problem. Take, for example, the problem of delivering goods to various locations by a fleet of vehicles based at a depot. If there are no *a priori* restrictions on delivery times and if all goods can be delivered within a short period of time (say, three hours), then one may ignore temporal considerations to obtain a pure vehicle routing problem (an example for this scenario may be newspaper delivery). If, however, the times of visits to various locations are of primary importance, as in the case where customers only accept deliveries within a given span of time, then temporal characteristics may no longer be ignored and in fact the time restrictions guide the routing and scheduling activities. The resulting problem must then be attacked by methods that are different from those used for the earlier example. In general, the

Location	Activity	Customer Number	Time	Number on Vehicle
3405 Powhatan Avenue	Pick-up	25	12:00	1
2901 Strickland Street	Deliver	25	12:15	0
861 Park Ave. -Waxter Center	Pick-up	26	13:45	1
5604 Woodmont Avenue	Deliver	26	14:10	0
1111 E. Coldspring Lane	Pick-up	5	14:35	1
1701 Bloomingdale Road	Deliver	5	14:58	0
Levindale Avenue	Pick-up	29	15:15	1
6514 Eberle Drive	Deliver	29	15:27	0
1111 E. Coldspring Lane	Pick-up	12	16:00	1
1818 N. Collington Avenue	Deliver	12	16:12	0
1111 E. Coldspring Lane	Pick-up	51	16:24	1
1111 E. Coldspring Lane	Pick-up	48	16:24	2
1111 E. Coldspring Lane	Pick-up	47	16:24	3
1111 E. Coldspring Lane	Pick-up	50	16:24	4
1111 E. Coldspring Lane	Pick-up	49	16:24	5
1600 Mount Royal Avenue	Deliver	47	16:56	4
301 McMechen Street	Deliver	48	17:00	3
1032 W. Franklin Street	Deliver	50	17:12	2
2013 Madison Avenue	Deliver	51	17:23	1
708 North Gilmore	Deliver	49	17:34	0
1600 Charles-Penn Station	Pick-up	63	17:47	1
500 Calvert Street	Pick-up	34	18:01	2
500 Calvert Street	Pick-up	33	18:01	3
1616 Melby Court	Deliver	63	18:46	2
3006 Pinewood Avenue	Deliver	33	18:56	1
6239 Northwood Drive	Deliver	34	19:10	0

Fig. 1.1. Sample route and schedule for vehicle

characteristics and restrictions associated with various service activities lead to different categories of problems that require different modeling assumptions. The following discussion focuses on certain frequently-encountered problem characteristics.

A routing and scheduling system deals with a collection of entities requiring service. A *precedence relationship* between two such entities states that one of these is to be serviced before the other. For example, in dial-a-ride systems, a passenger must be picked up at one location and delivered to his destination location. The pick-up activity must precede the delivery activity thus establishing a precedence relation between these two activities. An entity requiring service is said to have a *definitive time for service* if the starting and ending times of the service are specified in advance. In a mass transit system, for instance, a timetable provides definitive times for the start and termination of each trip in the timetable, requiring the vehicle to cover the entire trip without interruption. Somewhat less restrictive time constraints may be imposed by *time windows*. A *two-sided window* $[s, t]$ restricts the service time of an entity to fall into a specified interval of time from s to t. For example, a warehouse may only accept deliveries between 9 : 00 AM and 4 : 00 PM. The time window associated with such deliveries is then 9 :00 AM to 4 : 00 PM. Clearly, if the two endpoints of a time window coincide, a definitive time for service will result. Thus an activity that must be carried out at 9 : 00 AM may receive a time window with $s = t = 9 : 00$ AM. A *one-sided* time window is of the form $[-\infty, t]$ or $[s, \infty]$. The first window requires that the service be provided *before* time t and the second restricts the service to occur *after* time s. Some dial-a-ride systems (see Section 4.9) have one-sided time windows.

Table 1.1. Characteristics of routing and scheduling problems.

CHARACTERISTICS	POSSIBLE OPTIONS
1. Size of Available Fleet	one vehicle multiple vehicles
2. Type of Available Fleet	homogeneous (only one vehicle type) heterogeneous (multiple vehicle types) special vehicle types (compartmentalized, etc.)
3. Housing of Vehicles	single depot (domicile) multiple depots
4. Nature of Demands	deterministic (known) demands stochastic demand requirements partial satisfaction of demand allowed
5. Location of Demands	at nodes (not necessarily all) on arcs (" " ") mixed
6. Underlying Network	undirected directed mixed euclidean
7. Vehicle Capacity Restrictions	imposed (all the same) imposed (different vehicle capacities) not imposed (unlimited capacity)
8. Maximum Route Times	imposed (same for all routes) imposed (different for different routes). not imposed
9. Operations	pickups only drop-offs (deliveries) only mixed (pick ups and deliveries) split deliveries (allowed or disallowed)
10. Costs	variable or routing costs fixed operating or vehicle acquisition costs common carrier costs (for unserviced demands)
11. Objectives	minimize total routing costs minimize sum of fixed and variable costs minimize number of vehicles required maximize utility function based on service or convenience. maximize utility function based on customer priorities

If the entities to be serviced have no temporal restrictions and there are no precedence relations among these entities, then we have a *routing* problem. Routing problems form the subject of Chapter 2. If each entity has a definitive service time, then a *scheduling* problem results. Scheduling problems are discussed in Chapter 3. Otherwise, one is dealing with a *combined routing and scheduling* problem. Generally, routing and scheduling problems involve both precedence relations and time windows. This class of problems is studied in Chapter 4.

In addition to the division of problems into the three major classes given above, one may further characterize routing and scheduling problems through a more detailed list of their characteristics. Table 1.1 presents some broad characteristics in which various routing and scheduling problems may differ and is based on a similar enumeration by Bodin and Golden[85].

The entries in Table 1.1 may be used to provide a quick description of a routing or scheduling problem to be studied. Taking different combinations of options within various characteristics on the l.h.s. of Table 1.1 results in a large number of possible problem settings. Two examples, couched in the terminology of the table, should illustrate this classification approach. Consider a problem with a single domicile, a single vehicle of unlimited capacity, deterministic (i.e. known) demands that must all be serviced at nodes of an undirected network, no restriction on route time, and the objective of minimizing routing costs alone in the form of total distance traveled. This problem is the celebrated traveling salesman problem discussed extensively in Chapter 2. To take another example, the Dilworth problem (see Ford and Fulkerson[227]) may be characterized as a multi-vehicle scheduling problem with a homogeneous fleet, deterministic demands at all nodes of a directed network, unrestricted vehicle capacity and route time, fixed vehicle acquisition costs, and the objective is to minimize the number of vehicles required to service all demands. This problem is utilized in the UCOST procedure for scheduling vehicles in mass transit systems (see Section 3.3 and Bodin, Rosenfield, and Kydes[92]). By varying the choices for the costs and the objective, that is, options within categories 10 and 11, other versions of this scheduling problem for mass transit systems may be obtained. For example, minimizing routing costs resulted in the RUCUS formulation of mass transit vehicle scheduling[69] whereas the minimization of the sum of fixed and variable costs yields the formulation of Bodin and Dial[83].

1.3. COMPLEXITY AND COMPUTATIONAL BURDEN OF ROUTING AND SCHEDULING PROBLEMS

An important consideration in the formulation and solution of routing and scheduling problems is the computational burden associated with various solution techniques for these problems. The computational burden of solving a given problem clearly increases as the size of the problem becomes larger. The nature of this growth in computation time as a function of problem size is an issue of both theoretical and practical interest. If this growth is too rapid, the computational burden soon becomes prohibitive even for moderate problem sizes thereby limiting the applicability of a solution technique in a realistic environment where the problems encountered are typically large scale.

Most routing and scheduling problems of interest may be formulated as network problems. A measure of the problem size is then available in the number of nodes (and possibly arcs) of the resulting network. Table 1.2 lists a number of standard network problems that are frequently encountered in solving routing and scheduling problems. To impart some feel for the problem sizes that are currently manageable, the table provides the network size that can be solved within a few minutes on a computer comparable to the UNIVAC 1108. Here, an upper limit of 5000 nodes is chosen since larger problems may run into storage difficulties. Thus, when 5000 appears in the table, it should not necessarily be taken to mean that larger problem sizes are impossible to handle. Although the actual numbers in Table 1.2 (first presented in Golden *et al.*[298]) are subject to frequent changes as advances in designing computational procedures are made, they still serve to indicate the "relative" burden of solving the problems listed.

While Table 1.2 represents a practical way of comparing the computational burden of various network problems, this issue may also be approached by means of a theoretical schema that involves the notion of "polynomially-bounded" algorithms. A polynomially-bounded algorithm for a problem is a procedure whose computational burden increases only polynomi-

Table 1.2. Selected network problems and algorithms

Problem Name	Heuristic Algorithm		Exact Algorithm	
	Size Handled Easily	References	Size Handled Easily	References
Shortest Path from s to t	NN		5000	Golden & Ball [297]
Shortest Path from s to all other nodes	NN		5000	Denardo & Fox [173], Golden [288], Pape [538], Gilsinn & Witzgall [277], Dial et al. [186]
Shortest Paths Between All Nodes	NN		500	Kelton & Law [382]
K Shortest Paths	NN		500 (K<5)	Shier [597],[598]
Minimal Spanning Tree	NN		5000	Kershenbaum & Van Slyke [385]
Capacitated Minimal Spanning Tree!	1000	Kershenbaum [384]	40	Chandy & Lo [117]
Transportation Problem	NN		5000	Mulvey [498], Bradley et al. [100], Glover et al. [280]
Max Flow	NN		5000	Cheung [122], Glover, et al. [280]
Min Cost Flow	NN		5000	Bradley et al. [100], Barr et al. [41]
Matching	NN		500	Cunningham & Marsh [157], Derigs [174], Derigs & Kazakidis [177]
Traveling Salesman Problem!	1000	Webb [667], Golden & Bodin [299], Golden et al. [302]	100	Miliotis [480],[481], Held & Karp [337], [338], Padberg & Hong [528], Balas & Christofides [29]
Vehicle Routing Problem !	750	Golden et al. [306]	30	Christofides et al. [143],[144]

! indicates problem is NP-hard
NN indicates heuristic or approximate algorithms are not necessary

ally with problem size in the worst case. The class of all problems for which polynomially-bounded algorithms are known to exist is denoted by *P*. Problems in the class *P* can generally be solved to optimality quite efficiently. In Table 1.2, all problems for which the symbol ! does *not* appear belong to the class *P*. The practical implication of obtaining a polynomially-bounded algorithm for a problem may be illustrated through a small example.

Consider two network problems, labeled *A* and *B*, both defined on a network of size *N* (say, the number of nodes). Assume that Problem *A* belongs to *P* and that an algorithm requiring $1000N^2$ computations in the worst case is available for solving an instance of *A* with size *N*. For Problem *B*, however, suppose that the best known algorithm is not polynomial and requires 2^N computations in the worst case. For $N = 15$, Problem *A* will be significantly more time-consuming to solve than Problem *B*. However, as the problem size *N* increases, this situation is rapidly reversed. For $N = 30$, Problem *B* would require over one billion computations as opposed to 900,000 computations for *A* (this latter number corresponds to a couple of seconds of Univac 1108 computer time). One may conclude that solving large-scale problems with an algorithm of exponential computational burden is impractical. Consequently, it is useful to obtain some theoretical insight into the class of problems for which no polynomially-bounded algorithm may be expected to exist.

In contrast to the class *P*, there is a large class of network and combinatorial problems for which no polynomially-bounded algorithm has yet been found. Problems in this class are called "*NP*-hard". Loosely speaking, this class is characterized by the property that if a polynomially-bounded algorithm exists for any particular problem in this class, then all other problems of the class are also solvable in polynomial time. As such, the class of "*NP*-hard" problems may be viewed as forming a hard core of problems that polynomial algorithms have not been able to penetrate so far. This suggests (but does not prove) that the effort required to solve "*NP*-hard" problems increases exponentially with problem size in the worst case. Although there are some problems that have not yet been classified as either *NP*-hard or members of the class *P*, most problems reviewed in this paper fall into one of these two classes. In fact, all problems with the symbol ! in Table 1.2 are *NP*-hard. Research into the computational complexity of various combinatorial problems has been an area of intense activity in the recent years. The interested reader may consult the works of Garey and Johnson[242], Karp[374], Lenstra and Rinnooy Kan[434], Lewis and Papadimitriou [443], Papadimitriou and Steiglitz[536], and Tarjan[642] for precise definitions, further details, and extensive classifications of *NP*-hard problems. In the area of routing and scheduling, Lenstra and Rinnooy Kan[436] provide a concise overview of complexity results known to date. As expected, most routing and scheduling problems of interest are *NP*-hard. The practical utility of the available complexity results therefore lies in identifying components or subproblems of *NP*-hard routing and scheduling problems that belong to the class *P* and thus may be used effectively in attacking the larger problem. To take one example, Section 3.5.3 describes an approach for the crew/vehicle scheduling problem (itself *NP*-hard) that utilizes a matching subproblem. It may be cautioned that, even among problems that belong to the class *P*, apparently minor changes in problem characteristics may result in radical changes in the computational complexity of the resulting problems. For example, although the Chinese Postman Problem (CPP) (defined in Section 2.9) on directed undirected networks belongs to the class *P*, the mixed-CPP (when a mixture of directed and undirected arcs is allowed) is *NP*-hard. (For a recent survey of the mixed CPP, see Minieka[487]). The Dilworth problem mentioned in the previous section is also in *P* as long as all vehicles are housed at the same depot. The multiple depot case, however, is *NP*-hard[436].

Given that most routing and scheduling problems are *NP*-hard, known approaches for solving these problems optimally suffer from an exponential growth in computational burden with problem size. When faced with an *NP*-hard problem, one frequently resorts to heuristic or approximate procedures to obtain near-optimal solutions in lieu of seeking optimal solutions. A heuristic algorithm is a procedure that uses the problem structure in a mathematical (and usually intuitive) way to provide feasible or near-optimal solutions. A heuristic is considered effective if the solutions it provides are consistently close to the optimal solution. In many cases it is possible to obtain bounds on the deviation of the heuristic solution from the optimal one in the worst case. Examples of such bounds for traveling salesman heuristics may be found in

Section 2.3. The design of effective heuristics with known average of worst-case error bounds is an active area of current research (see the survey papers by Fisher[215], Graham *et al.*[314], Klee[390], Sahni[576], and Silver *et al.*[599]). The two papers by Lenstra and Rinnooy Kan[435] and Ball and Magazine[39] discuss approximation algorithms and issues in the design of effective heuristics in the specific context of routing and scheduling problems.

1.4. SET PARTITIONING AND SET COVERING PROBLEMS

Many routing and scheduling problems can be formulated as instances of a special class of zero-one integer programs known as set partitioning or set covering problems. Basically, a set covering problem involves a given 0–1 matrix with costs attached to all columns. The objective is to choose a minimum-cost collection of columns such that the number of 1's appearing in each row of the selected columns is at least one. If this number is required to be exactly one, the set partitioning problem results. In this case the rows are partitioned into a number of subsets each "covered" by exactly one selected column. Set covering and set partitioning problems have been studied extensively over the last two decades (see Balas and Padberg[31], Garfinkel and Nemhauser[246], [247]).

The idea of using set partitioning or covering in routing and scheduling problems dates back to at least 1961[610]. In these applications the rows represent the entities that require service whereas the columns of the integer program correspond to ways in which service may be provided to a subset of the demand entities. For example, in vehicle routing from a central depot for delivery to a set of customers, the rows correspond to the customers and the columns stand for feasible round trips based at the depot through a given cluster of customers. The column cost would then equal the total cost associated with such a trip. By the same token, in air crew scheduling, the rows represent the flight legs to be flown and the columns enumerate possible round trips that a crew might fly (see Marsten and Shepardson[471]).

While set covering and partitioning provides a valid conceptual framework for the formulation of many routing and scheduling problems, its practical utility may be limited if the

Table 1.3. Representative applications of set covering and partitioning to routing and scheduling problems.

I. Air Crew Scheduling Arabeyre et al. [11], Baker et al. [23], Marsten et al. [470], Marsten and Shepardson [471], Rubin [449], Spitzer [610].
II. Airline Fleet Planning Levin [440], [441]
III. Mass Transit Crew Scheduling Koljonen and Tamminen [396], Mitra and Welsh [490], Parker and Smith [539], Ryan and Foster [574], Ward et al. [665]
IV. Vehicle Routing and Scheduling Balinski and Quandt [32], Cullen, Jarvis, and Ratliff [156], Dantzig and Ramser [167], Fisher, Jaikumar, and Bell [221], Foster and Ryan [228], Pierce [545].

resulting integer program is too large. In the last decade, it has been possible to solve set partitioning problems of some 150 rows and a few thousand columns optimally (see, for example, Marsten[469]). However since all columns corresponding to feasible options must be enumerated, in most cases a full-scale representation of a routing and scheduling problem as an integer program leads to problems of enormous size. Moreover, even the enumeration of all feasible columns is often a difficult and time consuming task. As a result, researchers have employed heuristics or clustering approaches for decomposing set partitioning problems (Marsten and Shepardson[471]), or have embedded these problems within an interactive computing environment (Cullen, Jarvis, and Ratliff[156]). Clearly, heuristics can also be used to provide approximate solutions to set partitioning or set covering problems—an approach that is frequently resorted to in practice (see Baker *et al.*[23]). Further discussion of the application of set partitioning and covering problems may be found in Section 3.5. A representative (but not exhaustive) list of applications in the area of routing and scheduling is presented in Table 1.3. The reader may consult this table as a guide to further references in this research.

1.5. ORGANIZATION OF THIS REPORT

We wish to conclude this chapter with a brief review of the plan of this paper. Chapters 2 and 3 focus on routing and scheduling problems respectively. The major problems of interest and the associated solution techniques are discussed. Chapter 4 describes combined routing and scheduling problems where routing and scheduling features are present simultaneously. This is an area of great potential for future research and the approaches discussed in Chapter 4 must be viewed as first steps along a relatively unexplored path. Chapter 5 deals with implementation issues for both routing and scheduling systems. The discussion in this chapter reviews the difficulties associated with data collection and explores database design issues and the options provided by manual, automated, and interactive man–machine systems. Finally, Chapter 6 presents conclusions and directions for future research. The report concludes with a comprehensive list of references on all aspects of routing and scheduling problems.

A report of this scope cannot avoid making certain disclaimers. The authors must emphasize that the main focus of this report is vehicle and crew routing and scheduling. The main routing and scheduling decisions draw upon and interact with a variety of "higher-level" and peripheral planning issues. These issues can not be fully treated if the size of this report is to be kept within reasonable limits. For instance, a variety of related problems involving location, districting, and aggregation or clustering questions are not discussed in any detail within this report.

The authors realize that even in the specific area of routing and scheduling problems, the report may have suffered from certain inadvertent omissions. In particular, some degree of provincialism is unavoidable as the authors tend to dwell upon the research best-known to them in a research area of vast scope and much recent activity. The authors wish to apologize for all potential shortcomings in advance and hope that the readers consider whatever omissions they might detect as being due to mere oversight.

CHAPTER 2
ROUTING OF VEHICLES

The basic routing problem is easy to state. We are given a set of nodes and/or arcs to be serviced by a fleet of vehicles. There are no restrictions on when or the order in which these entities must be serviced. The problem is to construct a low-cost, feasible set of routes—one for each vehicle. A route is a sequence of locations that a vehicle must visit along with an indication of the service it provides.

In Fig. 2.1, a set of vehicle routes that services the 13 demand points is presented. Each node has a demand of one unit, the vehicle capacity is three units, and each vehicle must return to the depot it originated from. Each route can be traversed in either direction.

The routing of vehicles is primarily a spatial problem. It is assumed that no temporal or other restrictions impact the routing decision except (possibly) maximum route-length constraints. This is in contrast to scheduling problems to be described in Chapter 3, where the movement of each vehicle must be traced through both time and space.

Due to the relatively unconstrained nature of these problems and their inherent complexity, they have tantalized and challenged combinatorial analysts and operations researchers for many years. (See Magnanti[455] for an extensive discussion.)

The basic vehicle routing problems discussed in this chapter are:

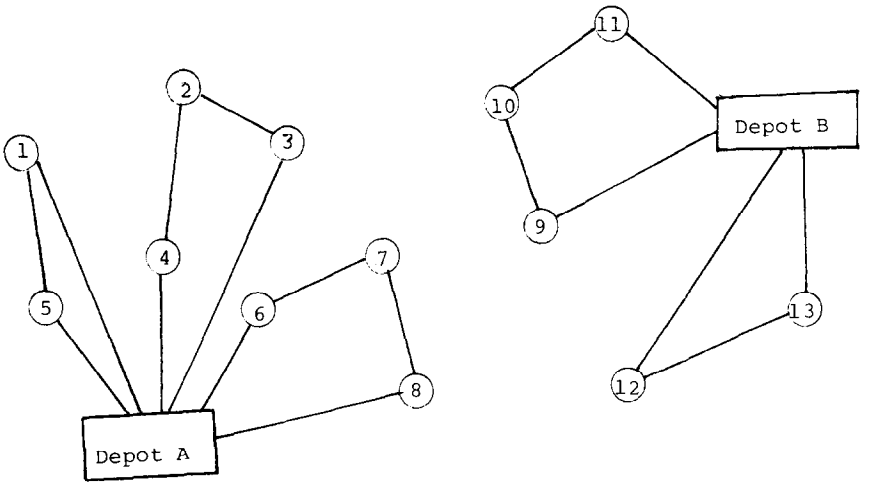
*The Traveling Salesman Problem
*The Chinese Postman Problem
*The M-Traveling Salesman Problem
*The Single Depot, Multiple Vehicle, Node Routing Problem
*The Multiple Depot, Multiple Vehicle, Node Routing Problem
*The Single Depot, Multiple Vehicle, Node Routing Problem with Stochastic Demands
*The Capacitated Chinese Postman Problem.

By way of introduction, we now discuss each of the preceding problems informally. A figure illustrating the spatial configuration of a feasible solution is provided where appropriate. In addition to these basic problems, a number of variants are also described in this chapter.

The *traveling salesman problem* requires the determination of a minimal cost cycle that passes through each node in the relevant graph exactly once. If costs are symmetric, that is, if the cost of traveling between two locations does not depend on the direction of travel, we have a symmetric traveling salesman problem; otherwise, we have an asymmetric or directed traveling salesman problem. A feasible solution to a symmetric traveling salesman problem is given in Fig. 2.2. Every feasible solution has two arcs incident to each node. A feasible solution to an asymmetric traveling salesman problem is given in Fig. 2.3. In this case, note that there is one arc into and one arc out of every node.

The *Chinese postman problem* requires the determination of the minimal cost cycle that passes through every arc of the graph at least one time. A Chinese postman problem is called directed or undirected, depending on whether arcs of the graph are directed or not. Both of these variations can be solved by polynomially-bounded algorithms. The mixed Chinese postman problem has some undirected and some directed arcs; this problem is *NP*-hard. An illustration of a feasible solution to an undirected Chinese postman problem is given in Fig. 2.4.

The *multiple traveling salesman problem* is a generalization of the traveling salesman problem where there is a need to account for more than one salesman (or vehicle). The M vehicles in the fleet are to leave from and return to a common depot. There are no restrictions on the number of nodes that each vehicle may visit except that each vehicle must visit at least one node. An illustration of a feasible solution to a multiple traveling salesman problem with $M = 3$ is given in Fig. 2.5.



Route 1: Depot A - 1 - 5 - Depot A

Route 2: Depot A - 3 - 2 - 4 - Depot A

Route 3: Depot A - 6 - 7 - 8 - Depot A

Route 4: Depot B - 9 - 10 - 11 - Depot B

Route 5: Depot B - 12 - 13 - Depot B

Fig. 2.1. Illustration of routes.

The *single depot, multiple vehicle, node routing problem* (classical vehicle routing problem) asks for a set of delivery routes for vehicles housed at a central depot, which services all the nodes and minimizes total distance traveled. The demand at each node is assumed to be deterministic and each vehicle has a known capacity. The multiple traveling salesman problem is a vehicle routing problem with a fleet of M vehicles each of which has, for all practical purposes, infinite capacity (so that no capacity problems arise). In the example in Fig. 2.6, the demand at each node is one, and vehicle capacity is 3.

The *multiple depot, multiple vehicle, node routing problem* is a generalization of the previous problem in that the fleet of vehicles now must serve D depots rather than just one. All other constraints from the classical VRP still apply. In addition, each vehicle must leave from and return to the same depot. A feasible solution to this problem with two depots using the same data as in Fig. 2.6 is given in Fig. 2.7. In this example, we assume that each depot can house up to two vehicles.

The *single depot, multiple vehicle, node routing problem with stochastic demands* is identical to the classical vehicle routing problem except that the demands are not known with certainty,

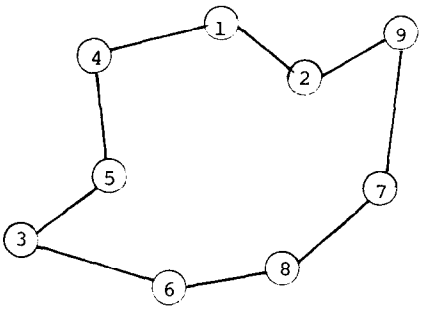


Fig. 2.2. Symmetric case.

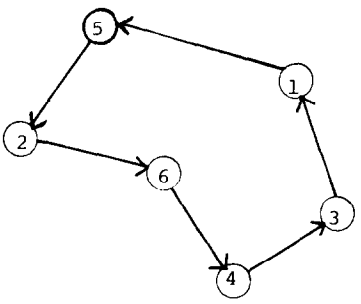


Fig. 2.3. Asymmetric case.

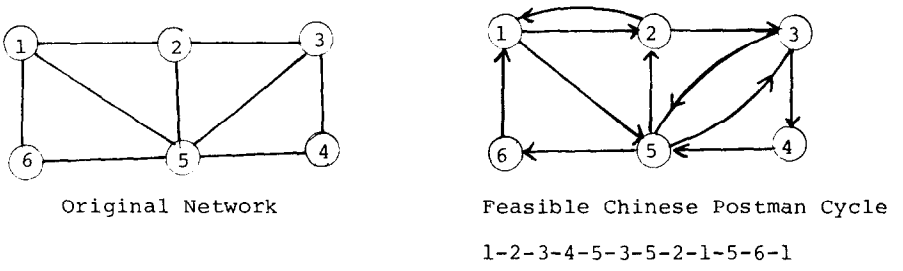


Fig. 2.4. Chinese postman problem.

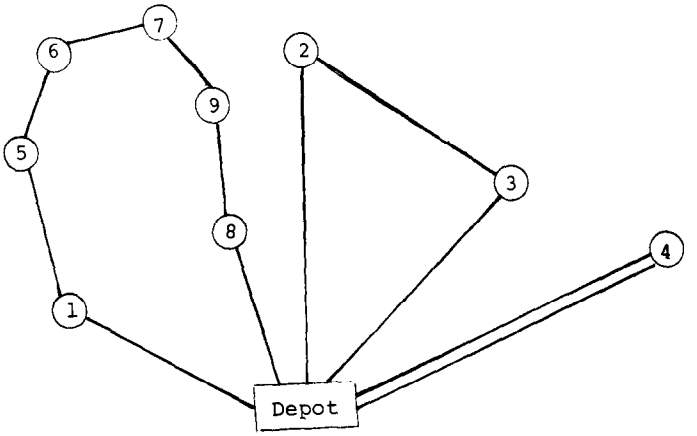


Fig. 2.5. The multiple traveling salesman problem.

but instead come from a specified probability distribution. For example, it may be that node i has demand described by a Poisson distribution with mean λ_i.

The *capacitated arc routing problem* is as follows: given an undirected network with arc demands $q_{ij} \geq 0$ for each arc which must be satisfied by one of a fleet of vehicles, each of capacity W, find the vehicle cycles, each passing through the depot, that satisfy all demands at minimal total cost. In Fig. 2.8, each arc has unit demand and $W = 3$.

In this chapter, we discuss many of the approaches that have been proposed for solving the routing problems already described and others as well. For some of these algorithms we present worst case bounds. Worst case bounds tell us that the worst value of the objective function resulting from the algorithm under consideration is guaranteed not to exceed $K(n)$ times the optimal objective value where $K(n)$ is a function of n (the number of nodes). For optimal algorithms, $K(n)$ is equal to 1 and for heuristic algorithms, $K(n) > 1$ and generally increases with n. Since the determination of worst case bounds usually requires sophisticated mathematical analysis, worst case results are stated and referenced in this chapter, but are not derived in any detail.

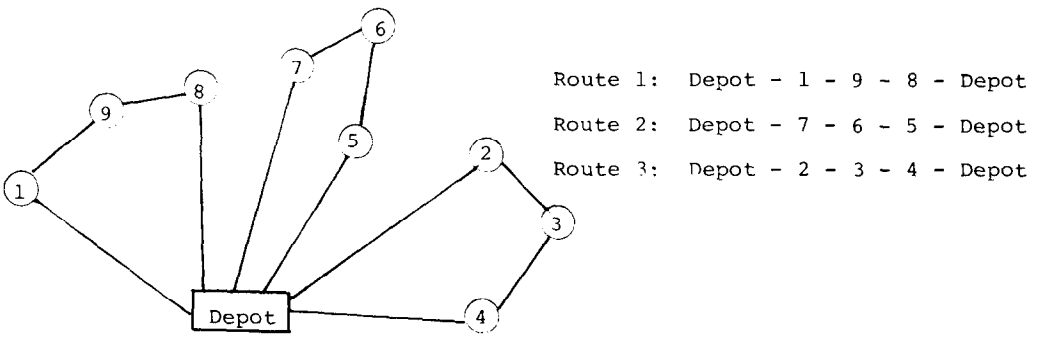
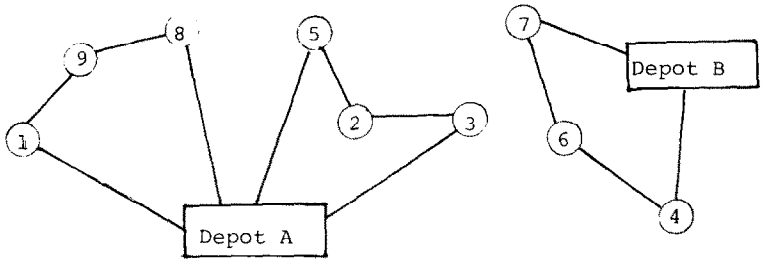


Fig. 2.6. Vehicle routing problem.

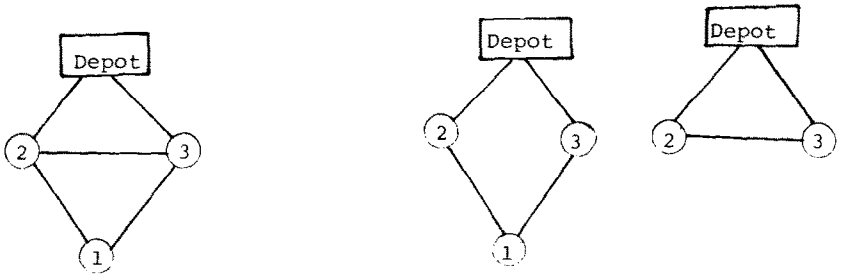


Route 1: Depot A - 1 - 9 - 8 - Depot A

Route 2: Depot A - 5 - 2 - 3 - Depot A

Route 3: Depot B - 7 - 6 - 4 - Depot B

Fig. 2.7. Multiple depot problem.



Original Network Feasible Solution

Fig. 2.8. Capacitated arc routing problem.

The next section discusses the celebrated traveling salesman problem (TSP) and focuses on the routing of a single vehicle. Sections 2.2 and 2.3 treat the solution methodology for this problem including both exact and heuristic techniques. Sections 2.4 and 2.5 discuss single depot multiple vehicle routing problems. Subsequent sections deal with extensions of this problem in the presence of multiple depots, arc routing, and various other scerarios.

2.1. THE TRAVELING SALESMAN PROBLEM (TSP)— INTRODUCTION

Let a network $G = [N, A, C]$ be defined with N the set of nodes, A the set of branches, and $C = [c_{ij}]$ the matrix of costs. That is, c_{ij} is the cost of moving or the distance from node i to node j. The traveling salesman problem requires the Hamiltonian cycle in G of minimal total cost (a Hamiltonian cycle is a cycle passing through each node $i \in N$ exactly once). Many interesting aspects of this problem have been discussed by Bellmore and Nemhauser[61] and Christofides[131]. In addition, connections between statistical mechanics (the study of how systems with many degrees of freedom behave in thermal equilibrium at a finite temperature) and the TSP are explored in a recent paper by Kirkpatrick *et al.*[389].

Karp[377] has shown that the traveling salesman problem (TSP) is *NP*-complete. This remains true even when additional assumptions such as the triangle inequality or Euclidean distances are invoked (see Garey *et al.*[241] and Papadimitriou[532]). These results imply that a polynomially bounded exact algorithm for the TSP is unlikely to exist. Although ingenious algorithms for the TSP have been proposed by Little *et al.*[447], Held and Karp[337], [338] Miliotis[480], Crowder and Padberg[155], and others, they all encounter problems with storage and running time for cases with more than about 100 nodes. An important exception is the recent work of Padberg and Hong[528].

Due to the difficulty of the TSP, many heuristic (approximate) procedures have been developed. These heuristics may be compared analytically, as in the ground-breaking paper of Rosenkrantz, Stearns, and Lewis[568] by studying their worst-case behavior. Alternatively,

computational experimentation may be used to compare the performance of these heuristics, as in the work of Golden *et al.*[302] and Stewart[627]. Golden *et al.*[302] consider a set of problems ranging in size from 25 to 150 nodes and give some guidelines as to which TSP procedure to use in a particular situation. If an approximate, but not an optimal solution is sought, they recommend that the user consider one of the quick tour construction procedures. For more accurate (but more costly) results, a composite procedure is suggested. Finding the optimal solution is feasible only for very small problems with, say, less than 100 nodes. Stewart[627] describes a number of new algorithms designed specifically to perform well on Euclidean problems.

We now proceed to review both optimal and heuristic approaches to the TSP. The next section focuses on an exact approach that is typical of a general methodology for solving combinatorial optimization problems.

2.2. TSP—OPTIMAL APPROACHES

Formulations. Optimal approaches to the TSP are based on mathematical programming formulations for this problem. Indeed, a particular way of formulating the TSP may provide special insights and motivate the exact approach to be used. The following discussion reviews some useful formulations of the TSP.

For the sake of simplicity, we assume that the costs are symmetric (that is, $c_{ij} = c_{ji}$) unless otherwise specified and set $c_{ii} = +\infty$ for $i = 1, 2, \ldots, n$. The problem is to form a tour over all the nodes beginning and ending at the origin, node 1, which gives the minimum total tour distance or cost. For notation, let

$$x_{ij} = \begin{cases} 1 & \text{if arc } i\text{–}j \text{ is in the tour} \\ 0 & \text{otherwise.} \end{cases}$$

An assignment-based formulation of the problem selects the matrix $X = (x_{ij})$ of decision variables so that exactly one arc (i, j) emanates from each node i and exactly one arc (i, j) is directed into each node j. This implies an assignment of each node to its successor node on the tour. It is well known that the assignment requirements, by themselves, do not ensure that the matrix X corresponds to a tour. Instead, the assignment may result in subtours as shown in Fig. 2.9.

To eliminate the possibility of subtours, further restrictions are imposed on the choices for the arc selection matrix X to give the formulation:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = b_j = 1 \qquad (j = 1, \ldots, n) \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = 1 \qquad (i = 1, \ldots, n) \tag{2.3}$$

$$X = (x_{ij}) \in S \tag{2.4}$$

$$x_{ij} = 0 \text{ or } 1 \qquad (i, j = 1, \ldots, n). \tag{2.5}$$

Fig. 2.9. Two subtours.

The set S can be any restrictions that prohibit subtour solutions satisfying the assignment constraints (2.2), (2.3), and (2.5). Such restrictions are called subtour-breaking constraints. Possible choices for S includes:

(1) $S = \{(x_{ij})\colon \sum_{i \in Q} \sum_{j \notin Q} x_{ij} \geq 1$ for every nonempty proper subset Q of $N\}$;

(2) $S = \{(x_{ij})\colon \sum_{i \in R} \sum_{j \in R} x_{ij} \leq |R| - 1$ for every nonempty subset R of $\{2, 3, \ldots, n\}\}$;

(3) $S = \{(x_{ij})\colon y_i - y_j + nx_{ij} \leq n - 1$ for $2 \leq i \neq j \leq n$ for some real numbers $y_i\}$.

Note that S contains nearly 2^n subtour-breaking constraints in (1) and (2), but only $n^2 - 3n + 2$ constraints in formulation (3). Observe that the configuration in Fig. 2.9 satisfies constraints (2.2), (2.3), and (2.5), but not (2.4). That is, it does not represent a tour. The first set of subtour-breaking constraints (1) states that every proper subset Q of nodes must be connected to the other nodes in the network in the solution X. They prohibit the solution in Fig. 2.9 when $Q = \{1, 2, 3\}$. The second set of subtour-breaking constraints (2) implies that the arcs selected in X contain no cycle, since a cycle on nodes R must contain $|R|$ arcs. It excludes the solution in Fig. 2.9 when $R = \{4, 5, 6\}$. The third set of subtour-breaking constraints is more complicated. Adding constraints (3) for arcs 4–5, 5–6, and 6–4 in Fig. 2.9 yields $3n \leq 3(n-1)$, a contradiction. A similar contradiction arises whenever the matrix X contains a subtour solution. Thus, each set of constraints (1), (2), and (3) prohibits subtours.

It is also easy to see that tours are feasible in each case. Constraints (1) and (2) present no difficulties. In constraints (3), let

$$y_i = \begin{cases} t & \text{if node } i \text{ is visited on the } t^{\text{th}} \text{ step in a tour} \\ 0 & \text{otherwise.} \end{cases}$$

Then, each constraint (3) associated with an arc in a tour states that

$$t - (t+1) + n \leq n - 1.$$

Other constraints in (3) only reveal that $y_i - y_j \leq n - 1$ whenever $y_i \leq n$ and $y_j \geq 1$.

A somewhat different formulation due to Gavish and Graves[257] introduces flow variables y_{ij} and assumes that $(n-1)$ units of a good are supplied at the origin, node 1, and that every other node in the network requires 1 of these units. The formulation is:

$$\text{Minimize} \quad \sum_{i=j}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.6}$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n) \tag{2.7}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n) \tag{2.8}$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=1}^{n} y_{ji} = -1 \qquad (i = 2, \ldots, n) \tag{2.9}$$

$$y_{ij} \leq U x_{ij} \qquad (i, j = 1, \ldots, n) \tag{2.10}$$

$$x_{ij} = 0, 1, \; y_{ij} \geq 0 \qquad (i, j = 1, \ldots, n) \tag{2.11}$$

where $U \geq n - 1$. Constraints (2.7) and (2.8) define an assignment problem, as before. The

"forcing constraints" (2.10) insure that the flow y_{ij} on arc (i, j) is zero if that arc is not selected in the assignment matrix X. The formulation prohibits subtours, since, if node i lies on a subtour not containing the origin, it could not receive a unit of demand from the origin. Formally, suppose that there is a subtour $\{i_1, i_2, \ldots, i_r, i_1\}$ that does not include node 1. If we set $y_{i_1,i_2} = f$, then (2.9) implies that

$$y_{i_2,i_3} = f - 1$$

$$y_{i_3,i_4} = f - 2$$

$$\vdots$$

$$y_{i_r,i_1} = f - r + 1.$$

But this gives $\sum_{j=1}^{n} y_{i_1 j} - \sum_{j=1}^{n} y_{j i_1} = f - (f - r + 1) = r - 1 > -1$, which violates (2.9). Thus, no subtours are admissible. Any tour $\tau = \{i_0 = 1, i_1, i_2, \ldots, i_{n-1}, i_n = 1\}$ is feasible since setting y_{i_{j-1},i_j} for $j = 1, 2, \ldots, n$ and $y_{\alpha\beta} = 0$ otherwise, satisfies all constraints.

These various formulations are valuable for a number of reasons. They help provide insight onto the complexity of the TSP and its relationship to other routing and distribution problems. They suggest algorithmic strategies and heuristic procedures, some of which are obtained by dualizing with respect to certain constraints. Finally, they serve as compact mathematical problem representations.

An exact solution procedure. In this section, we demonstrate a procedure for computing sharp lower bounds on the TSP solution and we describe how to embed this procedure within a branch and bound framework in order to solve the TSP exactly. This approach was first introduced by Held and Karp[337, 338] in a formal sense, and by Christofides[126] informally. Extensions have been proposed by Hansen and Krarup[327] and by Bazaraa and Goode[49]. For a comprehensive treatment of exact approaches to the TSP, see Christofides[134].

Consider the formulation (2.1)–(2.5). Constraints (2.2) and (2.3) can be replaced by

$$\sum_j x_{1j} = 2 \tag{2.12}$$

$$\sum_j x_{ij} + \sum_k x_{ki} = 2 \quad \text{for} \quad i = 1,2,\ldots,n \tag{2.13}$$

$$\sum_i \sum_j x_{ij} = n. \tag{2.14}$$

Constraints (2.12) and (2.13) state that each node has a degree of two. Constraint (2.14) guarantees that a total of n arcs are in the solution. Note that under these new constraints a solution with arcs (1, 2), (2, 4), (4, 3) and (1, 3) is acceptable. Assuming a symmetric distance matrix, the direction on arc (1, 3) can be reversed without changing the objective value in order to obtain a tour. Let problem (P) be the problem represented by (2.1), (2.4), (2.5), (2.12), (2.13) and (2.14).

A lower bound on the solution to problem (P) can be computed if we solve the Lagrangian relaxation obtained from including constraints (2.13) in the objective function. This yields a new objective function of

$$\underset{x}{\text{Minimize}} \quad \sum_i \sum_j c_{ij} x_{ij} + \sum_i \lambda_i \left\{ \sum_j x_{ij} + \sum_k x_{ki} - 2 \right\}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a vector of Lagrange multipliers. We call this problem (LR_λ).

Rearranging the objective function, we can rewrite it as:

$$(LR_\lambda): \quad \underset{x}{\text{Minimize}} \quad \sum_i \sum_j (c_{ij} + \lambda_i + \lambda_j) x_{ij} - 2 \sum_i \lambda_i$$

subject to: (2.4), (2.5), (2.12), and (2.14).

Let z and $z_D(\lambda)$ be the optimal objective values for problems (P) and (LR_λ), respectively. If x_{ij}^* is an optimal solution to (P), then

$$z_D(\lambda) \leq \sum_i \sum_j c_{ij} x_{ij}^* + \sum_i \lambda_i \left\{ \sum_j x_{ij}^* + \sum_k x_{ki}^* - 2 \right\} = z.$$

Thus, (LR_λ) provides an infinite family of lower bounds on the length of an optimum tour, one for each value of λ. As we will see, for a given λ, $z_D(\lambda)$ is easy to determine. The best of these lower bounds is given by

$$\max_\lambda z_D(\lambda).$$

A key observation, which motivates this approach, is that problem (LR_λ) is substantially easier to solve than problem (P). For any given λ, the arc costs are given by $c'_{ij} = c_{ij} + \lambda_i + \lambda_j$ and the problem is to find a least-cost 1-*tree*. A 1-*tree* is a tree having node set $\{2, 3, \ldots, n\}$ along with two distinct arcs incident to node 1. A least cost 1-tree can be determined in only $O(n^2)$ operations using a minimal spanning tree algorithm on nodes $\{2, 3, \ldots, n\}$ and attaching the two smallest arcs with respect to $C' = (c'_{ij})$ incident to node 1.

There are a number of effective iterative procedures for obtaining $\max_\lambda z_D(\lambda)$. The subgradient method has worked well in a variety of applications and is, perhaps, the most popular procedure for determining λ^*. Details may be found in papers by Held and Karp[337, 338].

In general, the bounds provided from Lagrangian relaxation and the subgradient method are extremely tight and, because of this fact, have been used successfully in branch and bound procedures to solve TSP's of moderate size efficiently. In this setting, we solve a Lagrangian relaxation in place of a linear programming relaxation at each node of the branch and bound tree. Each partitioning of a node in the branch and bound tree includes, and/or excludes, some arcs from the tour. That is, at each node, a set of arcs, A^1, has already been included and another set, A^2, has been excluded from the tour. If the least-cost 1-tree at a particular node is a tour, then it is the optimal tour for the given sets of included and excluded arcs and a feasible solution to problem (P). If, on the other hand, the lower bound at a node exceeds the value of a feasible solution (upper bound), then that node in the branch and bound tree may be discarded and it requires no further consideration.

The Lagrangian relaxation procedure outlined above typifies a general line of attack for combinatorial routing problems. For the TSP, it is possible to devise other Lagrangian relaxation schemes based on a different formulation and/or the relaxation of constraints other than the ones used above. For example, Christofides[134] discusses the assignment problem, matching problem, and shortest n-arc path TSP relaxation in his overview paper. In all cases the role of the relaxation is to provide tight bounds on the optimum value searched for. We will encounter this approach also in the context of the Vehicle Routing Problem discussed later in this chapter.

Perhaps, at this point, it is a good idea to indicate the size of the largest TSP's that have been solved to optimality. Crowder and Padberg[155], using a cutting plane algorithm that generates subtour-breaking and "comb" inequalities as needed, have solved a symmetric 318 node problem in under 50 minutes of computer time on an IBM 370/168. The above-mentioned comb inequalities are due to Chvatal[125] and Grötschel and Padberg[319]. When it comes to asymmetric TSP's the restricted Lagrangian approach of Balas and Christofides[29] seems to be the best available exact procedure known. This algorithm combines features of subtour-elimination and Lagrangian relaxation procedures and has been *remarkably* successful in solving up to 325 node problems. In particular, ten 325 node problems were solved in an

average of 49 seconds on a CDC 7600. The maximum time observed was only 82 seconds.

Since exact approaches to the TSP are, in general, computationally burdensome for large TSP's, a variety of heuristic approaches have found wide use. The next section reviews heuristic techniques for the TSP.

2.3. TSP—HEURISTIC APPROACHES

Heuristics examined. The heuristics we examine fall into three broad classes—tour construction procedures, tour improvement procedures, and composite procedures. *Tour construction procedures* generate an approximately optimal tour from the distance matrix. *Tour improvement procedures* attempt to find a better tour given an initial tour. *Composite procedures* construct a starting tour from one of the tour construction procedures and then attempt to find a better tour using one or more of the tour improvement procedures. Most of these procedures are described in the literature and, hence, will be sketched only briefly; but newer procedures will be studied in more detail. We assume for the sake of simplicity that the costs are symmetric, (i.e. $c_{ij} = c_{ji}$) satisfy the triangle inequality, and are defined for each (i, j) pair, unless otherwise specified.

2.3.1 *Tour construction procedures*

(*a*) *Nearest neighbor procedure* (Rosenkrantz, Stearns, and Lewis[568]).

Step 1. Start with any node as the beginning of a path.

Step 2. Find the node closest to the last node added to the path. Add this node to the path.

Step 3. Repeat step 2 until all nodes are contained in the path. Then, join the first and last nodes.

Worst case behavior:

$$\frac{\text{length of nearest neighbor tour}}{\text{length of optimal tour}} \leq \frac{1}{2}\left\lceil \lg(n) \right\rceil + \frac{1}{2}$$

where lg denotes the logarithm to the base 2, $\lceil X \rceil$ is the smallest integer $\geq X$, and n is the number of nodes in the network.

Number of computations. The nearest neighbor algorithm requires on the order of n^2 computations.

Comments. In a computational setting, the procedure outlined above may be repeated n times, each time with a new node selected as the starting node. The best solution obtained would then be listed as the answer. Notice that this strategy runs in an amount of time proportional to n^3.

(b) *Clark and Wright Savings* (Clark and Wright[145], Golden [291]).

Procedure

Step 1. Select any node as the central depot which we denote as node 1.

Step 2. Compute savings $s_{ij} = c_{1i} + c_{1j} - c_{ij}$ for $i, j = 2,3, \ldots, n$.

Step 3. Order the savings from largest to smallest.

Step 4. Starting at the top of the savings list and moving downwards, form larger subtours by linking appropriate nodes i and j. Repeat until a tour is formed.

Worst case behavior. The worst case behavior for this approach is known for both a sequential and concurrent version. Golden[289] demonstrates that for a sequential version of this algorithm where at each step we select the best savings from the last node added to the subtour, the worst case ratio is bounded by a linear function in $\lg(n)$. Ong[517] has derived a similar result for the concurrent version.

Number of computations. The calculation of the matrix $S = [s_{ij}]$ in step 2 requires about cn^2 operations for some constant c. Next, in step 3, savings can be sorted into nonincreasing order via the "Heapsort" method of Williams[674] and Floyd[226] in a maximum of $cn^2 \lg(n)$ comparisons and displacements. Step 4 involves at most n^2 operations since there are that many savings to consider. Thus, the Clark and Wright savings procedure requires on the order of $n^2 \lg(n)$ computations.

Comments. Each node may be selected as node 1 for the above procedure (yielding $n^3\lg(n)$ computations). The best solution obtained would be listed as the answer. In practice one can exploit geometric properties when distances are Euclidean; as a result, only a fraction of the almost n^2 saving are calculated which reduces running times by an order of magnitude (see Golden, Magnanti and Nguyen[306] for details).

(c) *Insertion procedures* (Rosenkrantz, Sterns and Lewis [568]).

An insertion procedure takes a subtour on k nodes at iteration k and attempts to determine which node (not already in the subtour) should join the subtour next (the selection step) and then determines where in the subtour it should be inserted (the insertion step). For the first four insertion procedures we discuss, each node in the network can be used as a starting node. Notice that when each node is used as a starting node, the complexity of the entire procedure increases by an order of magnitude (that is, the number of computations is multiplied by n).

(c1) *Nearest insertion*

Procedure

Step 1. Start with a subgraph consisting of node i only.

Step 2. Find node k such that c_{ik} is minimal and form the subtour i-k-i.

Step 3. *Selection step.* Given a subtour, find node k not in the subtour closest to any node in the subtour.

Step 4. *Insertion step.* Find the arc (i, j) in the subtour which minimizes $c_{ik} + c_{kj} - c_{ij}$. Insert k between i and j.

Step 5. Go to step 3 unless we have a Hamiltonian cycle.

Worst case behavior. $\dfrac{\text{length of nearest insertion tour}}{\text{length of optimal tour}} \leq 2.$

Number of Computations. The nearest insertion algorithm requires on the order of n^2 computations.

(c2) *Cheapest insertion*

Procedure

Same as for nearest insertion except that steps 3 and 4 are replaced by the following step.

Step 3′. Find (i, j) in subtour and k not, such that $c_{ik} + c_{kj} - c_{ij}$ is minimal and, then, insert k between i and j.

Worst case behavior: Same as for nearest insertion.

Number of computations. The cheapest insertion algorithm requires on the order of $n^2\lg(n)$ computations.

(c3) *Arbitrary Insertion*

Procedure

Same as for nearest insertion except that in step 3, arbitrarily select node k not in the subtour to enter the subtour.

Worst case behavior.

$$\frac{\text{length of arbitrary insertion tour}}{\text{length of optimal tour}} \leq \lceil \lg(n) \rceil + 1.$$

Number of computations. The arbitrary insertion algorithm requires on the order of n^2 computations.

(c4) *Farthest Insertion*

Procedure

Same as for nearest insertion except that in step 3 replace "closest to" by "farthest from" and in step 2 replace "minimal" by "maximal".

Worst case behavior. Same as for arbitrary insertion.

Comments. Since the worst case result holds for any arbitrary ordering of the nodes, it holds in particular for the ordering induced by the farthest insertion procedure. It is, however, quite possible that a tighter worst case bound can be derived.

(c5) *Quick Insertion or Nearest Addition*

Procedure

Step 1. Pick any node as a starting circuit T, with one node (and 0 edges).

Step 2. Given the k-node circuit T_k, find the node z_k not on T_k that is closest to a node, call it y_k, on T_k.

Step 3. Let T_{k+1} be the $k+1$—node circuit obtained by inserting z_k immediately in front of y_k in T_k.

Step 4. Repeat Steps 2 and 3 until a Hamiltonian circuit (containing all nodes) is formed.

Worst case behavior. Same as for nearest insertion. *Number of Computations*: Same as for nearest insertion.

(c6) *Convex hull insertion*

It has been shown that if the costs c_{ij} represent Euclidean distance and H is the convex hull of the nodes in two-dimensional space, then the order in which the nodes on the boundary of H appear in the optimal tour will follow the order in which they appear in H (see Eilon, Watson-Gandy and Christofides[201] for a discussion of the implications of this result). This observation serves as impetus for the convex hull heuristic procedure (which is discussed along with a host of variants in Stewart's doctoral dissertation[627]). See also the work of Golden and Stewart[308].

Procedure

Step 1. Form the convex hull of the set of nodes. The hull gives an initial subtour.

Step 2. For each node k not yet contained in the subtour decide between which two nodes i and j on the subtour to insert node k. That is, for each such k, find (i, j) such that $c_{ik} + c_{kj} - c_{ij}$ is minimal.

Step 3. From all (i, k, j) found in Step 2, determine the (i^*, k^*, j^*) such that $(c_{i^*k^*} + c_{k^*j^*})/c_{i^*j^*}$ is minimal.

Step 4. Insert node k^* in subtour between nodes i^* and j^*.

Step 5. Repeat Steps 2 through 4 until a Hamiltonian cycle is obtained.

Worst case behavior. Unknown.

Number of computations. This procedure involves, in the worst case, about as much computational complexity as the cheapest insertion algorithm—on the order of $n^2 \lg(n)$ computations.

Comments. Wiorkowski and McElvain[685] introduced an insertion algorithm also based on convexity ideas a number of years ago. However, their procedure performed with rather poor accuracy. More recently, Norback and Love[514] have proposed two geometric methods closely related to algorithm (c6).

(c7) *Greatest Angle Insertion*

Procedure

Step 1. From the convex hull of the set of nodes. The hull gives an initial subtour.

Step 2. Choose the node k^* not in the subtour and the arc (i^*, j^*) in the subtour such that the angle formed by the two arcs (i^*, k^*) and (k^*, j^*) is the largest possible.

Step 3. Insert node k^* in subtour between nodes i^* and j^*.

Step 4. Repeat Steps 2 and 3 until a Hamiltonian cycle is obtained.

Worst case Behavior. Unknown. However, Yang[696] has demonstrated the negative result that there does not exist a constant that bounds the ratio of "greatest angle" tour length to optimal tour length.

Number of computations. Same as for cheapest insertion. *Comments*: Norback and Love[514] seem to have first suggested this approach as well as a related procedure known as the eccentric ellipse method.

(c8) *Difference × ratio insertion* (*Or* [518])

Procedure

Same as for greatest angle insertion except that Step 2 is replaced by the following step.

Step 2'. Choose the node k^* not in the subtour and the arc (i^*, j^*) in the subtour such that the product $\{c_{i^*k^*} + c_{k^*j^*} - c_{i^*j^*}\} \times \{(c_{i^*k^*} + c_{k^*j^*})/c_{i^*j^*}\}$ is smallest possible.

Worst case behavior. Unknown

Number of computations. Same as for cheapest insertion.

(d) *Minimal spanning tree approach* (Kim [387])

Procedure.

Step 1. Find a minimal spanning tree T of G.

Step 2. Double the edges in the minimal spanning tree (MST) to obtain an Euler cycle.

Step 3. Remove polygons over the nodes with degree greater than 2 and transform the Euler cycle into a Hamiltonian cycle.

Worst case behavior.

$$\frac{\text{length of MST approach tour}}{\text{length of optimal tour}} \leq 2.$$

Number of computations. This approach requires on the order of n^2 computations.

(e) *Christofides' heuristic* Christofides[132] recently proposed the following interesting technique for solving TSP's.

Procedure

Step 1. Find a minimal spanning tree T of G

Step 2. Identify all the odd degree nodes in T. Solve a minimum cost perfect matching on the odd degree nodes using the original cost matrix. Add the branches from the matching solution to the branches already in T, obtaining an Euler cycle. In this subgraph, every node is of even degree although some nodes may have degree greater than 2.

Step 3. Remove polygons over the nodes with degree greater than 2 and transform the Euler cycle into a Hamiltonian cycle.

Worst case behavior.

$$\frac{\text{length of Christofides' tour}}{\text{length of optimal tour}} \leq 1.5.$$

Cornuejols and Nemhauser[151] have improved this bound slightly (although not asymptotically) in obtaining a tight bound for every $n \geq 3$.

Number of computations. Since the most time-consuming component of this procedure is the minimum matching segment which requires $O(n^3)$ operations, this heuristic is $O(n^3)$. In most cases, the number of odd nodes will be considerably less than n.

(f) *Nearest merger* (Rosenkrantz *et al.* [568])

The nearest merger method when applied to a TSP on n nodes constructs a sequence $S_1, \ldots, S_n$ such that each S_i is a set of $n - i + 1$ disjoint subtours covering all the nodes.

Procedure

Step 1. S_1 consists of n subtours, each containing a single node.

Step 2. For each $i < n$, find an edge (a_i, b_i) such that $c_{a_i b_i} = \min \{c_{xy}$ for x and y in different subtours in $S_i\}$. Then, S_{i+1} is obtained from S_i by merging the subtours containing a_i and b_i. (At each step in the procedure, the two subtours closest to one another are merged.)

Worst case behavior.

$$\frac{\text{length of nearest tour}}{\text{length of optimal tour}} \leq 2.$$

Number of computations. This approach requires on the order of n^2 computations.

Comments. To merge two tours T_1 and T_2 when one or both consist of a single node is trivial. If T_1 and T_2 each contain at least two modes, let (a, b) be an edge in T_1 and (d, e) an edge in T_2 such that

$$c_{ad} + c_{be} - c_{ab} - c_{de}$$

is minimized. Then MERGE (T_1, T_2) is the tour derived from T_1 and T_2 by deleting (a, b) and (d, e) and adding (a, d) and (b, e).

Other tour construction algorithms have been proposed recently by Stinson[634] and Karp[376]. Stinson's heuristic is a modification of Vogel's approximation method which has been used extensively in obtaining an initial feasible solution to the transportation problem.

Karp[376] presents a partitioning algorithm for the TSP in the plane and performs a probabilistic analysis in order to obtain some interesting theoretical results. From a practical point of view, his procedure is a decomposition algorithm which should be capable of solving extremely large TSP's. The key idea is to partition a large rectangle into a number of subrectangles and solve a TSP in each subrectangle. The outcome is an Euler cycle which can be transformed into a tour over all the nodes with minimal effort. A simpler, but somewhat related approach, is described by Platzman and Bartholdi[546] they include a BASIC program listing in the appendix.

2.3.2 *Tour improvement procedures*

Perhaps the best known heuristics for the TSP are branch exchange heuristics. The 2-opt and 3-opt heuristics were introduced by Lin[444] in 1965 and the k-opt procedure, for $k \geq 3$, was presented by Lin and Kernighan[446] more recently. These branch exchange heuristics work as follows:

Step 1. Find an initial tour. Generally, this tour is chosen randomly (this need not, however, be the case) from the set of all possible tours.

Step 2. Improve the tour using one of the branch exchange heuristics.

Step 3. Continue Step 2 until no additional improvement can be made.

The branch exchange procedure terminates at a local optimum. For a given k, if we define a k-change of a tour as consisting of the deletion of k branches in a tour and their replacement by k other branches to form a new tour, then a tour is k-optimal if it is not possible to improve the tour via a k-change. Steps 2 and 3 would generate this k-optimal tour. Since the 2-opt exchange procedure is weaker than the 3-opt exchange procedure, it will generally terminate at an inferior local optimum. Similarly, a k-opt exchange procedure will generally terminate with a better local optimum than will a 3-opt exchange procedure for $k \geq 4$. Figure 2.10 illustrates a 2-opt exchange. Note that the exchange is well-defined at each step. Once we have decided to drop arcs (A, B) and (E, F) from the current solution in order to reduce tour length, we must introduce arcs (A, E) and (B, F), since introducing (A, F) leads to a subtour solution and introducing (A, B) leads to the original solution. Note that symmetry is required here since the direction on arcs (B, C), (C, D), and (D, E) is reversed.

These branch exchange procedures are important since they illustrate a general approach to heuristics for combinatorial optimization problems. In addition, they have been used to generate excellent solutions to large-scale traveling salseman problems in a reasonable amount of time.

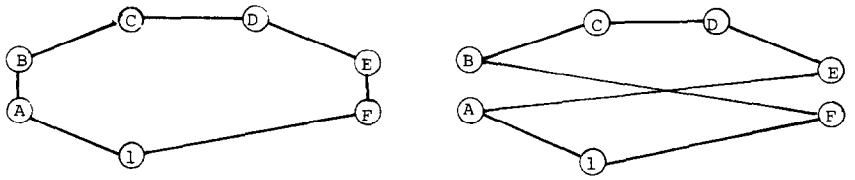


Fig. 2.10. Initial tour and feasible 2-opt swap.

In terms of worst case performance, Rosenkrantz, Stearns, and Lewis[568] give some partial results. For example, they show (assuming the triangle inequality) that a 2-optimal tour can be almost twice the length of an optimal tour.

Roughly speaking, a 3-opt procedure requires n times the work that a 2-opt procedure needs. To obtain close approximations to the length of the optimal tour using this strategy, one should repeat a 3-opt procedure for a number of starting tours[568]. Computationally, this approach becomes burdensome quickly. We, therefore, consider a class of heuristic methods which is computationally less expensive but as accurate as the repeated use of the 3–opt procedure. These are composite procedures.

2.3.3 *Composite procedures*

The basic composite procedure can be stated as follows:

Step 1. Obtain an initial tour using one of the tour construction procedures.

Step 2. Apply a 2-opt procedure to the tour found in Step 1.

Step 3. Apply a 3-opt procedure to the tour found in Step 2.

The composite procedure is relatively fast computationally and gives excellent results.

The idea behind the composite procedure is to get a good initial solution rapidly and hope that the 2-opt and 3-opt procedures will then find an almost-optimal solution. In this way, the 3-opt procedure which is the computationally most expensive step of the three, need only be used once.

The following are some possible variants of the composite procedure outlined above:

Variant 1. Run the composite procedure without Step 2.

Variant 2. Run the composite procedure without Step 3. This variant gives very fast running times and very accurate results but would not be expected to perform as accurately as the three-step composite procedure.

Variant 3. Run the composite procedure a few times, using different tour construction algorithms in Step 1. This variant, although more expensive computationally than the three-step composite procedure, is expected to give better results.

Variant 4. In Step 1 of the composite procedure, only run the tour construction procedure from a small number of starting nodes. Then perform Steps 2 and 3 from the best of these solutions.

All of the heuristic algorithms described in this section are intended for symmetric TSP's (although some can be applied to asymmetric problems as well). Algorithms specifically designed for asymmetric TSP's have been proposed by Akl[4], Frieze *et al.*[236], Karp[373], Van Der Cruyssen and Rijckaert[659], and Kanellakis and Papadimitriou[370]. We discuss only the first of these methods here.

2.3.4 *Akl's directed TSP approach*

The procedure described in this section was especially designed for directed (i.e. asymmetric) TSP's by Akl[4] and is closely related to Christofides' heuristic. Before we describe the procedure, we need to define a *minimal directed spanning graph* (MDSG). An MDSG is a subgraph spanning n nodes of a complete directed graph with weights $w(i, j)$, which has minimum weight and forms a tree when directions are ignored. In other words, when directions are ignored, the graph is a minimal spanning tree with arc lengths $l_{ij} = \min\{w_{ij}, w_{ji}\}$.

Given a directed TSP on graph G, the algorithm stated below computes a nearly optimal solution. First, we present the steps briefly and then we comment on them in more detail.

Table 2.1. Directed TSP distance matrix

	1	2	3	4	5	6
1	∞	7	65	68	34	81
2	19	∞	22	27	59	29
3	14	43	∞	62	77	65
4	76	53	64	∞	6	51
5	39	58	38	27	∞	13
6	46	67	27	11	38	∞

Procedure

Step 1. Find a MDSG of G.

Step 2. Add a set of arcs to the MDSG in order to make the directed graph thus obtained Eulerian.

Step 3. Find an Euler cycle in this directed graph.

Step 4. Transform the Euler cycle into a Hamiltonian cycle.

Step 1 involves solving a minimal spanning tree (MST) on the graph G which is now taken to be undirected with arc costs $l_{ij} = \min\{w_{ij}, w_{ji}\}$. If an arc (i, j) appears in the resulting MST it is given the direction (i, j) if $l_{ij} = w_{ij}$ and (j, i) if $l_{ij} = w_{ji}$. This directed subgraph is an MDSG of G. In Step 2, we identify nodes with excess in-degree as supply points and nodes with excess out-degree as demand points and solve an associated transportation problem in order to balance the graph. Step 3 is straightforward (see Nijenhuis and Wilf[512]). Step 4 is also quite easy: Assume the Euler cycle is n_1–n_2–. . .–n_{l-1}–n_l where the n_i's are not necessarily distinct nodes. A Hamiltonian cycle can be formed by starting at node n_1, moving to the right and introducing a node into the Hamiltonian cycle only if it appears for the first time. In order to generate all Hamiltonian cycles obtainable from the Euler cycle, two copies of the Euler cycle are placed contiguously, n_1–n_2–. . .–n_{l-1}–n_l–n_1–n_2–. . .–n_{l-1}–n_l and the method just described is repeated l times, every time using the next node in $\{n_1, n_2, \ldots, n_l\}$ as a start node.

We now illustrate the procedure by quickly working through the small example shown in Table 2.1. Note that in this example, which is taken from Akl[4], the triangle inequality is not preserved.

The results of Steps 1, 2 and 4 are displayed in Fig. 2.11–2.13. The length of the resulting tour is 94 units.

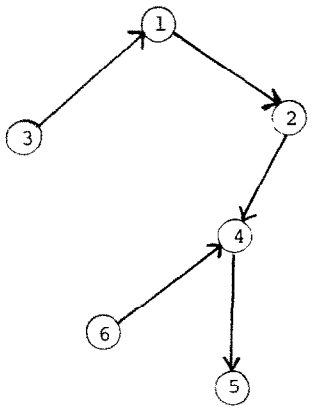


Fig. 2.11. Directed TSP step 1.

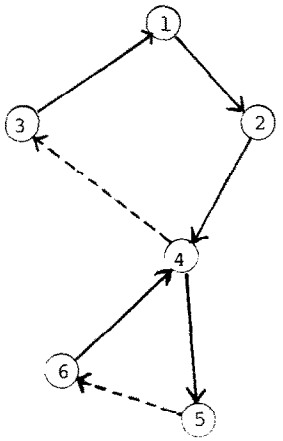


Fig. 2.12. Directed TSP step 2.

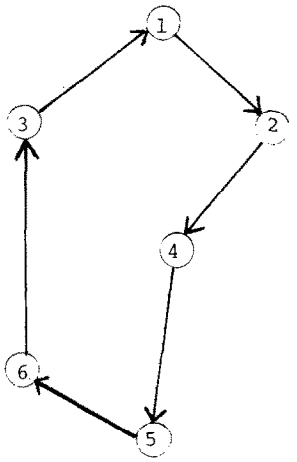


Fig. 2.13. Directed TSP step 4.

2.3.5 *Heuristic approaches to the time-constrained traveling salesman problem*

Suppose that each ordered pair (i, j) of nodes has associated with it a net profit and a travel time. In the time-constrained TSP, the objective is to find a subtour that begins and ends at the origin (node 1), which maximizes profit while requiring less than τ units of time. This is a much more realistic statement of the problem a traveling salesman faces and it has been the focus of research attention by Cloonan[146], Gensch[264], and, more recently, Golden *et al.*[304].

In particular, let $p = [p(i, j)]$ be the net profit matrix and $t = [t(i, j)]$ be the time matrix and let both of these be $N \times N$. We now present an iterative heuristic solution procedure. At each iteration l, we let P and T denote the total profit and total time of the subtour just generated. Also, ΔP and ΔT are the changes in total profit and time associated with each permissible insertion. Our initial idea was to use the ratio $\Delta P/\Delta T$ as a measure of the attractiveness of an insertion. However, since these changes might be negative, we sought an alternative approach.

Procedure TCTSP

Step 0. Set l, P, and T to 0. Initialize R_l. Choose α.

Step 1. For each node i, compute

$$\Delta T = t(1, i) + t(i, 1),$$

and

$$\Delta P - R_0 \Delta T = \{p(1, i) + p(i, 1)\} - R_0\{t(1, i) + t(i, 1)\}.$$

Find the node i^* such that $\Delta T \leq \tau$ and $\Delta P - R_0 \Delta T$ is maximized. If no such node exists, stop. Otherwise, form the subtour $1 - i^* - 1$ and record it. Set P to $P + \Delta P$ and T to $T + \Delta T$.

Step 2. $l \leftarrow l + 1$

$$R_l = \alpha(P/T) + (1 - \alpha) R_{l-1}.$$

Step 3. For each node k not in the present subtour and adjacent nodes i and j in the subtour, compute

$$\Delta T = t(i, k) + t(k, j) - t(i, j),$$

and

$$\Delta P - R_l \Delta T = \{p(i, k) + p(k, j) - p(i, j)\} - R_l\{t(i, k) + t(k, j) - t(i, j)\}.$$

Find the i, k, j triple i^*, k^*, j^* such that $T + \Delta T \leq \tau$ and $\Delta P - R_l \Delta T$ is maximized. If no such triple exists, go to Step 5. Otherwise, insert k^* between i^* and j^* in the subtour and record the subtour. Set P to $P + \Delta P$ and T to $T + \Delta T$.

Step 4. Go to Step 2.

Step 5. From all the subtours recorded, select the one with the largest P.

Several points still need clarification. First, the ratio P/T is the worth of a unit of time in the current subtour. R_l is the best estimate of the worth of a unit of time at iteration l. That is, R_l is an estimate which takes into account all previous ratios P/T but weights the more recent ones more heavily. R_o should represent an educated guess (possibly based on preliminary analysis) of the profit to time ratio for the optimal subtour. The expression for R_l is the standard exponential smoothing model that is used in a variety of forecasting situations. Finally, in order to promote flexibility and generate a number of heuristic solutions, one may vary α between 0 and 1 and select different initial values for R_0. We remark that each application of this insertion procedure requires on the order of N^3 operations in the worst case.

Consider a traditional traveling salesman problem with arc lengths $l(i, j)$, in which we seek a minimum-length Hamiltonian tour over the points. Procedure TCTSP will generate a near-optimal solution to the TSP if we set

$$p(i, j) = \tau - l(i, j), \text{ and}$$

$$t(i, j) = 0 \text{ for all } i \text{ and } j,$$

Table 2.2. TCTSP example

p(i,j)

	1	2	3	4	5
1	-	160	125	35	53
2	-48	-	86	2	58
3	-35	106	-	80	65
4	-50	97	152	-	60
5	-37	148	125	55	-

t(i,j)

	1	2	3	4	5
1	-	26	26	15	14
2	8	-	30	19	16
3	6	28	-	12	16
4	8	30	25	-	16
5	4	24	26	13	-

where $\tau \geq 2 \max\{l(i, j)\}$. In this case, the heuristic becomes the well-known cheapest-insertion algorithm. The transformation from $l(i, j)$ to $p(i, j)$ is needed in order to ensure that each node will be included in the subtour obtained.

An example of a time-constrained TSP with $\tau = 72$ is given in Table 2.2.
If we let $R_0 = 1$ and $\alpha = 1$, Procedure TCTSP determines the subtour 1–2–5–4–1 with a total profit of 223 and a total time of 63. The intermediate subtours are 1–2–1 and 1–2–5–1 with total profits of 112 and 181 and total travel times of 34 and 46 units. The *optimal* solution is 1–3–4–5–1 with a total profit of 228 and a total travel time of 58.

2.4. SINGLE DEPOT/MULTIPLE VEHICLE ROUTING

2.4.1 *Introduction*

In addition to its practical relevance, which has already been illustrated, the routing and scheduling of vehicles is an area of importance to operations research theoreticians as well. Recent research in this field has provided significant breakthroughs in problem formulations and in the design and analysis of algorithms. These advances have an important bearing on future research in routing and scheduling, as well as in related combinatorial and computational research endeavors.

In this section we focus primarily on *node* routing problems in which demands at *nodes* must be satisfied by a *fleet* of vehicles that is domiciled at a central depot. In a later section, we study multi-vehicle *arc* routing problems.

2.4.2 *The multiple traveling salesmen problem*

The multiple traveling salesmen problem (MTSP) is a generalization of the traveling salesman problem (TSP) that comes closer to accommodating real world problems where there is a need to account for more than one salesman (vehicle). Multiple traveling salesmen problems arise in various scheduling and sequencing applications. For example, this framework could be used to develop the basic route structure for a pickup or delivery service (perhaps a school bus or rural bus service—see Chapter 4). It has already proved to be an appropriate model for the problem of bank messenger scheduling, where a crew of messengers picks up deposits at branch banks and returns them to the central office for processing (see [638]).

In the multiple traveling salesmen problem, M salesmen are to visit the nodes of a given n node network in such a way that the total distance traveled by all M salesmen is a minimum. Each salesman must travel along a subtour of the nodes, which includes a common depot, and every node except the depot must be visited exactly once by exactly one salesman. The mathematical programming formulation of the (MTSP) displayed below is a natural extension of the assignment-based formulation of the traveling salesman problem.

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.15}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = b_j = \begin{cases} M & \text{if } j = 1 \\ 1 & \text{if } j = 2,3,\ldots,n \end{cases} \tag{2.16}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = \begin{cases} M & \text{if } i = 1 \\ 1 & \text{if } i = 2,3,\ldots,n \end{cases} \tag{2.17}$$

$$X = (x_{ij}) \in S \tag{2.18}$$

$$x_{ij} = 0 \text{ or } 1 \quad (i, j = 1), \ldots, n) \tag{2.19}$$

for any choice of the set S that breaks subtours which do not include the origin (node 1). In particular, any of the choices for S given earlier can be used. Note that (2.16) requires that *all* M salesman be used.

The apparent generalization that the MTSP provides is somewhat illusory, since any problem of this type can be converted into an equivalent TSP. Equivalent TSP formulations of the MTSP were derived independently by Bellmore and Hong[59], Orloff[524], Svestka and Huckfeldt[638], and Eilon *et al.*[201]. The equivalence is obtained by creating M copies of the depot denoted $D_1, D_2, \ldots, D_M$, each connected to the other nodes exactly as was the original depot (with the same lengths). The M copies are not connected, or are connected by arcs each with a length exceeding

$$\sum_{i=1}^{n} \sum_{j=1}^{n} |c_{ij}|.$$

In this way, an optimal single salesman tour in the expanded network (over node set $\{D_1, D_2, \ldots, D_M, 2, 3, \ldots, n\}$) will never use an arc connecting two copies of the depot. Then, by coalescing the copies back into a single node, the single salesman tour decomposes into M subtours as required in the MTSP.

For example, consider a four-node problem with two salesmen. Originally, we have nodes 1, 2, 3 and 4 with node 1 as the origin or central depot. Our expanded network is comprised of nodes D_1, D_2, 2, 3 and 4 where nodes D_1 and D_2 represent the two salesmen. In Fig. 2.14, we see that tour D_1–3–D_2–4–2–D_1 in the expanded network is equivalent in the two-salesmen interpretation to subtours 1–3–1 and 1–4–2–1.

This useful transformation therefore allows one to solve MTSP's by using any of the algorithms for the TSP described in Sections 2.2 and 2.3. We should remark, however, that Laporte and Nobert[414] report that it is more efficient to solve the MTSP directly using a constraint relaxation approach.

2.3.4 *Vehicle routing formulations*

The vehicle routing problem requires a set of delivery routes from a central depot to various demand points, each having service requirements, in order to minimize the total distance covered by the entire fleet. Vehicles have capacities and, possibly, maximum route time constraints. All vehicles start and finish at the central depot. We will refer to the following formulation for this problem, which is due to Golden *et al.* [306], as the generic vehicle routing problem. If the maximum route time constraints are omitted, we obtain the standard vehicle routing problem (VRP). The problem as stated is a pure delivery problem. In the case where only pick-ups are made instead, we have an equivalent problem. When both pick-ups and deliveries are present simultaneously, we face a more complex problem that we shall discuss later in Chapter 4.

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{v=1}^{NV} c_{ij} x_{ij}^{v} \tag{2.20}$$

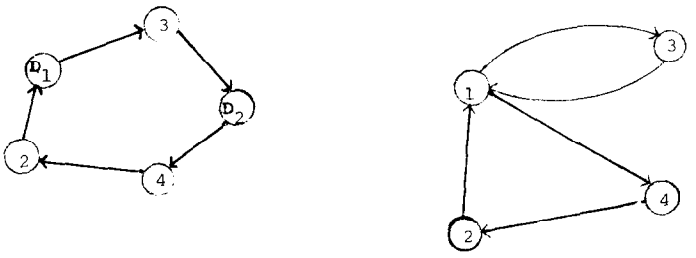


Fig. 2.14. A two-salesmen problem.

$$\text{subject to} \quad \sum_{i=1}^{n} \sum_{v=1}^{NV} x_{ij}^{v} = 1 \qquad (j = 2, \ldots, n) \tag{2.21}$$

$$\sum_{j=1}^{n} \sum_{v=1}^{NV} x_{ij}^{v} = 1 \qquad (i = 2, \ldots, n) \tag{2.22}$$

$$\sum_{i=1}^{n} x_{ip}^{v} - \sum_{j=1}^{n} x_{pj}^{v} = 0 \qquad \begin{matrix} (v = 1, \ldots, NV; \\ p = 1, \ldots, n) \end{matrix} \tag{2.23}$$

$$\sum_{i=1}^{n} d_i \left(\sum_{j=1}^{n} x_{ij}^{v} \right) \leq K_v \qquad (v = 1, \ldots, NV) \tag{2.24}$$

$$\sum_{i=1}^{n} t_i^{v} \sum_{j=1}^{n} x_{ij}^{v} + \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^{v} x_{ij}^{v} \leq T_v \qquad (v = 1, \ldots, NV) \tag{2.25}$$

$$\sum_{j=2}^{n} x_{1j}^{v} \leq 1 \qquad (v = 1, \ldots, NV) \tag{2.26}$$

$$\sum_{i=2}^{n} x_{i1}^{v} \leq 1 \qquad (v = 1, \ldots, NV) \tag{2.27}$$

$$X \in S \tag{2.28}$$

$$x_{ij}^{v} = 0 \text{ or } 1 \qquad \text{for all } i, j, v, \tag{2.29}$$

where n = number of nodes; NV = number of vehicles; K_v = capacity of vehicle v; Tv = maximum time allowed for a route of vehicle v; d_i = demand at node i ($d_1 = 0$); t_1^{v} = time required for vehicle v to deliver or collect at node i($t_1^{v} = 0$); t_{ij}^{v} = travel time for vehicle v from node i to node j($t_{ii}^{v} = \infty$); c_{ij} = cost of travel from node i to node j; $x_{ij}^{v} = 1$ if arc $i-j$ is traversed by vehicle v, 0 otherwise; X = matrix with components $x_{ij} \equiv \sum_{v=1}^{NV} x_{ij}^{v}$, specifying connections regardless of vehicle type.

Equation (2.20) states that total distance is to be minimized. Alternatively, we could minimize costs by replacing c_{ij} by a cost coefficient c_{ij}^{v} which depends upon the vehicle type. Equations (2.21) and (2.22) ensure that each demand node is served by exactly one vehicle. Route continuity is represented by equations (2.23), i.e. if a vehicle enters a demand node, it must exit from that node. Equations (2.24) are the vehicle capacity constraints; similarly, equations (2.25) are the total elapsed route time constraints. For instance, a newspaper delivery truck may be restricted from spending more than one hour on a tour in order that the maximum time interval from press to street be made as short as possible. Equations (2.26) and (2.27) guarantee that vehicle availability is not exceeded. Finally, the subtour-breaking constraints (2.28) can be, with slight modifications, any of the equations specified previously. Since (2.21) and (2.23) imply (2.22), and (2.23) and (2.26) imply (2.27), from now on we consider the model to include (2.20)–(2.29) excluding (2.22) and (2.27), which are redundant. We assume that $\max_{1 \leq i \leq n} d_i < \min_{1 \leq v \leq NV} K_v$. That is, the demand at each node does not exceed the capacity of any truck. In our model we have assumed that when a demand node is serviced, its requirements are satisfied completely, that is, partial servicing of demand is excluded.

Observe that when vehicle capacity constraints (2.24) and route time constraints (2.25) are nonbinding, and can be ignored, this model reduces to a MTSP as can be seen by eliminating constraints (2.23) and substituting $x_{ij} = \sum_{v=1}^{NV} x_{ij}^{v}$ in the objective function and remaining constraints.

Note that since $\sum_{j=2}^{n} x_{1j}^{v}$ is 1 or 0 depending upon whether or not the v^{th} vehicle is used, a fixed acquisition cost $f_v \sum_{j=2}^{n} x_{1j}^{v}$ can be added to the model when it is formulated with an objective function to investigate trade-offs between routing and vehicle acquisition (or fixed) costs.

We can, without difficulty, generalize the generic model to the multicommodity case where several different types of products must be routed simultaneously over a network in order to satisfy demands at delivery points for the various products.

Furthermore, we can incorporate timing restrictions (time windows) into the vehicle dispatching model. If we define a_j as the arrival time at node j, then restrictions on delivery deadlines $\bar{a}_j$ and earliest delivery times $\underline{a}_j$ can be represented by the following nonlinear equations:

$$a_j = \sum_v \sum_i \left(a_i + t_i^v + t_{ij}^v\right) x_{ij}^v \qquad (j = 1, \ldots, n)$$

$$a_i = 0$$

$$\underline{a}_j \leq a_j \leq \bar{a}_j. \qquad (j = 2, \ldots, n).$$

Alternatively, the above nonlinear constraints can be replaced by the linear constraints

$$\left.\begin{aligned} a_j &\geq (a_i + t_i^v + t_{ij}^v) - (1 - x_{ij}^v)\, T \\ a_j &\leq (a_i + t_i^v + t_{ij}^v) + (1 - x_{ij}^v)\, T \end{aligned}\right\} \text{for all } i, j, v.$$

where $T = \max_{1 \leq v \leq NV} T_v$. When $x_{ij}^v = 0$, these constraints are redundant. When $x_{ij}^v = 1$, they determine a_j in terms of the arrival time a_i at node i preceding node j on a tour, the delivery time t_i^v at node i, and the travel time t_{ij}^v between nodes i and j. Of course, these constraints add considerably to the size of the model (2.20)–(2.29). Their role is to introduce a vehicle scheduling component into the routing problem. Time windows are more fully discussed in Chapter 4.

2.4.4 *Classification of solution strategies in vehicle routing*

Most solution strategies for vehicle routing problems can be classified as one of the following approaches: (1) cluster first—route second; (2) route first—cluster second; (3) savings/insertion; (4) improvement/exchange; (5) mathematical-programming-based; (6) interactive optimization; (7) exact procedures. The first four approaches as well as the last one have been used extensively in the past. The other two approaches represent relatively recently developed ideas. A more general framework for heuristic algorithms is given by Ball and Magazine[39].

Cluster first-route second procedures group or cluster demand nodes and/or arcs first and then design economical routes over each cluster as a second step. Examples of this idea are given by Gillett and Miller[274], Gillett and Johnson[273], Chapleau *et al.*[120], and Karp[376] for the standard single depot vehicle routing problem.

Route first-cluster second procedures work in the reverse sequence. First, a large (usually infeasible) route or cycle is constructed which includes all of the demand entities (that is, nodes and/or arcs). Next, the large route is partitioned into a number of smaller, but feasible, routes. Golden *et al.*[296] provide an algorithm that typifies this approach for a heterogenous fleet size vehicle routing problem. Newton and Thomas[506] and Bodin and Berman[82] use this approach for routing school buses to and from a single school, and Bodin and Kursh[87, 88] utilize this approach for routing street sweepers. See also the work of Stern and Dror[623]. These applications are described in greater detail in Chapter 4.

Savings or *insertion* procedures build a solution in such a way that at each step of the procedure (up to and including the penultimate step) a current configuration that is possibly infeasible is compared with an alternative configuration that may also be infeasible. The alternative configuration is one that yields the largest savings in terms of some criterion function, such as total cost, or that inserts least expensively a demand entity not in the current configuration into the existing route or routes. The procedure eventually concludes with a feasible configuration. Examples of savings/insertion procedures for single depot node and arc routing problems are described by Clarke and Wright[145], Golden *et al.*[306], Norback and

Love[514], and Golden and Wong[310]. Hinson and Mulherkar[350] use a variation of this procedure for routing airplanes.

Improvement or *exchange* procedures such as the well-known branch exchange heuristic developed by Lin[444] and Lin and Kernighan[446] and extended by Christofides and Eilon[139] and Russell[572] always maintain feasibility and strive towards optimality. At each step, one feasible solution is altered to yield another feasible solution with a reduced overall cost. This procedure continues until no additional cost reductions are possible. Bodin and Sexton[94] modify this approach in order to schedule mini-buses for the subscriber dial-a-ride problem.

Mathematical programming approaches include algorithms that are directly based on a mathematical programming formulation of the underlying routing problem. An excellent example of mathematical-programming-based procedures is given by Fisher and Jaikumar[219]. They formulate the Dantzig-Ramser vehicle routing problem as a mathematical program in which two interrelated components are identified. One component is a traveling salesman (routing) problem and the other is a generalized assignment (packing) problem. Their heuristic attempts to take advantage of the fact that these two problems have been studied extensively and powerful mathematical programming approaches for their solution have already been devised. We discuss their heuristic in detail later. The algorithm due to Krolak and Nelson[404] is similar in spirit. Christofides *et al.*[143] and Stewart and Golden[628] discuss Lagrangian relaxation procedures for the routing of vehicles (see also Section 2.2). In addition, the article by Christofides, Mingozzi, and Toth[144]] represents a mathematical-programming-based (in particular, dynamic programming) approach for obtaining lower bounds in a variety of combinatorial optimization problems related to vehicles routing. Further discussion on this topic can be found in the article by Magnanti[455].

Interactive optimization is a general-purpose approach in which a high degree of human interaction is incorporated into the problem-solving process. The idea is that the experienced decision-maker should have the capability of setting the revising parameters and injecting subjective assessments based on knowledge and intuition into the optimization model. This almost always increases the likelihood that the model will eventually be implemented and used. Some early adaptations of this approach to vehicle routing problems are presented by Krolak, Felts and Marble[402] and Krolak, Felts and Nelson[403]. The paper by Cullen, Jarvis and Ratliff[156] introduces several rather novel interactive optimization heuristics.

Exact procedures for solving vehicle routing problems include specialized branch and bound, dynamic programming and cutting plane algorithms. Some of the more effective TSP approaches are described by Held and Karp[337, 338], Hansen and Krarup[327], Miliotis[480, 481], Balas and Christofides[29] and Crowder and Padberg[155]. Christofides *et al.*[143] discuss exact algorithms for the VRP. As noted before, any relaxation procedure that can provide good lower bounds on the optimal value of the vehicle routing problem can be embedded within a branch and bound approach to yield an exact procedure.

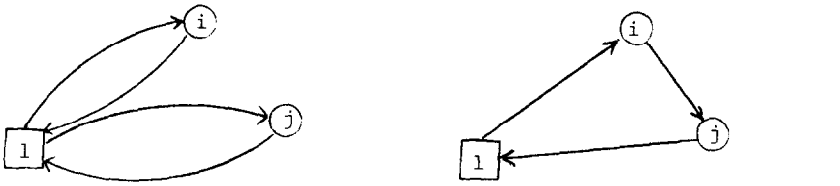
2.5. SELECTED SOLUTION TECHNIQUES FOR SINGLE DEPOT VRP's

As pointed out in the previous section, there are exact algorithms that solve the vehicle routing problem optimally by branch and bound techniques and there are heuristic algorithms that solve the problem approximately. Since the exact algorithms have been viable only for very small problems, we concentrate on heuristic algorithms. To our knowledge, the largest vehicle routing problem of any complexity that has been solved exactly involved about 25 customers[143]. We will now discuss several vehicle routing heuristic methods for a homogeneous fleet that have been used for problems up to 1000 customers.

The savings algorithm. The savings algorithm due to Clarke and Wright[111] is an "exchange" procedure in the sense that at each step one set of tours is exchanged for a better set. Initially, we suppose that every demand point is supplied individually by a separate vehicle (refer to Fig. 2.15 a).

Now, if instead of using two vehicles to service nodes i and j, we use only one, then we obtain a savings s_{ij} in travel distance of

$$(2c_{1i} + 2c_{1j}) - (c_{1i} + c_{1j} + c_{ij}) = c_{1i} + c_{1j} - c_{ij}$$



(a) Initial Setup (b) Nodes i and j have been linked

Fig. 2.15. Tour aggregation

(see Fig. 2.15 b). We note that if distances are asymmetric then $s_{ij} = c_{i1} + c_{1j} - c_{ij}$. For every possible pair of tour end points i and j there is a corresponding savings s_{ij}. We order these savings from largest to smallest and starting from the top of the list we link nodes i and j with maximum savings s_{ij} unless the problem constraints are violated.

The sweep approach. The sweep approach was originally devised by Gillett and Miller[274]. This approach, which is an efficient algorithm for problems of up to about 250 nodes, constructs a solution in two stages. First, it assigns nodes to vehicles and then it sequences the order in which each vehicle visits the nodes assigned to it. Rectangular coordinates for each demand point are required, from which we may calculate polar coordinates. We select a "seed" node randomly. With the central depot as the pivot, we start sweeping (clockwise or counterclockwise) the ray from the central depot to the seed. Demand nodes are added to a route as they are swept. If the polar coordinate indicating angle is ordered for the demand points from smallest to largest (with seed's angle 0) we enlarge routes as we increase the angle until capacity restricts us from enlarging a route by including an additional demand node (note that the procedure does not explicitly account for timing constraints); this demand point becomes the seed for the following route. Once we have the routes, we can apply TSP algorithms such as the 3-opt heuristic to improve tours. In addition, we can vary the seed and select the best solution. Figure 2.16 illustrates this procedure.

A penalty algorithm. Stewart and Golden[628] and Stewart[627] present a heuristic algorithm that treats the capacity constraints by moving them into the objective function and applying a multiplier in order to impose a penalty when demand on a route exceeds capacity. This explicit consideration of customer demand tends to provide better results.

We will assume that the vehicle fleet is homogeneous and that no timing restrictions are imposed. In this instance, the mathematical programming formulation of the general VRP given earlier becomes:

$$\text{Minimize} \quad \sum_{v} \sum_{i,j} c_{ij} x_{ij}^{v} \tag{2.30}$$

subject to

$$\sum_{i,j} d_i x_{ij}^{v} \leq K \quad \text{for all} \quad v = 1, \ldots, M \tag{2.31}$$

$$X = [x_{ij}^{v}] \in S^{*} \tag{2.32}$$

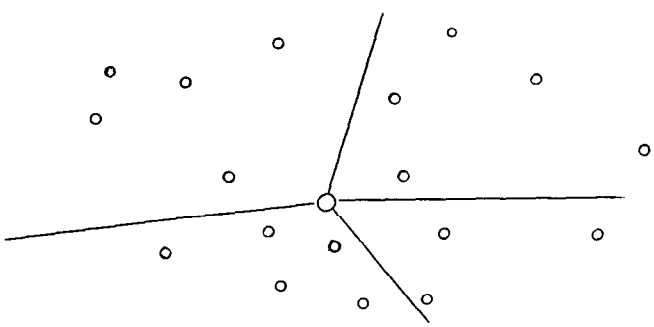


Fig. 2.16. Clusters generated by sweep procedure.

where c_{ij} = the distance or cost of traveling from i to j; $x_{ij}^v = 1$ if i and j are joined on route v, 0 otherwise; d_i = the demand at point i; K = the vehicle capacity; M = the number of vehicles; S^* = the set of all M-traveling salesman solutions. Without loss of generality, we can impose the additional redundant constraints

$$\sum_{i,j} d_i x_{ij}^k \geq \sum_{i,j} d_i x_{ij}^l \text{ for } l = k+1 \text{ and } k = 1, \ldots, M-1 \tag{2.33}$$

to the previous mathematical (integer) program. The effect is merely to assign the largest demand route—number 1, the second largest—number 2, and so on.

A Lagrangean problem associated with the VRP given in (2.30)–(2.33) is of the form

$$\text{Minimize} \quad \sum_{v} \sum_{i,j} c_{ij} x_{ij}^v + \lambda \sum_{i,j} d_i x_{ij}^l \tag{2.34}$$

subject to

$$\sum_{i,j} d_i x_{ij}^k \geq \sum_{i,j} d_i x_{ij}^l \text{ for } l = k+1 \text{ and } k = 1, \ldots, M\text{-}1 \tag{2.35}$$

$$X = [x_{ij}^v] \in S^* \tag{2.36}$$

where $\lambda \geq 0$ may be interpreted as a penalty for route failure.

We next present an algorithm that exploits this formulation. The idea is to start with a value of λ and solve problem (2.34)–(2.36) for $x(\lambda)$. Next, the algorithm provides a mechanism for varying λ in order to approach the optimal solution to the VRP. Each time λ is varied, problem (2.34)–(2.36) is solved for $x(\lambda)$ until "near optimality" is attained for the VRP. The procedure is heuristic due to the fact that (2.34)–(2.36) is solved approximately and not exactly. Moreover, even if (2.34)–(2.36) were solved exactly, the result might not be an optimal solution to the VRP since there may be a duality gap between the objective value for the "best" $x(\lambda)$ and the optimal solution to the VRP. The steps are described below:

Step 1. Set $\lambda = 0$. Solve the resulting MTSP. If the solution satisfies the capacity constraints, terminate. Otherwise, continue.

Step 2. Select $\lambda > 0$ and $\Delta > 0$.

Step 3. Solve the problem given in (2.34)–(2.36). This is an MTSP in which the largest demand route has penalty λ. If $x(\lambda)$ is feasible for the VRP, go to Step 5. Otherwise, continue.

Step 4. Route 1 is above vehicle capacity and has route demand y_1. Set $\lambda \leftarrow \lambda + \Delta$ and go to step 3.

Step 5. Search for the smallest value of λ, call it λ^*, between $\lambda - \Delta$ and λ such that $x(\lambda^*)$ is a feasible solution to the VRP. Then terminate with solution $x(\lambda^*)$.

The motivation underlying the penalty algorithm is that if λ is increased, the largest demand route is made smaller—thus driving infeasible solutions feasible. Conversely, if λ is decreased, the largest demand route becomes cheaper and is loaded more fully, thus, taking better advantage of vehicle capacity. This obvious relationship between λ and feasibility indicates why the redundant constraints (2.33) were introduced in the first place.

Some specific comments concerning the nature of the algorithm still need to be made. Step 1 can be implemented in a number of ways. One possibility is to solve the VRP using the savings procedure and then to combine small demand routes so as to form exactly M routes. These need not all be feasible. An alternative is to convert the MTSP into a TSP and apply one of the numerous heuristics for the TSP discussed previously. In step 3, a modified 3-opt procedure is used to solve the MTSP in which the largest demand route is penalized. The modification entails recalculating total demand on each route at each step, and associating λ with the largest demand route. In other words, the problem given in (2.34)–(2.36) is solved approximately for each different value of λ. Figure 2.17 describes the nature of step 3. The solid lines in Fig. 2.17 (a) and (b) define a feasible solution to a 3-salesman problem of the form (2.34)–(2.36). In (b) the three nodes D_1, D_2 and D_3 corresponding to the salesmen in (a) are coalesced. In (a) the dashed

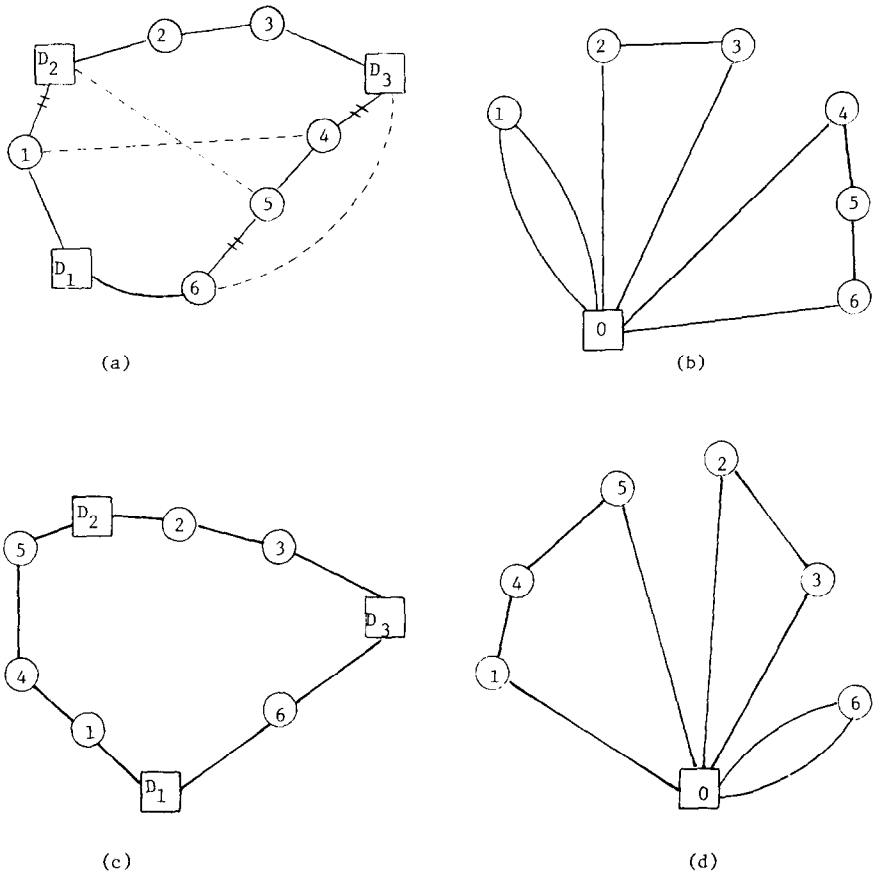


Fig. 2.17. Step 3—the swap operation

lines indicate a potential 3-swap. If the cost including the largest route penalty of λ times total demand associated with the new configuration (as shown in (c) and (d)) is less than the cost of the configuration in (a) and (b), the swap is performed. In step 3, we continue until it is no longer possible to reduce the cost (as measured by objective function (2.34)) via 3-swaps.

Step 4 is intended to help drive towards feasibility and in step 5 we strive for optimality. Finally, we point out that if different starting solutions are used in step 1, a number of different VRP solutions may be obtained, from which the best could then be selected. In particular, the savings algorithm can be reapplied t times using capacities $Q+\epsilon$, $Q+2\epsilon$, . . . , $Q+t\epsilon$.

M-Tour approach. This approach, due to Russell[572], is very similar to step 3 of the Penalty algorithm. The essential difference is that the M-tour algorithm requires a feasible solution to the VRP (with M vehicles) as input. This M-tour solution is expressed as a traveling salesman tour on an expanded network as in (a) and (b) of Fig. 2.17. Then, a modified 3-opt procedure or any other improvement scheme is used to reduce total cost. Before performing a profitable swap, however, a check for feasibility must be carried out. In that way, feasibility is preserved and total cost is improved at each step of the modified 3-opt procedure.

A generalized assignment heuristic. Now, we discuss a heuristic which views the VRP as consisting of two interrelated components. One component is a traveling salesman (routing) problem and the other is a generalized assignment (packing) problem. This heuristic was developed by Fisher and Jaikumar[219].

Before we discuss the mechanics of the heuristics algorithm, we formulate the VRP in such a way that the two components become self-evident. The formulation is given below.

$$\text{Minimize} \sum_{i,j,v} c_{ij} x_{ij}^{v} \tag{2.37}$$

subject to

$$\sum_i d_i y_{iv} \le K_v, \qquad v = 1, \ldots, \mathrm{NV} \tag{2.38}$$

$$\sum_v y_{iv} = \begin{cases} \mathrm{NV}, & i = 1 \\ 1, & i = 2, \ldots, n \end{cases} \tag{2.39}$$

$$y_{iv} = 0 \text{ or } 1, \quad \begin{matrix} i = 1, \ldots, n \\ v = 1, \ldots, \mathrm{NV} \end{matrix} \tag{2.40}$$

$$\left.\begin{aligned} \sum_i x_{ij}^v &= y_{jv}, \quad j = 1, \ldots, n && (2.41)\\ \sum_j x_{ij}^v &= y_{iv}, \quad i = 1, \ldots, n && (2.42)\\ \sum_{i,j \in L} x_{ij}^v &\le |L| - 1, \quad \begin{matrix} L \subseteq \{2, \ldots, n\} \\ 2 \le |L| \le n-2 \end{matrix} && (2.43)\\ x_{ij}^v &= 0 \text{ or } 1, \quad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, n \end{matrix} && (2.44) \end{aligned}\right\} v = 1, \ldots, \mathrm{NV}$$

where

$$y_{iv} = \begin{cases} 1 \text{ if customer } i \text{ is serviced by vehicle } v \\ 0 \text{ otherwise} \end{cases}$$

Constraints (2.38)–(2.40), which define a generalized assignment problem, guarantee that each customer i is assigned to a vehicle, that no vehicle load $\sum_i d_i y_{iv}$ exceeds its capacity and that each vehicle route contains the central depot (node 1). Whenever the variables y_{iv} are fixed to satisfy (2.38)–(2.40), the constraints (2.41)–(2.44) for each vehicle v define a TSP over the customers assigned to that vehicle.

An advantage of this formulation is that since the TSP and the generalized assignment problem have been studied extensively in the literature, this formulation of the VRP can benefit from the use of theory and algorithms already devised for the two component problems. In addition, the heuristic that we are to describe based upon this formulation will always find a feasible solution if one exists.

Now consider the following reformulation of the VRP as a nonlinear generalized assignment problem.

$$\text{Minimize} \quad \sum_v f(y_v) \tag{2.45}$$

subject to

$$\sum_i d_i y_{iv} \le K_v, \qquad v = 1, \ldots, \mathrm{NV} \tag{2.46}$$

$$\sum_v y_{iv} = \begin{cases} \mathrm{NV}, & i = 1 \\ 1, & 1 = 2, \ldots, n \end{cases} \tag{2.47}$$

$$y_{iv} = 0 \text{ or } 1, \quad \begin{matrix} i = 1, \ldots, n \\ v = 1, \ldots, \mathrm{NV} \end{matrix} \tag{2.48}$$

where $f(y_v)$ is the cost of an optimal TSP tour over all customers in $N(y_v) = \{i | y_{iv} = 1\}$. Since $f(y_v) = \text{Minimum } \{\sum_{i,j} c_{ij} x_{ij}^v |$ constraints (2.41)–(2.44) hold for fixed $v\}$, it is clear that $f(y_v)$ is

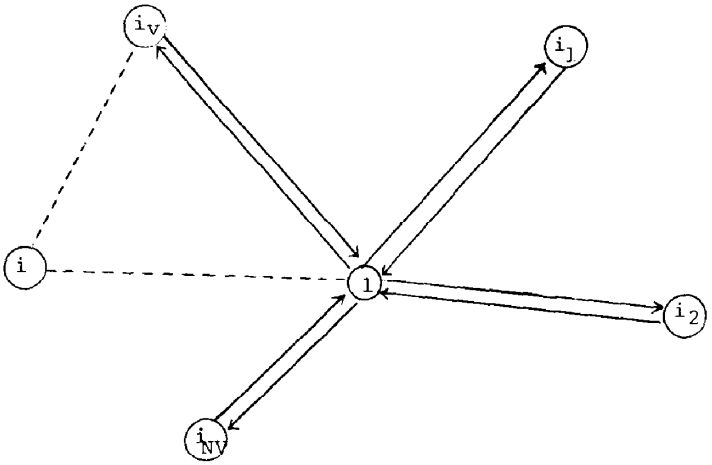


Fig. 2.18. Computation of $\tilde{c}_{iv}$.

indeed a complicated function. The generalized assignment heuristic replaces $f(y_v)$ in (2.45) with a linear approximation $\sum_i \tilde{c}_{iv} Y_{iv}$ and then solves (2.45)–(2.48).

The solution to the resulting linear generalized assignment problem defines a feasible assignment of customers to vehicles. Then, for each vehicle, we need to solve a TSP using either exact or approximate techniques. What remains to be shown is how one constructs an effective linear approximation of $f(y_v)$. We present one straightforward approach; there are a number of others that we do not pursue here.

Suppose we judiciously choose a set of "seed" customers $i_1, \ldots, i_{NV}$ to be assigned to vehicles $1, \ldots, NV$, respectively. We can set $\tilde{c}_{iv}$ equal to the cost incurred upon inserting customer i into route 1–i_v–1, as illustrated in Fig. 2.18. That is,

$$\tilde{c}_{iv} = \min\{c_{1,i} + c_{i,i_v} + c_{i_v,1},\ c_{1,i_v} + c_{i_v,i} + c_{i,1}\} - \{c_{1,i_v} + c_{i_v,1}\}.$$

Note that we are not assuming symmetry here. The seed customer may be varied and the procedure repeated. Also, the selection of seeds may be automated. Despite the simplicity of this linear approximation, the computational results have been extremely impressive.

2.6. MULTI-DEPOT VEHICLE ROUTING

Formulations. Minor alterations in the integer programming formulation of the vehicle routing problem given in (2.20)–(2.29) permit the modeling of multiple depot problems. Letting nodes $1, 2, \ldots, M$ denote the depots, we obtain the formulation by changing the index in constraints (2.21)–(2.22) to ($j = M+1, \ldots, n$) and ($i = M+1, \ldots, n$), respectively, by changing constraints (2.26) and (2.27) to

$$\sum_{i=1}^{M} \sum_{j=M+1}^{n} x_{ij}^{v} \le 1 \qquad (v = 1, \ldots, NV) \tag{2.26'}$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{n} x_{ip}^{v} \le 1 \qquad (v = 1, \ldots, NV), \tag{2.27'}$$

and be redefining our previous choices for the subtour breaking set S to be:

(1') $S = \{(x_{ij})$: $\sum_{i \in Q} \sum_{j \notin Q} x_{ij} \ge 1$ for every proper subset Q of $\{1, 2, \ldots, n\}$ containing nodes $1, 2, \ldots, M\}$;

(2') $S = \{(x_{ij})$: $\sum_{i \in R} \sum_{j \in R} x_{ij} \le |R| - 1$ for every nonempty subset R of $\{M+1, M+2, \ldots, n\}\}$;

$$(3')\quad S=\{(x_{ij}):\ y_i-y_j+nx_{ij}\le n-1 \quad \text{for } M+1\le i\ne j\le n \text{ for some real numbers } y_i\}.$$

Heuristic solution techniques for multi-depot VRP's. Whereas the single depot VRP has been studied widely, the multi-depot problem has attracted less attention. The relevant literature is represented by only a few papers. However, a number of relatively successful heuristic approaches, which we will discuss in this section, have been developed for the multi-depot problem.

The assignment— sweep approach. This method, proposed by Gillett and Johnson[273], is an extension of the sweep heuristic that solves the multi-depot problem in two stages. First, locations are assigned to depots. Then, several single depot VRP's are solved using the sweep heuristic. Each stage is treated separately.

Initially, all locations are in an unassigned state.
For any node i, let

$t'(i)$ be the closest depot to i, and $t''(i)$ be the second closest depot to i.

For each node i, the ratio

$$r(i)=c_{i,t'(i)}/c_{i,t''(i)}$$

is computed and all nodes are ranked by increasing values of $r(i)$. The arrangement determines the order in which the nodes are assigned to a depot: those that are relatively close to a depot are considered first. After a certain number of nodes have been assigned from the list of $r(i)$, a small cluster is formed around every depot.

Locations i such that the ratio $r(i)$ is close to 1, are assigned as follows. If two adjacent nodes j and k are already assigned to a depot, inserting i between j and k on a route linked to t creates an additional cost equal to $c_{ji}+c_{ik}-c_{jk}$ which represents a part of the total cost (or distance). In other words, the algorithm assigns location i to a depot t by inserting i between two nodes already assigned to depot t, in the least costly manner.

The sweep algorithm is used to construct and sequence routes in the cluster about each depot independently. A number of refinements are then made to improve the resulting solution.

A multi-depot savings approach. This procedure, due to Tillman and Cain[648] starts with the initial solution consisting of servicing each node exclusively by one route from the closest depot. Let c_{ij} = cost or distance between demand nodes i and j; and c_i^k = cost or distance between node i and depot k. The total initial cost of all routes is then $D=\sum_{i=1}^{N} 2\min\{c_i^k\}$ where N is the number of demand points. The method successively links pairs of nodes in order to decrease the total cost. One basic rule is assumed in the algorithm: the initial assignment of nodes to the nearest terminal is temporary, but once two or more nodes have been assigned to a common route from a terminal, the nodes are not reassigned to another terminal. In addition, as in the original savings algorithm, nodes i and j can be linked only if neither i nor j are interior to an existing tour.

At each step, the choice of linking a pair of nodes i and j on a route from terminal k is made in terms of the savings when linking i and j at k. Nodes i and j can be linked only if no constraints are violated.

The computation of savings is not as straightforward as in the case of a single depot. The savings s_{ij}^k are associated with every combination of demand nodes i and j and depot k, and represent the decrease in total cost or distance traveled when linking i and j at k. The formula is given by

$$s_{ij}^k=\tilde{c}_i^k+\tilde{c}_j^k-c_{ij}$$

where

$$\tilde{c}_i^k=\begin{cases}2\min_t\{c_i^t\}-c_i^k & \text{if } i \text{ has not yet been given a permanent assignment}\\ c_i^k & \text{otherwise.}\end{cases}$$

In the first case, node i is being reassigned from its closest depot and the previously assigned cost $2 \min_t \{c_i^t\}$ is saved; in the second case, node j is being inserted between depot k and node i and the link from i to k is dropped from the current solution.

The savings s_{ij}^k are computed for $i, j = 1, \ldots, N (i \neq j)$ and $k = 1, \ldots, M$ at each step. They can be stored in M matrices, each N by N. At each step, the link $i-j$ at k is chosen that maximizes s_{ij}^k and that yields a feasible solution (with regard to capacity and other restrictions).

A multi-depot savings algorithm for large problems. The multi-depot VRP can be viewed as being solved in two stages: first, nodes must be allocated to depots; then routes must be built that link nodes assigned to the same depot. Ideally, it is more efficient to deal with the two steps simultaneously as in the multi-depot savings algorithm just discussed. When faced with larger problems, with say 1000 nodes, however, this method is no longer tractable computationally. A reasonable approach would be to divide the problem into as many subproblems as there are depots and to solve each subproblem separately.

The algorithm presented in this section attempts to implement this strategy while using the ratios $r(i)$ introduced in the discussion of the assignment-sweep algorithm. The approach is due to Golden *et al.*[306]. If a given node i is much closer to one depot than any other depot, i will be served from its closest depot. When node i is equidistant from several depots, the assignment of i to a depot becomes more difficult. For every node i, we determine the closest depot k_1 and the second closest depot k_2. If the ratios $r(i) = c_i^{k_1}/c_i^{k_2}$ is less than a certain chosen parameter $(0 \leq \delta \leq 1)$, we assign i to k_1; in the case $r(i) \geq \delta$ we say that node i is a border node.

Clearly, if $\delta = 0$, all nodes are declared border nodes and if $\delta = 1$, all nodes are assigned to their closest depot. For a given problem, we can fix the number of border nodes as we wish, by varying δ.

The method proceeds as follows. In the first step, we ignore the nonborder nodes and the multi-depot savings algorithm is applied to the set of border nodes. The algorithm allocates the border nodes to depots and simultaneously builds segments of routes connecting these nodes. At the end of this first step, all nodes of our problem are assigned to some depot and all border nodes are linked on some routes. The solution to the VRP is produced depot by depot using single depot VRP techniques. The segments of routes that are built on border nodes are extended to the remaining nodes.

2.7. THE FLEET SIZE AND MIX PROBLEM

The goal of this section is to consider a situation in which the distributor decides, for the purposes of convenience and reliability, to *lease*, rather than *acquire*, vehicles for a certain period of time. The number of vehicles of each type needed to handle the system demand effectively must be determined together with a coherent routing policy for such vehicles. We assume that each vehicle type has a fixed leasing cost and a variable routing cost that is proportional to the distance traveled. The variable cost component encompasses the costs of fuel, maintenance, and manpower. This problem is formulated as a mathematical program by Golden *et al.*[296]. See Ball *et al.*[38] for a related study.

Golden *et al.*[296] also develop heuristics for solving this complex routing problem. We discuss some of these here. The most obvious approach for solving the fleet size and mix problem is to adapt the Clarke and Wright savings technique, discussed in Section 2.5. Remember that two distinct subtours containing customers i and j can be joined if: (a) i and j are not in the same subtour; (b) neither i nor j is an interior point in a subtour; and (c) the combined subtour containing both i and j has a total demand not in excess of vehicle capacity. The savings due to such a combination is $s_{ij} = c_{0i} + c_{0j} - c_{ij}$ and is computed for all $1 \leq i, j \leq n$ $(i \neq j)$. The Clarke and Wright procedure makes combinations in the order indicated by the savings (from largest to smallest), until no further feasible combinations exist.

This procedure is deficient for the fleet size and mix problem due to the fact that it ignores fixed vehicle costs. Since an infinite number of vehicles is assumed to be available, feasibility is certainly no problem. As a result, the savings method will tend to combine subtours until the capacity of the largest vehicle is reached. Unfortunately, a fleet composed almost entirely of largest vehicles may not be the least expensive. We denote this naive application of the Clarke and Wright approach to the fleet size and mix problem by CW.

It should now be clear that we need to extend the concept of savings to include fixed vehicle costs. Every subtour has a total cost equal to the sum of its routing costs and the cost of the vehicle servicing its customers. When two subtours combine to form a larger subtour, the total cost of this larger subtour can be calculated in a similar fashion. To determine which two subtours should combine at any point in the procedure, we need to find the subtours which generate the greatest total savings when combined. These savings must be calculated anew at every step, but all the other attractive features of the CW approach remain. This method, the Combined Savings (CS) approach, is the basis for the remaining savings algorithms that we consider in this section.

In order to give a precise representation of this algorithm, some additional notation is necessary. Let F be a real-valued function such that if z is the demand of a subtour, $F(z)$ is the fixed cost of the smallest vehicle that can service that demand. Clearly, F is a step function. Now consider a subtour I which has node i as one of its endpoints and a subtour J which has customer j as one of its endpoints. If the total demands on these two subtours are z_i and z_j, then the combined savings of joining customers i and j is

$$\bar{s}_{ij} = s_{ij} + F(z_i) + F(z_j) - F(z_i + z_j).$$

The algorithm terminates when $\bar{s}_{ij}$ is negative for all eligible i, j pairs.

The CS approach is a logical extension of the CW approach to the fleet size and mix problem. There is, however, and element of imprecision to the algorithm since $\bar{s}_{ij}$ represents only the *immediate* savings gained by joining customers i and j. Consider the following example. Suppose that subtours I, J and K each have total demand of 100 and are each serviced by a vehicle with a capacity of 100. Suppose further that the next largest vehicle has a capacity of 300 and is therefore considerably more expensive than the 100 capacity. It might be that having one 300 capacity vehicle service the combination of the I, J and K subtours is cheaper than having these routes individually serviced by three 100 capacity vehicles. But will this combined route form under the CS approach? Not necessarily. Since subtours can only combine two at a time, it is quite possible that the combined savings for, say, customers i and j (endpoints of subtours I and J) will be negative, due to the large fixed cost of the 300 capacity vehicle. In this situation, the CS approach ignores the fact that the 300 capacity vehicle would have an unused capacity of 100 units which could be used to absorb additional subtours later on.

These considerations motivated Golden *et al.*[296] to develop variants of the CS approach, which may be termed *opportunity* savings algorithms. Basically, these algorithms consider total savings to include savings in routing costs, savings in fixed vehicle costs, and opportunity savings. The first two savings are determined in the same manner as in the CS algorithm. Opportunity savings is a function of the unused capacity of the vehicle servicing the combined subtours.

The first opportunity savings algorithm is the Optimistic Opportunity Savings (OOS). In this algorithm, an opportunity savings is somewhat arbitrarily defined to be the cost of the smallest vehicle that can service the entire unused capacity of the new vehicle. For example, if the available vehicles have capacities of 50, 100 and 200 units, the total savings incurred by combining two subtours with total demands of 40 and 80 units into one with a total demand of 120 units would be $\bar{s}_{ij}$ plus the opportunity savings (the cost of a 100 capacity vehicle—the smallest vehicle that can service the unused capacity of 80 units). The OOS algorithm optimistically assumes that a subtour S with a total demand equal to the unused capacity on subtour R always exists and that at some future time subtour S will combine with subtour R and be absorbed by the vehicle servicing subtour R. The opportunity savings represents the fixed vehicle cost savings that would occur if this future combination materialized. The savings in routing cost from this future combination cannot be estimated as easily and is therefore ignored. Clearly, the OOS algorithm encourages subtours to combine more readily than the CS algorithm.

We now provide a formal description of the OOS algorithm. Let $P(z)$ be the capacity of the smallest vehicle that can service a subtour with a total demand of z. Thus, in the previous example, $P(40) = 50$, $P(80) = 100$, and $P(120) = 200$. The OOS algorithm is identical to the CS

approach, except that the savings upon linking customers i and j becomes

$$s_{ij}^{*} = s_{ij} + F(z_i) + F(z_j) - F(z_i + z_j) + F(P(z_i + z_j) - z_i - z_j).$$

Is the optimism of the OOS algorithm warranted? It would seem not. Overall, the CS approach outperforms the OOS algorithm. Actually, both methods are flawed. Essentially, the CS approach under-combines and the OOS algorithm over-combines subtours.

A remedial algorithm that comes to mind is the method of Realistic Opportunity Savings (ROS). This method is motivated by two basic assumptions. *First*, the sole purpose of opportunity savings should be to encourage the use of larger vehicles when it seems profitable to do so. Thus, opportunity savings should not be included in the savings formula unless combining two subtours forces the use of a larger vehicle. If the vehicle required to service the combined subtours is larger than each of the vehicles presently used, we say that a vehicle threshold has been crossed. Opportunity savings should be included only if a vehicle threshold is crossed. *Second*, based on empirical observations, the total demand of a subtour tends to be close to a vehicle capacity. Therefore, instead of assuming that the vehicle which might be absorbed in some future combination is the smallest vehicle that can service the unused capacity, it is perhaps more logical to use the largest vehicle that can be squeezed into the unused capacity. Note that this is the vehicle just smaller than the OOS vehicle, except when the unused capacity equals a vehicle capacity, in which case both algorithms use the same vehicle.

A full description of the ROS algorithm follows. Let $F'(z)$ be a function analogous to $F(z)$, except that $F'(z)$ represents the fixed cost of the largest vehicle whose capacity is less than or equal to z. If $z < a_1$, $F'(z) = 0$. Let I and J denote two subtours with total demands of z_i and z_j and let i and j be endpoints of subtours I and J. Assume, without loss of generality, that $z_i \geq z_j$. If $F(z_i + z_j) = F(z_i)$, then $\bar{s}_{ij}$ is the total savings from linking customers i and j as in the CS approach. If, however, $F(z_i + z_j) > F(z_i)$, then a vehicle threshold has been crossed and opportunity savings need to be considered. In this case, the total savings would be

$$s_{ij}' = \bar{s}_{ij} + F'(P(z_i + z_j) - z_i - z_j).$$

After total savings has been computed, the ROS algorithm proceeds in the same way as the CS approach.

Table 2.3. Summary of the savings algorithms

Algorithm	Savings	Savings Formula
CW	s_{ij}	$c_{0i} + c_{0j} - c_{ij}$
CS	$\bar{s}_{ij}$	$s_{ij} + F(z_i) + F(z_j) - F(z_i + z_j)$
OOS	s_{ij}^{*}	$\bar{s}_{ij} + F(P(z_i + z_j) - z_i - z_j)$
ROS	s_{ij}'	$\bar{s}_{ij} + \delta(w)\, F'(P(z_i + z_j) - z_i - z_j)$
ROS-γ	s_{ij}''	$s_{ij}' + (1-\gamma)\, c_{ij}$

where $F(z)$ = the fixed cost of the smallest vehicle that can service a demand of z

$P(z)$ = the capacity of the smallest vehicle that can service a demand of z

$F'(z)$ = the fixed cost of the largest vehicle that has a capacity less than or equal to z

$w = P(z_i + z_j) - P(\max\{z_i, z_j\})$

$$\delta(w) = \begin{cases} 0 & \text{if } w = 0 \\ 1 & \text{if } w > 0 \end{cases}$$

The computational results presented by Golden *et al.* [296] supported the claim that the ROS algorithm is superior to the OOS and CS algorithms. In seven of twelve problems tested, the ROS method performed at least as well as the other two algorithms. Its average and worst case behavior on the twelve problems was also superior to all other savings methods tested.

Still, none of the above methods are methods are powerful enough that single application will consistently generate good solutions. It is therefore necessary to vary these algorithms so that they produce a number of different solutions for each problem, with the least costly of these being selected as a final solution. A method used which introduces variety into the ROS algorithm is ROS-γ.

The ROS-γ algorithm uses a route shape parameter that changes the Clarke-Wright savings expression to $s_{ij} = c_{0i} + c_{0j} - \gamma \cdot c_{ij}$. In tests, Golden *et al.*[296] varied γ from 0.0 to 3.0, by tenths, for a total of 31 trials.

The logic behind the ROS-γ approach is fairly straightforward. A major flaw of the savings algorithms is that once a node is assigned to a tour, it cannot be reassigned to another at a later point. Two linkings may seem almost equivalent at an early stage of the algorithm and yet the choice between these can have enormous impact upon the final solution. Varying γ allows the approach to incorporate the variety which is essential to finding a good solution. The Realistic Opportunity Savings algorithm with route shape parameter seems to be the best of the savings algorithms that Golden *et al.* [296] have developed and tested. Additional heuristics and other aspects of the fleet size and mix problem are discussed by Golden *et al.*[296].

The savings algorithms described in this section are conveniently summarized in Table 2.3.

2.8. STOCHASTIC VEHICLE ROUTING

The vehicle routing problem (VRP), already discussed in detail, has been studied by many authors in recent years. This section treats a version of the stochastic vehicle routing problem (SVRP) which is a generalization of the deterministic VRP. The following modifications to the VRP are required:

(1) Customer demand is a random variable with a known probability distribution.

(2) Routes must be designed before the actual demands become known.

(3) The objective is to minimize expected travel distance, but other costs might be incurred if some customers cannot be serviced on a particular trip.

The SVRP is at least as difficult as the VRP. The presence of nonlinearities in the constraints and/or the objective function caused by the probabilistic demands may, in fact, make it considerably more difficult. For this reason, only heuristic solution methods have been considered for this problem.

Relatively little research has been conducted on the SVRP. Exceptions include the articles by Tillman[646], Stewart[625], Golden and Stewart[307], Golden and Yee[311], Yee and Golden[697], Stewart and Golden[631] and Cook and Russell[149]. Stewart's thesis[627] is the best source of material on this topic. Also see the recent work of Federgruen and Zipkin[208].

Some research that we describe has concentrated on transforming the SVRP to a VRP and then applying one of the existing VRP algorithms to this transformed problem. In what follows, artificial capacity will refer to a capacity to be used in a VRP algorithm in place of the true vehicle capacity. This artificial capacity will be less than the true capacity. Routing will be performed by viewing mean customer demands as artificial demands and filling vehicles up to their artificial capacity.

A mathematical formulation of the SVRP is now presented. When the demand at each customer location is a random variable, the problem becomes a stochastic vehicle routing problem and must be treated accordingly. In the formulation below, the demands d_i are random variables from a known probability distribution. The parameter α is the maximum allowable probability that any route fails.

Stochastic Vehicle Routing Problem (SVRP):

$$\text{Minimize} \sum_k \sum_{i,j} c_{ij} x_{ijk} \tag{2.49}$$

$$\text{Subject to Prob}\left\{\sum_{i,j} d_i x_{ijk} \leq Q\right\} \geq 1-\alpha \quad \text{for } k=1,\ldots,m \tag{2.50}$$

$$x=[x_{ijk}] \in S \tag{2.51}$$

where c_{ij} = the distance or cost of traveling from i to j; Q = the vehicle capacity; m = the number of vehicles; S = the set of all m-traveling salesman solutions; $x_{ijk}=1$ if i and j are joined on route k, 0 otherwise. Constraint (2.50) is referred to as a chance-constraint.

The chance-constrained model is useful when a penalty for violation of the capacity constraint cannot be explicitly defined. In this case, some subjective estimate of customer service in terms of the probability of route failure may be all that is obtainable.

Solution procedures. The solution procedures discussed in this section were designed so that existing heuristic algorithms for the VRP could be used to solve the SVRP with only slight modification. This is an important consideration since the existing VRP algorithms have been so well-studied in the literature. Other heuristics have been studied and are discussed in Stewart's thesis [627].

In the arguments that follow, we demonstrate that a chance constraint can be transformed into an equivalent deterministic constraint. In many cases, a closed-form expression for the artificial capacity can be derived from this equivalent constraint.

Consider a constraint of the form

$$\text{Prob}\left\{\sum_i a_i y_i \leq b\right\} \geq 1-\alpha. \tag{2.52}$$

If the a_i are independent and identically distributed random variables with mean $\bar{a}_i$ and standard deviation s_i and the y_i are decision variables, then the mean of $\sum_i a_i y_i$ is $M=\sum_i \bar{a}_i y_i$ and the standard deviation is given by $S=(\sum_i s_i^2 y_i^2)^{1/2}$. We now seek the constant T such that

$$\text{Prob}\left\{\left(\sum_i a_i y_i - M\right)/S \leq T\right\} = 1-\alpha. \tag{2.53}$$

If $b \geq M+TS$, then the l.h.s. of inequality (2.52) is also not less than $1-\alpha$. Hence, constraint (2.52) can be replaced by the deterministic constraint

$$M+TS \leq b. \tag{2.54}$$

This constraint is generally nonlinear in nature as seen below when we rewrite the constraint in terms of y_i:

$$\sum_i \bar{a}_i y_i + T\left(\sum_i s_i^2 y_i^2\right)^{1/2} \leq b \tag{2.55}$$

Under special conditions, however, this constraint can assume a linear form. We note that in the context of the SVRP the a_i's are probabilistic demands and the y_i's are 0–1 variables. Of course, b is the vehicle capacity. If the probability distributions are such that $S^2=\lambda M$ for some constant λ, which is indeed the case for the gamma, binomial, Poisson, and negative binomial distributions discussed by Golden and Yee [311], then (2.55) can be rewritten as

$$\sum_i \bar{a}_i y_i + T\left(\lambda \sum_i \bar{a}_i y_i\right)^{1/2} \leq b \tag{2.56}$$

or

$$M+T\sqrt{(\lambda M)} \leq b. \tag{2.57}$$

If this inequality is manipulated properly, it results in a simple expression of the form

$$M \leq \bar{b}. \tag{2.58}$$

Define $w = T\sqrt{\lambda}$ and note that

$$b - M \geq wM^{1/2}$$

or

$$(b - M)^2 \geq w^2 M$$

which yields

$$M \leq \frac{2b + w^2 - \sqrt{(w^4 + 4bw^2)}}{2} = \bar{b}. \tag{2.59}$$

If T is determined by Prob $(z \leq T) = 1 - \alpha$ where z is a unit normal variate, then this result is identical to the general expression derived by Golden and Yee[311].

Unfortunately, closed-form expressions for $\bar{b}$ (the artificial capacity) like eqn (2.59) do not exist for very many probability distributions. With this is mind, a more general algorithm is often required. Several such algorithms including a penalty function procedure are developed by Stewart[627] and Stewart and Golden[629].

2.9. THE CHINESE POSTMAN PROBLEM

Introduction. The problem of finding an optimal route for a single vehicle over a network is, as we have seen, a very difficult combinatorial problem. Routing problems can be classified as node routing problems, arc routing problems, or general routing problems. Bodin[79] and Bodin and Golden[85] provide a convenient taxonomy for these problems. The problem of visiting all nodes in a network in the minimal amount of time (node routing) is the classical Traveling Salesman Problem (TSP). The problem of covering all arcs of a network while minimizing the total distance traveled (arc routing) is the Chinese Postman Problem (CPP). The General Routing Problem (GRP) on network $G = G(N; A)$ (N is the set of all nodes, A the set of all arcs) is a generalization which includes the TSP and the CPP as special cases. Here we seek the minimum cost cycle which visits every node in subset $Q \subseteq N$ and covers every arc in subset $R \subseteq A$. Orloff introduces the GRP and later extends it to handle more realistic problem situations[523–525]. In this section, we concentrate primarily on the Chinese Postman Problem and arc routing problems in general. Applications include routing of street sweepers, snow plows, household refuse collection vehicles, postmen, the spraying of roads with salt-grit to prevent ice formation, the inspection of electric power lines, gas or oil pipelines for faults, etc. Since in many vehicle routing situations customers are so numerous that identifying them individually becomes cumbersome, we can consider the CPP to be the continuous counterpart to the discrete TSP. Here an arc replaces a number of customers.

Several variations of the CPP which have been studied are: the undirected postman problem (where all arcs are undirected), the directed postman problem (where all arcs are directed), and the mixed postman problem (where some arcs are directed and others are not). Historically, the CPP has been thought of as an undirected postman problem and in this chapter "CPP" will refer to the undirected case. Stricker[635], Christofides[128], Golden and Wong[310], and Golden *et al.*[303] consider an extension which we will call "the Capacitated Chinese Postman Problem" (CCPP), which reflects real-life situations more directly. Here, we are given arc demands $Q = [q_{ij}]$ which must be satisfied by vehicles of capacity W. A number of cycles must be formed which traverse every arc, satisfy demands, and yield minimum distance. This is an arc routing version of the VRP.

The undirected Chinese postman problem. The solution of the undirected Chinese postman problem relies on the notion of Euler tours. An Euler tour is a continuous path (tour) through a network that traverses each arc exactly once. Conditions for the existence of Euler tours in directed and undirected graphs were first given by Euler in 1736 in connection with the celebrated Konigsberg Bridge problem. Euler proved that such a tour exists in an undirected graph if and only if the degree (number of incident arcs) of every node is even. If a graph

contains odd nodes (nodes of odd degree) we must cover some arcs more than once in order to form a cycle covering all arcs.

Mei-Ko[477], in 1962, addressed the problem of minimizing the distance required to cover all arcs when odd nodes do exist (see also [476]). Because this work appeared in *Chinese Mathematics* the problem of finding the best postman tour on an undirected network has become known as the Chinese Postman Problem.

We present below the mathematical programming formulation for the CPP which has been given by Stricker[635]. Since Edmonds' matching theory results (discussed later) provide us with a graph-theoretic algorithm for solving this problem, the integer programming formulation is not solved by classical IP techniques. In fact, for most applications, these programs are too large for solution by anything but special purpose algorithms. However, the formulation is valuable in that it provides a framework for many similar problems, such as the CCPP, which are more complex.

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=i}^{n} c_{ij} x_{ij} \tag{2.49}$$

$$\text{subject to} \quad \sum_{k=1}^{n} x_{ki} - \sum_{k=1}^{n} x_{ik} = 0 \quad \text{for } i = 1, \ldots, n \tag{2.50}$$

$$x_{ij} + x_{ji} \geq 1 \quad \text{for all arcs } (i, j) \in A \tag{2.51}$$

$$x_{ij} \geq 0 \text{ and integer} \tag{2.52}$$

where n = the number of nodes in the network; A = the set of all arcs in the network; x_{ij} = the number of times arc (i, j) is traversed from i to j; c_{ij} = the length of arc (i, j).

We first observe that every graph must have an even number of odd nodes. If d_i is the degree of node i and m is the number of arcs, then $\sum_i d_i = \sum_{i \text{ even}} d_i + \sum_{i \text{ odd}} d_i = 2m$ because each arc adds unity to the degrees of its two end nodes. Since $\sum_{i \text{ even}} d_i$ is even, $\sum_{i \text{ odd}} d_i$ must also be even. But each $d_i (i \text{ odd})$ is odd so the number of odd nodes must be even.

Let n_0 be the set of odd nodes, N_e the set of even nodes, and N the set of all nodes. Since the number of odd nodes $|N_0|$ is even, we can partition these nodes into two groups and form $k = (1/2)|N_o|$ paths between disjoint pairs of nodes. The arcs $\bar{A}$ contained in these paths are added to the original graph G as artificial arcs to obtain the augmented graph $G_0(\bar{A})$. The augmented graph will be a multigraph (more than one arc can join two nodes) since the artificial arcs are duplicates of arcs which were in G to begin with.

A feasible solution to the CPP therefore stems from k paths which connect the k pairs of odd nodes. The problem then reduces to finding the best k paths.

Glover[278] and Murty[499] presented algorithms to solve the CPP in 1967; these were evaluated by Stricker[635]. Bellman and Cooke[56] have developed a dynamic programming algorithm which performs well only when there are fewer than about 15 arcs. The matching application to the CPP pointed out by Edmonds and Johnson[197] and Christofides[128], however, yields a good (polynomial) algorithm. We next discuss this computationally efficient approach. Christofides[128] proves the following theorem which formalizes the notion that we need only find the best set of arcs to traverse more than once.

THEOREM 2.1 (Christofides) For any cycle covering G, there is some choice of $\bar{A}$ for which $G_0(\bar{A})$ possess an Euler tour corresponding to the cycle of G. The correspondence is such that if a cycle traverses an arc (i, j) of G γ times, there are γ arcs (one real and γ-1 artificial) between i and j in $G_0(\bar{A})$ each of which is traversed exactly once by the Euler tour of $G_0(\bar{A})$; and conversely.

The CPP algorithm follows directly now, assuming we can solve a matching problem. All that is necessary is to find the set of arc $\bar{A}$ in paths between nodes of odd degree which produces the least additional cost. It should be noted that γ in Theorem 2.1 is never greater than 2. In other words, the optimal cycle never traverses any arc of G more than twice since

no artificial arc can be included in more than one path between odd nodes. If it were, we could do better by dropping that arc from both paths. The steps of the algorithm are displayed below.

CPP Algorithm

Step 1. Using a shortest path algorithm on the graph G with cost matrix $[c_{ij}]$ form the $|N_o|$ by $|N_o|$ matrix $D = [d_{ij}]$ where d_{ij} is the cost of the least-cost path from node $i \in N_o$ to another node $j \in N_o$.

Step 2. Use a polynomial-time minimum 1-matching algorithm to find $\bar{A}_{\min}$ according to the cost matrix D.

Step 3. If node α is matched to another node β, identify the least-cost path $\mu_{\alpha\beta}$ corresponding to the cost $d_{\alpha\beta}$ of step 1. Insert artificial arcs to obtain $G_0(\bar{A}_{\min})$.

Step 4. The sum of the costs from matrix $[c_{ij}]$ of all arcs in $G_0(\bar{A}_{\min})$ is the minimum cost of a cycle covering G.

A 1-matching on the complete graph formed from the odd nodes is a set of arcs E which cover all the odd nodes in such a way that no two arcs in E are adjacent (share a node). That is, each odd node is of degree 1 relative to the set E. A minimum weighted 1-matching is a 1-matching of minimum total length. Edmonds has, almost singly-handedly, developed the elegant combinatorial theory behind matching. In fact, Edmonds and Johnson[196, 197] demonstrate that the CPP can also be solved directly on G using a generalized matching algorithm; this saves the computation of all shortest paths between odd nodes. The basic references on matching theory results are Edmonds[193–195], and Edmonds and Johnson [196, 197]. Edmonds' matching algorithm requires on the order of n^4 computations in the worst case; Gabow[239] and Lawler[425] later improved the matching algorithm to perform on the order of n^3 computations in the worst case. Recent improvements in computational complexity and empirical running times are due to Derigs and Kazakides[177] and to Christofides[140].

Once the graph $G_0(\bar{A}_{\min})$ has been determined, an Euler tour on G_o corresponding to the minimum length cycle covering G can be found by starting with any node and traversing and then erasing an incident arc such that the removal of the arc does not divide the graph into two disjoint components. Subsequent nodes are treated similarly one at a time.

The directed Chinese postman problem. For directed connected networks, a sufficient condition for the existence of an Euler tour is that for each node the number of directed arcs into the node is the same as the number out of the node.

Beltrami and Bodin[63] provide an optimal algorithm for solving the postman problem over a directed network and a heuristic algorithm for solving the directed capacited capacitated Chinese Postman Problem. The second algorithm involves an application of the first algorithm. Then, the resulting giant tour is broken into a set of smaller tours which obey the capacity restrictions. It is fundamentally very similar to the approach suggested by Stricker[635].

In order to solve the postman problem on a directed network find all nodes n_j and m_k where the number of arcs into n_j exceeds the number out by s_j and the number of arcs out of m_k exceeds the number entering by d_k. It can easily be shown that $\sum_j s_j = \sum_k d_k$.

We can construct a bipartite graph between nodes n_j and m_k. Each distance c_{jk} is the shortest path distance from n_j to m_k and may be found by applying a shortest path algorithm to the original graph G. If there is a node $n_{j'}$ such that no path exists to any node m_k then the directed postman problem does not have a feasible solution. In order for an Euler tour to exist each node n_j must have additional arcs directed outward and each node m_k must have additional inward arcs. The s_j's can be viewed as supplies and the d_k's as demands. The following transportation problem naturally arises.

$$\text{Minimize} \quad \sum_{j,k} c_{jk} x_{jk}$$

subject to

$$\sum_k x_{jk} = s_j \quad \text{for all } j$$

$$\sum_j x_{jk} = d_k \quad \text{for all } k$$

$$x_{jk} \geq 0.$$

Now if we augment the original graph G with the arcs implied by the solution of the transportation problem we have an Euler tour which is the solution to the directed postman problem.

2.10. CAPACITATED ARC ROUTING PROBLEMS

Uncapacitated routing problems can be classified as node problems, arc routing problems, or general routing problems (see Chapter 1). We have studied the TSP and CPP in depth and mentioned the GRP. In each case we have assumed that all arcs are undirected and that there is a vehicle of unlimited capacity. We point out that the terms *cost* and *distance* are used interchangeably in this section.

Capacitated variations, of course, reflect real-life situations more directly. For example, the vehicle routing problem has been the focus of much research attention. On the other hand, capacitated arc routing problems, also extremely practical, have been comparatively neglected (the recent work by Stern and Dror[623], Golden and Wong[310], and Golden *et al.*[303] are exceptions).

The capacitated arc routing problem (CARP) that we focus on is as follows: given an undirected network $G(N, E, C)$ with arc demands $q_{ij} \geq 0$ for each arc (i, j) which must be satisfied by one of a fleet of vehicles each of capacity W, find cycles each of which passes through the domicile (node 1) which satisfy demands as minimal total cost. Several related problems are posed below.

(1) The Chinese postman problem (CPP).

(2) The rural postman problem (RPP). Find a minimum-cost cycle which traverses each arc in a given subset $R \subseteq E$ at least once. See Orloff[523, 525] for more discussion and Lenstra and Rinnooy Kan[432] for an NP-completeness proof.

(3) The capacitated Chinese postman problem (CCPP).

(4) The traveling salesman problem (TSP).

(5) The vehicle routing problem (VRP).

(6) The general routing problem (GRP).

Now we examine the relationship between the CARP and these other problems. If we assume $q_{ij} > 0$ for all $(i, j) \in E$, the CARP reduces to the CCPP. If, in addition, we have only one vehicle of capacity $W \geq \sum_i \sum_j q_{ij}$, we obtain the CPP. With one vehicle of capacity $W \geq \sum_i \sum_j q_{ij}$ and $q_{ij} > 0$ for all arcs $(i, j) \in R \subseteq E$, we have the RPP. Thus, the CCPP, the CPP, and the RPP are special cases of the CARP. Since we can incorporate the constraint that a node must be serviced by splitting that node into two nodes joined by a zero-length arc with a demand equal to the original node demand, the TSP, VRP, and GRP are also special cases of the CARP.

Golden and Wong[310] show that even an approximate, restricted version of the CARP is NP-hard. They also show how to obtain, in polynomial time, a valid lower bound for the CARP by considering a particular matching problem derived from the original Carp.

A general purpose heuristic for the CARP has been suggested by Golden and Wong[310] and modified and tested by Golden *et al.*[303]. Its steps are outlined below.

Step 1. INITIALIZE—All demand arcs are serviced by a separate cycle.

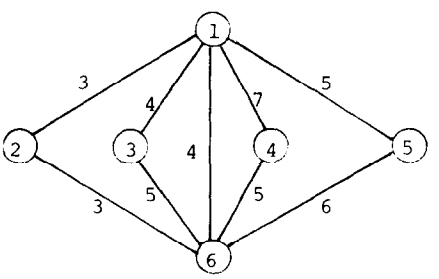


Fig. 2.19. Illustration of algorithm.

Step 2. AUGMENT—Starting with greatest length cycle available, see if a demand arc on a smaller cycle can be serviced on the larger cycle.

Step 3. MERGE—Subject to capacity constraints, evaluate the merging of any two cycles (possibly subject to additional restrictions). Merge the two cycles which yield the largest positive savings.

Step 4. ITERATE—Repeat Step 3 until finished.

Note that Steps 3 and 4 are closely related to the well-known Clarke-Wright vehicle routing algorithm. The details of the algorithm (in particular, the MERGE step) are explained by Golden *et al.*[303].

We demonstrate the algorithm on the small example given in Fig. 2.19. Here, vehicle capacity is 4 and each arc has unit demand. Initially, there are nine cycles—one for each demand arc. The augmentation step leaves the four cycles 1 2 6 1, 1 3 6 1, 1 6 4 1, 1 6 5 1. Step 3 merges 1 2 6 1 and 1 6 5 1 to obtain 1 2 6 5 1 reducing the number of cycles to three and we are finished. Note that merging 1 2 6 1 and 1 3 6 1 to obtain 1 2 6 3 1 or 1 3 6 1 and 1 6 5 1 to obtain 1 3 6 5 1 would have given the same total distance.

Much additional information regarding the CARP and the CCPP is contained in the papers by Christofides[128], Golden and Wong[310], and Golden *et al.*[303].

CHAPTER 3
VEHICLE AND CREW SCHEDULING PROBLEMS

3.1. INTRODUCTION

Vehicle and crew scheduling problems can be thought of as routing problems with additional constraints having to do with the times when various activities may be carried out. The routing problems reviewed in Chapter 2 give special importance to the spatial characteristics of the activities performed. In particular, the successive moves of a vehicle, generally referred to displacements in space. In scheduling problems, however, a time is associated with each activity. For example, each location may require a delivery at a specified time. Thus, the temporal aspects of vehicle movements now have to be considered explicitly. As a result, the activities are followed in both space *and* time. The feasibility of an activity is also influenced by both space and time characteristics, e.g. a single vehicle could not service two locations with identical delivery times. The sequencing of vehicle activities in both space and time is at the heart of the vehicle scheduling problem.

In crew scheduling, the primary concern is to sequence the movements of a crew in space and time so as to staff the desired vehicle movements. While crew scheduling problems are essentially similar to vehicle scheduling problems, the former generally involve more complicated restrictions, such as requirements for crew lunch breaks, union regulations, etc.

For example, consider the following bus operator schedule:

7 : 00 AM:	report to work at garage
7 : 03 AM:	walk to relief point A
7 : 11 AM:	board bus No. 1 at relief point A and relieve current operator, then operate bus No. 1 (includes both in-service operations and deadheading)
11 : 20 AM:	get off bus No. 1 at relief point B and take lunch break
11 : 52 AM:	travel via transit system from relief point B to relief point C
12 : 20 PM:	board bus No. 2 at relief point C and relieve current operator, then operate bus No. 2
3 : 30 PM:	get off bus No. 2, having driven it to garage, and go off duty.

This example illustrates some of the complexities involved in crew schedules. In particular, crew schedules almost always require rest periods such as the lunch break illustrated. Note that two types of deadheading take place: (1) with the operator driving a vehicle, and (2) with the operator walking or being a passenger on the transit system. Also note that an operator may be associated with more than one vehicle during the day.

In general, vehicle and crew scheduling problems interact with one another: the specification of vehicle schedules will set certain constraints on the crew schedules and *vice versa*. Ideally, therefore, one would want to solve the two problems simultaneously. However, models incorporating both problems into a single optimization problem are generally quite complex.

Consequently, most practitioners have opted for sequential procedures that solve one problem first, and then the other, with some mechanism for taking the interaction between the two into account.

Before we describe our formal representation of scheduling problems let us examine more closely the nature of the temporal constraints. Suppose we wish to deliver a package to Mr. Smith and we know that once we arrive at Mr. Smith's house the delivery will take exactly 10 minutes. If Mr. Smith put no further restrictions on delivery time, we could represent temporal considerations solely as a weight (10 minutes) associated with the delivery task together with a constraint that stated no route could last more than say 9 hours. We would use a routing algorithm to solve this problem, which would have the freedom to assign Mr. Smith's delivery to any time of the day. On the other hand, suppose Mr. Smith specified that the delivery must

take place at exactly 2 : 10 PM. In this case, a start time 2 : 10 PM and an end time 2 : 20 PM would be associated with the delivery task and we would use a scheduling algorithm to solve the problem. Our scheduling algorithm would have to assign Mr. Smith's delivery at 2 : 10 PM.

In general, the input to vehicle and crew scheduling problems is a set of tasks. Each task has a specified start time, end time, start location and end location. The cost function consists of components that might include vehicle operating costs, vehicle capital costs and crew operating costs. The fleet of vehicles and the set of crews may be limited and may be housed at one or more depots. The type of scheduling problem that evolves is a function of the constraints imposed upon the formation of schedules, the types of tasks being serviced and the locations where these tasks must be carried out.

In Fig. 3.1, a set of vehicle schedules servicing 10 tasks is presented. We assume in this example that each task has the same start and end location (the depot). The start and end times

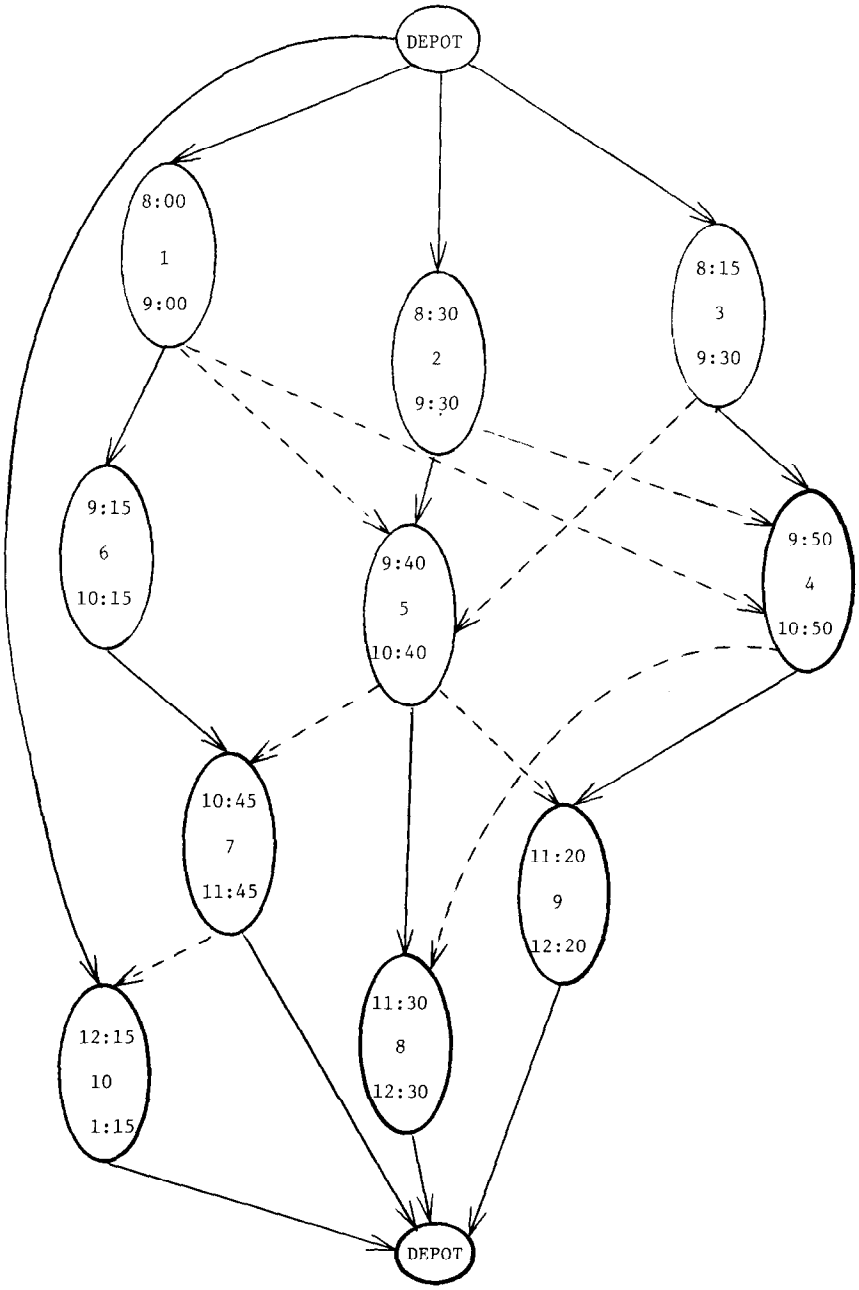


Fig. 3.1. Vehicle schedules with length of path restrictions.

of each task are given within the node representing the task. A solid branch between two nodes indicates that these two nodes are on the same vehicle schedule and that the vehicle schedule will follow the orientation of the branch. The dotted branches indicate feasible connections which are not used in the solution. In this example, a branch is drawn from node i to j if the start time of task j is greater than the end time of task i and if the start time of task j is less than or equal to the end time of task i plus one hour. Thus, in the example, there is no branch from node 6 to node 5 since the start time of node 5 is less than the end time of node 6 and no branch from node 6 to node 8 since the start time of node 8 is greater than the end time of node 6 plus 1 hour. Each vehicle schedule is assumed to be no longer than 5 hours. Note that this restriction prohibits the inclusion of node 10 on the left-most path.

We will consider the following problems in this chapter:

*the single depot vehicle scheduling problem

*the single depot vehicle scheduling problem with length of path restrictions

*the single depot vehicle scheduling problem with multiple vehicle types

*the multiple depot vehicle scheduling problem

*the fixed location worker scheduling problem

*the mass transit crew/vehicle scheduling problem

*the air crew scheduling problem

*the rostering and bid line problems.

In all formulations and algorithms we describe the primary input will be a set of tasks with each task characterized by start and end times and start and end locations.

Sections 3.2 and 3.3 give an overview of the problems discussed in this chapter. The remaining sections discuss algorithms for solving vehicle and crew scheduling problems.

3.2. VEHICLE SCHEDULING

Three real-world constraints commonly determine the complexity of the vehicle scheduling problem. These restrictions are: (a) a constraint on the length of time a vehicle may be in-service before it must return to the depot for servicing or refueling; (b) the restriction that certain tasks can only be serviced by certain vehicle types; and (c) the presence of a variety of depots where vehicles may be housed. Our description of vehicle scheduling problems is classified according to these restrictions: Section 3.2.1 describes the unconstrained case and Sections 3.2.2 through 3.3.4 handle cases (a) through (c) respectively.

3.2.1 *The single depot vehicle scheduling problem (VSP)*

The *single depot vehicle scheduling problem* (VSP) requires the partitioning of the nodes (tasks) in an acyclic network into a set of paths in such a way that a certain cost function is minimized. Each path corresponds to the schedule for a single vehicle. An objective function that minimizes the number of paths effectively minimizes capital costs since the number of required vehicles equals the number of paths. If a weight equal to the corresponding deadhead time is associated with each arc, an objective function that minimizes total arc weight effectively minimizes operating costs since these are proportional to total vehicle travel time. If capital and operating costs can be quantified, then a combined objective can be used to minimize total system costs. Figure 3.1 depicts a set of tasks and a solution to the vehicle scheduling problem with path length constraints; Fig. 3.2 gives a solution to the path minimization VSP without the length of path restrictions.

3.2.2 *Vehicle scheduling problem with length of path restrictions (VSPLPR)*

In the *vehicle scheduling problem with length of path restrictions* (VSPLPR), constraints are placed on the length of time a vehicle may spend away from the depot or the mileage a vehicle may cover without returning to the depot for service. This constraint commonly encountered in practice corresponds to fuel restrictions, maintenance considerations, etc. Whereas the VSP can be solved to optimality using a polynomially-bounded algorithm, the VSPLPR is *NP*-hard[33]. To our knowledge, no optimal vehicle scheduling algorithms have been designed explicitly for this problem. Heuristic algorithms that treat length of path constraints as well as other constraints are described in Section 3.4. In addition, this problem is also solved as a subproblem in an algorithm for generating crew schedules. This algorithm is discussed in

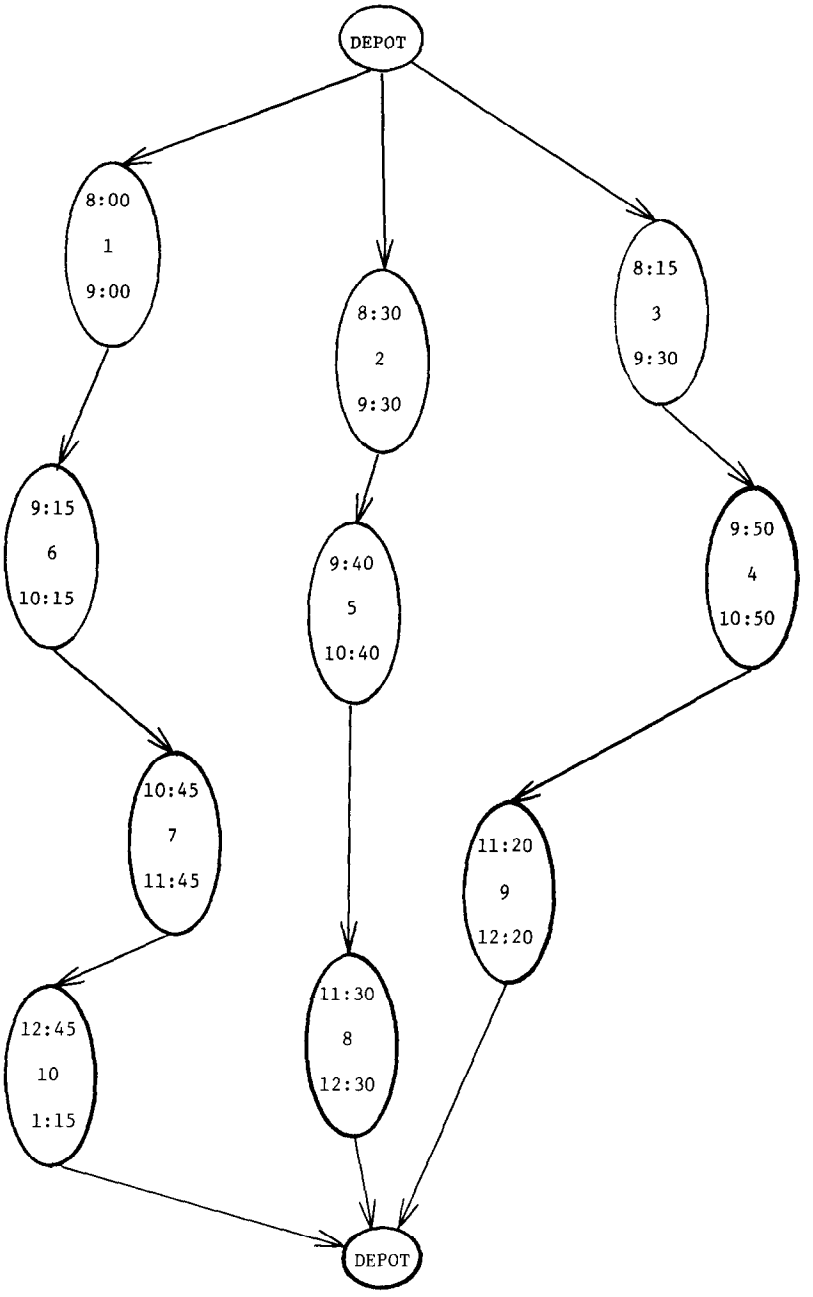


Fig. 3.2. Vehicle schedules with no length of path restrictions.

Section 3.5. A solution to this problem is illustrated in Fig. 3.1 where the length of path restriction is 5 hours.

3.2.3 *Vehicle scheduling problem with multiple vehicle types (VSPMVT)*

The *vehicle scheduling problem with multiple vehicle types* (VSPMVT) allows the possibility that vehicles with different characteristics are available to service the tasks. In most cases, the characteristic is vehicle capacity. For example, in transit systems, mini-buses can service the lines with low demand, regular buses can service the lines with high demand and either vehicle can service lines with medium demand. For each task the set of vehicles that may service it is specified. Consider the example in Fig. 3.3 where nodes 2, 6, and 7 can be serviced by type 1 vehicles, nodes 1, 3, 4, and 5 by type 2 vehicles, and nodes 8, 9, and 10, by either vehicle. There is no length of path restriction. A minimum path solution uses 4 vehicles rather than the 3 vehicles given in the single vehicle case (Fig. 3.2).

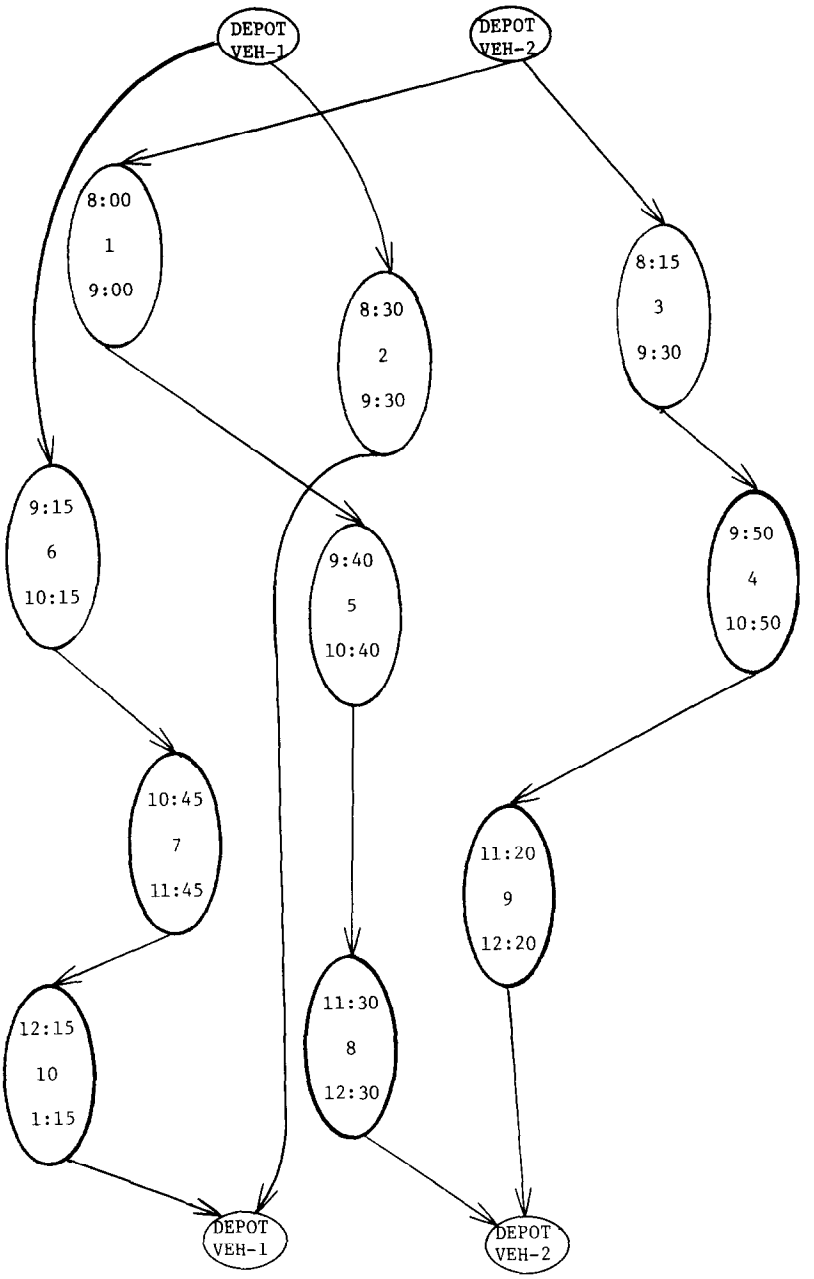


Fig. 3.3. Vehicle schedules with 2 vehicle types.

3.2.4 *Vehicle scheduling problem with multiple depots (VSPMD)*

In the *vehicle scheduling problem with multiple depots* (VSPMD), tasks may be serviced out of more than one depot. As with the vehicle routing problem, each vehicle must leave and return to the same depot and the fleet size at each depot must range between a specified minimum and maximum. An illustration of a feasible solution to the VSPMD is given in Fig. 3.4. In the solution given, nodes 1, 5, 6, 7, 8 and 10 are serviced by vehicles out of depot 1, nodes 2, 3, 4 and 9 are serviced by vehicles out of depot 2. This solution might be optimal in a situation where the objective function included operating costs and the start locations of tasks 1 and 6 and the end locations of tasks 8 and 10 were close to depot 1 and the start locations of tasks 2 and 3 and the end locations of tasks 3 and 9 were close to depot 2.

3.3. CREW SCHEDULING

Before describing crew scheduling problems it is worthwhile to examine more closely the relationship between crew and vehicle scheduling. Figure 3.5 illustrates a set of vehicle

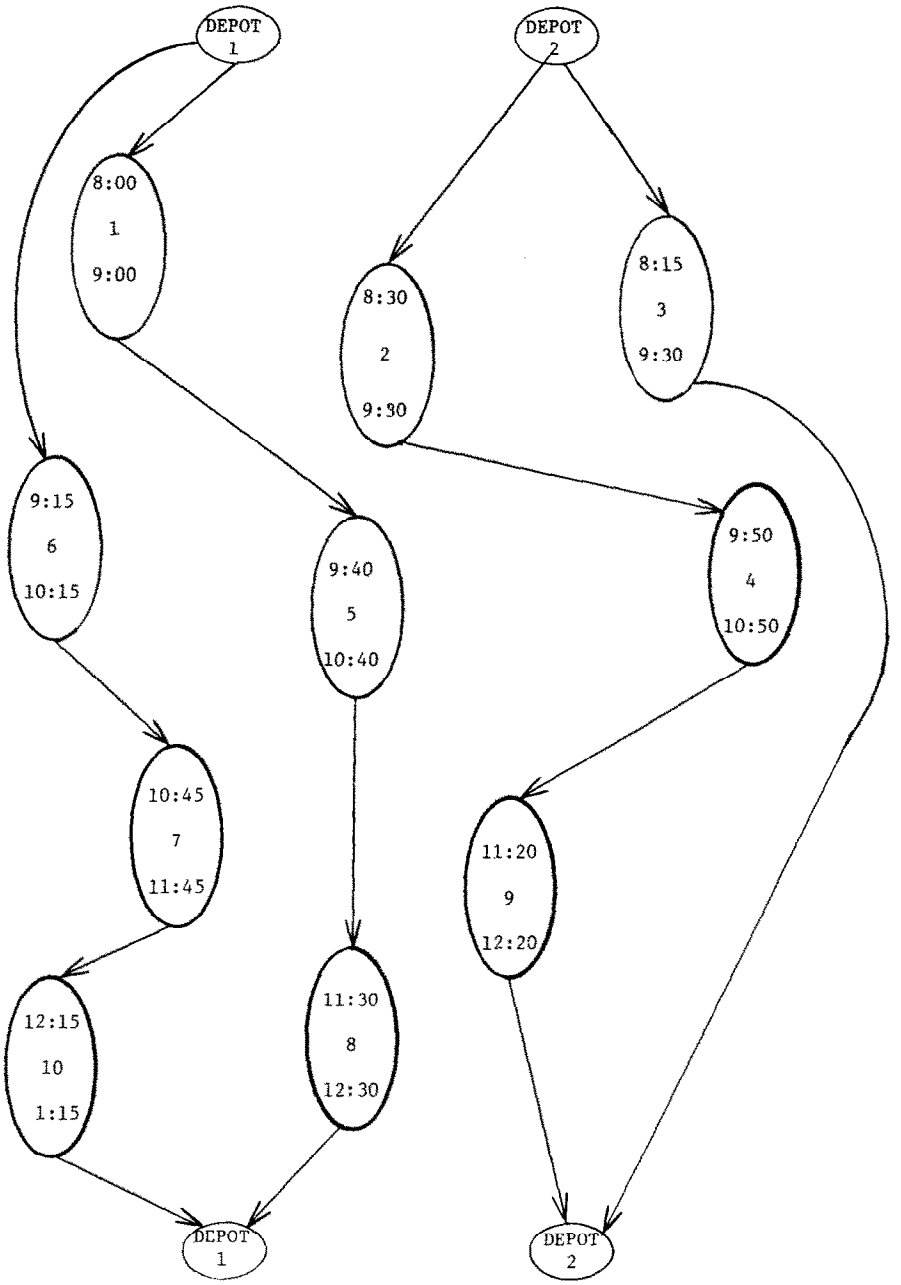


Fig. 3.4. Vehicle schedules with 2 depots.

schedules. These schedules describe the vehicle movements required for a certain transportation system. Each individual schedule has a set of points where one crew can relieve another. In the mass transit setting, a relief point is a designated stop along a transit line and in the airline case it is the end of any flight leg. To obtain crew schedules each vehicle schedule is split into pieces at one or more of the relief points. As Fig. 3.5 shows, an individual crew schedule is then obtained by grouping one or more of these pieces together. Of course, the feasibility of joining one piece with another depends not only on the end time of the first piece relative to the start time of the second, but also on the end location of the first piece relative to the start location of the second. The cost function will almost never be linear since the total cost of the crew schedule would rarely equal the sum of the costs of its component pieces.

Although we have described crew schedules in terms of vehicles, this by no means suggests that vehicles should be scheduled prior to and separately from crews. Unfortunately, in nearly all practical settings this is the approach taken. In the case of aircraft and air crew scheduling,

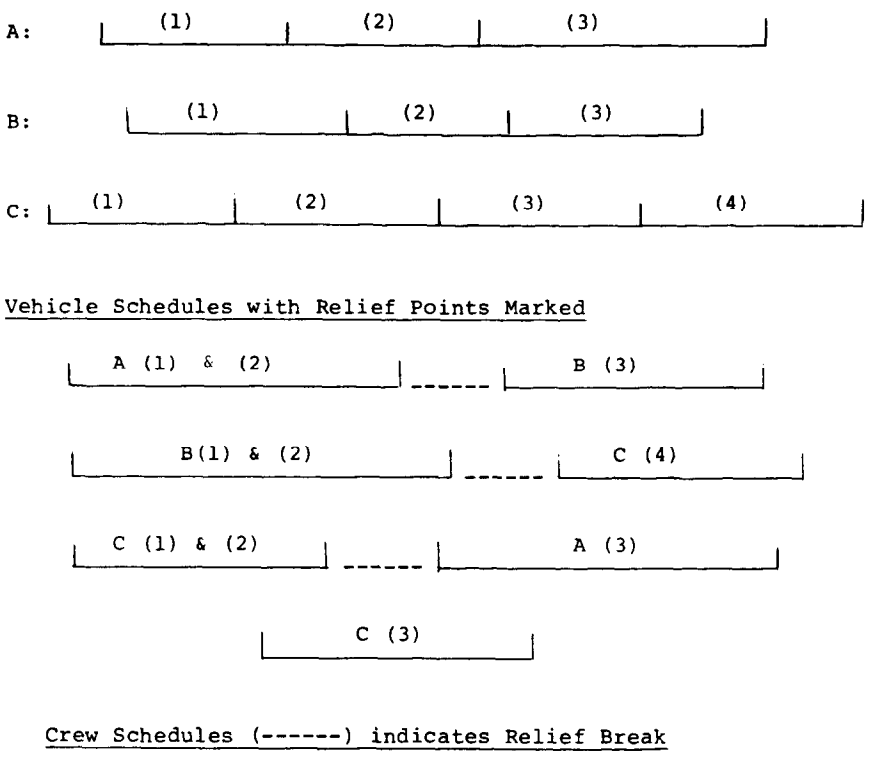


Fig. 3.5. Vehicle and crew schedules.

this approach is justifiable since aircraft costs significantly dominate air crew costs. In the mass transit case, however, crew costs dominate vehicle operating costs and in some cases reach as high as 80% of total operating costs. Under such circumstances, it seems foolish to allow vehicles to be scheduled independently of, and without regard to crews. For this reason, in this section rather than describing a "pure" mass transit crew scheduling problem, we describe a combined crew/vehicle scheduling problem.

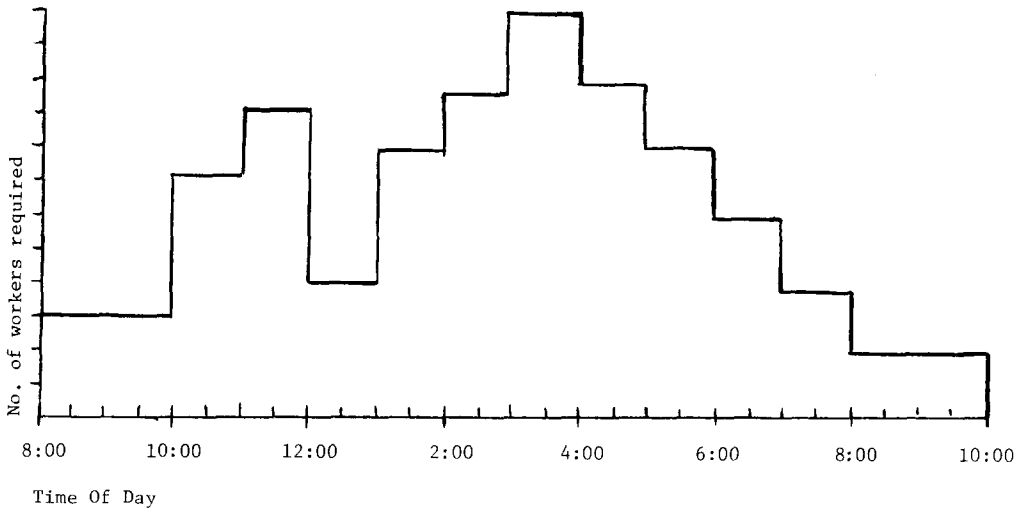
Although the primary thrust of this chapter is toward scheduling crews to operate vehicles, we start with a simpler case, that of scheduling workers at a fixed location, e.g. sales clerks or telephone operators. Considering this case will help in the conceptualization of more complicated situations where crew and vehicle schedules interact.

3.3.1 *Scheduling workers at a fixed location*

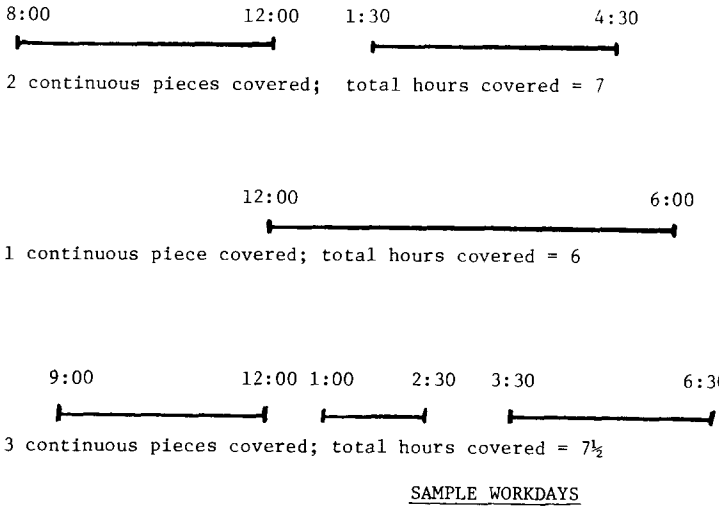
For this problem we do not start out with a set of tasks as described previously, but rather divide the workday into T time intervals and specify a demand for workers, d_t associated with each time interval $t = 1, 2, \ldots, T$. (In Section 3.4.2 we show how these specifications can be converted into a set of tasks). The worker scheduling problem is to find a set of worker schedules that cover all the required work. It is assumed that workers are interchangeable and that any worker can be relieved at the end of any time period and that any worker can start at the beginning of any time period. Figure 3.6 illustrates the demand for workers via a histogram and gives sample workdays for an individual. A cost is associated with each possible workday and the problem is to cover the histogram with a minimum cost set of workdays. In the case where workdays do not contain breaks this problem has a network flow structure. The problem just described applies to several real world situations including telephone operator scheduling[589]. It has also been used to estimate crew requirements for urban mass transit systems[76, 89]. In this case, the spatial considerations of transit crew scheduling are de-emphasized in order to get a clear picture of the temporal considerations.

3.3.2 *Mass transit crew and vehicle scheduling*

Passenger service for a mass transit system is defined by means of a set of *lines* and *line schedules* as illustrated in Fig. 3.7(A). Each traversal of a line must be performed by a single



WORKER DEMAND HISTOGRAM



SAMPLE WORKDAYS

Fig. 3.6. Fixed location worker scheduling.

vehicle so that continuous passenger service can be provided. We refer to each one-way traversal of a line as a *trip*. As Fig. 3.7(B) illustrates, we may characterize trips by their start time and start location and end time and end location. If vehicles are to be scheduled independently of crews, the appropriate problem description from Section 3.2 can be applied with the set of trips serving as the set of input tasks.

To define a combined crew/vehicle scheduling problem we must consider the nature of crew movements. Each line has one or more *relief points* which are stops along the line where one crew may relieve another. Relief points do not necessarily coincide with the start or end of a trip (see Fig. 3.7(A)). Thus, a crew's period of work on a single vehicle starts and ends at either a relief point or at the garage. We create the set of *d-trips* (driver trips) by partitioning each trip at its relief points. These are illustrated in Fig. 3.7(C). The set of *d*-trips makes up the set of input tasks for the mass transit crew/vehicle scheduling problem. Whereas a trip represents the minimal amount of work that must be performed by a single vehicle, a *d*-trip represents the minimal amount of work that must be performed by a single crew.

FIGURE 3.7 A: LINE SCHEDULE

College Park (I)	Mount Ranier*(II)	North Capitol & New York Ave.	Potomac Park (III)
9:00	9:20	9:55	10:15
9:15	9:35	10:10	10:30
⋮	⋮	⋮	⋮
10:20	10:00	9:25	9:05
10:35	10:15	9:40	9:20
⋮	⋮	⋮	⋮

* indicates relief point

FIGURE 3.7B: TRIPS (indicated by [(start time, start location),(end time, end location)])

[(9:00,I),(10:15,III)], [(9:15,I),(10:30,III)], [(9:05,III),(10:20,I)],

[(9:20,III),(10:35,I)].

FIGURE 3.7C: D-TRIPS

[(9:00,I),(9:20,II)], [(9:20,II),(10:15,III)], [(9:15,I),(9:35,II)],

[(9:35,II),(10:30,III)], [(9:05,III),(10:00,II)], [(10:00,II),(10:20,I)],

[(9:20,III),(10:15,II)], [(10:15,II),(10:35,I)]

Fig. 3.7. Input to mass transit crew/vehicle scheduling problem.

A crew operates a single vehicle during a continuous portion of work called a *piece*. A sequencing of pieces into a single crew schedule is called a *run*. As was illustrated at the beginning of the chapter, if the end location of one piece and the start location of another do not coincide, then the crew must walk or use the transit system to go from one location to another. In general, crews are paid for this dead time. Figure 3.8 illustrates a solution to the mass transit crew/vehicle scheduling problem. It consists of: (a) a set of *blocks*, which specify the work plan for vehicles over the day and which are made up of a set of *d*-trips; (b) the decomposition of blocks into pieces: and (c) the sequencing of pieces into runs.

Figure 3.9 illustrates a typical service profile for an urban mass transit system. Part of the challenge of mass transit crew scheduling results from the fact that this profile is rather ill-suited for decomposition into desirable crew runs. In particular, it is clear that, compared to mid-day, there will be a large number of pieces required during the AM peak and during the PM peak. It is natural to try to join these pieces into *split runs*, i.e. a run with a very long break between pieces. Transit crews generally prefer *straight runs* which consist of a single piece or contain a short break between pieces. The typical union contract places cost penalties on split runs or limits the percentage of straight runs used. To complicate matters further, the nature of the union regulations and the service profile can vary significantly from one municipality to another.

3.3.3 *Air crew scheduling*

The vehicle and crew scheduling problems which perhaps have received the most attention in recent years are the air crew and aircraft scheduling problems. As with mass transit crew and vehicle scheduling, a fixed timetable is assumed for the air crew scheduling problem. Aircraft, on the other hand, are almost always scheduled at the same time timetables are formed. The combined aircraft scheduling/timetable formation problem is a combined routing and scheduling problem which is discussed in Chapter 4.

The crew scheduling problem is broken down into two parts: generating pairings and

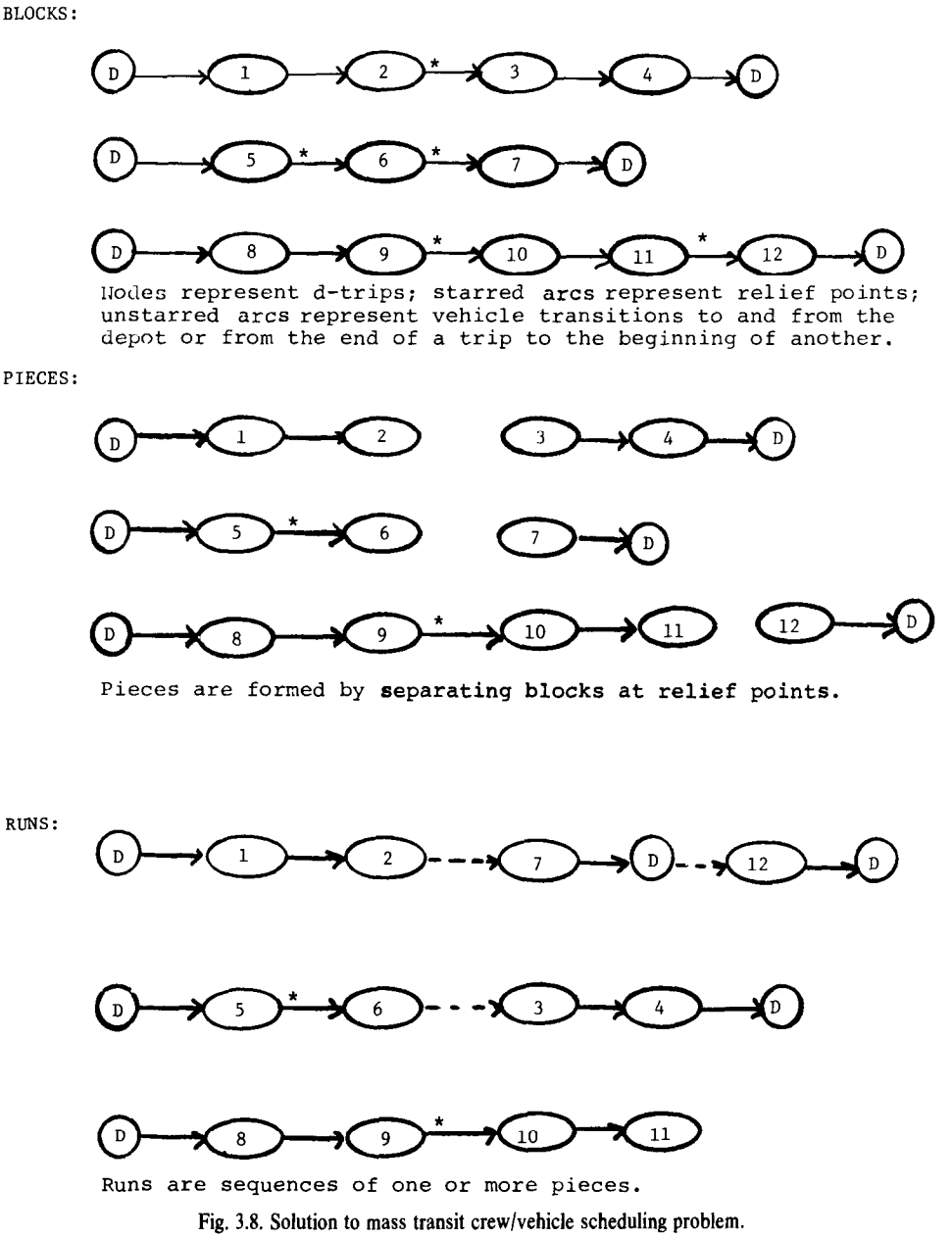
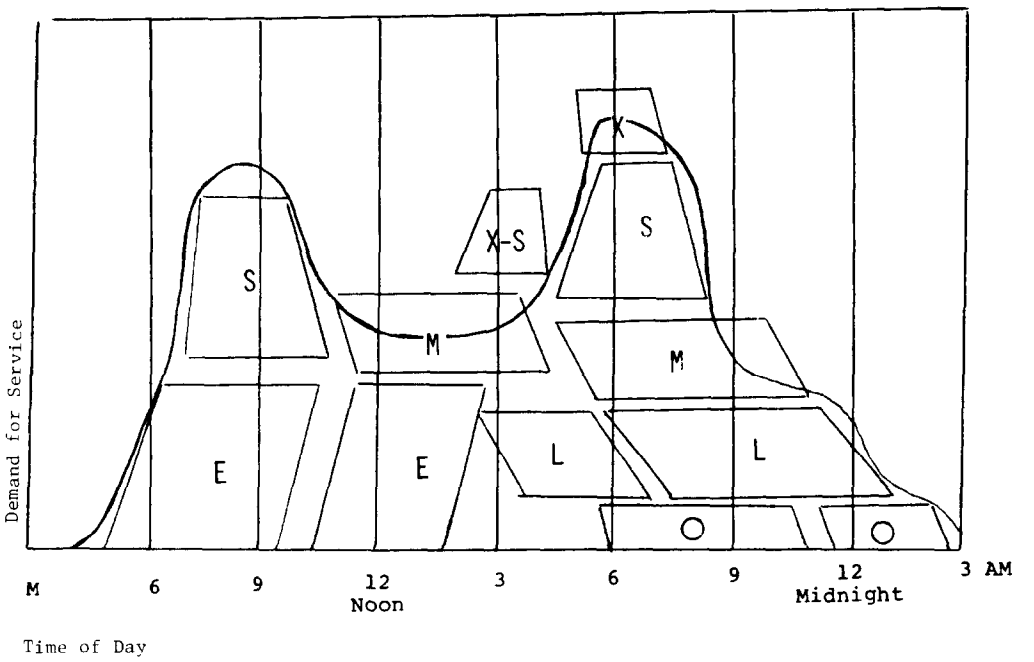


Fig. 3.8. Solution to mass transit crew/vehicle scheduling problem.

constructing bid lines. A *pairing* is a sequence of required service entities that a crew must complete which begins and ends at the same domicile. The construction of minimum cost pairings that cover all tasks arising from the timetable and follow the airplanes, has generally been referred to as the air crew scheduling problem.

Once the pairings have been constructed, bid lines are formed. The *bid lines* are sets of pairings that represent monthly work schedules for the crews. An overview of the pairings problem is now presented. The formation of minimum cost bid lines is described in Section 3.3.4.

The set of input tasks for the air crew scheduling problem consists of a set of flight legs. A *flight leg* corresponds to a flight between two cities, departing one city at a specified time and arriving at a second city at a specified time. A single crew pairing consists of a set of *duty periods*, which are continuous blocks of time when a crew is on duty. A duty period is in turn made up of a set of flight legs. Federal aviation regulations, union rules and company policies together impose a complex set of constraints in the formation of pairings. In particular, duty periods have a maximum length, 12 hours and a minimum rest time is required between duty



Run Types:

E - Early Straight Runs
L - Late Straight Runs
M - Middle Straight Runs
O - Owls (late late straight Runs)
S - Split Runs
X - Extra pieces (for use as Trippers, i.e., short one-piece runs, or the 3rd piece of work on a straight run)

Fig. 3.9. Service profile and decomposition into run types.

periods, 9.5 hours. Further, there are constraints on the manner in which flight legs can be combined into duty periods, e.g. a duty period can contain no more than 7 flight legs. Restrictions may also apply to the manner in which duty periods can be combined into pairings, e.g. no crew can fly more than 16 hours in any 48 hour interval. From a conceptual standpoint the air crew scheduling problem and the mass transit crew scheduling problem are similar. Roughly, flight legs correspond to d-trips, duty periods correspond to pieces and pairings correspond to runs. However, the time frames involved, the constraints, and the cost functions all differ significantly. These differences lead to differences in the solution algorithms as well.

Figure 3.10 illustrates a solution to an air crew scheduling problem. It consists of (a) a set of duty periods, which are in turn made up of flight legs and (b) a set of pairings, which consist of groups of duty periods. As we mentioned earlier, a final problem must be solved to group duty periods into bid lines. This problem is described in the next section.

3.3.4 *Rostering and bid line problems*

The crew schedules output by the procedures described in Sections 3.3.2. and 3.3.3 cover a time period of restricted length. For urban mass transit systems this time period is typically one day; for airline systems this period covers a few days. In certain instances, this is a "final" solution. For example, in most North American transit systems, a single driver follows the same crew schedule each day. This system is generally preferred by both the drivers and the transit companies since it alllows drivers to become familiar with particular routes and riders. However, each crew schedule has a different cost and desirability and drivers bid on schedules

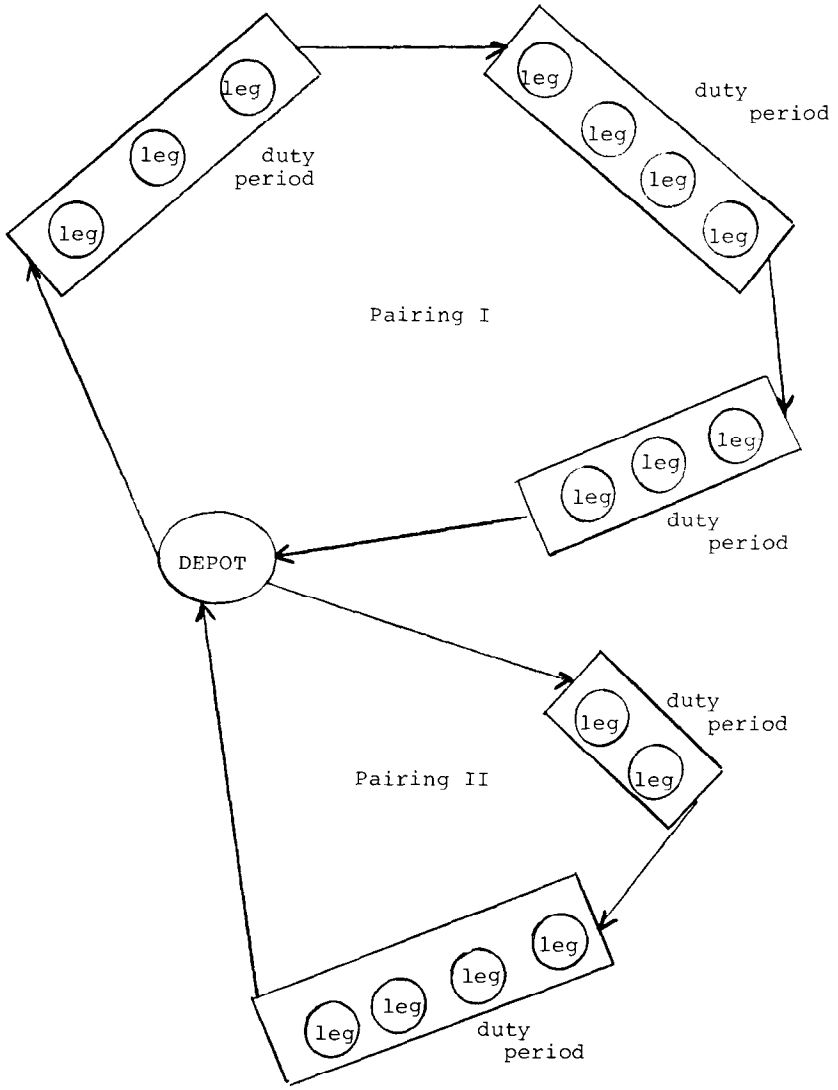


Fig. 3.10. Solution to air crew scheduling problem.

using a priority system based on seniority. Operating in this manner builds some inequities into the system. In most European transit systems each driver within a particular category receives the same pay and there is a need to insure that all schedules are of "equal difficulty". This is accomplished by changing the daily schedule that a particular driver follows every day so that all schedules are of about the same difficulty. This combining of daily schedules into weekly or monthly schedules is called the *crew rostering problem.*

The airline industry also must solve a very similar problem since the crew schedules output by the procedures outlined in Section 3.3.3 are for limited time periods. Just as North American transit crews bid on daily schedules, airline crews bid on monthly schedules. In the airline industry, the problem of combining pairings into monthly work schedules is called the *bid line problem.*

The constraints on these schedules are usually quite simple. The total number of hours worked on each bid line must be between some minimum and maximum. In some applications, this is a "hard constraint", while in other applications, there is a penalty attached to having a bid line fall below the minimum number of hours or exceed the maximum number of hours. For example in the airline industry, a crew is guaranteed a minimum number of hours per month even if the crew's bid line contains fewer than the minimum number of hours. There are also constraints on the minimum and maximum amount of time that may exist between two pairings (or crew schedules) on the same bid line (or roster).

3.4 VEHICLE SCHEDULING ALGORITHMS

All vehicle scheduling problems we discuss can be formulated as optimization problems on appropriately defined networks. In the case of the VSP, this formulation leads to an efficient solution using a minimum cost flow algorithm. In the other cases, complicating constraints make the problems much more difficult. Section 3.4.1 describes the network formulation of the VSP and shows how this problem may be efficiently solved using a network flow algorithm. Section 3.4.2. gives formulations for the constrained vehicle scheduling problems and presents approximate algorithms for solving these problems.

3.4.1 *Solving the VSP*

Recall that as input to the vehicle scheduling problem we start with a set of tasks with each task i characterized by a start location, SL_i, a start time, ST_i, an end location, EL_i, and an end time, ET_i. For any pair of locations L1 and L2 we denote by TM(L1, L2) the time to travel from L1 to L2. DL denotes the location of the depot. We construct a network similar to the one illustrated in Fig. 3.1. The node set N consists of a node representing each task together with a super source node s and a super sink node t. The arc set A is obtained by inserting an arc from task node i to task node j if it is feasible for a single vehicle to service both tasks, i.e. if $ST_j - ET_i \geq TM(EL_i, SL_j)$. Further, an arc is inserted from s to each task node and from each task node to t. These arcs represent trips to and from the depot. Each (s, t)-path through this network represents a possible schedule for a single vehicle. The solution to the vehicle scheduling problem is a set of (s, t)-paths that cover all task nodes. If we associate a cost c_{ij} with each arc $(i, j) \in A$, we formulate the partitioning problem as a minimum cost flow problem as follows:

$$\text{V1:} \quad \text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{s.t.} \sum_{i:(i,j)\in A} x_{ij} - \sum_{i:(j,i)\in A} x_{ji} = 0 \text{ for all } j \in N - \{s, t\} \tag{3.1}$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \text{ for all } j \in N - \{s, t\} \tag{3.2}$$

$$0 \leq x_{ij} \leq 1 \text{ and integer for all } (i, j) \in A \tag{3.3}$$

We interpret

$$x_{ij} = \begin{cases} 1 \text{ if some vehicle traverses arc } (i, j), \\ 0 \text{ otherwise.} \end{cases}$$

Constraint (3.1) is the standard conservation of flow constraint; constraint (3.2) insures that exactly one path (vehicle schedule) will pass through each task node. We may represent vehicle costs fairly accurately within this formulation. We assume that operating costs are a linear function of vehicle milage and travel time and that a fixed capital cost may be associated with each vehicle used. We then define

$$c_{ij} = \begin{cases} \text{the operating cost associated with deadheading from } EL_i \text{ to } SL_j \text{ for } i, j \in N - \{s, t\}, \\ \text{the operating cost associated with deadheading from } EL_i \text{ to DL for } j = t, \text{ the vehicle} \\ \text{capital cost plus the operating cost associated with deadheading from DL to } SL_i \text{ for} \\ i = s. \end{cases}$$

This cost function definition is based on the assumption that the number of paths (arcs in the solution out of node s) equals the number of required vehicles. Depending on how arcs and the problem's time horizon are defined the number of vehicles may or may not equal the number of paths. For example one path could represent a vehicle leaving the depot at 7 AM and returning at 10 AM and another path could represent a 3 PM to 6 PM vehicle schedule. It is clear these

could be covered by the same vehicle. Note from the definition of the arc set A given previously that there would be a "long arc" from the end task on the 7–10 AM schedule to the start task on the 3–6 PM schedule. Thus, a single path could cover these two work intervals (see Fig. 3.11).

However, in practice the vehicle might return to the depot between these work intervals and thus the cost of this long arc should be adjusted to reflect deadheading to and from the depot. The number of such long arcs can be quite large, i.e. proportional to the number of nodes squared, and consequently many times they are not included in the network. Rather, as was suggested in Section 3.1, arcs whose time duration is greater than some upper bound are not included in the network so that each path explicitly represents a vehicle trip away from the depot. In such situations many times, capital costs can be accurately included by taking into account temporal problem structure. For example, in mass transit systems, where demand for service has a dual peak structure (see Fig. 3.9), it is assumed that the number of vehicles required will equal the number required in the largest peak. If this were the AM peak then

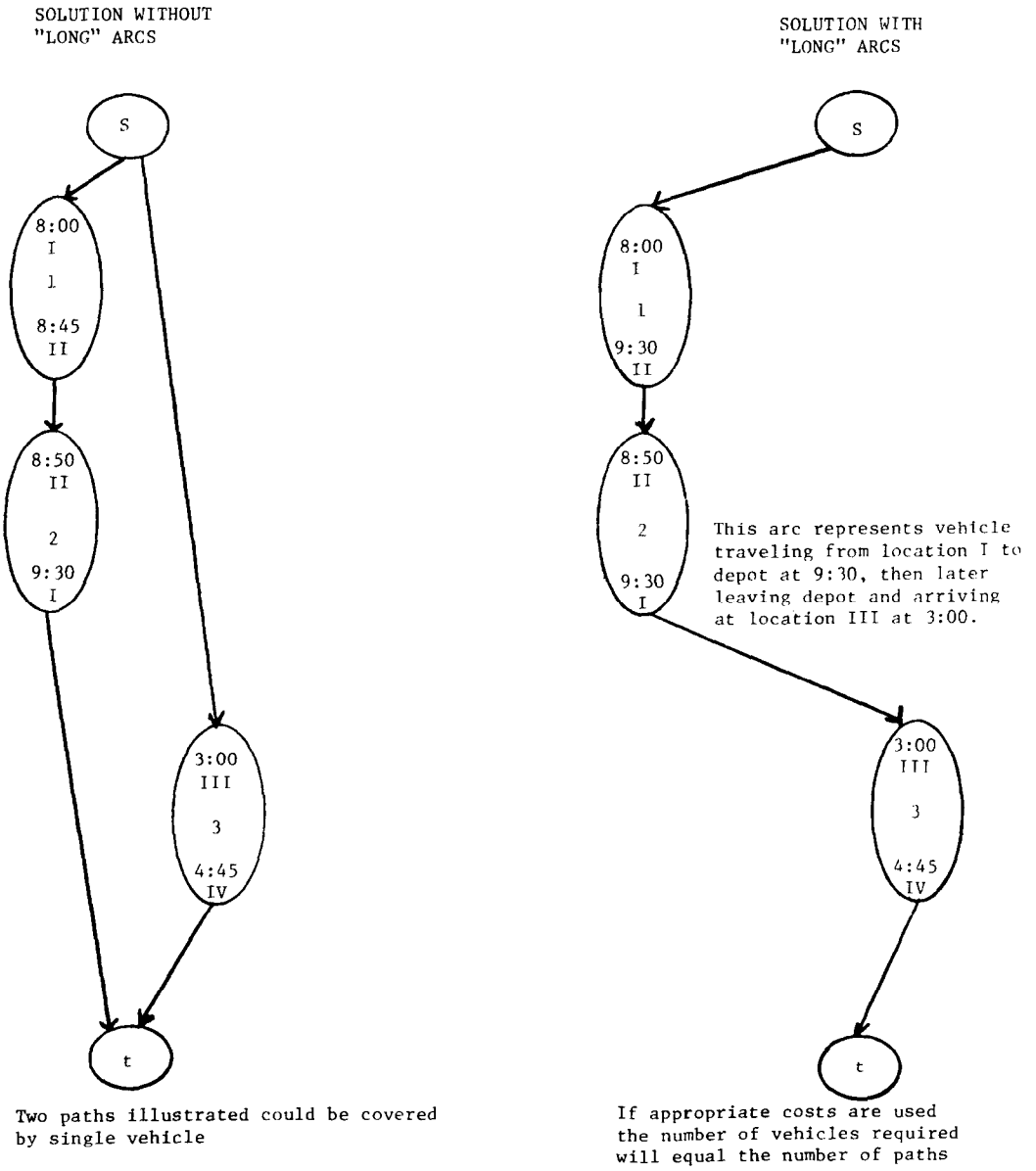


Fig. 3.11. Long arcs in a vehicle scheduling network.

capital cost could be included in the costs of arcs from s to tasks starting between say 5 AM and 9 AM. This is a fairly accurate approximation of total capital costs.

The preceding discussion has assumed that good estimates of capital and operating costs are available that allow the two types of costs to be compared. If this is not the case, then either operating costs alone or capital costs alone can be minimized by minimizing deadhead mileage or the number of paths (subject to previous reservations), respectively. A third alternative is to find the solution that minimizes the number of paths and then resolve the problem minimizing deadhead mileage subject to the restriction that the number of paths used equals the number found in the previous solution. The solution that results would be the least vehicle solution that minimized deadhead mileage.

The mathematical program V1 can easily be seen to be a minimum cost flow problem and consequently can be efficiently solved. The use of this formulation to solve vehicle scheduling problems was first devised by Dantzig and Fulkerson[164]. This formulation with cost per unit flow on the arcs between two nodes being deadheading time is the basis of the vehicle scheduling algorithm contained in the RUCUS package[69]. This formulation with a cost of one on all arcs pointing out of s, which is sometimes referred to as the *Dilworth formulation* (Ford and Fulkerson[227]) can be solved as a maximum flow problem. In this form, it has served as the basis for the vehicle scheduling algorithm in UCOST (Bodin, Rosenfield and Kydes[92]).

3.4.2 *Solving the VSPLPR, VSPMVT and VSPMD*

All the constrained vehicle scheduling problems discussed here, VSPLPR, VSPMVT and VSPMD, are known to be *NP*-hard ([33] and [436]). We can generalize formulation V1 for the VSP to include these problems, however, the generalizations involve complications that preclude them from being solved as minimum cost flow problems. In this section we give formulations for the constrained scheduling problems and present several approximate solution algorithms. We should note that these problems are treated as a group because each of the solution approaches discussed applies to two or more of the problems.

Formulations for the VSPLPR, VSPMVT and VSPMD. We first show how to extend formulation V1 to the VSPLPR and then give a single formulation that applies to the VSPMVT and VSPMD. Recall that the VSPLPR is essentially the same as the VSP except that no paths with time duration greater than some constant T_{MAX} are allowed. We start with the network defined for the VSP but do not include the nodes s and t. Rather we define a set of "back arcs" that are used to characterize feasible paths. A back arc is inserted from task node j to task node i if paths *starting at* i and *ending at* j are feasible, i.e., if $\mathrm{TM}(\mathrm{DL}, \mathrm{SL}_i) + (\mathrm{ET}_j - \mathrm{ST}_i) + \mathrm{TM}(\mathrm{EL}_j, \mathrm{DL}) \leq T_{MAX}$. We denote by A_1 the set of forward arcs defined previously and by A_2 the set of back arcs. Variables x_{ij} represent arcs in A_1 and y_{ij} arcs in A_2. We associate a cost c_{ij} with all $(i, j) \in A_1 \cup A_2$. We may now define VSPLPR as

$$\text{V2:} \quad \text{Min} \sum_{(i,j)\in A_1} c_{ij}x_{ij} + \sum_{(i,j)\in A_2} c_{ij}y_{ij}$$

$$\text{s.t.} \sum_{i:(i,j)\in A_1} x_{ij} + \sum_{i:(i,j)\in A_2} y_{ij} - \sum_{i:(j,i)\in A_1} x_{ji} - \sum_{i:(j,i)\in A_2} y_{ji} = 0 \qquad \text{for all } j \in N, \tag{3.4}$$

$$\sum_{i:(i,j)\in A_1} x_{ij} + \sum_{i:(i,j)\in A_2} y_{ij} = 1 \qquad \text{for all } j \in N, \tag{3.5}$$

$$\sum_{(i,j)\in A_1\cap C} x_{ij} + \sum_{(i,j)\in A_2\cap C} y_{ij} \leq |C| - 1 \text{ for all cycles } C \text{ with } |A_2 \cap C| \geq 2, \tag{3.6}$$

$$0 \leq x_{ij} \leq 1 \text{ and integer for all } (i, j) \in A_1, \tag{3.7}$$

$$0 \leq y_{ij} \leq 1 \text{ and integer for all } (i, j) \in A_2. \tag{3.8}$$

Where for any set G, $|G|$ is the number of elements in G. Constraints (3.4), (3.5), (3.7) and (3.8)

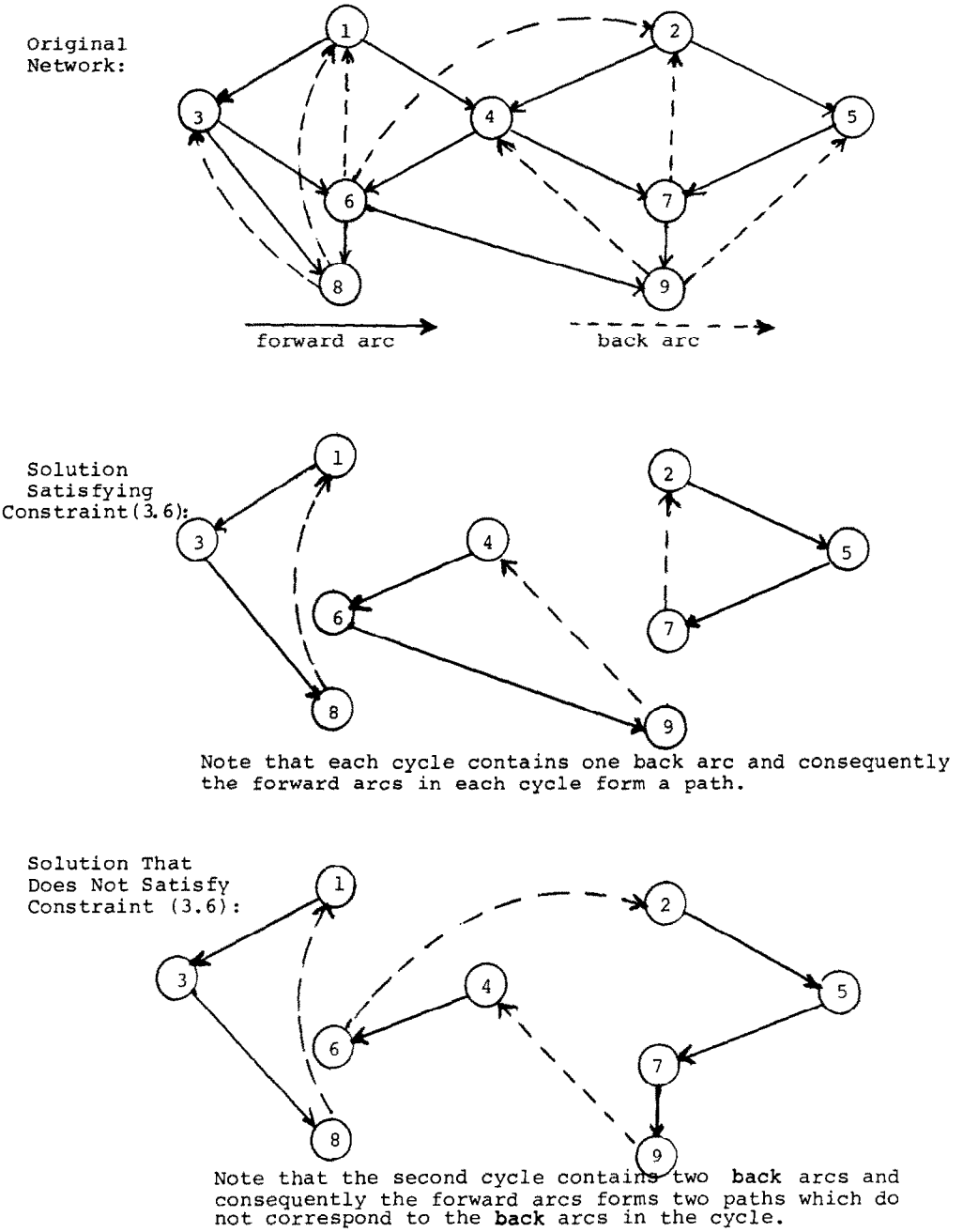


Fig. 3.12. Feasible solutions to V2 with and without constraint (3.6).

constitute a network flow formulation that seeks a feasible circulation. Constraint (3.6) is required to ensure that no cycle includes more than one back arc. Cycles with more than one back arc do not correspond to paths and consequently are not allowed. Figure 3.12 illustrates a feasible solution to V2 and also illustrates a feasible solution to V2 with constraint (3.6) dropped. The previous discussion concerning determination of the c_{ij}'s applies to V2 except that vehicle capital costs and costs for trips to and from the depot are associated with the appropriate back arc.

Although more compact formulations of the VSPLPR exist, the preceding formulation captures some of this problem's structure. In particular, it will be used in describing and solving certain crew scheduling problems in Section 3.5 where, due to the structure of the network considered, constraint (3.6) may be dropped.

We may formulate both VSPMVT and VSPMD as multicommodity flow problems. We start

with the same node set defined for the VSP but rather than adding the single pair of nodes s and t, we include several source/sink pairs, $\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_k, t_k\}$. In the case of VSPMVT each pair corresponds to a vehicle type and in the case of VSPMD each pair corresponds to a depot. We define arcs as in VSP except that the arcs defined out of s and into t are defined out of each s_l and into each t_l. A set of flow variables $\{x_{ij}^l\}$ and costs $\{c_{ij}^l\}$ is associated with each source/sink pair $\{s_l, t_l\}$. Further, we associate lower and upper bounds $\underline{b}_l$ and $\bar{b}_l$ with each vehicle category l. In this case, an x_{ij}^l variable is only associated with arc (i, j) if traversal of the arc is allowed by a vehicle in category l. We denote by A_l all those arcs for which a traversal by a vehicle in category l is allowed. The multicommodity flow formulation is:

$$\text{V3:} \quad \text{Min} \sum_{l=1}^{k} \sum_{(i,j)\in A_l} c_{ij}^l x_{ij}^l$$

$$\text{s.t.} \quad \sum_{i:(i,j)\in A_l} x_{ij}^l - \sum_{i:(j,i)\in A_l} x_{ji}^l = 0$$

$$\text{for all tasks } j \text{ and categories } l, \tag{3.9}$$

$$\underline{b}_l \le \sum_{j:(s,j)\in A_l} x_{sj}^l \le \bar{b}_l \text{ for } l = 1, 2, \ldots, k \tag{3.10}$$

$$\sum_{l=1}^{k} \sum_{i:(i,j)\in A_l} x_{ij}^l = 1 \quad \text{for all task nodes } j \tag{3.11}$$

$$0 \le x_{ij} \le 1 \text{ and integer for } l = 1, 2, \ldots, k \text{ and all } (i, j) \in A_l. \tag{3.12}$$

For VSPMVT, an arc from task node i to task node j is included in A_l if a type l vehicle can service both tasks i and j. For VSPMD, an arc from task node i to task node j is included in A_l if both tasks i and j can be serviced by a vehicle housed at depot l. In both cases arcs out of s_i nodes and into t_i nodes are only included in A_l for $i = l$. We should note that the "driving force" of this formulation for VSPMVT is the definition of the feasible arc set A_l and the bounds $\underline{b}_l$ and $\bar{b}_l$ whereas the driving force for VSPMD is the definition of the arc costs for arcs out of each s_l and into each t_l.

The remaining sections describe algorithms for solving these problems.

The concurrent scheduler. The concurrent scheduler is a simple intuitive approach that has proved quite successful in practice for solving a variety of constrained scheduling problems. Bodin, Rosenfeld and Kydes [92] describe this algorithm as follows:

Step 1. Order tasks by starting time. Assign task 1 to vehicle 1.

Step 2. For $k = 2$ to the number of tasks. If it is feasible to assign task k to an existing vehicle then assign it to the vehicle that involves the minimum deadhead time. Otherwise, create a new vehicle and assign task k the new vehicle.

Depending on whether VSPLPR, VSPMVT, or VSPMD are involved, the appropriate feasibility check would be applied in Step 2. For VSPMD, the new vehicles created in step 2 would be assigned to the nearest depot.

We have found in our discussions with practitioners that this heuristic is widely used in practice. In addition, since it is easy to code and computationally efficient, it is useful for generating starting solutions for other algorithms. It can be interpreted as a member of the class of "greedy" heuristics, which have become popular in recent years because of their simplicitly and analyzability [150, 502].

Two step approaches. VSPMVT and VSPMD can both be viewed as vehicle scheduling/clustering problems in the sense that the output of each is a set of vehicle schedules clustered either by vehicle type or depot. This interpretation suggests two classes of approaches: one which clusters tasks and then schedules vehicles over each cluster and another which schedules vehicles and then clusters vehicle schedules. These approaches are not documented in the literature formally but have nonetheless been widely used by practitioners.

For the task clustering approach, we associate a weight with each task/cluster pair that measures how appropriate it is to include the task in the cluster. For VSPMVT, this weight

would be a 0/1 value depending on whether or not the vehicle type is feasible for that task. For VSPMD, the weight would measure the proximity of the task to the corresponding depot. Given these weights, a minimum weight assignment that satisfies lower and upper bounds on vehicle availability can be found by solving a transportation problem. The VSP is then solved over each cluster.

For the second approach, the VSP is solved over the entire task set. Subsequently, a transportation problem is set up in a manner similar to the one described in the previous paragraph. In this case, the transportation problem assigns entire vehicle schedules to clusters. This second approach has been applied successfully to VSPMDs, however, it is less appropriate for VSPMVTs since feasibility cannot be guaranteed and may be difficult to obtain.

It is interesting to note the similarity in philosophy of the approaches with multiple vehicle node routing approaches described in Section 2.4.4. The first procedure can be thought of as a "cluster first" approach while the second procedure can be thought of as a "route first" approach. In both the routing and scheduling problems we either group the required tasks into clusters, one cluster representing each depot, and then solve a single depot routing or scheduling problem or we form routes or schedules first and then assign these routes or schedules to the appropriate depot.

Finally, we should note that in many cases, a natural clustering exists within the physical system. For example, in mass transit systems, lines provide a natural clustering for trips. This clustering may be quite appropriate to use for either the VSPMVT or the VSPMD since trips arising from a single line have common characteristics with respect to locations and vehicle types required. We should caution, however, that very often clustering by line is enforced by transit agencies to an undesirable extent due to convenience or historical reasons.

An interchange heuristic. In this section, we describe an interchange heuristic [605] that has been applied very successfully in several mass transit agencies in England. It can handle a wide variety of cost functions and constraints. Side constraints are typically handled by using Lagrangean relaxation in the manner described in Section 2.5.

The procedure can be viewed as an adaptation of the 2-opt algorithm for the traveling salesman problem [445] (see Section 2.3). A starting solution can be found using a variety of approaches such as the concurrent scheduler. Assuming that a starting solution is already available, the procedure effects interchanges between the components of this schedule to improve costs. An interchange affects only two vehicle schedules, say, 1 and 2. It joins the first half of vehicle schedule 1 with the second half of vehicle schedule 2 and the first half of 2 with the second half of 1. In this case, the cost of the interchange cannot be evaluated by simply looking at the four arcs involved, but rather, the costs of the two new vehicle schedules must be compared to the cost of the two old vehicle schedules. This more global evaluation strategy will allow a variety of complex cost functions to be considered. Two cost functions that relate to the problems mentioned in this section are now described.

The first cost function is appropriate for handling the VSPMVT. The cost of a vehicle schedule is:

$$\underset{\text{vehicle type } l}{\text{Min}} \quad \sum_{i \in B(l)} (\mathrm{ET}_i - \mathrm{ST}_i)$$

where $B(l)$ is the set of tasks i in the schedule that are not feasible for type l vehicles. Note that this cost is zero if the vehicle schedule is feasible for some vehicle type. This objective function does not consider vehicle availability ($\underline{b}_l$ and $\bar{b}_l$). If the interchange heuristic produces a set of schedules that violate the vehicle availability, then a transportation problem is solved that assigns the vehicle schedules generated to vehicle types using the cost function described above and the restrictions on vehicle availability ($\underline{b}_l$ and $\bar{b}_l$). If the value of the transportation solution obtained is greater than zero (i.e. solution to vehicle scheduling problem is infeasible) then the interchange heuristic is reentered where certain vehicle types are penalized based on the shadow prices obtained from the transportation problem solution. The procedure continues to iterate in this fashion until a feasible solution is obtained or no further improvement is found.

The following cost function is appropriate for the VSPMD. The cost of a vehicle schedule is

$$\min_{\text{all depots } l} \{\text{cost to travel from } EL_j \text{ to depot } l \text{ plus the cost of travel from depot } l \text{ to } SL_i \text{ where } i \text{ is the first trip on the schedule and } j \text{ is the last}\}$$

Although reference [605] does not describe explicitly how to handle depot capacities ($\underline{b}_l$ and $\bar{b}_l$), presumably these could be handled by modifications in the technique described in the previous paragraph.

The two objective functions described above are aimed only at obtaining feasible solutions. In a practical application of this procedure, a weighted objective that includes the objective function components described above as well as components related to vehicle capital and operating costs would be appropriate.

3.5 CREW SCHEDULING ALGORITHMS

This section is divided into five subsections. Section 3.5.1 gives a general formulation that applies to all the crew scheduling problems discussed. Sections 3.5.2, 3.5.3, 3.5.4 and 3.5.5 present algorithms for scheduling workers at a fixed location, mass transit crew/vehicle scheduling, air crew scheduling, and rostering and bid line problems, respectively. As we noted in Section 3.3, fixed location worker scheduling provides insight and algorithmic intuition for the more complicated crew scheduling problems. When this problem is discussed in Section 3.5.2, we also discuss its application to estimating mass transit crew costs.

3.5.1 *General formulation for crew scheduling problems*

As with all scheduling problems, we start with a set of tasks as input. We represent the crew scheduling problem with a hierarchical set of constraints. The first level defines continuous crew work periods (CWPs). The key assumption concerning CWPs is that the crew pay depends only on the starting and ending times of the CWP. The second level of constraints groups sets of CWPs into full work schedules (FWSs). For fixed location worker scheduling, Figure 3.6 illustrates three FWPs consisting of two, one and three CWPs. In mass transit crew scheduling a CWP corresponds to a piece and an FWS corresponds to a run. In air crew scheduling a CWP corresponds to a duty period and an FWS corresponds to a pairing.

The first level of constraints consists of the constraint set for V2 (see 3.4 through 3.8). We start with a node set N that represents each task by a node. The arc set A_1 includes the arc (i, j) if a crew can perform task i followed by task j in a single CWP. Arc set A_2 includes all "back arcs" (j, i) for which a CWP starting at task i and ending at task j is feasible. The constraint set for V2 applied to this network yields the first level of constraints.

To complete the formulation, we must describe the manner in which CWPs are formed into FWSs. An FWS is simply a set of CWPs. Typically an FWS consists of a small number, e.g. 2, 3 or 4, of CWPs. We call a potential FWS a *pattern* and define ρ = the set of all FWS patterns, $p(i, j)$ = the set of all patterns that cover a CWP starting with task node i and ending with task node j.

The constraint required to insure that all CWPs are covered by an FWS is

$$\sum_{l \in p(i,j)} z_l - y_{ji} = 0 \quad \text{for all } (j, i) \in A_2 \tag{3.13}$$

$$0 \leq z_l \leq 1 \text{ and integer} \quad \text{for all } l \in \rho \tag{3.14}$$

where

$$z_l = \begin{cases} 1 & \text{if pattern } l \text{ is used,} \\ 0 & \text{otherwise.} \end{cases}$$

To define the mathematical program completely, we associate costs d_l with each $l \in \rho$ as

well as c_{ij}'s with each $(i, j) \in A_1 \cup A_2$. We now have

$$\text{CI: Min} \sum_{l \in \rho} c_l z_l + \sum_{(i,j) \in A_1} c_{ij} x_{ij} + \sum_{(i,j) \in A_2} c_{ij} y_{ij}$$

$$\text{s.t. (3.4), (3.5), (3.6), (3.7), (3.8), (3.13), (3.14).}$$

Generally, all crew costs can be captured in the d_l's. In Section 3.5.3 we apply this formulation to mass transit crew/vehicle scheduling. In that case, the c_{ij}'s are used to handle vehicle costs.

We should note that it typically is impractical to use this formulation explicitly to solve scheduling problems. One reason is that the amount of data required to "write it down" can be quite large. In particular, the number of patterns can grow in proportion to n^{2k} where n is the number of tasks and k is the number of CWPs per FWS. Even with k equal to 2 or 3 this growth rate can be prohibitively large.

We will describe several algorithms in terms of this formulation. In most cases, the algorithms employ two or more phases, with the early phases assigning values to a subset of the variables. Once certain variables are fixed, the resultant problem becomes smaller and much more tractable.

3.5.2 *Algorithms for scheduling workers at a fixed location*

As we discussed in Section 3.3.1, this problem is characterized by a demand histogram and a set of possible work days. We may specify the demand histogram by defining

T = number of time intervals,

d_t = number of workers required during interval t for $t = 1, 2, \ldots, T$.

To make this demand specification compatible with the model given in the previous section we must define a set of tasks. In the most direct representation, each time interval t corresponds to a set of d_t tasks whose start and end times are the start and end times of interval t. Rather than explicitly representing each of these tasks, we represent each time interval t with a single node in N and replace constraint (3.5) with the following constraint which states that task t must be covered by at least d_t workers,

$$\sum_{i:(i,j) \in A_1} x_{ij} + \sum_{i:(i,j) \in A_2} y_{ij} \geq d_t \quad \text{for } t = 1, \ldots, T. \tag{3.15}$$

We also allow all variables to take on values greater than one which means that several crews may follow the corresponding arc or pattern. This is accomplished by dropping the upper bounds from (3.7), (3.8) and (3.14). As Fig. 3.13 illustrates in the network associated with specification of CWPs for the fixed location worker scheduling problem, the set of all forward arcs forms a single path. It is not difficult to see that in such networks constraint (3.6) may be deleted to obtain a set of network flow constraints. This observation does not mean that the entire problem can always be solved as a network flow problem. However, if each FWS consists of exactly one CWP, i.e. if all workdays contain no rest or lunch breaks, then since constraints (3.13) and (3.14) are not required, we may solve the fixed location worker scheduling problem as a network flow problem.

When breaks in the workdays are allowed, the problem remains quite difficult even though the CWP constraints have a network flow structure. In this case, we give another formulation of the problem [76]. We define

n = number of possible workdays,

$$a_{tj} = \begin{cases} 1 & \text{if work day } j \text{ covers period } t \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for } j = 1, 2, \ldots, n,$$

c_j = cost of work day j, for $j = 1, 2, \ldots, n$,

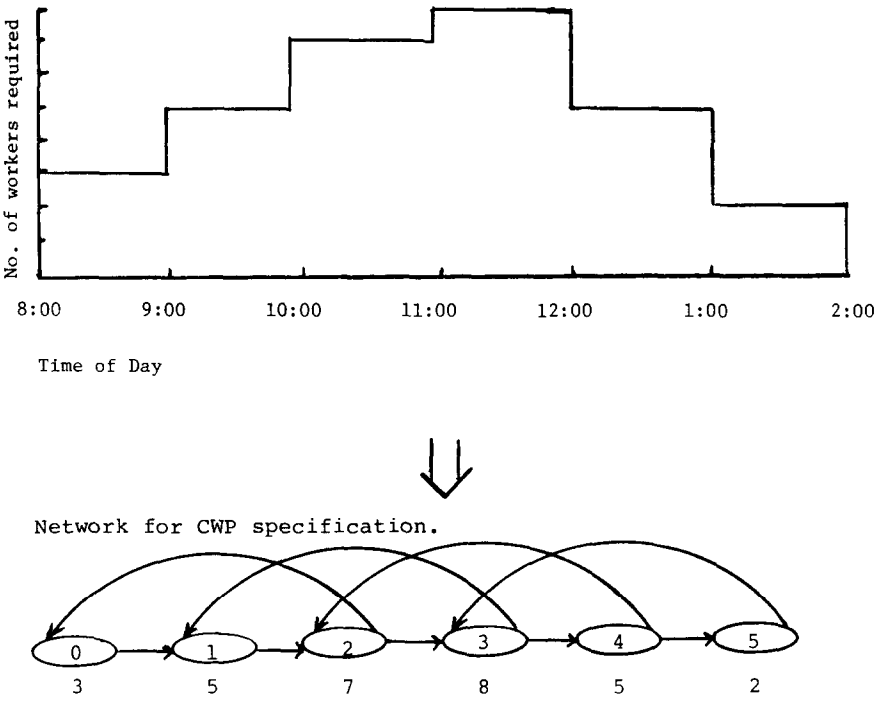


Fig. 3.13. Network associated with CWP specification for fixed location worker scheduling problem.

then we may write the fixed location worker scheduling problem as,

F1: Minimize $\sum_{j=1}^{n} c_j x_j$

subject to

$$\sum_{j=1}^{n} a_{tj} x_j \geq d_t \quad \text{for } t = 1, 2, \ldots, T, \tag{3.16}$$

$$x_j \geq 0 \text{ and integer for } j = 1, 2, \ldots, n. \tag{3.17}$$

Note that this formulation is, in fact, a generalization of the set covering problem.

Examples of the use of the model discussed in this section can be found in Bennett and Potts[67], Bodin *et al.*[91] and [92], Heller[340], Monroe[494], Guha and Browne[321], Jenkins[365], Miller[484] and Segal[589].

This model has also been used in conjunction with mass transit crew scheduling to estimate crew requirements[76], [92] and as a first step in a crew scheduling algorithm[437]. In this case, a time interval length is arbitrarily chosen with accuracy increasing as the time interval length decreases. We assume that vehicles have already been scheduled and define

d_t = the number of vehicles that operate during the entire time interval t.

Note that d_t is a lower bound on the number of crews required to be on duty during time period t. Even with time intervals defined as small as possible, e.g. 1 minute, this is still an approximate model in that it ignores crew movements required to get from the end location of one piece to the start location of another.

This model has been used as part of the UCOST system developed by the Department of Transportation[92] for long range planning. In this case, the histogram was partitioned into continuous $\frac{1}{2}$ day shifts no more than about $4\frac{1}{2}$ hours in length. It was assumed that each shift

could be paired with some other shift to form a full day run. In this way an estimate of the total number of required crews (half the number of shifts) was obtained using a flow algorithm in a situation where non-continuous work days were allowed.

Jean Marc Rousseau and his colleagues [76, 437] have used the model both to estimate crew requirements and as a first step in crew scheduling. Their model allows work days to contain explicit breaks and therefore rules out using a flow algorithm. Instead, they use formulation F1 and relax the integrality restriction on the x_j's so that the problem may be efficiently solved using a linear programming code. In this case, an additional set of linear constraints of the form

$$\sum_{j=1}^{n} b_{ij}x_j \leq e_i \quad \text{for } i = 1, 2, \ldots, m \tag{3.18}$$

is added to the model. These constraints serve two purposes: first, to enforce a variety of different union restrictions and crew work day requirements, and second to make the model itself conform more closely to reality. A constraint of the first type might be that at least 50% of all runs must be straight runs. A constraint of the second type might be that if there are 10 short vehicle blocks starting at 7:30 AM and ending at 9:00 AM then there must be 10 runs with pieces starting at 7:30 AM and ending at 9:00 AM. This procedure has been used to estimate the economic impact of new union regulations proposed during union negotiations. The additional set of constraints (3.18) is particularly useful in this regard.

3.5.3 *Mass transit crew/vehicle scheduling algorithms*

Two major workshops dealing with mass transit crew and vehicle scheduling have been held: one in Chicago in 1975 [74] and the other in Leeds, England in 1980 [692]. The proceedings of the latter [692] provide an accurate representation of the state of the art in this area. The first attempts at computerized crew scheduling either followed closely the techniques used by manual schedulers ("run cutting") or viewed the problem explicitly as a set partitioning problem. Initially, both of these approaches were unsuccessful. However, over the past few years, a large amount of research has been directed toward the two approaches and both have produced good results. In addition to discussing these two areas, we present two new approaches, one which initially solves the crew estimation problem described in Section 3.5.2 [437] and one which simultaneously generates crew and vehicle schedules using several matching-based heuristics [35]. We should mention that aside from this last approach, all other approaches described start with a solution to the vehicle scheduling problem (i.e. operate sequentially).

Before describing individual procedures we interpret, in terms of the formulation given in Section 3.5.1, the effect of solving the vehicle scheduling problem prior to scheduling crews. Recall that in the mass transit crew/vehicle scheduling problem, the set of input tasks is the set of d-trips. The start of each d-trip is either the beginning of a trip (BT) or a relief point (RP) and the end of each d-trip is either the end of a trip (ET) or a relief point. Forward arcs (arcs in A_1) are drawn from each ET to BT exactly as in the vehicle scheduling network except that no "long arcs" are allowed. One forward arc is drawn from each d-trip ending in an RP to the d-trip that starts with the corresponding RP. The use of an ET to BT arc has the same interpretation as in the VSP, namely that the crew and vehicle proceed from the end of the first trip to the beginning of the next. The use of an RP to RP arc simply means that a crew does not exit the vehicle at that relief point. Back arcs represent feasible pieces, for example, a back arc from a node ending in an RP to a node beginning with a BT would represent a piece which started with the crew pulling the vehicle out of the garage and ended with the crew exiting the vehicle at a relief point.

The effect of scheduling vehicles is to specify values for the BT to ET arc variables. This greatly simplifies the resulting network. As Fig. 3.14 illustrates, after scheduling vehicles the network consists of a set of isolated subnetworks, each one corresponding to a single vehicle. Each of these subnetworks has the network structure of the fixed location worker scheduling problem of Section 3.3.2. This structure is used in solving a subproblem in the algorithm of [437]. A further effect of scheduling vehicles is that the number of potential patterns is greatly reduced. Set partitioning algorithms must enumerate all or a substantial number of the patterns.

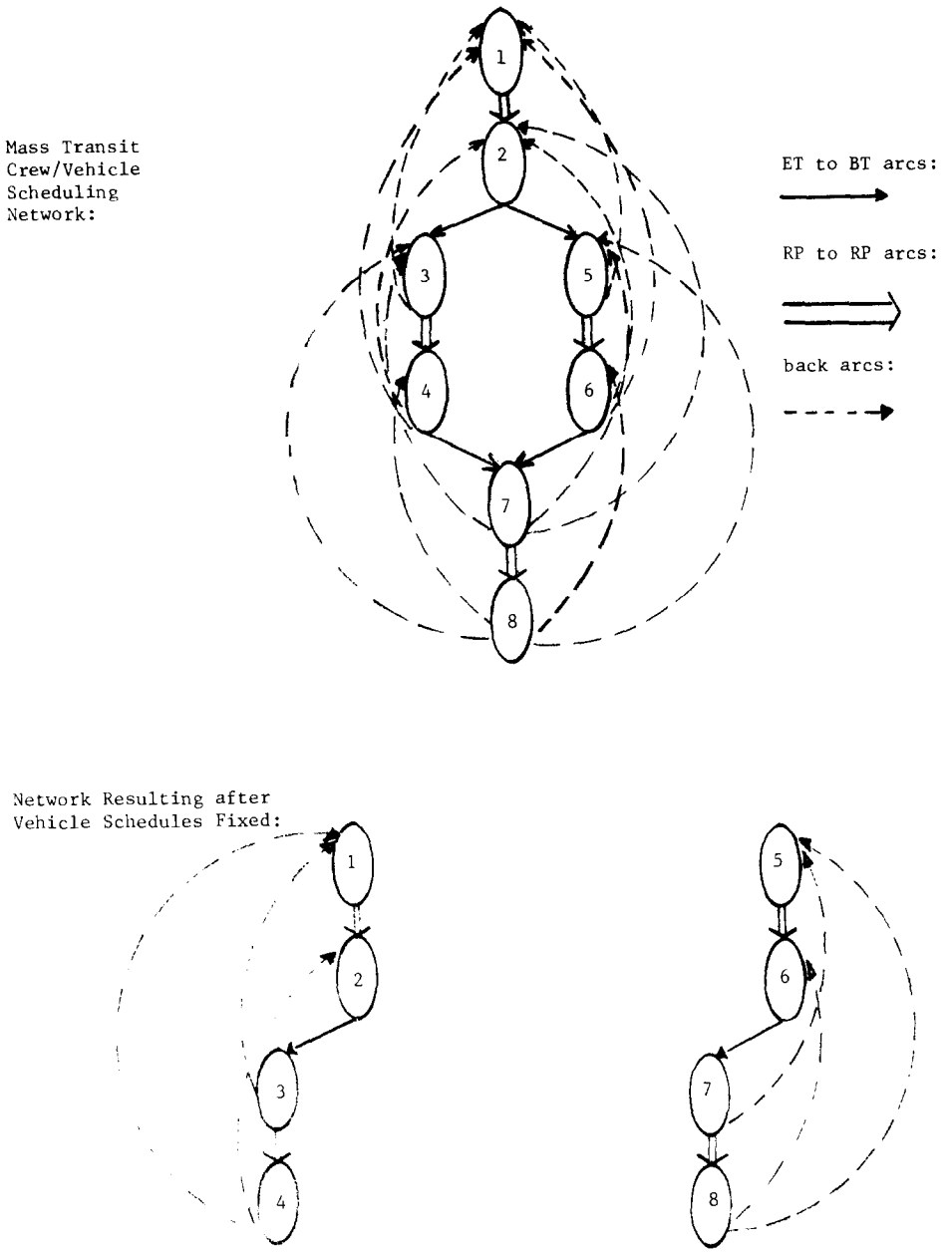


Fig. 3.14. The effect of fixed vehicle schedules on mass transit crew scheduling problem.

Thus, this reduction is extremely important. Of course, it is possible that scheduling vehicles first can eliminate many cost effective solutions (including the optimal solution).

Set partitioning based approaches. Set partitioning and covering methods have generally not been used in operational settings within mass transit agencies. However, a large amount of recent research[490, 539, 574, 665] has been directed toward limiting problem size by decomposing the problem or restricting the set of columns generated. To understand these approaches we must reexamine the service profile graph given in Fig. 3.9. Ward *et al.*[665] use a decomposition approach which classifies runs according to where their component pieces fit into the service profile. They then solve a sequence of set partitioning problems with the first problem generating the runs that end the latest (owls), the second problem generating the runs that end the second latest (late straights) and so forth until all trips are covered. Of course, the set of columns generated at any step will be constrained by the runs generated during previous steps. Since all columns are not considered simultaneously the approach cannot guarantee an optimal solution to the set partitioning problem.

Parker and Smith[539] include all trips in the service profile within one set covering problem, however, they use a variety of criteria to restrict the set of columns considered. These include: (a) lower and upper bounds on piece length; (b) lower and upper bounds on length of breaks; (c) restrictions on piece start and end times based on run classification similar to the one shown in Fig. 3.9.

It is widely felt that the service profile contains valuable information that should be used in generating crew schedules. These two approaches attempt to use this information. The approach of Lessard *et al.*[437], which is described later in this section, includes profile information within the crew scheduling algorithm by solving the estimation problem described in Section 3.5.2. The two set partitioning/covering methods just described have undergone some computational testing and the results appear promising. However, they have not yet been used in an operational environment.

The matching problem is the set partitioning problem with no more than two ones per column. An efficient exact solution procedure exists for this problem[193]. Several approaches (e.g. [35], [437]) to mass transit crew scheduling initially generate a set of pieces or a subset of the runs together with some "left-over" pieces and then solve the problem of generating runs from pieces using a matching algorithm.

Run cutting approaches. It was generally felt that manual schedulers produced extremely good solutions, with the primary disadvantage of the manual approach being the time it took to produce a new solution (several weeks) and the time it took to train new schedulers (several years). While this assessment is true to a certain extent, there have been significant cost savings produced in recent years[413, 437, 582] using computerized techniques. We refer to the general set of heuristic approaches used by manual schedulers as *run cutting*. The first large-scale computerized implementation of run cutting was called RUCUS; it was developed under sponsorship of the Urban Mass Transportation Administration. The results were mixed. The system was cumbersome to use, rather inflexible and operated in a "batch" environment. Consequently, it became very difficult to incorporate minor changes in work rules and problem parameters. Some of the earlier computational experience with RUCUS is presented in Goeddel[285].

A recent implementation[345], which is used in an operational environment, is based on heuristic methods inspired by the manual schedulers. This method can be viewed as a refinement of the initial RUCUS technique. Some of the main embellishments that contributed to the success of this package include: (a) a highly flexible interactive user interface; (b) a large set of parameters for specifying the problem and for fine-tuning the algorithm; (c) a flexible system for manipulating input data.

We will now describe a procedure which is typical of the methods employed in [673] and [345]. The algorithm "cuts" pieces of work from a pair of vehicle schedules and fixes this pair as a run. To choose the first piece the algorithm scans all vehicle blocks for the earliest relief point beginning a piece of unscheduled work (block A in Fig. 3.15). It then determines a set of possible relief points along this same vehicle block for ending the piece. These relief points are determined based on maximum and minimum piece lengths for both the first piece on the run being generated and the leftover piece. For each one of these relief points, candidate pieces from other blocks (block B in Fig. 3.15) are found for forming possible two piece runs. Each candidate piece is evaluated in terms of the run formed and in terms of the quality of the segments remaining after the piece is cut out of the vehicle schedule. For example, having a very short piece among the remaining segments would be considered undesirable since it would be difficult to form a good run using this piece. This procedure depends heavily on the fine-tuning of several subjective cost parameters (e.g. cost of left-over segments).

An algorithm based on service profile decomposition. Lessard *et al.*[437] use a two step approach for solving the mass transit crew scheduling problem: the first step generates pieces and the second merges pieces into runs. They use information contained in the service profile to guide the piece generation phase. This information is obtained by solving the linear programming relaxation of the histogram partitioning problem described in Section 3.5.2. We should note that this relaxation can be shown to be a relaxation of the crew scheduling problem itself.

A solution to the linear programming relaxation in effect specifies a "shape" for the crew scheduling solution. For example, it might specify that there should be about 10 runs consisting

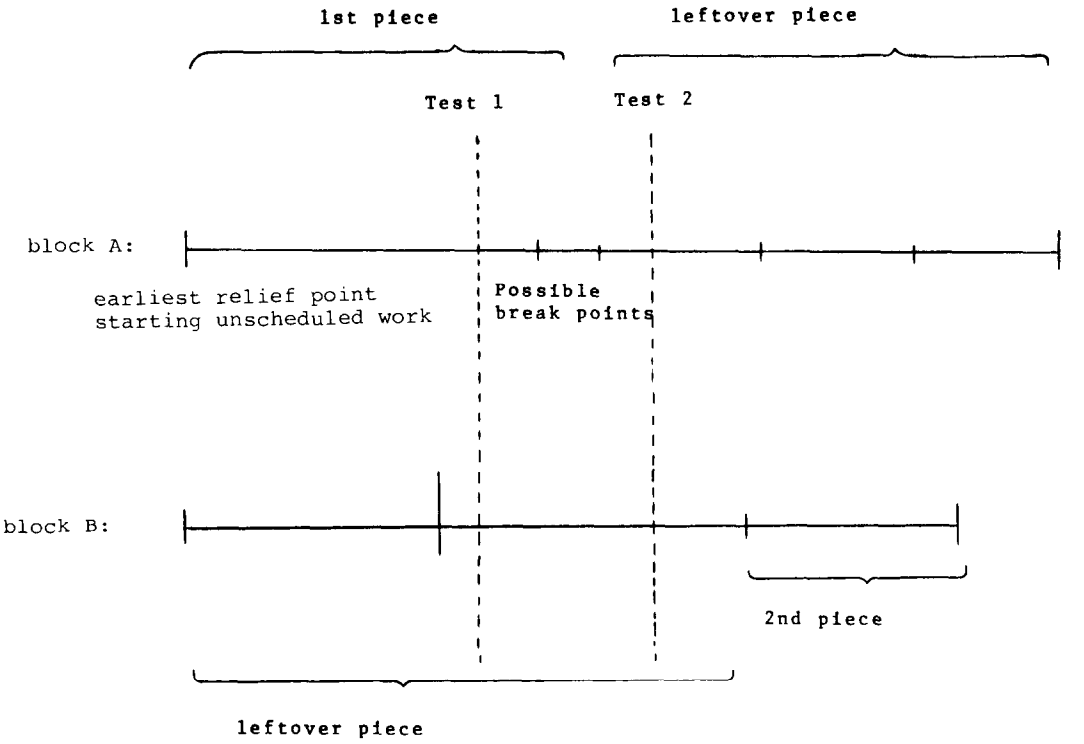


Fig. 3.15. Creation of two-piece runs using a run cutting algorithm: first piece of block A and second piece of block B are joined subject to tests 1 and 2.

of a short peice during the AM peak and a long peice during the PM peak, about 15 runs with two equal size pieces, one in the AM peak and one during the mid-day, etc. This in turn may be viewed purely as information concerning pieces, e.g. 10 pieces should start at 7:00 AM and end at 10:15 AM, 5 pieces should start at 7:30 AM and end at 11:30 AM, etc. Since this set of pieces makes up the solution to a relaxation of the crew scheduling problem the corresponding cost is a lower bound on the optimal solution. Consequently, if a real set of pieces is found that is "close to" this lower bound piece set, its cost should be close to the optimal solution. A quadratic objective function is used for the piece generation phase which minimizes the discrepancy between the linear programming solution and the piece set obtained. The overall solution procedure can now be outlined:

Step 1. Solve the linear programming relaxation of the histogram partitioning problem.

Step 2. Partition the vehicle blocks into pieces, so that the resulting piece set conforms as closely as possible to the set of pieces specified by the solution to the linear program.

Step 3. Combine peices into runs.

Step 4. Try to improve the solution by making marginal changes in the partition of vehicle schedules into pieces.

Step 1, of course, can be solved efficiently using a linear programming code. A heuristic is used for Step 2 which solves a sequence of shortest path problems. This algorithm makes use of the network structure illustrated in Fig. 3.14. Note that, in terms of the formulation given in Section 3.5.1, once Step 2 is completed the only unspecified variables are the z_l's. Thus the only relevant constraints are (3.13) and (3.14) where in (3.13) the y_{ji} values are fixed. This remaining problem can be viewed as a matching problem. Since it is rather large, Step 3 solves it using a heuristic that makes use of a matching code. This procedure is being successfully used in an operational environment in Quebec City and Montreal. In Montreal it produced a savings of 3% in crew costs. Experimentation with using other relaxations in Step 1 are given in [113].

A matching based algorithm. In Ball, Bodin and Dial[35, 36], the crews and vehicles are scheduled simultaneously. The procedure consists of two phases corresponding to the two levels of constraints in the formulation of Section 3.5.1. The first phase specifies pieces, i.e. the values for the x_{ij} and y_{ij} variables and the second phase merges pieces into runs, i.e. specifies values for the z_l variables.

The network used in the first phase does not include the back arcs. Rather the problem is viewed as partitioning the nodes in an acyclic network into paths restricted in length. This path length restriction, T, would be the maximum piece time, e.g. 4–1/2 hours.

The major steps of the algorithm are:

Step 1. Form the crew/vehicle scheduling network defined at the beginning of this section but do not include back arcs. Partition the nodes in this acyclic network into a set of paths (pieces) whose lengths are less than or equal to T.

Step 2. Improve the set of pieces by performing combination/splits of pairs of pieces.

Step 3. Combine pairs of pieces into runs or crew schedules.

Step 1 starts by partitioning the nodes into a set of levels and initially defines a set of "partial paths" as the nodes on level 1. It iteratively augments the partial paths by the nodes on the next higher numbered level using a matching based heuristic. Step 2 uses a matching based interchange heuristic. As was mentioned earlier, the problem of combining pieces into runs (Step 3) can also be formulated and solved as a matching problem. This matching problem usually has side constraints associated with it such as a constraint that at least 50% of all runs must be straight runs.

Although this procedure has not been used in an operational environment, it has been tested on a data base from the city of Baltimore and has produced significant cost savings (see Ball, Bodin and Dial[35]).

3.5.4 *Air crew scheduling algorithms*

A scheduling problem which has tantalized combinatorial analysts for many years is the air crew scheduling problem, i.e. the problem of forming a set of pairings which cover all the flight legs in the timetable. As is demonstrated in Section 3.5.1, the air crew scheduling problem and the mass transit crew/vehicle scheduling problem have a number of similarities. However, while a variety of methods have been proposed for the mass transit problem, research on air crew scheduling has centered on using the set partitioning model. The primary reason for this is that air crew scheduling problems are typically much smaller. In particular, problems with 100 to 300 flight legs are considered quite realistic in air crew scheduling whereas in mass transit crew scheduling a small to moderately sized problem might have 1500 d-trips and larger problems could have several thousand d-trips. Other differences are that: (1) the air crew scheduling problem does not have the highly peaked demand structure illustrated in Fig. 3.9, (2) the air crew scheduling problem covers a longer time horizon, i.e. a set of pairings covers a month whereas a set of runs a day, and (3) a duty period is not necessarily carried out on a single airplane whereas a piece is always carried out on a single vehicle.

The set partitioning model was first used to solve airline crew scheduling problems a number of years ago[610]. However, the early attempts to solve these problems were largely unsuccessful due to two difficulties: (1) the problems attempted were simply too large for existing exact codes, e.g. 400 rows and 30,000 columns, and (2) effective heuristic codes were not available. More recently, the situation has improved dramatically[471] due to the development of better algorithms and codes and more importantly, as a result of an explicit consideration of problem size in choosing an approach. Current set partitioning codes can solve problems with up to approximately 150 rows. Heuristics have been used to solve problems of larger size. Problems within this range are obtained by looking at smaller airlines or by restricting one's attention to smaller aircraft fleets. In addition, in many cases, larger problems can be decomposed into components in such a way that it suffices to solve a few small set partitioning problems.

In this section, we present an overview of the exact set partitioning based approaches of Rubin[570] and Marsten and his colleagues[470, 471]. In addition, we describe a set of heuristic set partitioning based procedures and solution improvement methods developed by Baker and others[20, 23–25].

Exact set partitioning approaches. The two approaches described in this subsection are based on solving a set partitioning problem using an exact set partitioning algorithm, e.g. branch and bound. However, since in many cases the set partitioning problem solved is an aggregation or restriction of the original crew scheduling problem, optimality is not necessarily guaranteed.

One of the first uses of the set partitioning approach for air crews was the algorithm developed by Rubin[570]. It proceeds as follows:

Step 0. Find an initial feasible set of pairings P.

Step 1. For $i = 2$ to $\bar{k}$. For each subset S of P of size i let R be the set of legs (rows) covered by S. Find the set, A, of all pairings that cover exactly the set of rows R. Solve the set partitioning problem with row set R and column set A. If the cost of the solution is less than the cost of S, then replace S in P with the solution found.

Generally, $\bar{k}$ is chosen to be rather small, i.e. 2 or 3. This procedure became a standard approach in the airline industry and several adaptations of this procedure are still in use by various airline companies. This procedure is very slow computationally, but it tends to give good answers in most applications attempted.

Marsten and Shepardson[471] and Marsten, Muller and Killion[470] report on three highly successful uses of Marsten's linear programming-based set partitioning algorithm[469] for solving the air crew scheduling problem. The airlines involved in the study were The Flying Tiger Line, Pacific Southwest Airways, and Continental Airlines. In most cases, problems with fewer than 130 rows and 30,000 columns were solved to optimality or to within 1% of optimality. The yearly cost savings produced by these solutions were on the order of hundreds of thousands of dollars. The problem sizes involved are fairly modest due to the fact that the airlines involved had relatively small fleets and that the schedulers created *resolved legs.* A resolved leg is a sequence of regular flight legs that have to be flown together as a unit. On a resolved leg, the trips are so grouped that there is no opportunity for the crew to change planes. Also, a sophisticated column generation procedure was used to eliminate columns that were not likely to appear in an optimal solution. This use of an exact set partitioning algorithm in this context does not guarantee optimality since the resolved leg represents an aggregation of the rows in the overall set partitioning problem and since the elimination of columns could discard the optimal solution from the set of solutions considered by the algorithm.

Heuristic set partitioning. Heuristics are the only recourse for most of the larger crew scheduling problems encountered in practice. Baker[20] and Baker, Bodin and Fisher[24] describe the use of heuristic set covering algorithms for solving air crew scheduling problems. These algorithms, together with solution improvement methods which are described in the next subsection, are being used by the Federal Express Corporation for both single and multiple depot crew scheduling. Similar approaches have been used by the Selective Bidding Corporation. These algorithms are quite fast and have been compared favorably with competitive procedures.

The Federal Express crew scheduling system contains three steps: (i) a pairings generator used to enumerate candidate crew schedules, (ii) a heuristic set covering algorithm, used to obtain an initial covering solution, and (iii) several solution improvement techniques used to reduce the total cost of the covering solution. Steps (i) and (ii) are described below and Step (iii) is described in the next subsection.

The pairings generator enumerates new pairings which are feasible vis-a-vis the constraints described in Section 3.3.3 and computes the cost of each pairing. In this enumeration, subjective reduction procedures are incorporated to limit the number of pairings generated and to reduce the computer run time of the generator module. The imposition of subjective reduction procedures in the enumeration process was used to enforce desired operational policies associated with the crew pairings (e.g. maximum time away from base, maximum layover time, maximum time away from base to flight time ratio, etc.).

Four distinct heuristics for finding feasible solutions to the set covering problem were evaluated. The first procedure, termed Algorithm A, begins with an empty solution set and adds to the solution the crew schedule that minimizes the cost per uncovered flight. The procedure continues until all flights are covered. The second procedure, Algorithm B, considers the crew schedules in ascending order of cost per flight leg covered. The algorithm begins with an empty solution set and adds the first crew schedule from the sorted list that contains no previously covered flights. If a solution is not obtained after a complete pass through the crew schedules, the process is repeated allowing candidate crew schedules to contain one previously covered flight. This process is repeated, allowing increased overcoverages until a complete solution is obtained.

The third procedure considered, Algorithm *C*, was proposed by Johnson[367] for the case where all columns have costs equal to one; this gives the minimum number of pairings needed to cover all the legs. Algorithm *C* executes in a manner identical to Algorithm *A* except that crew schedules are selected based on the maximum number of uncovered flights. The rationale of this approach is that if the column generator contains subjective tests that produce only efficient crew schedules, then a minimum set of such schedules may produce a near-optimal solution. The fourth procedure, termed Algorithm *D*, is Johnson's technique for the exact covering problem. This procedure starts with an empty solution set and adds to the solution the crew schedule that minimizes the ratio of covered to uncovered flight segments. The rationale of this algorithm is similar to that of Algorithm *C*.

Baker experimented with several data bases from Federal Express Corporation. The computational results indicate that Algorithm *A*, the selection of crew schedules according to a minimum cost per uncovered flight leg criterion, dominates the other set covering heuristics considered. Additional computational evidence in support of Algorithm *A* as an effective set covering heuristic may be found in the recent papers of Balas and Ho[30]. In addition, in Baker[20] and in Ho[351], discussions of worst case bounds for set covering heuristics may be found.

Solution improvement procedures. Baker, Bodin, Finnegan and Ponder[23] and Baker, Bodin and Fisher[24] propose three solution improvement procedures: an extended 2-opt algorithm, a solution merge technique and a hub turn determination procedure. A brief description of these solution improvement procedures is given below.

The extended 2-opt procedure, which assumes a single feasible solution to the set covering problem as input, is an adaptation of the 2-opt procedure proposed by Lin[443] for the traveling salesman problem (see Section 2.3). In this interpretation of the air crew scheduling problem, each flight leg to be scheduled is viewed as a city to be visited. The exchange of arcs connecting the cities on a traveling salesman's route is analogous to changing the sequence in which the flight legs are scheduled. Since any such exchange must take place at a stop common to all the candidate pairings, each city is examined for possible exchanges of pairings completions at that point. If k pairings transit a particular city, then a $k \times k$ assignment problem may be solved to determine the optimal reconfiguration of the crew schedules.

The second solution improvement procedure employed was a solution merging technique. Input to the merge procedure is a *set* of feasible solutions. Solution merging, which was originally proposed by Hinson and Mulherkar[350] for the Federal Express aircraft routing problem (see Chapter 4), exploits the availability of feasible covering solutions provided by the set covering heuristics. The first solution found becomes the incumbent. As subsequent solutions are found they are compared to the incumbent to determine if a reduced cost composite solution can be formed. Since this technique compares only two solutions at a time, the determination of the lower cost composite solution can be made very efficiently. The merge problem can be viewed as a restricted set partitioning problem where the rows are the legs and the columns are the pairings from the two solutions. This problem is restricted in the sense that only a fraction of the set of all possible columns are considered.

The final solution improvement technique employed involved joining two pairings together into a single pairing which is hubturned. A hubturn occurs when a crew transits the crew base on a pairing. All pairings start and end at the crew base. It is often possible to exploit the nonlinearity of the crew scheduling objective function by allowing two individual pairings to be performed sequentially as a reduced cost hubturn pairing. A matching algorithm can be used to find a hubturned solution. The nodes in the matching problem are the pairings in the best solution found so far and the arcs represent joining two pairings into a pairing with a hubturn. The costs on the arcs are the actual cost for a hubturned solution. The node costs are the costs of corresponding pairings without hubturns.

These procedures together with a heuristic set partitioning algorithm are being used by Federal Express Corporation to solve single hub crew scheduling problems. Computational results may be found in [23] and [24].

3.5.5 *Algorithms for rostering and bid line problems*

As described in Section 3.3.4, the output of the algorithms described in Section 3.5.3 and 3.5.4 may or may not be the final solution to the corresponding crew scheduling problem. The

output of the algorithms in Section 3.5.3 are mass transit crew schedules (runs) that cover all work on a particular day. The rostering problem [67, 321, 341, 112] refers to the problem of sequencing runs over a week, month or longer period to form the work plans for individual crews over that period. The output of the algorithms in Section 3.5.4 are air crew schedules (pairings) lasting a few days that typically cover all work in a given month. The bid line problem [214] refers to the problem of sequencing sets of these pairings into monthly work plans for individual crews. An important distinction between these two problems is that in the rostering problem the grouping of runs by day provides important structure to the problem, namely, that no crew can perform two runs on the same day and that the same runs are repeated each day with variations occurring only on weekends. No such structure exists in the bid line problem.

We may describe these problems in the same general terms with which we have characterized other scheduling problems. In this case, the input set of tasks is the set of runs or set of pairings defined over the appropriate time period. Note that with each run or pairing we may associate start and end times and start and end locations. In this case, however, the start and end locations are always the depot. Thus, many of the spatial issues associated with previous problems are no longer relevant. The precise nature of rostering and bid line problems varies significantly depending on the particular real world setting. In addition, these problems have not been as well studied as some of the other scheduling problems. We present two different approaches here that apply to two different well-structured problem settings. We summarize some of the other references that generally do not introduce new mathematical models or algorithms but rather give experience in solving these problems in real-world settings.

A large body of research has been directed toward so-called cyclic scheduling problems [26, 27, 45]. The model used fits into the class of fixed location worker scheduling problems described in Sections 3.3.1 and 3.5.2 and may be applied to certain rostering problems. In this case the time interval, t, is one day and the demand d_t used in constraint (3.16) is the number of runs required for day t. In the simplest form of this problem, the time horizon is one week and the d_t's are equal for weekdays. We wish to cover all requirements with consecutive sets of 5 day periods where the last day (Saturday) and first day (Sunday) are considered consecutive. The associated matrix $A = \{a_{ij}\}$ is illustrated in Fig. 3.16. This particular case of the fixed location worker scheduling problem with all $c_j = 1$ can be easily solved optimally in polynomial time [26, 27] by finding the solution to a set of equations. Several slight variations of this problem can also be efficiently solved. The key property of the matrix A is that ones appear consecutively in either the rows or columns. For the case shown to be solvable by network flow methods in Section 3.5.2, the ones always appear consecutively in the columns of A. Such matrices are known to be totally unimodular [246], which, of course, is related to the fact that they are solvable using network flow methods. The columns in the matrix in Fig. 3.16 have consecutive ones where the last and first columns are defined to be consecutive. Such matrices,

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Each column represents a possible worker schedule;
the 7 rows correspond to the 7 days of the week;
the "consecutive ones" property found in this matrix
arises from the requirement that workers must work
in consecutive days (Saturday and Sunday are consec-
utive).

Fig. 3.16. Cyclic scheduling matrix for single week.

in general, are not totally unimodular, however, in [45] it is shown that in most cases these problems can be solved in polynomial time by iteratively solving several network flow problems. Reference [44] solves further generalizations using linear programming roundoff heuristics.

The model just described essentially considered all runs in a particular day with equal weight. This assumption is generally not a good one. The typical objective of rostering for European mass transit systems is to balance work requirements among all crews since each crew receives the same pay, which is independent of the particular work performed. In such cases the objective is to produce rosters that are as equitable as possible in terms of total hours worked. In [112], this problem is addressed by using an objective function that minimizes the maximum weight (time) roster. In this case an n-level network, (N, A), is defined where n is the number of days in the time period over which the roster is to extend. We assume that the same set of runs is to be repeated on each day. A node in the network denoted by $\langle i, k\rangle$ represents run i in day (level) k. An arc from node $\langle i, k\rangle$ to node $\langle j, k+1\rangle$ is included in the arc set A if runs i and j can be performed on the same roster in succeeding days (see Fig. 3.17). We denote such an arc by $(i, j; k)$ and associate 0/1 variables x_{ij}^k with each such arc. A weight w_i is associated with each run i. This weight represents the time duration of the run i. The rostering problem may now be formulated as a multi-level bottleneck assignment problem [112]:

R1:

$$\text{Min } z$$

$$\text{s.t.} \quad \sum_{i:(i,j;k)\in A} x_{ij}^k = 1 \quad \text{for} \quad k = 2, \ldots, m \text{ and all runs } j \tag{3.19}$$

$$\sum_{i:(j,i;k)\in A} x_{ji}^k = 1 \quad \text{for} \quad k = 1, \ldots, m-1 \text{ and all runs } j \tag{3.20}$$

$$z_j^1 = w_j \quad \text{for all runs } j \tag{3.21}$$

$$z_j^k = w_j + \sum_{i:(i,j;k-1)\in A} z_i^{k-1} x_{ij}^{k-1} \quad \text{for } k = 2, \ldots, m \text{ and all runs } j \tag{3.22}$$

$$z_j^n \le z \quad \text{for all runs } j \tag{3.23}$$

$$0 \le x_{ij}^k \le 1 \quad \text{for all } (i, j; k) \in A. \tag{3.24}$$

Constraints (3.19) and (3.20) imply that the 0/1 x_{ij}^k variables will represent a partition of the nodes into paths of length n, where each path passes through exactly one node on each level. Constraints (3.21), (3.22) and (3.23) allow us to interpret z_j^k as the weight of the partial path starting with a node on level l and ending with a node on level k and z as the weight of the maximum weight path.

In [112], it is shown that his problem is *NP*-complete. A heuristic solution algorithm is presented which involves iteratively solving several bottleneck assignment problems over nodes on level k and $k+1$. Efficient algorithms exist for solving the bottleneck assignment problem [178], which is an assignment problem with a mini–max objective function. Theoretical as well as computational results are given in [112] which indicate that this algorithm is quite effective.

We should point out that neither of the approaches just described provides a complete solution to the rostering problem. The objective function used in the first model is not that realistic and the second model does not treat the issue of days off. Neither model is easily extendable to time horizons larger than one week whereas in most realistic settings a single

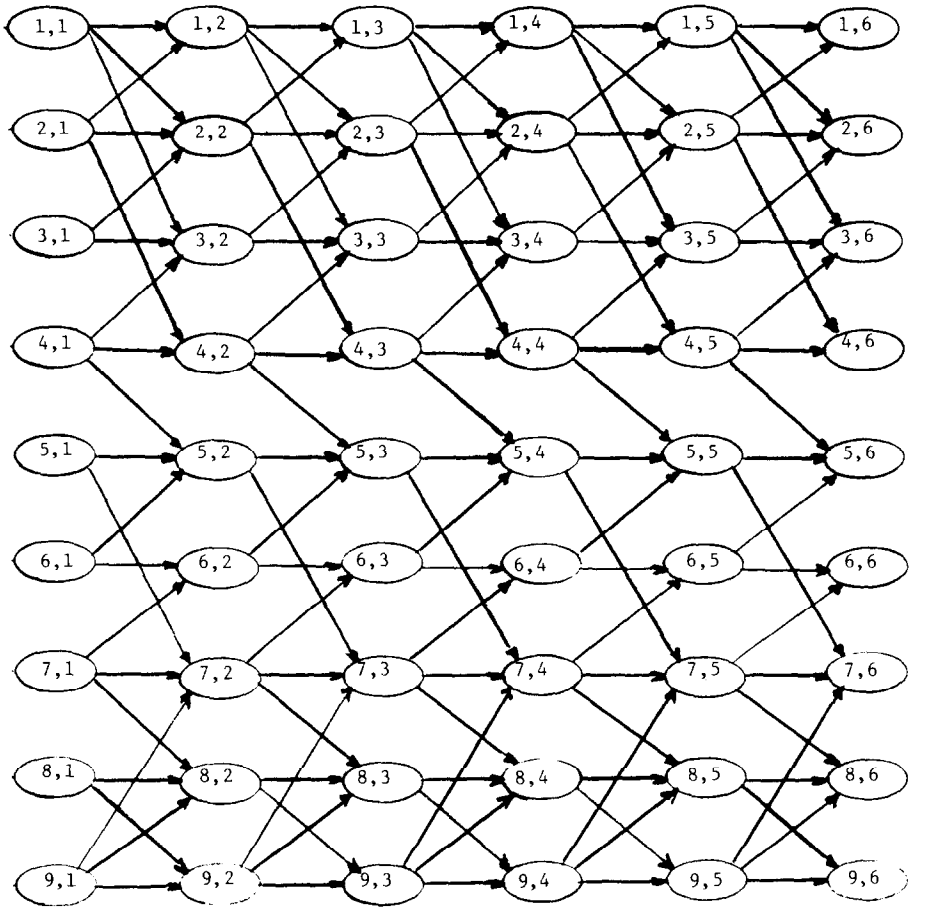


Fig. 3.17. Multi-level network for rostering problem.

roster covers several weeks. We should point out that both of these models could be quite useful when combined with procedures that treated other aspects of the problem.

References [67] and [321] give more comprehensive solutions to the rostering problem. In particular, [321] uses the first model presented in this section as a subprocedure in determining rosters. Other references which give practical experience with solving rostering problems are [341], [397] and [361]. This last reference describes a dynamic system which allows roster changes based on employee absentees.

CHAPTER 4
COMBINED ROUTING AND SCHEDULING

4.1. INTRODUCTION

Most combined routing and scheduling problems occur as applications and are characterized by task precedence and time window constraints. Task precedence relationships force the pickup activity for a task to precede the delivery activity for the task and the pickup and delivery tasks must be on the same vehicle. The difficulties due to precedence constraints are illustrated in the following example.

In Fig. 4.1, the optimal tour for the traveling salesman problem is *D*–1–2–3–4–*D* with a length of 12 hours. If the precedence relationship forces node 1 to be serviced before node 3 and node 4 to be serviced before node 2 (e.g. deliveries are required from node 1 to node 3 and from node 4 to node 2), then the optimal tour of length 13 hours is *D*–1–4–3–2–*D*. If a driver cannot work more than 12 hours a day, then two tours *D*–1–3–*D* and *D*–4–2–*D* each 9 hours in duration result and, hence, two vehicles are needed.

A second set of constraints involves the servicing of tasks within specified time windows. A time window on a service task requires that the task be serviced within the specified time interval. For example, in a delivery problem, a particular delivery may be constrained to occur between 10:00–11:30 AM. Thus, any route which involves this particular task must ensure that the delivery time fall within these time bounds. In this discussion, the time interval is a contiguous period of time and this type of window is called a *simple time window.*

The example in Fig. 4.2 shows how time windows may complicate a routing problem. Node *D*, the depot, provides service to the three points 1, 2, and 3 each requiring 1/3 of a truck-load. The vehicle cannot leave the depot until 8:00 AM and must return by 5:00 PM. The travel times (in hours) between nodes are shown in Fig. 4.2. An optimal tour of length 8 is *D*–1–2–3–*D*. If nodes 1 and 3 must be serviced between 8–12 AM and node 2 after 12 AM, then the above tour is no longer feasible and the optimal solution is to have two tours *D*–1–2–*D* and *D*–3–*D* with lengths equal to 7 and 4 hours, respectively. Hence, in this case two vehicles are required.

With no time windows, the set of tasks that may follow a particular task can be specified *a priori* and one can construct a network (directed or undirected depending upon the application) including all the tasks. With time windows, the complete set of tasks that can feasibly follow a given task cannot, in general, be specified beforehand since the exact time of service for a given task cannot be ascertained in advance. It is possible for a task to follow a given task, say task *t*, on a route if task *t* is serviced at the beginning of its time window but not in the case where task *t* is serviced at the end of the time window. Hence, with time windows, it becomes very difficult to construct the complete network of possible connections *a priori.* The works of Swersey and Ballard[639], Orloff[526], Wren and Holliday[695], Christofides *et al.*[144], and Baker[21] comprise the only literature on the routing problem with simple time windows that we are aware of.

A more complicated version of a time interval is when the time interval involves a noncontiguous period of time. Under such a requirement, a task may require service a specified number of times over a time duration of say, a week, and there may be constraints on the pattern of days for servicing these tasks. For example, there may be a separation of at least two days required between consecutive services of the task or it may be that a task can only be serviced on Monday, Tuesday, or Friday. Once the days of the week are assigned, there may be additional simple time window and task precedence constraints on the times for servicing the tasks during the days scheduled. The assignment-routing problem of Russell and Igo[573] is one example of a problem with these kinds of constraints. Having assigned the tasks to days of the week, the problem for a given day becomes a daily routing problem which may be solved by one of the routing procedures presented in Chapter 2. However, the complicated interaction of the routing and assignment decisions is a major source of difficulty in this problem and can

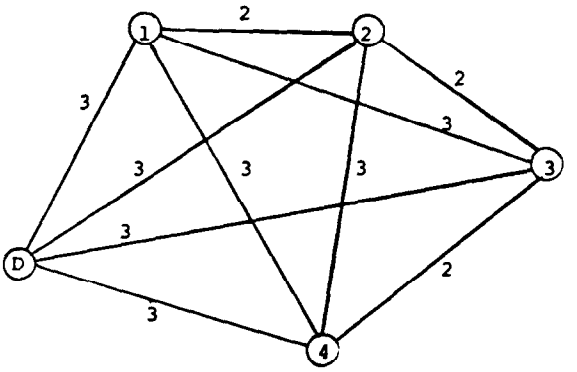


Fig. 4.1. A routing problem with precedence relations.

seriously effect the solution to the problem. To our knowledge, there has been almost no work done on solving the combined assignment-routing problem when the time windows involve noncontiguous periods of time and there exist precedence constraints on the servicing of the tasks on the days assigned.

The pickup and delivery problems considered in this chapter fall into the class of routing and scheduling problems known as "many to many". The "many to many" means that each item to be serviced can have a different origin (or pickup) location x_i and a different destination (or delivery) location y_i, although these origins and destinations need not be unique. In general, however, these problems can have "many" origins and "many" destinations. In the single depot vehicle routing problems considered in Chapter 2, all items at each pickup point are being delivered to a central depot so that all the destinations for all the items are the same. We call this problem, the "many to one" routing problem since these problems have many origins and one destination (the depot). Similarly, the "one to many" problem has one origin and many destinations. In this problem, the items are loaded on the vehicles at a central depot and delivered to the "many" customers. The "many to few" and "few to many" routing problems correspond to the multiple depot routing problems considered in Chapter 2. We should remark that in "many to many" problems, "many" refers to the pickup and delivery locations of the tasks being serviced and the trips to and from the depots or garages are additional. In "many to one", "one to many", "many to few" and "few to many" problems, the "many" refers to the servicing of the required tasks (either pickup or delivery) and the "one" (or "few") refers to the beginning or ending locations of the routes at the depot (garage) or depots (garages).

In numerous combined routing and scheduling problems, we talk about the pickup and delivery locations of the items being serviced and do not explicitly worry about the garages where the vehicles are housed. The deadhead times to go from the depots or the garages to the first stop on the routes and from the last stop on the routes to the garages or depots are generally added to the length of the routes after the routes are formed, and they are not considered part of the optimization. If the distance to and from the garage (or garages) were to

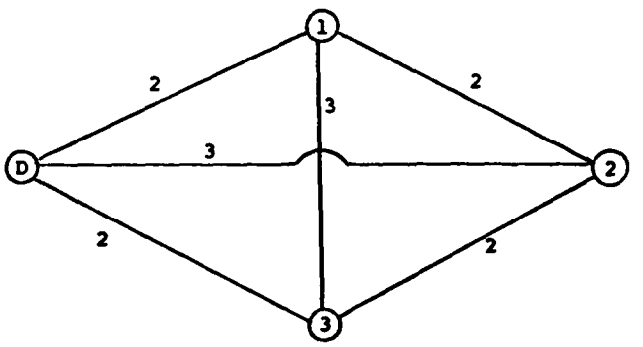


Fig. 4.2. A routing problem with time windows.

be taken into account in the optimization, then the routes might be altered somewhat. This consideration may complicate the models and affect the solution. In many cases, we have found that the total deadheading time to and from the garages is marginally small when compared to the total deadheading or travel times on the routes themselves.

In this chapter, the following routing and scheduling problems are discussed:

(1) school bus routing and scheduling
   *no mixed loads
   *mixed loads
(2) tractor-trailer routing and scheduling with full loads
(3) tractor-trailer routing and scheduling with partial loads
(4) routing and scheduling of street sweepers and household refuse collection vehicles
   *no constraints
   *parking regulation constraints
(5) airplane scheduling
(6) dial-a-ride routing and scheduling
   *without delivery time constraints
   *with delivery time constraints.

In the next section, a description of each of the above problems is presented. Most of these problems involve precedence relationships or time windows, or both. In each application, the situation and the constraints are described. In the remaining sections of this chapter, algorithms for solving these problems are presented. Practical considerations regarding implementation are brought out in this chapter and are developed further in Chapter 5. All of these problems are *NP*-hard. In most cases, the complications in these problems are such that exact algorithmic approaches based on mathematical programming formulations have not proven successful.

4.2 DESCRIPTION OF VARIOUS ROUTING AND SCHEDULING PROBLEMS

In this section, a description of a variety of routing and scheduling problems is given. In Sections 4.3–4.10, algorithms for solving these problems are presented.

4.2.1 *School bus routing and scheduling*

In the school bus routing and scheduling problem, there are a number of schools. Each school has a set of bus stops with a given number of students assigned to each stop and a fixed starting and ending time with corresponding time windows for the pickup and delivery of the students. The time window before the starting time of the school involves the allowable time interval for the delivery of students to the school in the morning and the time window after the ending time of the school is the allowable time interval associated with the pick-up of the students at the end of the school day. The primary objective of this problem is to minimize the "transportation costs" for the district. For a district with a leased fleet of vehicles, a convenient surrogate objective to transportation costs is to minimize total number of vehicles required. For a district-owned fleet of vehicles, a convenient surrogate objective is to minimize a combination of the fleet's operating cost and the number of vehicles used.

The term "transportation costs" merits some discussion. In many states, some of the student transportation costs are borne by the school district while other student transportation costs are subsidized by the state (up to 90% in New York State). However, the aid formula under which the subsidy is disbursed may involve factors other than the number of students being transported. As such, these aid formulas can further complicate the problem of minimizing the total student transportation costs for a district. For example, the formula used by New York State only subsidizes a district for those riders who live more than 1.5 miles from the school and this formula takes into account the length of the route as well as the number of students who live over 1.5 miles from the school. Since the local school districts set up the bus routes for their transportation programs, it may be more economical for a local school district to utilize more buses than absolutely necessary and have less efficient routes. As a further complication, the transportation cost for a school district is not necessarily a multiple of the number of buses used by the district. For example, in 1974, the Brentwood School District on Long Island, New York, paid about $11,000 per year for a bus and driver leased for 6 hours a

day, $14,000 per year for a bus and driver leased for 11 hours a day, and $15,000 per year for a bus and driver leased for 12 hours a day. This cost has almost doubled since then.

For many districts, the starting and ending times of the schools fall into relatively distinct time periods. For example, all high schools might start about 7:30, middle schools about 8:15 and elementary schools about 9:00. In this type of situation, the problem of minimizing the capital cost for the district becomes one of minimizing the maximum number of buses needed in any time period. If the starting and ending times of the schools do not fall into distinct time periods, then the problem of minimizing capital costs becomes the problem of minimizing the maximum number of buses needed at any point in time.

The first problem one must solve in scheduling school buses is to select the proper starting and ending times of the schools. If the school starting and ending times can be changed so as to substantially reduce the number of students requiring transportation in the peak time period, then a reduction in the number of buses needed for peak time operation can be realized. As an example, if time period 1 requires 20 buses and time period 2 calls for 40 buses, then a minimum of 40 buses will be used in the district. If, however, the starting and ending times of the schools are changed so that 32 buses are needed in time period 1 and 28 in time period 2, then the minimum number of buses required is only 32—a savings of 8 buses. Of course, the scheduling component may find a set of schedules for the buses requiring more than 32 buses. However, if capital cost minimization is a prime objective, it is better to begin the routing and scheduling of the school buses with a requirement of 32 buses rather than a requirement of 40 buses. If the starting and ending times of the schools do not fall into distinct time periods, then this problem becomes the problem of determining the appropriate starting and ending times in order to minimize the number of vehicles on the street at any point in time. This problem has been analyzed in detail by Desrosiers[180] and is discussed in Section 4.3.1.

Having determined the starting and ending times for the schools, the routing and scheduling algorithms must go to great lengths to ensure that no bus is idle in a peak time period; that is to say, every bus utilized in a time period immediately preceding or immediately following the peak time period must also service a route during the peak time period. Consequently, the routes for the buses in the off-peak time periods need not be as efficient as the routes in the peak time periods. As long as each bus utilized in an off-peak time period can be assigned a route during the peak time period, the capital requirements for the system are unchanged. Therefore, routes in the off-peak period can be shortened in duration to pick up fewer students than would ordinarily be prescribed by an "optimal seeking" routing procedure such as those described in Chapter 2. The scheduling component must assure that the routes in the off-peak fit together with the routes in the peak time periods to form bus schedules which minimize capital requirements.

In a suburban area, a bus is generally filled to capacity before the time to service the route is exhausted so that the routing problem for each school becomes a clustering problem which is relatively easy to solve. The clusters are the bus stops that form a route and it is required that the sum of all students to be picked up at the stops in the cluster does not exceed the bus capacity. In a suburban area, a bus can generally service many routes in a day (6–14 for the Brentwood, New York school district). Under these conditions, therefore, the scheduling problem is extremely important with regard to the effective utilization of the buses. In a rural area, on the other hand, a bus typically handles very few routes in a day. Each of these routes tends to be long and bus capacity may not become saturated on any of these routes. Hence, in scheduling school buses in a rural area, the routing problem becomes very important.

Most papers on the routing and scheduling of school buses focus primarily on the routing component (Bennett and Gazis[66], Angel *et al.*[8], and Tracz and Norman[653]). Newton and Thomas[508], and Bodin and Berman[82] discuss procedures for forming daily bus schedules with distinct time periods for the starting and ending time of the schools, as well as methods for routing the buses. In Swersey and Ballard[639], two integer programming formulations and a solution procedure based on linear programming are described for solving the bus scheduling problem when the starting and ending times of the schools do not fall into distinct time periods. In [639], the routes for each of the schools and the starting and ending times of the schools are assumed given.

Most papers in this area are concerned with the *single load problem*, which is the problem in

which the routes either pick up or deliver students to a single school. In rural areas, the *mixed loading problem* can occur. In the mixed loading problem, students from more than one school are picked up by a single bus, some students are delivered to one school while others remain on the bus and are transported to the next school along with additional students who are picked up in between the two schools. This problem resembles problems described in other sections of this chapter (see Sections 4.5 and 4.9). Much of the work on the mixed load routing and scheduling problem that we are aware of has been carried out by Orloff, Lockfeld, and members of the Center for Urban Analysis of Santa Clara County in California[116]. Our discussion in Section 4.3 will focus on the single load problem.

4.2.2 *Tractor-trailer routing and scheduling with full loads*

A common commercial distribution problem entails the routing and scheduling of the tractor portion of tractor trailer trucks where the trailers are assumed to have a full load. The term *full load* means that a trailer is attached to the tractor and is transported directly from a pickup point (the origin) to a delivery point (the destination). The load on a trailer has a unique and specified destination and is not to be split among different destination locations. The capacity of a tractor is one trailer. Since each trailer is transported from its origin to its destination, this problem obviously involves precedence constraints. A typical route for a tractor may be as follows (see Fig. 4.3)

> "leave domicile, pick up a trailer, deliver a trailer, deadhead, pick up a trailer, deliver a trailer, pick up a trailer, deliver a trailer, . . . , deliver a trailer, return to domicile."

The demands are specified in terms of the number of trailer trips between origin/destination pairs. Given this demand data, one may address the following two decision problems:

Problem 1. Minimize the total distribution cost for handling *all* origin–destination demands.

Problem 2. Determine the optimal fleet size required to service a subset of the origin–destination demands given that the remaining demand is to be serviced by common carrier.

Thus, Problem 2 involves an additional decision as to which subset of required demands is to be serviced. This subset is chosen using a profit-maximizing criterion with the unprofitable demand requirements being turned over to a common carrier. The paper by Ball *et al.*[38] addresses this problem in the context of the distribution activities of a large chemical firm. It is

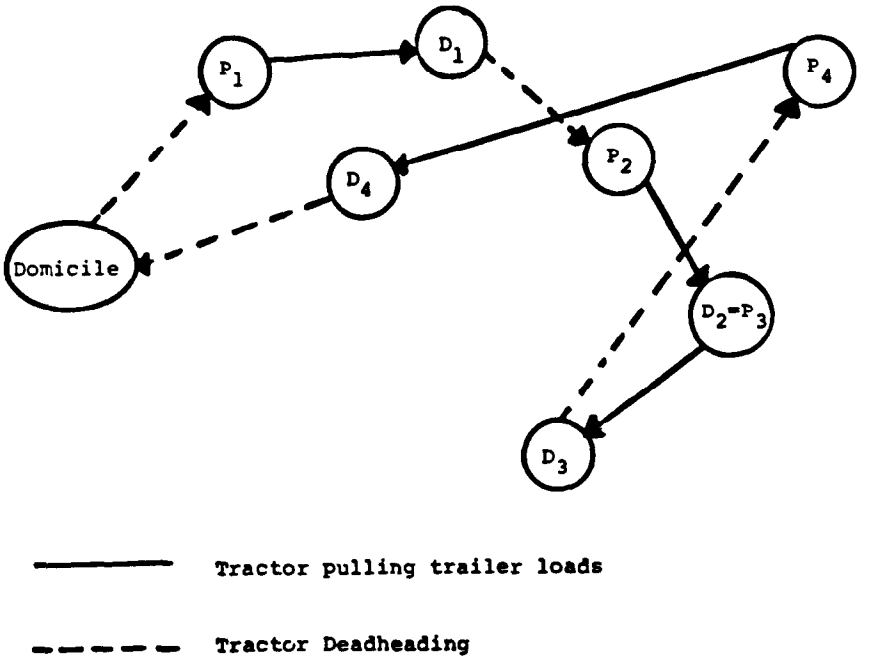


Fig. 4.3. Example of route for tractor trailer with full load problem.

also possible for a common carrier to have this problem in the sense that a common carrier may turn over undesirable or unprofitable trailer trips to a second common carrier (see Goren[312]).

In the application studied by Ball *et al.*[38], severely restrictive window constraints were not present. Nevertheless, the following timing constraints had to be observed: (a) Certain pickups and deliveries had to be carried out during specific hours of the day; (b) Each route served by two-driver crews could not exceed 120 hours in total driving time and each route served by one driver could not exceed 60 hours in total driving time; (c) multiple trips between the same origin-destination pair had to be spread out over the week. For example, if there were 7 trips between points A and B over the course of a week, constraint (c) requires that not all of these be scheduled for the same day, but rather that they be approximately evenly distributed over the days of the week (for example, two trips on Monday and on Thursday and one trip on Tuesday, Wednesday, and Friday). In Section 4.4, three algorithms for solving this problem are given.

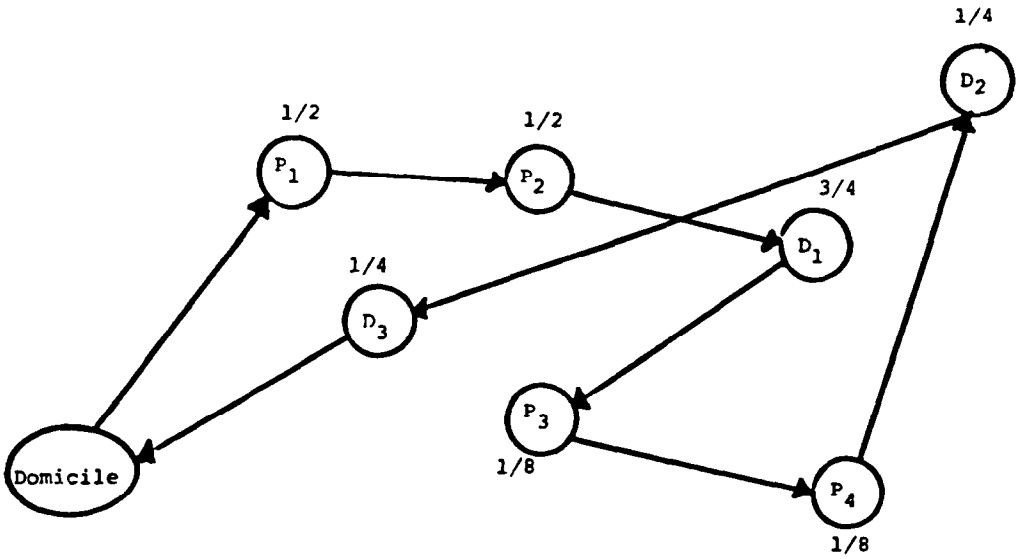
4.2.3 *Tractor trailer routing and scheduling with partial loads*

This problem is similar to the full load problem of Section 4.2.2 except that the demand for each origin–destination pair need not be a full trailer load. Consequently, the load on a trailer may be split among different origins and destinations. In this case, a typical route may look like the following:

> "leave domicile, pick up part of a load, deadhead, pick up part of a load, deadhead, deliver part of a load, pick up part of a load, deadhead,..., deliver part of a load, return to domicile."

Such a route is illustrated in Fig. 4.4.

The constraints on this problem are the precedence relationships for each origin-destination pair, time windows (most pickup and delivery locations have specific hours of operation), and crew workrules such as maximum number of hours/week that a crew can drive and break-time restrictions. Although many of the crew workrule constraints were considered explicitly in the algorithm that we devised, the firm assumed responsibility for forming the final set of crew



Note: The numbers outside of the nodes indicate the % of trailer either picked up or delivered at the node. Thus, at P_2 we pick up 1/2 a trailer load.

Fig. 4.4. Example of route for the tractor trailer with partial route problem.

schedules associated with the generated vehicle routes. The crew workrules incorporated by the algorithm into vehicle schedules were meant to facilitate this task for the firm.

In Assad *et al.*[18], this tractor trailer routing problem is discussed in greater detail. Problems of a similar structure can occur for the routing and scheduling of messenger services, package delivery within a local area, deliveries between banks, etc. This problem is very general and is deserving of further analysis.

It is interesting to draw an analogy between this problem and the school bus routing and scheduling problem with mixed loads (see Section 4.2.1). For the school delivery problem (that is, students being delivered to the schools), the number of destination locations (number of schools in the district) is very small compared with the number of origin locations (number of bus stops where students are to be picked up). The time window on the pickup time for a student is an interval [start time of the school—T, start time of the school] where T is a constant like 15 minutes. The loads at a bus stop are equivalent to the partial loads at the pickup or delivery points. The capacity of the vehicle is equal to the number of students that can be carried on it, and there are virtually no driver workrule constraints since school bus scheduling has virtually no mid-day required service. To our knowledge, nobody has utilized an algorithm such as described in [18] for solving the school bus routing and scheduling problem with mixed loads.

4.2.4 *Street sweeper and household refuse collection routing and scheduling*

The problems of scheduling street sweepers and household refuse collection vehicles are applications of the rural postman problem (see Chapter 2). For both of these problems, a set of street segments is specified as needing service. The basic problem is to cover every street segment by a vehicle in such a way that the minimum number of vehicles is used. A highly correlated objective is to minimize the total deadhead time of the vehicles for either a fixed or variable fleet size. There are no precedence relationships on the entities to be serviced. The time windows for the street sweeper routing and scheduling problem correspond to the parking regulations on the streets. In most cases, there are no time windows on the demands for the household refuse collection problem. If the network of streets to be serviced in a time period is connected, then these problems reduce to the Chinese postman problem of Chapter 2 which is very simple to solve.

The basic approaches for solving these problems are variants of the "route first-cluster second" or "cluster first-route second" procedures described in Chapter 2 as applied to the Chinese postman problem. The "route first" approach for routing and scheduling problems will also be encountered for the tractor trailer routing and scheduling problem with full loads (see Section 4.4). The algorithm for street sweepers will utilize the procedure for solving the directed rural postman problem whereas the algorithm for the tractor trailer routing and scheduling problem with full loads will use the algorithm for the undirected rural postman problem.

The key confounding constraint for street sweeper routing and scheduling is the constraint that sweeping can only take place when parking is banned from the street since only during this time can a sweeper come close enough to the curb to be effective. In New York City, for example, there are about seventy different parking regulations ranging from "no parking during peak traffic hours" to "parking on alternate sides of the street". Also, many New York City street segments† have more than one parking code. For example, a street segment can have the following two parking regulations: no parking 8–9 AM on part of the segment and no parking any time on the remainder of the segment. The "no parking 8–9" is to restrict parking during the AM rush hour while "no parking any time" is to restrict parking in front of some special building (for example, a school or church). The only time that the entire segment can be swept is 8–9 AM since we can only sweep the entire segment when parking is banned from the street. Therefore, in the network representation for this region, we set the parking regulation for this street segment to "no parking 8:00–9:00 AM".

A basic problem in household refuse collection is the statistical estimation of demand for

†A street segment is the street between two intersections. A street segment will become a branch in the corresponding network constructed for this problem.

service on a street. Based on the characteristics of the street (type of housing, number of houses, etc.) the only estimation procedure that we know of that works is based on a count of the number of houses on the street (Schuster and Schur[587]). As long as the neighborhood is homogeneous (like single family housing), this procedure is fine. However, in locations with many different types of buildings (such as Manhattan), the procedure breaks down. To our knowledge, no successful procedure for estimating demand other than the one cited above has been developed. The discussion in Section 4.6 concentrates on the street sweeper routing and scheduling problem. The household refuse collection routing and scheduling problem is quite simple once the demand on each street segment is known.

4.2.5 *Airplane scheduling*†

Airlines coordinate the scheduling of their airplanes with the process of generating their timetables. The generation of a timetable takes into account such factors as the expected number of passengers traveling between cities, frequency of service desired, nonstop versus multiple stop service, etc. The scheduling of the airplanes also gives an estimate of the capital and operating costs attributable to that timetable. This process takes into account the problems of generating pairings and bid lines for the crews (see Chapter 3) in that the airlines might change their timetables and vehicle schedules if a savings in crew costs can be realized. However, the operating costs attributable to running and maintaining the airplanes are significantly greater than the crew operating costs. In spite of this, most scheduling of airplanes for commercial airlines is carried out on a manual basis or in an interactive computing mode with little algorithmic sophistication.

Federal Express Corporation has the only computerized procedure that we are aware of in the area of plane scheduling. The timetable at Federal Express is changed every four to five weeks to take into account changes in demand, new cities, seasonality factors, etc. Input to the process is a matrix which gives the estimated package count between each pair of cities. The package count at a city is converted into a percent of capacity of the plane. Every city is to be visited once (we are talking about Federal Express Corporation's Priority 1 packages only). In general, each package leaves its origin city on a plane, travels to Memphis, Tennessee where it get sorted by its destination location and is then delivered to its destination location on a second plane. Each city is regarded as both an origin and a destination and each city can have a one-sided time window for pickups and deliveries. One time window specifies the earliest time that a plane can leave the city with its pickups. The second time window gives the latest time that a plane can get to the city with its deliveries. There are also constraints on the total length of an incoming and outgoing route and the capacity of a plane.

Two routing problems are solved. One routing problem is for the pickups and the second routing problem is for the deliveries. A sample set of delivery routes is given in Fig. 4.5. These routes look incomplete in the sense of the routes displayed in Fig. 2.1 in that the routes do not return to Memphis, the depot. The reason for this is that the planes layover at the end of the outgoing or delivery routes and start their pickups from the layover cities. The routes, when formed, give the time for delivery and pickup; thus, the timetable is generated as the routes are created. An analogous algorithm is available for the pickup problem. In Section 4.7, a description of the algorithm for solving the delivery problem is given.

4.2.6 *Dial-a-ride routing and scheduling problems*

In recent years, the area of dial-a-ride routing and scheduling has received considerable attention. In the dial-a-ride problem, customers call a dispatcher or scheduler requesting service. Each customer specifies a distinct pickup and delivery point and, perhaps, a desired time for pickup or delivery. If all customers demand immediate service, then routing and scheduling is done in real time and the problem is referred to as the *dynamic* or *real time dial-a-ride problem.* If all customers call in advance, so that a complete database of customer demand is known before any routing or scheduling is carried out, then this problem is referred to as the *subscriber or static dial-a-ride problem.* Both the dynamic and static dial-a-ride

†Virtually all that is discussed in this section is due to Joe Hinson of Federal Express Corporation. We are indebted to him for his assistance in providing us with the background for this section.

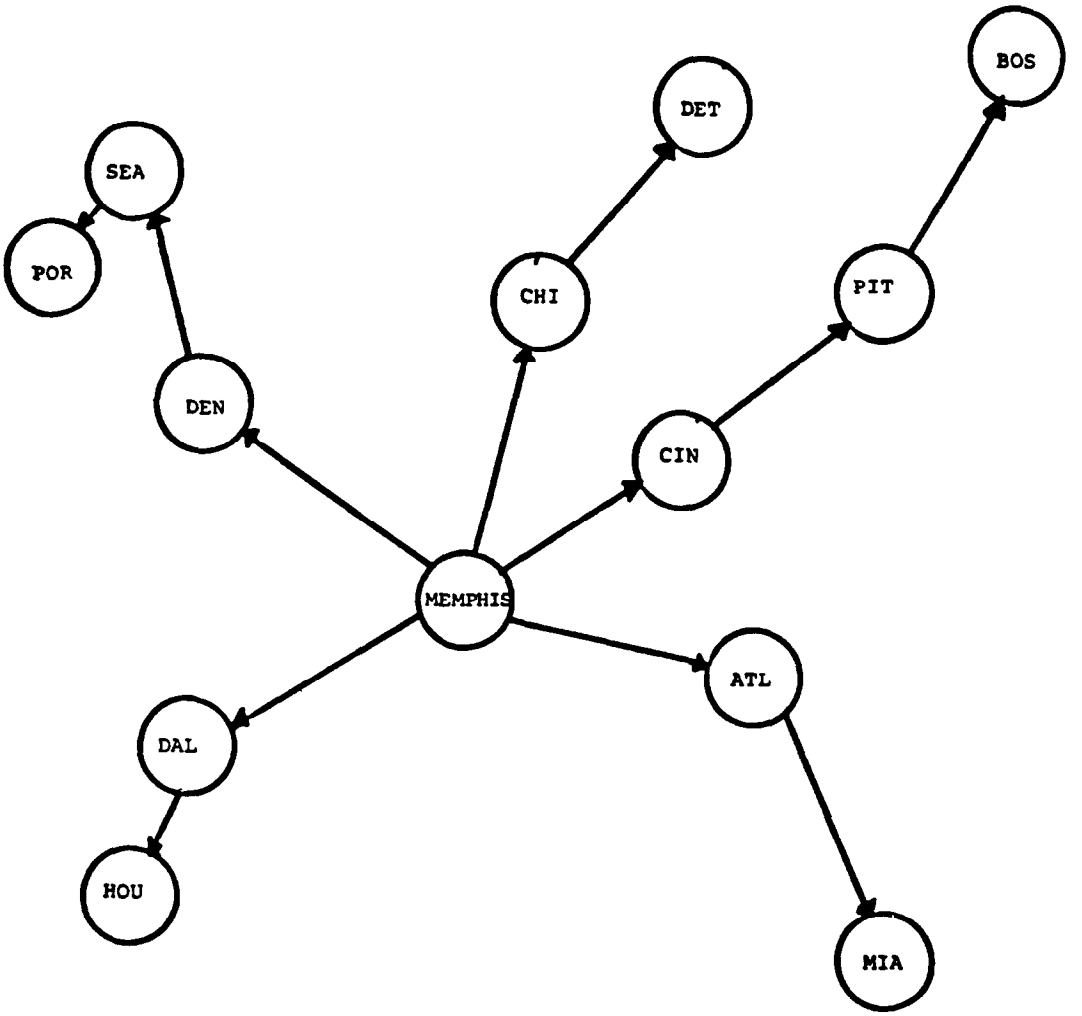


Fig. 4.5. Delivery routes for airplanes for an airplane routing and scheduling problem. (Memphis is the domicile for the planes.)

problems have precedence relationships since a customer must be picked up before he is delivered. In some situations, a desired time of pickup or delivery is specified in advance and the "other service" (either delivery or pickup) must be carried out within a given number of minutes from either the desired or the actual time of delivery or pickup. This situation introduces in a certain sense, a two-sided time window on the "other service".

A two-sided window introduces feasibility problems since it may be impossible (or very difficult) to handle the "other service" within its two-sided window. In [94] and [591], the two-sided window on the "other service" is disregarded and if a desired delivery time is specified, the delivery can take place any time before the desired delivery time and the pickup can be made at any time preceding the delivery time. Similarly, if a desired pickup time is stated, the pickup can be made at any time following the desired pickup time and the delivery can be made at any time following the pickup time. In essence, therefore, two-sided windows are replaced by a one-sided window on the activity with a desired pickup or delivery time and no time window is placed on the other activity. With one-sided windows, the potential exists, for customers to be on a vehicle for an excessive amount of time. However, experimentation with the algorithms (see [94], [591]) has shown that in this case most customers are still delivered within a reasonable amount of time.

Dial-a-ride problems and their extensions occur in many applications such as shared cab rides, package delivery, bank deliveries, etc. Perhaps the application that has received the most attention is the routing and scheduling of vehicles for social services such as for the elderly and

handicapped. In [103], Bunt *et al.*, tabulate some of the computerized para-transit operations in the United States, Canada and Europe. It is interesting to note that only a few of these communities actually route and schedule their vehicles by computer; most of the computerized operations in these communities are concerned with reservation and management information systems.

Despite the lack of actual implementation of computerized scheduling in this area, there has been a considerable amount of research into the routing and scheduling of dial-a-ride vehicles. Research into this area has been well-received over the past few years. Stein[616], Psaraftis[549], and Sexton[590], the winners of the 1977, 1978, and 1979 Dissertation Prize in Transportation awarded by the Transportation Science Subsection of the Operations Research Society of America, have all studied this problem area.

Our discussion of dial-a-ride problems is broken down into three parts. In Section 4.8, the dial-a-ride routing problem without time windows is disucssed and in Section 4.9 the static dial-a-ride routing and scheduling problem with time windows is addressed. In Section 4.10, procedures for solving the dynamic dial-a-ride routing and scheduling problem with time windows is described.

4.3. AN ALGORITHM FOR THE SCHOOL BUS ROUTING AND SCHEDULING PROBLEM

In this section, we describe a composite algorithm for solving the school bus routing and scheduling problem with single loads. This algorithm is broken down into the following three parts: (i) a procedure for selecting the starting and ending times of the schools (see Section 4.3.1), (ii) a procedure for forming partial vehicle routes (see Section 4.3.2), (iii) a procedure for forming daily schedules for the buses (see Section 4.3.3). Each of these procedures is described in more detail below. In this algorithm, we will assume that we can easily determine the student load at each bus stop. The mini-stop procedure first described in Bodin and Berman[82] and extended slightly by Desrosiers *et al.*[180] still represents the best approach for automatically determining the student loads at each of the stops for a school knowing the bus stops for the schools. The school bus scheduling problem is about the most data intensive routing and scheduling problem that we have encountered. To have a procedure to quickly determine student loads is a necessity, especially in an interactive computing environment (see Section 4.3.4).

4.3.1 *A procedure for selecting the starting and ending times of the schools*

As noted in Section 4.2.1, a key problem in minimizing transportation costs is to reduce the maximum number of buses needed at any time of the day. The procedure by Desrosiers *et al.*[180] can be utilized to solve this problem. In Desrosiers' approach, the starting and ending times of the schools are selected so that the maximum number of routes needed in any time interval is minimized. If the time of day is broken down into ten minute time intervals and a route requires 25 minutes, then it is assumed that the route will need three time intervals to service it. Thus, if the school is to start at 9:00 AM and the route needs three time intervals, then it is assumed that a bus is needed in the 8:30–8:40 time interval, the 8:40–8:50 time interval and the 8:50–9:00 time interval; that is to say, a route will exist from 8:30–9:00. If the school can start at either 8:40, 8:50 or 9:00, then there are three possible situations that can occur: (1) a bus is needed in time intervals 8:10–8:20, 8:20–8:30 and 8:30–8:40 if the school starting time is 8:40 (i.e. a route will exist from 8:10–8:40); (2) a bus is needed in time intervals 8:20–8:30, 8:30–8:40 and 8:40–8:50 if the school starting time is 8:50 (i.e. a route will exist from 8:20–8:50); (3) a bus is needed in time intervals 8:30–8:40, 8:40–8:50 and 8:50–9:00 if the school starting time is 9:00 (i.e. a route will exist from 8:30–9:00). Similar enumerations are carried out for each incoming and outgoing route. Moreover, the enumeration of the incoming and outgoing routes are combined to ensure that the length of the school day falls between feasible limits (say 6–6$\frac{1}{2}$ hours). An enumeration of a combined incoming and outgoing route is called a *possible run combination.* All possible run combinations are then enumerated for all the routes and a min max 0–1 programming problem is solved in order to determine the starting and ending times of the schools which minimize the maximum number of buses needed in any time interval. The details of this analysis are presented in [180].

This problem can also be solved approximately manually. For each school, a rectangle is cut

out. This rectangle has a length equal to the maximum number of time intervals to be covered by a route to service the school and a width equal to the number of routes for the school. These rectangles are then maneuvered on a big piece of paper in order to determine the starting and ending times for the schools which minimize the peak number of buses needed in any time period. From our experiences and the experiences of others, most school districts have less than 50 schools, and the allowable time intervals for the starting and ending times of the schools are quite small. Hence, this problem is quite tractable to solve manually. We have used this approach in studying the impact of regional bus transportation and have been able to show that a four school district region would achieve a savings of over 20% from a situation where each district had their own buses. Moreover, this manual approach corresponds to a certain extent to the run cutting approaches for crew scheduling for mass transit systems described in Chapter 3.

4.3.2 *The routing component*

In most algorithms, the routing component is a variant of the single depot routing problems described in Chapter 2. In [508], Newton and Thomas indicated that the "route first—cluster second" approaches worked better than other approaches for the types of problems generally encountered in the school bus routing and scheduling area for fairly densely populated suburban locations. The basic approach is as follows:

(i) Solve a traveling salesman problem using one of the composite heuristic procedures given in Chapter 2. An optimal traveling salesman tour is not needed since this tour is going to be broken up into feasible vehicle routes.

(ii) Break the tour up into feasible partial vehicle routes where the vehicle routes are formed in the manner described in Section 4.2.1.

In many cases, the routing problem is no more complicated than a clustering problem since the vehicle generally gets filled before the time constraints are violated and most students near the schools are not bused. The time constraints are expressed as the maximum time allowed to cover a route and deliver the students to the school, since a route for a school generally originates at another school. The routing component need only construct partial routes in a form displayed in Fig. 4.6. These partial routes have the form: bus stop j_m, bus stop $j_{m-1}, \ldots,$ bus stop j_1, school i. The scheduling component then determines the school k from which the bus route originates. Discussions on this approach are given in [82] and [180].

Furthermore in [180], a discussion of the routing problem with transshipment points is presented. In this problem, students are bused to specified transshipment points where they are released. A second bus then takes them to their schools. This problem is encountered in rural areas where each bus travels great distances and picks up few students. The routing algorithm utilized in this problem is a variant of the one used to solve the rural postman problem.

A second way to handle students in rural areas is by busing students from more than one school on the same bus (the mixed load problem). In the single load problem, a bus is emptied at a school before it can take on other students. In the mixed load problem, students are continually picked up and delivered so that the vehicle may never be emptied until the end of the morning or evening deliveries. In this case, the routing and scheduling problems become

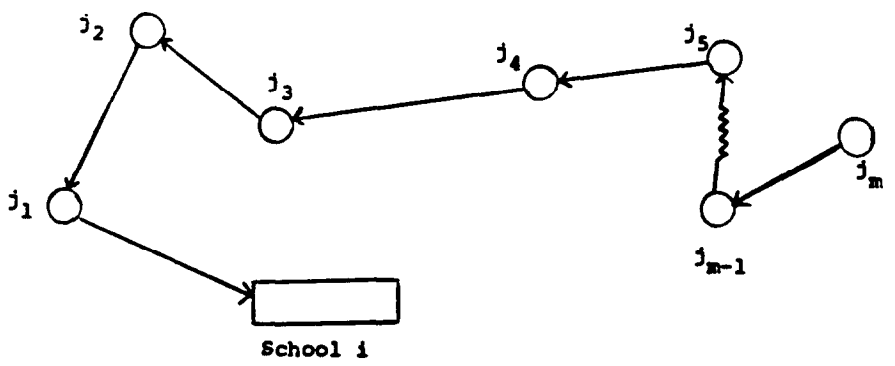


Fig. 4.6. Partial route for a school.

more complex combined routing and scheduling problems. We are not aware of any work on this problem outside of that reported to us by the Center for Urban Analysis of Santa Clara County in California[116]. We believe this problem has strong similarities to the multi-vehicle static dial-a-ride problem (Section 4.8) and the tractor trailer routing and scheduling problem with partial loads (Section 4.5).

4.3.3 *The scheduling component*

The scheduling component organizes the partial routes for each of the schools into daily schedules for the buses. The starting and ending times for each of the schools and the maximum allowable travel time for the students at each school have to be specified (this maximum may vary from school to school). As the student loads on each of the partial routes have already been satisfied in the routing component, they no longer serve as constraints in the scheduling component. The objective used in the scheduling component is to minimize the total travel time of all the buses in the district. If the starting and ending times of the schools fall into distinct time periods, the problem reduces to a sequence of transportation problems[82]. These transportation problems connect the partial bus schedules formed through the $t-1$st time period with the partial routes which are formed in the routing component that have to be covered in the tth time period. The cost c_{ij} for joining partial route j onto partial vehicle schedule i is either the total time to cover route j, the total deadheading time on route j or some other similar function.

A minor modification to the transportation algorithm approach presented in [82] is given in [180] in order to solve the scheduling problem when the starting and ending times of the schools do not fall into distinct time periods. A second approach for solving this problem with non-distinct time periods is presented by Swersey and Ballard[639]. As with the other approaches, their approach assumes that all the routes or partial routes for the district have been specified. They propose two mathematical models for determining the minimum number of buses needed to service all the routes. One model is a mixed integer program whereas the other model is a 0–1 integer program. The formulation of the 0–1 integer program is now given.

Suppose that route i as found in the routing component can take place between $E_{s(i)}$ and $L_{s(i)}$ and let $F^p_{s(i)}$ represent the only times when the deliveries of students can be made to the school where

$$F^p_{s(i)} = E_{s(i)} + \frac{(L_{s(i)} - E_{s(i)})(p-1)}{k-1} \quad \begin{matrix} p = 1, 2, \ldots, k \\ i = 1, 2, \ldots, t, \end{matrix} \tag{4.1}$$

k denotes the number of possible partitions of the time interval and t is the number of routes.

Define

$$x^{pq}_{ij} = \begin{cases} 1 & \text{if a bus assigned to route } i \text{ at } F^p_{s(i)} \\ & \text{is then assigned to route } j \text{ at arrival} \\ & \text{time } F^q_{s(j)} \\ 0 & \text{otherwise,} \end{cases}$$

$$y^q_j = \begin{cases} 1 & \text{if the arrival time of route } j \text{ is set} \\ & \text{to } F^q_{s(j)} \\ 0 & \text{otherwise.} \end{cases}$$

Let route 0 denote the depot. Then the formulation of this problem as given in [639] is the following:

$$\text{Minimize} \sum_{j,p,q} x^{pq}_{0j} \tag{4.2}$$

$$\text{subject to} \quad \sum_{i,p} x^{pq}_{ij} = y^q_j \quad j = 1, \ldots, r, \quad q = 1, \ldots, k \tag{4.3}$$

$$\sum_{j,q} x_{ij}^{pq} \leq y_i^p \quad i = 1, \ldots, r, \quad p = 1, \ldots, k \tag{4.4}$$

$$\sum_{q} y_j^q = 1 \quad j = 1, \ldots, r \tag{4.5}$$

$$x_{ij}^{pq} \in \{0, 1\}, \quad \text{all feasible } i, j, p, q. \tag{4.6}$$

Swersey explains that constraints (4.3) and (4.5) specify that each route is covered by exactly one bus, either from the depot or another route. Constraint (4.4) allows that bus to cover a later, feasible route. It is not necessary to specify in (4.6) that the y_j^q's are integers because this is assured by the integrality of the x_{ij}^{pq}'s. Note that if we eliminate the time windows, by setting one of the y_j^q variables equal to 1 for each route j, the problem reduces to a network flow problem which is equivalent to the tanker scheduling problem formulated by Dantzig and Fulkerson[164].

In [639], procedures for eliminating some of the variables in the problem and for solving this problem are given. The procedure for solving (4.2) to (4.6) begins by solving the linear programming relaxation to the problem. If the linear program does not terminate with integer x's then the objective has a value Z^0 which is not integer. The constraint

$$\sum_{j,p,q} x_{0j}^{pq} = \lceil Z^0 \rceil \tag{4.7}$$

is then added to the linear program (4.2) to (4.6) where $\lceil Z^0 \rceil$ is the smallest integer greater than or equal to Z^0. This new linear program is then solved. In over 75% of the cases, this process gave an integer solution and in the other cases, it was easy to manually find an integer solution requiring exactly $\lceil Z^0 \rceil$ buses.

This approach relies on a discretization of the time window surrounding each school's delivery and pickup times. As each time interval is discretized further, Swersey reports more trouble in finding a completely integer solution. This approach, however, may have applicability to routing and scheduling problems with both precedence and time window constraints such as the multi-vehicle subscriber dial-a-ride problem. Whether a formulation based on this idea can lead to a practical and computationally tractable algorithm for these other problems is a question for future research.

If each school would accept arriving students at any time before its starting time (that is, have a one-sided time window), then the scheduling problem for the delivery of the students to the schools would become one large minimum cost flow problem. Each node in the minimum cost flow problem would represent a partial vehicle schedule for a school. An arc in the minimum cost flow problem would be drawn from node j to node k if a partial vehicle schedule serving node j could also service node k (implying that the school associated with node k has a later start time than the school associated with node j). The cost of this arc would be set to the travel time required to cover the bus stops represented by node k, that is to say, the minimal travel time needed to leave school j, to cover all the bus stops on the route represented by node k, and to travel from the last bus stop to the school represented by node k. These arc costs are found by solving a Hamiltonian path problem (see Chapter 2).

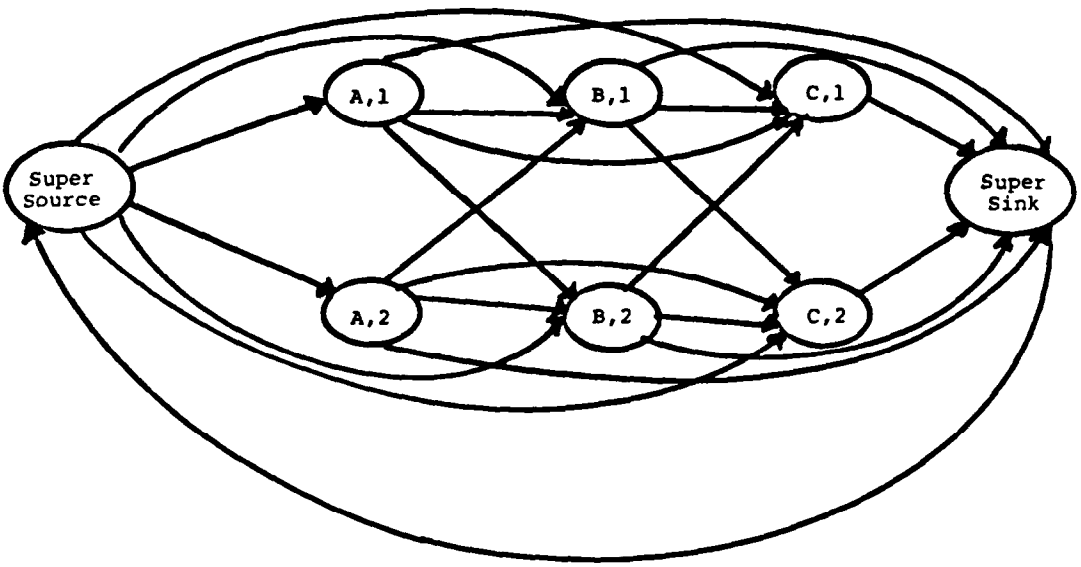
Each node in this minimum cost flow problem has a lower and upper bound on flow of 1 and a cost of 0. Each node is connected to a supersource and supersink by an arc. Each of these arcs has a lower bound on flow of 0, an upper bound on flow of 1 and a cost/unit flow of 0. The supersink is connected to the supersource by an arc. This arc has a lower bound and upper bound on flow equal to the minimum number of vehicles required if this minimum is known. If the minimum number is not known, then the lower bound on flow on this arc can be set to 0, upper bound on flow set to ∞, and cost per unit flow set to M, a large number. The solution to the minimum cost flow problem finds the minimum travel time solution for a fixed number of vehicles. Each path in the solution of the minimum cost flow problem represents a vehicle route and a vehicle schedule is then found. These paths ensure that no student is delivered to his school before the starting time. A similar problem would be defined for the delivery of students

to their homes after school closing each day. This formulation is quite similar to the RUCUS formulation for vehicle scheduling for mass transit systems (see Chapter 3).

The above-mentioned formulation disregards the time window surrounding the delivery time to a school so that students could be delivered to their school before the school's permitted delivery time. Scheduling by means of a sequence of assignment problems of the Swersey procedure type guarantees that all time windows are satisfied; however, Swersey's approach is not as efficient computationally.

The following example illustrates the difference between the school bus scheduling problem with one-sided and two-sided time windows on the delivery time to the schools. Suppose that schools *A*, *B* and *C* have two bus routes needed to service all their students. School *A* starts at 7:30 AM, school *B* starts at 8:15 AM and school *C* starts at 9:00 AM. Each route requires 45 minutes to pick up all the students and it takes 20 minutes to get from any school to the beginning stop of any route. If the time windows on the delivery times to schools *A*, *B* and *C* are $[-\infty, 7:30]$, $[-\infty, 8:15]$ and $[-\infty, 9:00]$ respectively, then 2 buses can service all six routes. Each bus chedule would be as follows:

Arrive at first stop to School *A*	6:05
Deliver student to School *A* & leave	6:50
Arrive at first stop for School *B*	7:10
Deliver students to School *B* & leave	7:55
Arrive at first stop on route to School *C*	8:15
Deliver students to School *C*	9:00



Note: Branch from supersink to supersource has lower and upper bounds on flow equal to 2 and cost/unit flow of 0.

All nodes (A,1), (B,1), (C,1), (A,2), (B,2), and (C,2) have a lower bound and upper bound on flow of 1 and cost/unit flow of 0.

All branches from supersource to node or node to supersink have a lower bound on flow of 0, upper bound on flow of 1 and cost/unit flow of 0.

All other branches have a lower bound on flow of 0, upper bound on flow of 1 and cost/unit flow of 0.

Fig. 4.7. Example of network for 3 period scheduling problem for the school bus routing and scheduling problem.

The network for this case is displayed in Fig. 4.7. The lower bound on the flow from the supersink to the supersource is 2 units.

In this example, a student would be delivered to school A 40 minutes before the starting time for the school. If waiting time before school were restricted to 15 minutes, then school A would have a two-sided window of (7:15, 7:30), school B would have a two-sided window of (8:00, 8:15), and school C would have a two-sided window of (8:45, 9:00). In this case, we would need four buses to service all the routes; two buses would service the routes for school A and C while two other buses would service the routes for school B. In this case, all students would be delivered at about the starting time of the schools.

4.3.4 *Interactive school bus routing and scheduling*

The school bus routing and scheduling problem can be solved by an interactive optimization approach. All three of the subproblems—the selection of the proper starting and ending times of the schools, the routing problem for each school, and the scheduling problem—are generally quite small from an algorithmic standpoint and, hence, are amenable to repetitive solutions. We believe that an effective way to solve school bus routing and scheduling problems would be to embed the above procedures within an interactive computing environment. In Fig. 4.8, a

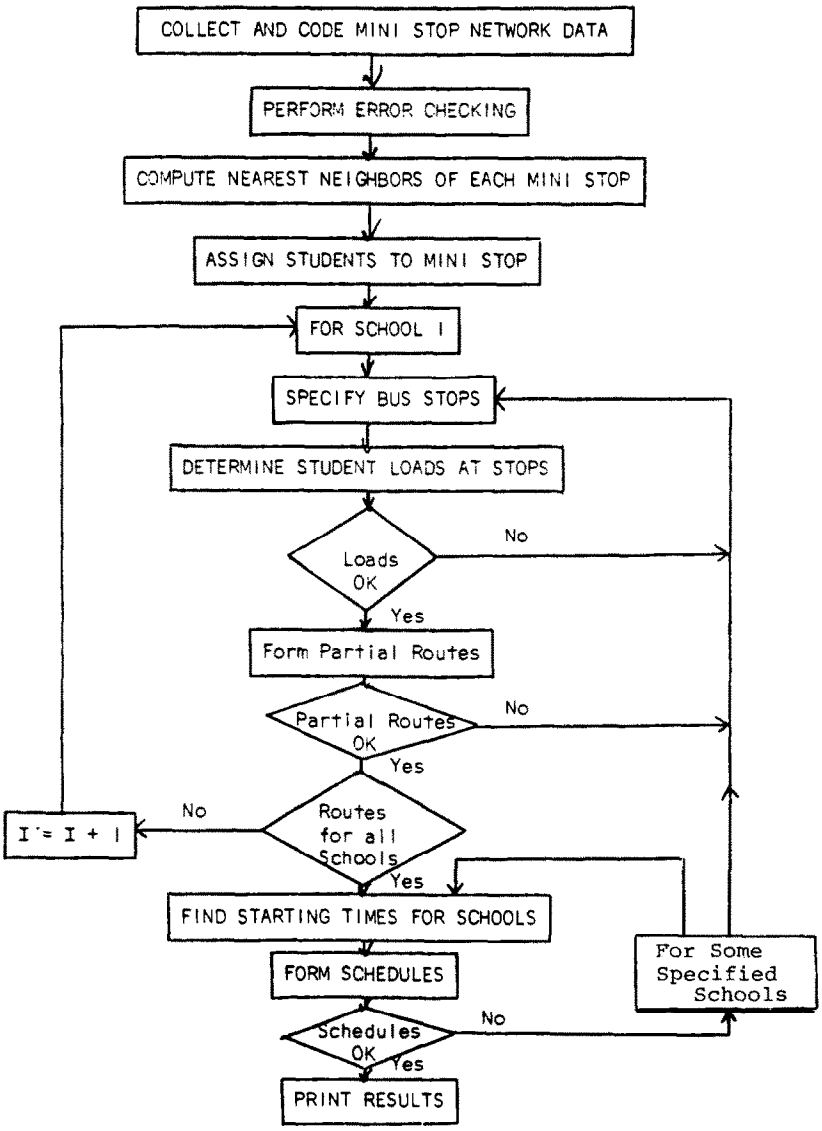


Fig. 4.8. Flow chart for school bus routing and scheduling procedure.

proposed flow chart for an interactive optimization procedure for the single load problem is presented. Although the use of an interactive approach was noted in [82] for the single load problem and in [116] for the mixed load problem, we know of no portable software which has been developed for solving this problem which takes into account all of the features cited in this section.

4.4. ALGORITHMS FOR SOLVING THE TRACTOR TRAILER ROUTING AND SCHEDULING PROBLEMS WITH FULL LOADS

Three solution procedures for solving the tractor-trailer problem with full loads are detailed in [38] and these procedures are outlined below. The first two procedures are variants of the "route-first" approach presented in Chapter 2. The demands are expressed in terms of the number of tractor trailer trips/week required between pairs of cities.

A large route is generated for a single vehicle with unlimited capacity which covers all the required tractor trailer trips. This route is then partitioned into smaller routes which can be feasibly handled by a vehicle. Feasibility means that all workrules and time windows are satisfied. These two procedures are best suited for the first decision problem mentioned in Section 4.2.2, that is, the problem where all demands are to be serviced by the fleet. For the second decision problem, where some of the demands are to be handled by the fleet and other demands are to be assigned to a common carrier, a *greedy* insertion scheme was developed. The results obtained by this insertion approach are described in [38] and were utilized by a major chemical firm in deciding on the size of its leased fleet. The precedence relations in this problem are not needed since a delivery of a trailer immediately follows the pickup of the trailer.

4.4.1 "*Route first*" *algorithms*

Both "route first" procedures consider a related single vehicle routing problem. The first step in a route first procedure is to find a single minimum cost route that covers all demand. The first of these procedures requires the solution of a directed Chinese postman problem. Each city becomes a node in the network and the number of arcs between each pair of cities is equal to the number of tractor trailer loads between these pair of cities over the week. The cost between any pair of cities is equal to the deadhead time between the cities. In the second procedure, each origin/destination demand is viewed as a node and each arc in the network has an associated deadheading cost. In this case, we find a minimal length traveling salesman problem which covers all of the nodes using the heuristics described in Chapter 2. Both procedures are practical since the directed Chinese postman problem can be solved exactly as an equivalent transportation problem and the TSP can be solved near-optimally. In the second step, this giant tour is partitioned into feasible tractor trailer routes. In this partitioning, the multiple trips between the same origin–destination pair are spread out over the week, the time windows on pickup and deliveries are satisfied, and no crew exceeds its maximum total driving time.

4.4.2 *The greedy insertion procedure*

In the greedy sequential insertion approach, a route is constructed in an iterative fashion by adding demand arcs representing origin–destination pairs to the route one pair at a time. The procedure uses the ratio of marginal increase in savings divided by marginal increase in route time, $\Delta S/\Delta T$, as a criterion for adding a demand arc. This "bang-for-buck" ratio is analogous to a ratio used in the well-known greedy algorithm for the knapsack problem [577]. For a leased vehicle to service a demand arc (i, j), let

t_{ij} = load time at i + travel time from i to j + unload time at j,

c_{ij} = crew wait cost at i + vehicle and crew travel cost + crew wait cost at j.

For a common carrier servicing a demand arc (i, j), let

r_{ij} = the tariff rate.

For a leased vehicle traversing a deadhead arc (i, j), let

s_{ij} = travel time, and

b_{ij} = vehicle and crew travel cost.

Then, to insert (i, j) between (h, k) we have

$$\Delta S(i, j) = (r_{ij} - c_{ij}) - b_{hi} - b_{jk} + b_{hk} \tag{4.8}$$

$$\Delta T(i, j) = t_{ij} + s_{hi} + s_{jk} - s_{hk}. \tag{4.9}$$

The demand arc (u, l) is added to the route if

$$\Delta = \frac{\Delta S(u, l)}{\Delta T(u, l)} = \text{MAX}\left\{\frac{\Delta S(i, j)}{\Delta T(i, j)}, 0\right\} \tag{4.10}$$

where the maximum is taken over all (i, j) pairs which are feasible to add to the route.

In this procedure, one route is constructed at a time. A route is completed if no more origin-destination pairs can be added to the route, i.e. $\Delta = 0$. After one route is completely formed, all origin destination pairs on the route are eliminated from the set of demands and the process is repeated for the next route. The procedure terminates when the algorithm determines that no additional advantageous routes can be formed.

4.4.3 *Discussion of the problem*

This problem is interesting from a research standpoint since it is one of the few routing and scheduling problems that we have encountered that can be formulated as either a node routing or an arc routing problem and for which an effective solution procedure can be designed using either formulation. Moreover, the sequential insertion procedure, which is a heuristic rather than an optimal-seeking procedure, found the best solution when only a subset of the nodes were to be covered.

4.5. AN ALGORITHM FOR THE TRACTOR TRAILER ROUTING AND SCHEDULING PROBLEM WITH PARTIAL LOADS

An algorithm for solving the tractor trailer routing and scheduling problem with partial loads is described in [18] and outlined below. The algorithm consists of three basic components.

* An aggregation component joins sequences of O–D pairs which are to be serviced as a group. This sequence of *O–D* pairs is called a "string".
* A routing component generates routes based on total distance.
* A scheduling component determines the time for each pickup and delivery.

In the aggregation component, certain *O–D* pairs are aggregated together into indivisible groups in order to provide continuous stretches of work which are not interrupted by rest periods and which satisfy time window and crew workrule constraints. These strings are generally of short duration (less than 1 day's worth of work for the crews) and are treated as an indivisible unit to be serviced as a single task in the remainder of the algorithm. The net effect of the aggregation step is to provide the remainder of the algorithm with feasible portions of a vehicle route that are attractive from the viewpoint of minimizing total distance traveled.

The routing component forms a route and an ordering of the locations to be visited. Each route generated in the routing component satisfies the following minimal feasibility conditions: (1) the precedence relations implied by the *O/D* pairs are observed (that is, the origin of each load must be visited before its destination); (2) the vehicle capacity is not exceeded; (3) the maximum route length is not exceeded. Other feasibility conditions which primarily relate to the timing constraints are checked in the scheduling components.

The routes are generated using a generalized sequential insertion procedure. Suppose that a partial route is found and a new *O–D* pair is to be inserted onto the route. Every possible insertion permutation is checked for feasibility (precedence relations, exceeding the vehicle capacity, etc.) and the one selected is the one with minimal insertion cost. A route is complete when no more insertions are possible. The *insertion cost* is defined to be the additional distance traveled when adding the new *O–D* pair on the route. Since this insertion procedure checks every possible permutation, the procedure is more general than most insertion procedures in that it allows for a rearrangement of the order of servicing the *O–D* pairs on the route.

The scheduling component takes the route generated by the routing component and attaches

starting and finishing times to all the activities. The time phasing is carried out so that the total time it takes to complete the route is minimized and additional constraints concerned with workrules and time windows are maintained.

Once the route has been scheduled, the route is rejected if the time it takes to complete the route is greater than the maximum allowable route time $\bar{T}$ or if the time to complete the route is large relative to the distance traveled. The latter condition prohibits accepting routes where the scheduling components had introduced large waiting times.

Once a route has been accepted, all O–D pairs serviced on the route are removed from the list of available $O-D$ pairs and the process is repeated until all $O-D$ pairs are serviced.

This three step approach of aggregation, routing, and scheduling has been discussed previously in this Chapter and in the matching-based approach for forming crew schedules for mass transit systems (Chapter 3) and will be encountered again in Section 4.9.2. Of course, since each of these applications are different, the computational steps are different. However, the overall approach may be a useful place to start in tackling some of the routing and scheduling problems involving precedence relations and time windows frequently encountered in practice.

4.6. A PROCEDURE FOR THE STREET SWEEPER ROUTING AND SCHEDULING PROBLEM

Without parking regulations, the basic procedure in routing street sweepers over a single time period is the directed Chinese postman algorithm. As noted in Chapter 2, the directed Chinese postman algorithm is a polynomially bounded algorithm which is, essentially, the transportation algorithm. Because of the parking regulation constraints, however, the following two complications arise. The first complication is that after all deadhead arcs have been added to ensure that all nodes are in balance, the network may not be connected. Hence the result may consist of disjoint cycles. In that event, extra deadhead arcs must be added to the solution to force an Euler tour. Since the standard algorithm for the Chinese postman problem assumes that the network is connected, the algorithm for solving this problem is a variant of the algorithm for solving the directed rural postman problem. The algorithm for the directed rural postman problem is described in [87] and [88]. The algorithm for the routing and scheduling of street sweepers is a "route first" type of algorithm since it first solves a rural postman problem and then partitions the "giant tour" into feasible routes for each of the vehicles. The partitioning if carried out in a reasonable manner should leave out long "strings" of deadheading arcs (see [88]).

The second complication occurs in forming the routes (Euler tours) for each of the vehicles. In generating these routes, the standard approach of Edmonds and Johnson [197] creates a cycle with undesirable turns (such as U-turns or left hand turns). Some work has gone into finding ways to reduce the number of bad turns through devices such as solving an assignment problem at each node [87, 88] or through the insertion of additional deadheading arcs (McBride [474]). In enumerating the tour when utilizing the assignment problem approach, one may derive a number of cycles rather than a single cycle, even if the network is connected. These cycles have to be "patched together" to form a route at the cost of additional deadheading (see [87, 88]).

In developing a set of routes and schedules for the entire day, a sequence of rural postman problems (one for each time period) is solved, and these routes are partitioned in the manner just described. The routes and schedules for each time period are then pieced together using the same transportation algorithm described previously for scheduling school buses to form a daily work schedule for the sweepers. The time windows for servicing the streets correspond to the length of the time period in which the street can be serviced and these windows are automatically satisfied since no route for a vehicle in a time period can be longer than the length of the time period. In this problem, there are no precedence relationships since any street can be serviced at any time during the time period.

The problem of the routing and scheduling of household refuse collection vehicles is similar to the problem of the routing and scheduling of street sweepers. Since household refuse collection vehicles have an effective capacity, demand estimates for every street have to be computed. There are no complications due to parking regulations since it is assumed that household refuse collection can be carried out at any time. Unless there are specific restrictions imposed by the community under consideration, there is only one time period in a day.

Therefore, there is no scheduling component to the routing and scheduling of household refuse collection vehicles. Finally, the safety considerations concerning the types of turns allowed, mentioned earlier for street sweepers, can also apply to household refuse collection (McBride[474]).

Schuster and Schur[587] give a manual approach for solving this problem where the demand on a street is assumed to be linear with the number of houses on the street and the vehicle capacity is expressed in terms of the number of houses on the route. This procedure is applicable in a suburban, homogeneous housing area. This approach might run into problems in an area with mixed housing such as apartments, stores, single family houses, etc. No other successful approach for demand estimation has been developed to the best of our knowledge.

Other work on these problems has been carried out by Stricker[635] and Orloff[526]. Stern and Dror[623] consider a variant of this problem, namely the problem of routing meter readers. They utilize the "route first" approach. In partitioning the tour into feasible routes, an arc can be broken in the middle if that is where the route should end. Thus, part of a street could be covered by a meter reader on one day and the remainder of the street on a subsequent day. This is the only arc routing problem that we know about which allows for the breaking up of an arc among two (or more) routes.

4.7. ALGORITHM FOR THE AIRPLANE SCHEDULING PROBLEM

The Federal Express Corporation's airplane scheduling algorithm is a variant of the Clarke and Wright savings approach for vehicle routing (see Chapter 2). For this problem, we assume we have N cities. City i has a known demand in terms of the forecasted number of packages at the city and a time window $[\underline{t}_i, \bar{t}_i]$ during which service is permitted. The time window is set to reflect special conditions at the city such as the time by which all packages are to be collected, the time needed to deliver the packages and meet the guarantee that all packages are at their destination by a specified time, weather conditions, times when the airport is open, etc. For the priority/class of flights, two separate routing and scheduling problems—one for the departing flights and one for the incoming flights are defined.

The algorithm has the following steps.

Step 0: *definition of the savings.* The savings for having cities i and j on the same route where j can follow i on the route is computed by:

$$s_{ij} = d_{0j} - d_{ij} \tag{4.11}$$

where

d_{0j} is the travel time from the depot to city j
d_{ij} is the travel time from city i to city j.

The savings are ordered giving

$$s^{(1)} \geq s^{(2)} \geq \ldots \geq s^{(K)} \geq 0. \tag{4.12}$$

This definition of savings is different than in the ordinary Clarke and Wright algorithm because we are dealing with one way flights (either outgoing or incoming aircraft).

Step 1: *iteration step.* Suppose for the first S iterations we have r partial routes $R_1, R_2, \ldots, R_r$. A savings is then randomly selected by using a uniform distribution from the next t savings on the list. This savings involves the joining of the partial route beginning with city w to the partial route ending with city v. If the partial route beginning with city w can be feasibly attached to the partial route ending with city v then this attachment is made forming a new and longer partial route. Feasibility means that all time windows are satisfied, vehicle capacity (in terms of package count) is not exceeded, and the length of the tour is not too long.

The iteration step is repeated until the fleet size is met or all savings are exhausted.

Step 2: *merge step.* The new solution is merged into the present best known solution using the merge routine which was described previously for scheduling air crews (see Chapter 3).

Step 3: Steps 0, 1, and 2 are repeated until a satisfactory solution is found.

This algorithm is in use on a monthly basis at Federal Express Corporation. Furthermore, Hinson and Mulherkar[350] report that it has been successfully applied to the standard vehicle routing problem as an alternative to the standard Clarke and Wright algorithm with the savings defined as in the Clarke and Wright procedure. The use of randomization in selecting the next entity in a heuristic routing and scheduling algorithm has not been explored extensively. Randomization may be a potentially useful approach to solving certain routing and scheduling problems.

4.8. STATIC DIAL-A-RIDE PROBLEMS WITHOUT TIME WINDOWS

In this section, we discuss the various approaches which have been proposed for solving dial-a-ride problems without time windows. This section is divided into two parts—the single vehicle case and the multiple vehicle case.

4.8.1 *The single vehicle case*

In the single vehicle dial-a-ride routing problem, we have a single vehicle of capacity C and n customers specified in advance. Each customer has a pickup and delivery location, the pickup of the customer must precede the delivery of the customer (the precedence relationship), and there are no time windows. Because of the precedence relations, this problem is a constrained traveling salesman problem. In the ensuing section, we describe the approaches of Psaraftis[550, 552] and Stein[615] for solving this problem.

An optimal algorithm for solving the single vehicle problem. In [550], Psaraftis developed a dynamic programming approach using a linear objective function for solving this problem. Since the algorithm required $0(n^2 3^n)$ time to find the optimal solution, only problems with fewer than 9 or 10 customers (18 to 20 pickup and delivery points) can be solved to optimality in a reasonable amount of computer time. Tharakan and Psaraftis[643] utilize an exponential disutility objective function in a formulation aimed at making this problem even more dynamic or usable in real time. Their algorithm will be explained further in Section 4.10.

In [550], Psaraftis assumes that customers request service by telephone. The customers are numbered according to the order in which they have telephoned for service and are arranged on a list in the same order. Thus, customer i holds the ith position on the list. Let N be the total number of customers, "$+i$" be the pickup point (origin) of customer i, "$-i$" be the delivery point (destination) of customer i, and let A be the starting point of the vehicle (location of vehicle at $t = 0$). Furthermore, let $t(i, j)$ be the time to go from any one of the $2N + 1$ points of our problem, point i, directly to any of the other $2N + 1$ points in the problem, point j. The problem is to design a route for a vehicle which will feasibly service all the customers known at time $t = 0$. The objective Psaraftis used in [550] is as follows:

$$\text{Minimize } w_1 \sum_{j=1}^{2N} T_j + w_2 \sum_{i=1}^{N} [\alpha \text{WT}_i + (2-\alpha)\text{RT}_i] \tag{4.13}$$

where w_1, w_2 = given "weights"; α = customer's time preference constant $(0 \le \alpha \le 2)$; T_j = duration of the jth leg of the route; WT_i = waiting time of customer i, from $t = 0$ until his time of pickup; RT_i = riding time of customer i, from his time of pickup until his time of delivery.

The constraints on the route are the following: (i) every customer is picked up before he is delivered; (ii) the vehicle capacity C cannot be exceeded; (iii) the maximum position shift constraints are satisfied; (iv) it is feasible to travel between adjacent activities for the route j.

The maximum position shift constraints have the following explanation. If p_i is the position customer i holds in the sequence of pickups and d_i the position he holds in the sequence of deliveries, then for a given nonnegative integer MPS,

$$|i - p_i| \le \text{MPS for } i = 1, 2, \ldots, N \tag{4.14}$$

$$|i - d_i| \le \text{MPS for } i = 1, 2, \ldots, N. \tag{4.15}$$

Thus, a customer holding the 5th position in the initial list of customers is the 3rd customer picked up on the route, then his *pickup position shift* is $+2$. Similarly, if the same customer is the 6th customer delivered on the route, then his *delivery position shift* is -1.

The maximum position shift constraints are used to cut down on the number of computations necessary for the dynamic programming algorithm and to act as "service guarantees" for the customers. For example, the customer who holds the ith position in the initial list is guaranteed that his position in the sequence of pickups and in the sequence of deliveries will be between $i - \text{MPS}$ and $i + \text{MPS}$. In this way it is hoped that the customer will not have to ride on the vehicle for too long a time.

Other remarks given in [550] concerning this formulation are listed below:

(1) $\sum_{j=1}^{2N} T_j$ is the time to service all customers. In fact, a special case ($w_1 = 1$, $w_2 = 0$) of the generalized objective calls for the minimization of $\sum_{j=1}^{2N} T_j$, an objective identical to that of the classical TSP. However, the constraints in this problem are different than in the classical TSP.

(2) The quantity $\sum_{i=1}^{N} [\alpha \cdot \text{WT}_i + (2-\alpha) \cdot \text{RT}_i]$ is the assumed form of the total degree of "dissatisfaction" experienced by the customers until their delivery. The waiting and riding times are weighted unequally in general (except if $\alpha = 1$, when customers are indifferent between the two). The restriction $0 \leq \alpha \leq 2$ causes no loss of generality, since the ratio $\alpha/(2-\alpha)$ can take any value between zero and infinity and thus, the *relative* values of the customers' waiting and riding times can have any specified ratio. A special case ($w_1 = 0$, $w_2 = 1$) of the generalized objective calls for the minimization of the total (or average) customer "dissatisfaction". Other measures of customer dissatisfaction will be used as an objective in other dial-a-ride problems (see Section 4.9.1).

(3) The generalized objective function does not include terms reflecting the customers' waiting times from the instant of call until the instant the vehicle becomes available ($t = 0$). This apparent omission has no effect on the formulation since these terms, one for each customer, would appear as a constant (sunk cost) in the objective and would not affect subsequent decisions. The linearity of the objective function is crucial in the validity of the above argument. Linearity could be a problem for the real time dial-a-ride problem but is not a problem for the static dial-a-ride problem.

(4) The values of MPS in (4.14) and (4.15) can be customer dependent and thus, the inequalities 4.3 and 4.4 need not be two-sided or symmetric. Each customer i may have his own upper and lower bounds for his MPS.

Based on the above, Psaraftis in [550] then develops a dynamic programming algorithm for finding an optimal solution to this formulation. Details of this algorithm are not presented here but can be found in [550].

Minimum spanning tree heuristics. In [552], Psaraftis developed two heuristics for solving this problem. Both heuristics are based on the minimal spanning tree algorithm of Christofides [132] for the traveling salesman problem (see Chapter 2). The first heuristic, called Algorithm 1.1, does not include the origin (or depot) in its set of nodes, whereas the second heuristic, called Algorithm 1.2, does.

Both algorithms begin by finding a traveling salesman tour through all the nodes representing the pickup and delivery points but disregarding the precedence relations so that on this initial route, the delivery of customer i could precede its pickup. If there are n customers, Algorithm 1.1 finds a traveling salesman tour through $2n$ nodes while Algorithm 1.2 finds a traveling salesman tour through $2n + 1$ nodes (since it includes the origin). The remaining steps in both algorithms consist of various improvement procedures on this traveling salesman tour in order to find a route that is approximately of minimal length and that satisfies the precedence relationships. Although Psaraftis does not go into detail on these improvement procedures, it appears that he utilizes a k-opt interchange procedure such as described by Lin and Kernighan [446] for the traveling salesman problem. Psaraftis's k-opt procedure is described in [551].

Algorithm 1.1 repeats these improvement procedures utilizing all pickup nodes as possible starting points for the route; Algorithm 1.2 finds one improved route starting with the origin or depot. Psaraftis shows that Algorithm 1.2 is $0(n^3)$ and has a worst case performance bound of 3.0. That is, in the worst case it provides solutions that are 200% longer than the optimal solution.

Algorithms 2.1 and 2.2 are the same as Algorithms 1.1 and 1.2 except that the initial traveling salesman tour only utilizes the minimal spanning tree portion of the Christofides algorithm. The minimal spanning tree is then duplicated (as in Kim's approach[387]) and then polygons are removed to form a tour. Psaraftis shows that Algorithm 2.2 with a specified starting point is $0(n^2)$ and finds a tour of length no more than four times the length of the optimal tour. A third algoithm, which is a variant of Algorithm 2.1 is also provided in [552].

Although Psaraftis utilizes the Christofides heuristic algorithm for finding a traveling salesman tour, it appears that any of the heuristic traveling salesman procedures given in Chapter 2 could be utilized to find an initial solution. Some, of course, are more amenable to worst case and average case analyses than others. How accurate an initial solution is needed to get the best final solution is as yet unresolved.

A probabilistic analysis of this problem. Stein[617] conducted a probabilistic analysis of certain aspects of this problem. His work assumes n customers demanding service. Each customer has an origin and destination and each of these origins and destinations are drawn independently from a uniform probability distribution over some planar region R with area a (an assumption which he carries throughout his work). The problem is to develop a minimum distance single vehicle route that services all n customers such that each customer is picked up before delivered. For this problem, Stein shows the following:

THEOREM (Stein[617])

Let y_n^* be the length of the optimal route through n random demand pairs in a region of area a. Then,

$$\lim_{n\to\infty} \frac{y_n^*}{\sqrt{n}} = 1.89b\sqrt{a} \text{ with probability 1} \tag{4.16}$$

where b has been estimated to be 0.765 by Monte Carlo experiments. It can also be shown that $\lim_{n\to\infty} (E(Y_n^*)/\sqrt{n}) = 1.89\ b\sqrt{a}$. (See Larson and Odoni[421] for the analogue for the traveling salesman problem.)

Stein also gives the following simple algorithm for this problem: (a) first solve a traveling salesman tour through the n origins; (b) next solve a traveling salesman tour thorugh the n destinations; (c) join these two tours together. Let Y_n' be the length of the total tour. Stein shows that $\lim_{n\to\infty} (Y_n'/\sqrt{n}) = 2b\sqrt{a}$ with probability 1. This tour is approximately 1.06 longer than the optimal tour if the n demand points are randomly generated from a uniform distribution in a region of area a. Stein also shows that dividing a region into m equal subregions (as displayed in Fig. 4.9), visiting each one sequentially, and servicing all customers with a minimum distance tour within the region is asymptotically optimal. Of course, this result is again based on the fact that all points are randomly generated from a uniform distribution in a region R. How these results would change if a different distribution were used for generating the points is unclear.

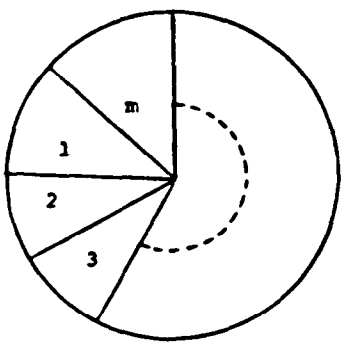


Fig. 4.9. Stein sectoring of a region for the school bus routing problem.

Stein continues this analysis for a multi-vehicle fleet and a multi-vehicle fleet with transfers allowed. He states that to minimize total travel time without transfers, one tour would suffice. If transfers are permitted and the objectives are to minimize time to completion as well as total travel time, then passengers are to be serviced in parallel by a fleet of k vehicles and the maximum length of any tour is shown to be

$$Y_n^k \approx \frac{4}{3}\sqrt{(2)}\frac{b}{k}\sqrt{(a)}\sqrt{(n)}.$$

Additional algorithms and results in the spirit of the above mentioned are developed by Stein for other problems. Note that in Stein's formulas the length of the tour is proportional to $\sqrt{n}$. The $\sqrt{n}$ relationship also occurs in many of the Monte Carlo simulations described by Psaraftis[552] for more complex and more realistic dial-a-ride problems without delivery times (see Beardwood *et al.*[50]).

4.8.2 *Multiple vehicle dial-a-ride*

To our knowledge, the only published papers that address the multiple vehicle dial-a-ride routing problem without time windows are the interactive optimization paper by Cullen, Jarvis and Ratliff[156] and the probabilistic analysis paper by Stein[617]. In an interactive optimization approach, the user is embedded within the optimization algorithm, interrupting the algorithm to guide its steps by aggregating certain nodes together, forcing certain nodes apart, and so forth. Cullen, Jarvis and Ratliff use an interactive approach that utilizes the concept of dual prices within a set partitioning algorithmic framework to indicate various promising alternatives to the user. Their procedures have been applied to the ordinary vehicle routing problem (see Chapter 2) as well as to the dial-a-ride routing problem. The results to date have been more encouraging on the vehicle routing problem than on the dial-a-ride routing problem. An interactive approach which combines Psaraftis's algorithm with the Cullen, Jarvis and Ratliff approach may hold promise for solving the multiple vehicle dial-a-ride problem without time windows. As yet, this approach is untested.

Stein also carried out some analyses for the multiple vehicle problem similar to his work for the single vehicle problem. Details of his analysis are not presented here but can be found in [617].

It will be very difficult to utilize the "giant tour" approach described for the vehicle routing problem (Chapter 2) for solving this class of dial-a-ride problems because there is no guarantee that the pickup and delivery points of the customers will be on the same route when the "giant tour" is broken up into clusters. The only reason the giant tour approach worked for the tractor trailer routing and scheduling problem, with full loads (Section 4.4) is that every trailer was picked up and then immediately dropped off so that the feasibility of the routes was easy to maintain.

4.9. STATIC DIAL-A-RIDE ROUTING AND SCHEDULING PROBLEMS WITH TIME WINDOWS

In this section, we consider approaches for solving the static or advanced reservation dial-a-ride routing and scheduling problem with time windows. In contrast to the dial-a-ride problems described in the previous section, the time of the pickups and deliveries play a significant role. This section is broken down into two parts—the single vehicle static case and the multi-vehicle static case.

4.9.1 *Single vehicle case*

In the single vehicle static case, we have a set of n customers known in advance, each of whom possesses an origin location, a destination location, and a desired delivery time. We seek a route and a schedule for a single vehicle with known capacity C. This route and schedule is to optimize a specified objective. This objective can vary depending upon the formulation of the problem. We consider the approaches of Sexton and Bodin[591] and Baker[21] in this section.

An approach based on Benders' decomposition. The objective used by Sexton and Bodin[591] minimizes the total inconvenience which a customer may experience. Total inconvenience is expressed as a linear combination of excess ride time and deviation from the desired delivery time. The following definitions are important.

* The *excess ride time* of a customer is defined to be his actual ride time minus his direct ride time.

* The *delivery time deviation* of a customer is defined to be his desired delivery time minus his actual delivery time. In this model, since late deliveries are not allowed, delivery time deviation is always non-negative.

* The *total inconvenience of a customer* is defined to be a weighted sum of his excess ride time and his delivery time deviation. The weights are assumed to be the same for all customers.

* The *total inconvenience in the schedule* is defined to be the sum of the total inconveniences of the customers. The objective in this problem is to minimize the total inconvenience in the schedule.

Each customer is assumed to have specified a desired delivery time. However, the approach described by Sexton and Bodin[591] is also applicable if all customers specify a desired pickup time rather than a desired delivery time since this pickup problem may be converted to a delivery problem in the following manner. For each customer, subtract the desired pickup time (in hours and minutes) from 24 hours and call it a desired delivery time. Call the pickup location for this customer the delivery location and the delivery location for this customer the pickup location. A "mixed" problem may arise if certain customers specify a desired pickup time and other customers require a desired delivery time. This situation is not handled by the technique presented in [591] and is considered to be an area requiring further research.

In [591], a mathematical formulation of the single vehicle dial-a-ride routing and scheduling problem and an algorithm for solving this problem are presented. The formulation is repeated here and the algorithm is briefly described.

Formulation of the single vehicle static case. Let n denote the number of customers to be serviced. Let x_i be the actual pickup time of the ith customer, y_i his actual delivery time and τ_i his desired delivery time (τ_i is assumed given). Thus, x_i and y_i are decision variables for the problem to be solved.

Define four $n \times n$ distance matrices D_{00}, D_{01}, D_{10}, and D_{11} whose entities in the ith row and jth column are, respectively:

$d_{00}(i, j)$ = minimum travel time from the *origin* of customer i to the *origin* of customer j,
$d_{01}(i, j)$ = minimum travel time from the *origin* of customer i to the *destination* of customer j,
$d_{10}(i, j)$ = minimum travel time from the *destination* of customer i to the *origin* of customer j,
$d_{11}(i, j)$ = minimum travel time from the *destination* of customer i to the *destination* of customer j.

It is assumed that $d_{01}(i, j) = d_{10}(j, i)$, $d_{00}(i, j) = d_{00}(j, i)$, $d_{11}(i, j) = d_{11}(j, i)$ and that the triangle inequality holds. Also, for the ith customer $\tau_i - y_i$ is the delivery time deviation and $y_i - x_i - d_{01}(i, i)$ is the excess ride time. Now, let ρ_i be the latest possible pickup time of the ith customer, calculated as

$$\rho_i = \tau_i - d_{01}(i, i). \tag{4.17}$$

Let C denote the capacity of the vehicle and A and B be the objective function weights on delivery time deviation and excess ride time.

Finally, define three sets of zero-one decision variables as follows:

$$u_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is picked up } \textit{before or at the same time} \text{ as customer } j \text{ is picked up,} \\ 0, & \text{otherwise,} \end{cases}$$

$$v_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is picked up } \textit{before} \text{ customer } j \\ & \text{is delivered,} \\ 0, & \text{otherwise,} \end{cases}$$

$$w_{ij} = \begin{cases} 1, & \text{if customer } i \text{ is delivered } \textit{before or at the same} \\ & \textit{time} \text{ as customer } j \text{ is delivered,} \\ 0, & \text{otherwise,} \end{cases}$$

for $i, j = 1, 2, \ldots, n$ and $i \neq j$.

The single vehicle dial-a-ride routing and scheduling problem (called the SVRS) may then be formulated as follows:

$$\text{Min } z = A \sum_{i=1}^{n} (\tau_i - y_i) + B \sum_{i=1}^{n} (y_i - x_i - d_{01}(i, i)) \tag{4.18}$$

subject to

$$x_i \leq \rho_i \quad i = 1, 2, \ldots, n \tag{4.19}$$

$$y_i \leq \tau_i \quad i = 1, 2, \ldots, n \tag{4.20}$$

$$y_i - x_i \geq d_{01}(i, i) \quad i = 1, 2, \ldots, n \tag{4.21}$$

$$\sum_{\substack{j=1 \\ j \neq 1}}^{n} u_{ji} v_{ij} \leq C - 1 \quad i = 1, 2, \ldots, n \tag{4.22}$$

$$d_{00}(j, i) \leq x_i - x_j + M u_{ij} \leq M - d_{00}(i, j) \quad i, j = 1, 2, \ldots, n \quad i \neq j \tag{4.23}$$

$$d_{10}(j, i) \leq x_i - y_j + M v_{ij} \leq M - d_{01}(i, j) \quad i, j = 1, 2, \ldots, n \quad i \neq j \tag{4.24}$$

$$d_{11}(j, i) \leq y_i - y_j + M w_{ij} \leq M - d_{11}(i, j) \quad i, j = 1, 2, \ldots, n \quad i \neq j \tag{4.25}$$

$$x_i \text{ and } y_i \text{ free} \quad i = 1, 2, \ldots, n \tag{4.26}$$

$$u_{ij}, v_{ij} \text{ and } w_{ij} = 0 \text{ or } 1 \quad i, j = 1, 2, \ldots, n \quad i \neq j. \tag{4.27}$$

In the above, M is a sufficiently large positive number. This formulation involves $2n$ continuous variables and $3n^2$ binary integer variables. There are $4n + 6n(n-1)$ constraints including n nonlinear constraints.

Inequalities (4.19) and (4.20) guarantee that no customer is picked up or delivered too late while (4.21), (4.23), (4.24), and (4.25) ensure that sufficient travel time is built into the schedule. The constraints (4.22) reflect the capacity of the vehicle. These are derived by counting the number of customers already on the vehicle at the pickup epoch of each customer. Although inequalities (4.19) are redundant with respect to (4.20) (4.21) and the fact that $\rho_i = \tau_i - d_{00}(i, i)$, $i = 1, \ldots, n$, it is useful to keep (4.19) in the ensuing analysis.

The following variables are used in the development of the algorithms for the single vehicle static case.

$$\phi_i = \rho_i - x_i \quad i = 1, 2, \ldots, n \tag{4.28}$$

$$\delta_i = \tau_i - y_i \quad i = 1, 2, \ldots, n \tag{4.29}$$

$$\begin{aligned} \sigma_{00}(i, j) &= d_{00}(i, j) + \rho_j - \rho_i \\ \sigma_{01}(i, j) &= d_{01}(i, j) + \tau_j - \rho_i \\ \sigma_{10}(i, j) &= d_{10}(i, j) + \rho_j - \tau_i \\ \sigma_{11}(i, j) &= d_{11}(i, j) + \tau_j - \tau_i. \end{aligned} \tag{4.30}$$

Here ϕ_i represents the slack in the ith pickup time constraint (4.19), δ_i is the ith delivery time deviation, i.e. the slack in the ith delivery time constraint (4.20), and the quantities $\sigma_{00}(i, j)$, $\sigma_{01}(i, j)$, $\sigma_{10}(i, j)$, and $\sigma_{11}(i, j)$ represent indicators of the 'space-time' separation of the corresponding tasks. For example, $\sigma_{01}(i, j)$ measures the spatial separation from the pickup of customer i to the delivery of customer j in the $d_{01}(i, j)$ term and the temporal separation in $\tau_j - \rho_i$. A new formulation using the above variables is presented in [591]. This formulation is then utilized in the development of an algorithm based on Benders' decomposition for solving this problem.

This algorithm operates as follows. An initial feasible route is established. A *route* is the order of picking up and dropping off the customers. Given this route, an optimal schedule is formed. A *schedule* specifies the exact times of pickup and delivery for the customers. From this optimal schedule, the objective function coefficients of the routing problem are changed. The order of servicing the customers on the route is then altered if the schedule based on the new ordering improves the objective function. Each new route and schedule produced yields a lower upper bound on the value of the objective of the (SVRS). This process continues until no new improved route and schedule can be found.

Key aspects of this algorithm are as follows.

(1) This algorithm is heuristic in the sense that a global optimum need not be found. As demonstrated in [591], however, the results found by this procedure are excellent and the algorithm itself is quite robust. The algorithm gave virtually the same result using any feasible starting route.

(2) The key to this approach is the scheduling algorithm. Given a feasible route, the resulting linear program is shown in [591] to be the dual of a maximum profit network flow algorithm. A noniterative algorithm was developed for finding an optimal solution to this scheduling problem. The total approach is computationally tractable because the scheduling algorithm is noniterative.

(3) A good lower bound has not been found for this problem. Hence, one can not measure how far from optimality the solution lies.

A branch and bound algorithm. In a recent paper, [21], Baker has proposed the following formulation for the single vehicle static dial-a-ride problem.

$$\text{Min } t_{2n} - t_0 \tag{4.31}$$

subject to

$$|t_j - t_i| \geq d_{ij} \qquad i = 0, 1, 2, \ldots, 2n;\ j = 0, 1, 2, \ldots, 2n;\ i \neq j \tag{4.32}$$

$$t_i \geq 0, \qquad i = 0, 1, 2, \ldots, 2n \tag{4.33}$$

$$w_{i1} \leq t_i \leq w_{i2} \qquad i = 1, 2, \ldots, 2n \tag{4.34}$$

where t_i is the time that stop i is visited. (t_0 is the time the vehicle leaves the depot); d_{ij} is the time to travel from city i to city j; and $[w_{i1}, w_{i2}]$ is a two-sided time interval around the time to visit stop i. w_{i2} can be thought of as the desired pickup or delivery time and w_i gives a definitive lower bound on how early stop i can be visited.

Suppose that t_i is the time that the pickup of customer i is to take place, t_{n+i} is the time that the delivery of customer i is to take place, and $w_{i,2} \leq w_{n+i,1}$, then any feasible solution to (4.32)–(4.34) will guarantee that customer i is picked up before he is delivered (the precedence constraints). The objective is to minimize the length of the route. This objective contrasts with Sexton's objective which is to minimize total customer inconvenience. There is no constraint on vehicle capacity; however, this omission does not seem to be a serious problem for dial-a-ride routing and scheduling problems.

Baker then analyzes the following four general approaches for solving this problem without

constraint (4.34)—nonlinear programming, dynamic programming, Lagrangean relaxation, and classical branch and bound methods. Baker shows that a penalty method which transforms the constrained optimization problem into an unconstrained optimization problem failed to find an optimal solution in over 90% of the several thousand cases tested. Since the model contains nondifferentiable functions, Baker developed a derivative-free penalty method to search for possible solutions.

Baker also discarded dynamic programming as an approach for this problem because of the enormous storage requirements imposed by dynamic programming. Similarly, Baker concluded that Lagrangean relaxation is not appropriate because this problem does not possess the mathematical structure necessary for using this approach. Baker found that branch and bound methods can be effectively utilized for solving this problem.

In developing the branch and bound algorithm, Baker shows that the dual to the model without (4.34) is a longest path problem on an acyclic network. Problems of this type can be solved by very efficient network algorithms. As a result, a nine-node problem was solved to optimality in 0.02 seconds on a UNIVAC 1100/40.

At this time, variants of the formulation have not been tested using the branch and bound algorithm so that we do not know of complications that might occur when two-sided windows on servicing the pickup or delivery of a customer are utilized or whether the algorithm will slow down significantly when a large number of customers demand service at or about the same time. The formulation is simple, however, and the solution approach is straightforward so that it does offer hope for solving problems of this type.

4.9.2 *The multi-vehicle static case*

An area that is currently receiving a significant amount of attention is the multi-vehicle static dial-a-ride problem. Over the last year, five new approaches to this problem have been developed and virtually none of these approaches are far enough along to be either completed or well documented. A meeting was held at the Urban Mass Transportation Administration in May, 1982 to review four of these approaches and to come up with future theoretical and applications-oriented directions for algorithmic development in this area. With the implementation of computerized scheduling systems for advanced reservation dial-a-ride sytems in Brockton, Massachusetts, and Dade County, Florida, and the soon-to-be-developed system in Toronto, Canada, the future looks bright for a significant breakthrough in this area shortly.

The five algorithms can be called the NEIGUT/NBS procedure, the MIT procedure[364], the Bodin/Sexton procedure[94, 78], the "Taxi" procedure[399], and the University of Montreal procedure[562]. Of these five approaches, the first three procedures are most alike in terms of algorithmic philosophy and are discussed below. There is a disclaimer on [399] which prohibits its discussion herein and the University of Montreal approach[562] is still embryonic and undergoing significant changes.

A sequential insertion procedure. The NEIGUT/NBS procedure which can be classified as a sequential insertion procedure has the following four key features:

(i) It seeks to maximize vehicle productivity by determining the minimum number of vehicles necessary to provide service.

(ii) It seeks to meet a specified quality of service for all the customers serviced by ensuring that all customer services have a maximum excess ride time and that they are picked up and delivered within given time intervals.

(iii) Not all customers have to be serviced since some customers can be serviced by a supplemental service made up of exclusive ride taxi and/or the demand responsive fleet.

(iv) The supplemental service should be at the least cost to the service agency.

The NEIGUT/NBS procedure initially determines time windows for the pickup and delivery of each customer. The time windows for a customer are a function of the exclusive ride time of the customer (the minimum time to go from the customer's origin to his destination) and specified parameters such as maximum on board time. Service for each customer will be guaranteed to fall within these time windows. Once these windows are defined, it is irrelevant to the algorithm whether a desired pickup time or desired delivery time has been specified or not.

Since the NEIGUT/NBS procedure schedules one vehicle and determines the best customer to insert on the route at a time, the algorithm falls into the class of sequential insertion

procedures described in Chapter 2 and in Section 4.5. Assume that a partial vehicle schedule has been created and that this partial vehicle schedule ends at time t^0 with the vehicle empty. Furthermore, assume that a customer c_k with pickup location p_k, delivery location d_k and earliest possible pickup time $\bar{t} > t^0$ exists and that it is possible to deadhead from the end of the partial vehicle schedule to p_k (i.e. the deadhead time $\langle \bar{t} - t^0 \rangle$. This customer c_k is eligible to be selected as the base trip. The algorithm then searches over all of these eligible base trip customers and selects the "best". The algorithm then attempts to build a cluster of customers and a partial route and schedule using customer c_k as the seed element and attach this partial route and schedule to the partial vehicle schedule which has already been created.

Any customer whose earliest pickup time falls between $\bar{t}$ and the latest possible delivery time of c_k, t^*, is selected for a possible pairing with c_k, and all possible route permutations of these two customers with p_k coming first are enumerated.

In this case, there are two permutations. If c_l is the other customer, then the two possible route permutations are p_k, p_l, d_l, d_k, and p_k, p_l, d_k, d_l. Of all possible permutations, the permutation selected is the one that is feasible in terms of the pickup and delivery windows for c_k and c_l and that minimizes the travel time from p_k to d_k. In this way, a grouping of two customers is found. Having this two customer grouping, a third customer is then joined to the cluster using the above ideas. Of course, with a three customer cluster, there are more route permutations to check. When no more customers can be added to the cluster, the partial vehicle schedule is expanded to include this cluster and its partial route and schedule.

This process continues until no more customers can be added to the route. This route is then frozen and the process is repeated over the remaining set of unassigned customers. As with most insertion procedures, the routes generated first are the most productive and the routes generated last tend to be inferior in quality. In the NEIGUT/NBS procedure, there is no route improvement routine. One, however, could certainly be applied.

A batch concurrent procedure. As described in [364], the assumptions in the MIT procedure regarding customer service are similar to the NEIGUT/NBS procedure in that:

(1) A customer's ride time cannot exceed his direct ride time by more than a specified percentage or the ride time can be no longer than a specified maximum total time.

(2) A customer must be delivered (or picked up) at his destination (origin) within a specified time window; The time window is actually a time interval defined in the manner described below; Because of the way in which the time window is defined, the MIT algorithm is the only procedure described in this section which allows either late deliveries or early arrivals; that is to say, a delivery later than the customer's desired delivery time or a pickup earlier that the customer's desired pickup time. The objective is to minimize total vehicle length and the fleet size is specified in advance.

The MIT procedure is classified as a batch concurrent algorithm since it can assign customers to more than one vehicle at a time and it can assign more than one customer to a vehicle at a time. The day is split *a priori* into time intervals of 20–30 minutes duration. Customers are identified as falling into certain time intervals. Within a time interval, the customers are categorized as follows.

Category 1. The desired pickup time of the customer falls into the time interval.

Category 2. The desired pickup and delivery time of the customer falls into the time interval.

Category 3. The desired delivery time of the customer falls into the time interval.

In the MIT algorithm, all customers falling in a time interval can be serviced at any time during the time interval. Thus, if the time interval is 8:00 to 8:30 and a customer in Category 3 who has a desired delivery time of 8:01 is scheduled for delivery at 8:30, then this delivery is 29 minutes late. No other algorithm makes this assumption. Furthermore, the MIT algorithm assumes that customers in Category 1 in time interval n will be Category 3 customers in time interval $n+1$. Assume that partial vehicle schedules exist over the first $n-1$ time intervals. The MIT algorithm then forms clusters for the customers for each of the vehicles in time interval n. This is accomplished via a special spanning tree algorithm. Basically, a set of spanning trees are formed over all the customers who are to be serviced in time interval n. Each spanning tree is formed over a subset of the customers who are to be serviced in time interval n and no customer can lie in more than one spanning tree. With some exceptions, the number of spanning trees found in time interval n is equal to the number of the partial vehicle schedules

found over the first $n-1$ time intervals. In [364], various rules are given for this tree formation. A look-ahead feature is incorporated in this step to ensure that Category 1 customers in the same spanning tree in time interval n go "well together" in time interval $n+1$. The minimal spanning tree heuristic described in Section 4.8.1 is then used to give a route for each spanning tree over the interval. At this time, no route improvement procedure is utilized.

A clustering/concurrent insertion procedure. The set of assumptions embedded within the Bodin/Sexton procedure is somewhat different than the assumptions contained in the other procedures. The basic assumptions in the Bodin/Sexton procedure are as follows.

(1) As with the MIT procedure, the Bodin/Sexton procedure assumes that a fleet size is specified in advance.

(2) Every customer has specified a desired time of delivery or a desired time of pickup. At this stage, the algorithm can not handle the problem of some customers having a desired time of pickup and others having a desired time of delivery.

(3) There are no upper bounds on the delivery time deviations (or pickup time deviations) or excess ride time for a customer although these constraints are currently being added to the algorithm.

(4) The objective is to minimize total customer inconvenience over all the customers in a manner analogous to the single vehicle problem as defined in Section 4.9.1.

As with the other procedures, the Bodin/Sexton procedure begins by finding small subsets of the customers who look like they should be serviced by the same vehicle. Furthermore, it is assumed that all customers in each of these subsets, are picked up by the vehicle before any customer is delivered. Various heuristic rules are employed to ensure that all customers within a cluster have about the same desired delivery time, the delivery locations are "close enough" and the pickup locations are not too widely scattered. These subsets of customers serve as seeds for the formation of the vehicle routes.

Other customers are then attached to these vehicle clusters via the iterative use of a transportation algorithm. The supply nodes in the transportation algorithm are the vehicle clusters (with a supply of 1), the demand nodes are the usassigned customers (with a demand of 1) and dummy supply and demand nodes are defined to ensure feasibility. In the transportation problem, the cost of joining supply node i with demand node j is the increase in total customer inconvenience as defined by (4.9) in assigning customer j to cluster i. Stringent tests are employed to ensure that the customers who are assigned first affect the objective cost the least and that their desired pickup or delivery times are close to the pickup or delivery times of at least one of the customers within the cluster.

A swapper algorithm (a 1-opt procedure) is then utilized to move customers between routes in order to reduce total customer inconvenience. The Benders' procedure (described in Section 4.9.1) is utilized throughout the entire process to reorder the pickups and deliveries of the customers on a route in order to improve the total customer inconvenience of the customers within a vehicle cluster. Furthermore, customers who adversely affect the routes are either not assigned to a vehicle cluster or are removed from their vehicle cluster. These customers are assumed to be serviced by an exclusive cab or a vehicle in the demand responsive fleet.

4.10. DIAL-A-RIDE DEMAND RESPONSIVE ALGORITHM

In a demand responsive system, we examine the system when a new customer calls into the dispatch office requesting service. At the time of this request, some of the earlier customers have been delivered to their destinations and, hence, are no longer considered in this problem. The remaining earlier customers who have requested service have been assigned to a vehicle and are either on board the vehicle waiting to be delivered to their destination or are waiting to be picked up. Moreover, at this time, the route and schedule for each vehicle is known. The problem is to determine the assignment of the new customer to a vehicle and the new route and schedule for the vehicle that the customer is assigned to. Because this assignment is carried out in real time, the algorithms are heuristic and very rapid.

Virtually all of the work that we are familiar with on this problem has been carried out by Wilson, Psaraftis, Odoni, and their colleagues at MIT. Some of these approaches have been extensively tested in the Rochester, New York Dial-A-Ride Demonstration Project, and in

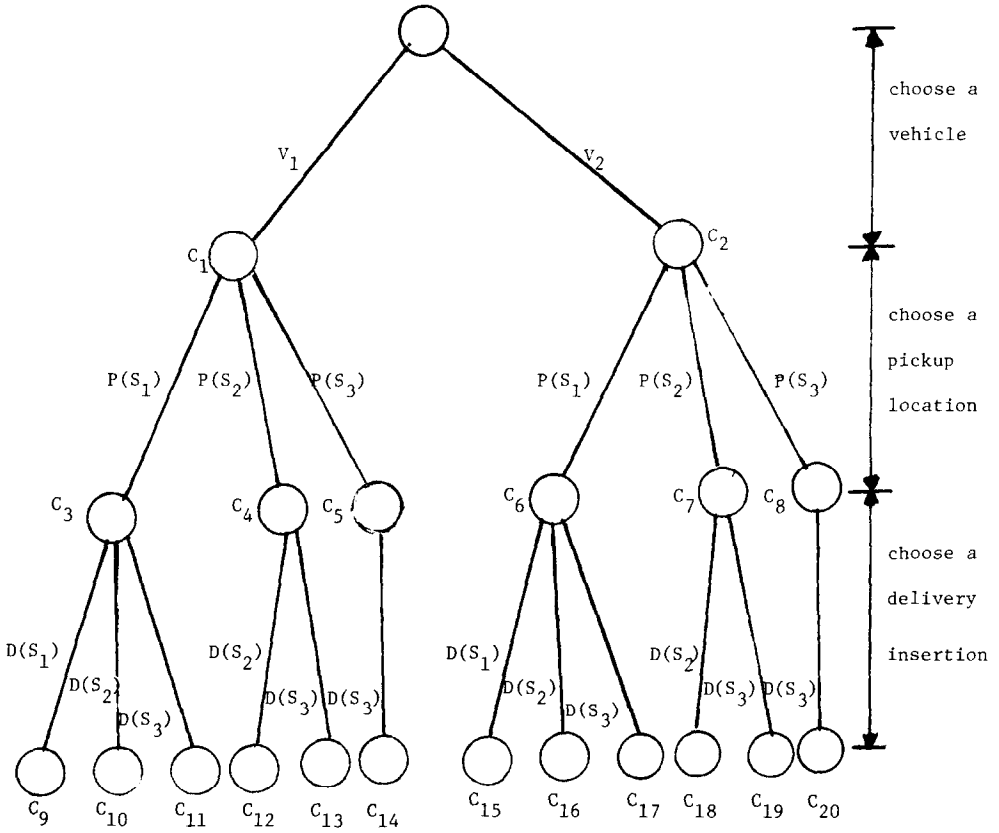
other locations. In this section, we describe the approaches of Wilson [680], Psaraftis [550], and Tharakan and Psaraftis [643].

An insertion algorithm. The immediate assignment algorithm of Wilson's selects the best way to incorporate the new passenger into the existing set of customers by use of an insertion algorithm (see Chapter 2) as soon as the customer demands service. All possible combinations of insertions of pickup and delivery stops for this customer into the routes for all available vehicles are investigated. This assignment is carried out as a three level decision tree as illustrated in Fig. 4.10 for two vehicles. Each branch in the tree represents a particular choice and each level of the tree enumerates the choices available. A cost is assigned to each node and a branch and bound algorithm is utilized to carry out the tree search.

Customers are broken down into classes depending upon the service demanded. Let D_{ij} be the disutility for the ith customer in class j. Then

$$D_{ij} = a_j w_i^2 + b_j R_i^2 + c_j P_i^2 \tag{4.35}$$

where, a_j, b_j and c_j are parameters; w_i = pickup time for customer i—time for request for service for customer i; R_i = ride time from pickup to delivery for customer i; P_i = actual pickup time—promised pickup time for customer i. Thus, if it is proposed that customer p in class q be



V_j = the j^{th} vehicle is chosen for the assignment of the new demand

$D(S_k)$ - the new demand's delivery is inserted after the k^{th} stop of the given vehicle

$P(S_k)$ - the new demand's pickup is inserted after the k^{th} stop of the given vehicle

C_n - the "cost" or "bound" associated with the n^{th} node

Fig. 4.10. Solution tree representation of the assignment of a new demand for the dynamic dial-a-ride problem.

assigned to a vehicle then, the total disutility for all customers is

$$D_{pq} + \sum_i \sum_j (D_{ij}^1 - D_{ij}) \tag{4.36}$$

where D_{ij}^1 is the disutility for customer i in class j after customer p in class q is assigned to a vehicle and D_{ij} is the same before customer p in class q is assigned to the vehicle. Another term in the objective function attempts to spread the tour length equally among all the vehicles. This term, for vehicle v is

$$(\mathrm{TL}_v^1 - \mathrm{TL}_v)(d\overline{\mathrm{TL}} + e\mathrm{TL}_v) \tag{4.37}$$

where d and e are parameters, TL_v^1 is the tour length for the vehicle under consideration after the new passenger is inserted into the vehicle, TL_v is the tour length for the vehicle under consideration before the new passenger is inserted onto the vehicle, and $\overline{\mathrm{TL}}$ is the mean tour length after assignment. Thus, the total objective function seeks the best vehicle v to assign the new demand. This vehicle v is the one which minimizes z where

$$z = D_{pq} + \sum_i \sum_j (D_{ij}^1 - D_{ij}) + (\mathrm{TL}_v^1 - \mathrm{TL}_v)(d\overline{\mathrm{TL}} + e\mathrm{TL}_v). \tag{4.38}$$

Wilson further extends the algorithm in the following ways. He develops an automatic reassignment capability (that is, the ability to systematically reconsider previous assignments) as part of the overall control procedure. Such a procedure, implemented in Rochester, has the following features:

(1) Every 20 minutes, the system is examined. All customers in the system who have not as yet been picked up by a vehicle are reassigned to another vehicle. The reassignment goes as follows. First, customers assigned to vehicle 1 are reassigned, then vehicle 2, etc. The order in which customers are reassigned from a vehicle corresponds to the times when they are assigned to that vehicle. All customers on a vehicle or already delivered are not considered for reassignment.

(2) Whenever a driver makes a stop out of sequence or whenever a vehicle is more than 15 minutes late in carrying out its route, all customers waiting to be served by that vehicle are considered for reassignment. Again, all customers on a vehicle or those already delivered are not considered for reassignment.

An alternative approach developed by Wilson involves the notion of deferred assignments. In this case, the algorithm does not assign a demand immediately upon receiving its request for service, but defers a decision in the hope that future information on other customers may assist in developing routes which can better service the customers. An extension to Wilson's algorithm incorporates a deferred assignment capability into the immediate assignment algorithm, and keeps track of the two best vehicle assignment alternatives until needed by the algorithm. Details of the deferred assignment and optimal reassignment algorithms along with many other extensions of the above type are described by Wilson and Miller [680].

A dynamic programming algorithm. In [550], Psaraftis extends the dynamic programming algorithm described in Section 4.8.1 to the single vehicle demand responsive case. Assume that the decision is made to assign a new customer to a vehicle. The maximum position shift constraints eliminate the possibility of indefinite deferrment in the case when a customer is being continually assigned to his last position in the pickup and delivery sequence because of his unfavorable geographical characteristics with respect to other customers. In Wilson's work described above, this is accomplished in part by the quadratic objective function and through heuristic criteria. The objective used here is the same as in Psaraftis's static case where the customer sets considered are all customers on board the vehicle along with customers waiting to be picked up. The origin of the route consists of the new location where the vehicle will be when the route is implemented. All changes in the value of the objective function concerned with the past are regarded as "sunk costs" (costs already incurred) and since these costs are additive, they are eliminated from further consideration. The updating of the route is carried

out every time a request for service is received and the customer is assigned to the route in question. The maximum position shift constraints are appropriately updated.

As noted in Section 4.8.1, this algorithm is slow when there are more than 8 or 9 customers. Computational speed may not be much of a problem in a dynamic environment since the number of customers to be served by a vehicle at any point in time is small. We do not know if the algorithm has been tested as yet in a real-time environement.

Dynamic programming algorithm with exponential disutility function. In [643], Tharakan and Psaraftis show that the dynamic programming approach of Psaraftis given in Section 4.8.1 can be extended to this problem with the objective

$$\underset{s \in S}{\text{Minimize}} \sum_{i=1}^{N} a_i \exp\left(\lambda \sum_{n=0}^{2N-1} t_n^i(s)\right) \tag{4.39}$$

where the following notation holds: S = the set of all feasible dial-a-ride sequences of stops; s = an element of S; a_i = constant for customer k $(i = 1, \ldots, N)$; λ = constant, same for all customers; $t_n^i(s)$ = incremental time customer i "suffers" from either waiting for the vehicle or riding on board it, when the vehicle travels between the nth and the $n+1$st stops of sequence s.

The a_i's are not necessarily assumed to be the same for each customer i. This is done not so much to reflect differences in time preference among customers, but rather to take into account the time each customer has been waiting for service up to the time the vehicle starts its route. Thus, if for each customer i, T_i is being counted from the time the vehicle is at $n = 0$, if all customers are assumed to have identical time preferences, and if T_i^0 is the interval from the moment customer i requests service until the moment the vehicle starts its route, a_i can be set as $a_i = \exp(\lambda T_i^0)$. In that way, customers who have requested service earlier than others are "favored" by the routing procedure. With this formulation, Tharakan and Psaraftis [643] can take into account how long a particular customer has been waiting for service prior to the routing of the vehicle. This is particularly useful in a "dynamic" or "real time" environment, where routing consists of an open-ended sequence of solution updates, each performed upon the appearance of a new customer request. On the other hand, as noted in Section 4.8.1 and in this section, a linear disutility function considers the time a particular customer has been waiting for service *prior to any update* as a "sunk cost", hence this time is irrelevant for routing decisions at subsequent updates.

Tharakan and Psaraftis then show that the dynamic programming algorithm of [643] can be applied to this problem. The same comments made in Sections 4.8.1 and previously in this section regarding the computational efficiency of the dynamic programming algorithm hold for this algorithm as well.

CHAPTER 5
IMPLEMENTATION ISSUES

As in all areas of operations research, there can be major difficulties involved in using routing and scheduling models and algorithms to solve real-world problems. We feel that research in the area of routing and scheduling should place special emphasis on developing methodologies that are particularly suited for solving real problems. In this chapter, we present an overview of some of the implementation issues that can occur when real routing and scheduling problems are tackled. More than the other chapters, this chapter draws on informal discussions with our colleagues and our personal experiences and opinions. This orientation is necessitated by the unfortunate fact that many details concerning implementations do not find their way into published papers. Some exceptions are the following books and papers, which we have used in preparing this chapter: Mercer, Cantley and Rand[478], Kursh[409], Larson and Odoni[421], and Eilon, Watson-Gandy and Christofides[201].

In Section 5.1, we discuss one of the most obvious and prominent of implementation obstacles: obtaining "real data". Section 5.2 discusses the use of man–machine interaction in computer implementations of routing and scheduling algorithms. Many researchers have found that for a variety of reasons the users of operations research tools desire to have a degree of "on-line control" of the progress of algorithms. Section 5.2 gives some of the reasons for considering interactive systems and discusses recent successes and some of the current work in this area. In Section 5.3 we discuss "reliability" considerations in routing and scheduling systems. Reliability in this case means the ability of a strategy specified by an algorithm to stand up to component failures, e.g., a tightly optimized set of crew schedules is of little value when 20% of the crews are sick. In Section 5.4 we consider some of the problems involved with handling the large database associated with many routing and scheduling problems. In Section 5.5, we discuss issues related to obtaining organizational acceptance of proposed solutions. In Section 5.6, we discuss factors involved in choosing the most appropriate model for a particular real-world setting.

5.1. DATA REQUIREMENTS

The following data requirements are associated with a wide variety of routing and scheduling problems: (1) distances and/or travel times; (2) costs; (3) service requirements. Many routing and scheduling problems have characteristics that fall into all three categories and consequently generate data requirements in all three categories. However, in many cases, a model is chosen with a primary consideration being the minimization of data required. For example, the classical vehicle routing problem has, as its objective, the minimization of distance traveled by a fixed size fleet of limited capacity vehicles subject to the restriction that customer demands are satisfied. The data required are distances, customer demands, vehicle capacity and fleet size. All of these quantities are relatively easy to obtain. However, the use of distances in the objective function and the fleet size constraint in many cases serve as a surrogate for a complex cost function that includes fixed costs that vary with the number of vehicles and costs that vary with distances and the times vehicles are in service. The constraints, also, might be a simplified representation of the restrictions occurring in practice. Just as the requirement of an efficient solution technique may limit a model's applicability to real-world problems, extensive data requirements can also impede the usefulness of certain models.

The first data category, distance, is probably the easiest to handle. A variety of sources exist containing highway and airline mileages between cities. Alternatively, given longitudinal coordinates, it is a trivial matter to compute straight line distances between two points. On a more microscopic level, analysis routines such as UTPS[658], are available for computing highway or street distances between pairs of locations by executing a shortest path algorithm.

Distances, however, are in many cases a surrogate for travel times. The travel time required between a pair of locations is a random variable that depends on (a) time of day, (b) location of travel. In some cases, distance divided by a mean travel speed yields an acceptable travel time. However, we have encountered several instances where this is not the case. The dial-a-ride problem discussed in [94] for the city of Baltimore used straight line distances plus a speed that varied by location within the city to produce times compatible with those experienced in practice. In [38], one speed was used for inter-city travel and another for intra-city; again acceptable times were obtained. For mass transit systems, travel times typically vary with time of day. This consideration becomes important when passenger demand levels change significantly with time during a single day.

The operating costs associated with vehicles can be difficult to estimate exactly. For this reason surrogate objectives, e.g. travel time or distance, number of vehicles, etc., are used in many cases. However, where vehicle operating costs must be traded off against other costs, e.g. capital costs or crew costs, the vehicle costs must be explicitly included. Typically, per mile factors and per vehicle factors are used to convert distance and number of vehicles to costs. While the accuracy of such estimates might be questioned, it should be noted that decision-making processes and maintenance policies in many cases use such factors, especially when they may be easily incorporated into the model used.

Crew costs can be determined quite precisely since well-defined procedures exist for determining them. However, it should be noted that these cost functions are usually very complex and due to differences in the union agreements they can vary widely from one problem to another.

In many cases, such as vehicle and crew scheduling problems arising from timetables and some vehicle routing problems, the demands or service requirements are well-defined and easy to obtain. However, in a large number of applications, e.g. garbage collection and fuel delivery, demand is stochastic in nature and at best rough estimates are available. The random factors in these problems should be handled explicitly. As discussed in Chapter 2, work in this direction has been undertaken in recent years and certainly deserves more attention.

5.2. INTERACTIVE COMPUTER SYSTEMS

One of the most significant impediments to user acceptance of computer systems is a feeling of the user's loss of control over the underlying physical system. Many computer-generated solutions are rejected based on relatively minor issues that could be corrected if certain controls over the computer system were given to the user. By including man–machine interaction dynamically in computer systems better solutions have been obtained and, more importantly, wider user acceptance has resulted.

The two major contributions a user can make dynamically during the execution of an optimization algorithm are—(1) including implicit constraints and (2) using human intuition to aid a heuristic algorithm. The first contribution directly impacts the acceptability of the final answer. In many real-world situations it is impractical to include all possible constraints in an optimization model. There may be too many constraints or they may be rather nebulous and hard to define. An interactive system would start to solve a model that leaves out many of the constraints and periodically shows solutions or partial solutions to the user. If at any time a violated constraint were observed, the user could then change the problem to include the constraint and the automatic solution algorithm would continue on. There are several desirable features to such an approach. First of all, in many cases only a small percentage of the constraints would be violated by the solution algorithm. Thus, a large amount of effort would be saved over trying to include all the constraints at the outset and, in addition, the smaller problem could have more structure and be more amenable to solution by a particularly efficient algorithm. Secondly, in many cases it would be difficult for a user to identify a constraint *a priori*, whereas when a solution that violated that constraint is given to the user, the violated constraint could be easily identified. Such as approach was used by Bodin *et al.*[86] to solve a railroad blocking problem.

The second major contribution a user can make dynamically within an optimization algorithm is to help direct a heuristic algorithm using human intuition. There are many situations in which human intuition cannot be easily programmed into a heuristic algorithm. An

important instance is that of spatial considerations. In [156], Cullen, Jarvis and Ratliff describe a set partitioning-based interactive system for solving dial-a-ride problems. User intuition is used to aid the generation of candidate vehicle routes (columns for the set partitioning problem). The user generates a candidate route after examining the current set of routes illustrated on a map. This method allows the user to provide spatial intuition that otherwise could not be obtained. In [402], Krolak, Felts and Marble employ a human's spatial intuition in the solution of Euclidean traveling salesman problems. Here the automatic portion of the procedure clusters points by solving assignment problems and presents these partial solutions to the user on a Euclidean map. The human is then able to take into account spatial considerations in completing and altering the solution given.

The discussion at the beginning of this section laid out two well-defined objectives of man–machine interaction. In many cases, however, a given technique could be used for either of these two reasons. In [36], Ball, Bodin and Dial describe a framework for using interaction in a crew scheduling system. This algorithm is based on the procedure described in Section 3.5.3. It iteratively builds up a set of paths corresponding to pieces in a crew schedule. The user can break up or fix certain partial paths at any iteration. The user could invoke this capability either to break up paths that violated certain constraints or to encourage the algorithm to head in a direction the user felt was promising. In the latter case the user would be using intuition obtained primarily from the service profile (see Fig. 3.9).

The urban mass transit crew and vehicle scheduling system described in [380] has a flexible easy-to-understand user interface. Before and after optimization algorithms are invoked both for crew and vehicle scheduling, the user has the option of fixing and/or altering the solution. As in the case mentioned above, this capability can be used either for invoking implicit constraints or for forcing certain solutions the user feels are desirable. In addition, the crew scheduling component itself, allows the user to intervene at each step to adjust a variety of problem parameters or to "roll back" the algorithm to a previous step. This feature is important to this approach because it requires that several parameters be set to tailor the algorithm to the particular problem environment. The interactive features allow these parameters to be altered dynamically or reset when the algorithm is performing badly.

5.3. RELIABILITY CONSIDERATIONS

All human or physical systems experience failures. The system designer must decide in the modeling phase how explicitly to take reliability considerations into account. At one extreme, system reliability could be included explicitly within the objective function or the constraint set. Alternatively, once a solution which did not account for component failures is obtained, the result of failure of certain components could be evaluated and contingency plans formulated. In this section, we point out areas where component failures could affect the effectiveness of a solution to routing and scheduling problems and give our own judgements on how important reliability considerations are in the design of routing and scheduling systems. As a rule, few models in this area explicitly take reliability considerations into account. Rather, certain operational procedures are used during the implementation phase to ensure reliable operation.

The most significant failures in any routing and scheduling system involve one of its three major components: vehicles, crews, and the network the system uses. All types of vehicles experience unscheduled breakdowns from time to time. A typical cause of network failures is bad weather, which could eliminate certain roads (arcs) from a truck delivery network or certain airports (nodes) from a passenger airline network. Crew failures occur any time a crew is unavailable at an unscheduled time. The most typical cause would be crew illnesses. There could be a variety of secondary system failures, such as failure of a power supply, failure of communications equipment, etc. We will ignore this third class since such failures are fairly rare and should be appropriately handled by controls on the underlying physical system.

Analysis of the reliability of the system consisting of crews or vehicles is fairly easy because a large number of components in the system are interchangeable. A typical urban mass transit system has hundreds of identical buses so that when one fails another can easily take its place. A similar situation exists with respect to crews. This is the classical K out of N system and the reliability of such systems can be ensured by always having enough available spare vehicles or crews. In particular, let K = No. of vehicles or crews required to operate the system,

p = the failure probability of a vehicle or crew, ρ = the system reliability desired, then the number of vehicles or crews required, N, can be found by choosing the smallest value for N that satisfies

$$\sum_{i=0}^{N-K} \binom{N}{i} p^i (1-p)^{N-i} \geq \rho.$$

This type of system can be readily analyzed and implemented. However, some care must be taken to insure the interchangeability of vehicles and crews. In particular, if a system contains a wide variety of vehicle types that are not interchangeable or if a system has several different categories of crew skills then the pools of interchangeable vehicles or crews could be very small. The obvious design consideration suggested is to ensure that to as great an extent as possible, crews and vehicles are interchangeable. Of course, this policy might not be compatible with other design considerations such as varying vehicle size depending on route demand.

The geographic dispersion of the system can also affect the ability to interchange components. For example, if depots and/or routes are spread over a wide geographic region then it may be impractical to replace a failed vehicle even if a spare exists somewhere in the system. Such considerations are most appropriately taken care of in depot location studies.

A variety of practical problems can arise in the allocation of replacement crews since typically absent (failed) crews must be replaced on short notice. Mass transit agencies typically retain a pool of "extra board operators" who are not guaranteed work on a daily basis; rather they are set aside as replacement operators for absent crews or other special assignments, e.g. charter buses, special event service, etc. In [583] an interactive system is described for quickly allocating extra board operators to work made available by absent crews according to a seniority based criterion.

Network failures are probably most vividly illustrated in the case of airline systems where safety is of utmost importance. Here reliability factors must be considered in vehicle routing in that in the event of a node (airport) failure an alternative landing site must be available. Fortunately, the world is now fairly dense in airports so nearby landing sites are generally available. However, the capability to reach alternate sites affects routing considerations in that the aircraft always must have sufficient fuel to get to an alternative site. This implies that the constraint on maximum flight length is tighter than it might be otherwise.

5.4. DATABASES FOR ROUTING AND SCHEDULING

As we have discussed previously the typical routing and scheduling problem requires input data from several sources and its output may impact several different groups in an organization. If a one-time solution to a problem is desired, then, while gathering and organizing data can be cumbersome, special file or database organization techniques are not required. On the other hand, operational software packages that must repeatedly solve a routing or scheduling problem for a particular distribution system require user-oriented, flexible data handling capabilities. In fact, as we discuss below, the successful application of optimization models in an operational setting can depend more heavily on the data handling routines than on the optimization algorithms. Software packages that are difficult to use and understand for non-computer professionals will rarely be accepted. Typically, the problem that is most evident and most obviously in need of computerization is the organization of a myriad of data. If this is not solved efficiently, then users will reject the entire package. To illustrate these difficulties we discuss the data handling problems associated with mass transit crew and vehicle scheduling.

The RUCUS system for scheduling urban mass transit crews and vehicles [285], [69] was developed in the late 1960's, field-tested in the early 1970's, and made available for industry use in 1973. It was developed under the sponsorship of the Urban Mass Transportation Administration (UMTA) of the U.S. Department of Transportation and, as such, was intended to be a general package available to and usable by a wide variety of transit agencies. The experience with RUCUS through the middle 1970's was disappointing in the sense that few agencies were able to use it successfully and those that did, only used a subset of its components. In addition, it was never easily installed on a site but rather only became

```
MINI-SCHEDULER II      DOCUMENT : TRIPS      SIZE :   19 23      USER : SAGE
MODE 1          DIV  1          LINE 1          SER  1

? GET DOCUMENT TRIPS

 01   02   03  -04- -05- -06-  -07-  -08- -09-   10   11   -12-  -13-  -14-
ROW  TRIP BLOK PAT- DIR  POG    POT   PIT  PIG  TRIP TRIP  1ST/  1STA  1STB
     HOOK NUM  NUM                                TIME MILE  MAIN  MAIN  MAIN
0001 S     2   14      9999  806A                  60    5  815A  830A
0002 S     4   41      9999  759A                  60    5
0003 S     1   14      9999  836A                  60    5  845A  900A
0004 S     6   41      9999  829A                  60    5
0005 +     2   41                                  60    5
0006 +     4   14                                  60    5  915A  930A
0007 +     1   41                                  60    5
0008 E     6   14                1059A 9999        60    5  945A 1000A
0009 +     2   14                                  60    5 1015A 1030A
0010 +     4   41                                  60    5
0011 E     1   14                1159A 9999        60    5 1045A 1100A
0012 +     2   21                                  30    5
0013 +     4   12                                  30    3 1125A 1135A 1140A
0014 +     2   12                                  30    3 1155A 1205P 1210P
```

Fig. 5.1. Tabular representation of trip file.

operational after significant modifications by agency personnel or more usually by consultants with a high level of computer expertise.

Two of the primary difficulties with the system were that the large number of data files that had to be input were represented as card files and the entire system was run in a batch environment. When minor modifications were required or minor errors discovered the entire procedure has to be restarted. Errors were common and restarts were difficult since specifying the correct programs and data files involved manipulation of JCL commands that were less than easy to understand especially for transit agency personnel not versed in computer know-how. Any computerized system for analysis of mass transit systems would have to include a trip file, a vehicle deadhead file, a crew inter-relief travel time file as well as files related to particular crews, lines, vehicles and garages. It is clear that these files are highly interrelated. Thus, a single change to the mass transit system could induce changes in several files. It seems clear that the complexity of this file system together with the rather primitive manner in which RUCUS handled the data was the primary reason for the initial failures of the RUCUS system.

In recent years, enhancements based on the RUCUS system [413, 380, 582] have met with a high degree of success. The optimization routines in these enhancements are similar to those in RUCUS. However, a principal difference between the original RUCUS system and the enhanced versions lies in the data handling routines. In particular, the relationships among files are more explicitly handled in that the entire system is treated as a database containing interrelated information rather than a set of independent files. In [380], a flexible interactive user interface that is suggestive of the now popular relational data model [168] was provided for user control over the data and all optimization processes as well. Figure 5.1 gives an example of the table representation of data presented to the user. A flexible command structure allows for easy deletion of rows and/or columns and changes to individual data values. We believe that the emphasis placed on data handling by the enhanced systems forms the primary reason for their success.

5.5. ORGANIZATIONAL ACCEPTANCE OF RESULTS

In any operations research study the organizational acceptance of the results depends not only on the quality of the solution obtained and the techniques used but also on a variety of factors involving how the study team relates to the organization. In this section, we briefly describe five factors which we have found to be important in our own experience: (a) involving personnel within the organization; (b) designing a friendly computer interface; (c) gaining acceptance by impacted workers; (d) assessing the organization's true objectives; (e) identifying all of the solution's benefits.

In the following paragraphs we describe each of these factors in detail.

(a) *Involving personnel within the organization.* When any outside study team comes to analyze a system, the persons intimately involved with the system always initially have a much greater knowledge of its operation. To gain the required knowledge of the system the team must interact closely with the internal personnel. However, more importantly, the internal personnel will tend to have a natural resistance towards the "outside" experts. To overcome this feeling and to ensure the accuracy of the model used, the internal personnel should be intimately involved in the study, e.g. in setting forth model assumptions, in deciding on the model's objectives, in gathering data, etc. In this manner, the study itself will not only belong to the study team but also to the organization being studied. In addition, the fact that the organization considers the model its own will greatly increase its chances for acceptance. A much more accurate model will also result.

This point is illustrated well in [409]. A street sweeping model was developed and implemented both on data from New York City, and Washington, D.C. In New York, sanitation department personnel did not actively support or participate in the study whereas in Washington they did. In both cases, the computer model produced solutions which substantially reduced department costs. However, in New York the solution was never used whereas in Washington it was. Although the New York personnel gave technical reasons for rejecting the solution it seems clear that if they had been more actively involved, the final solution would have been implemented.

(b) *Designing a friendly computer interface.* Very often when a computer system is delivered to a customer for periodic generation of routes and schedules, it replaces a manual system. The users probably were quite comfortable with the manual system and had little background in computer systems. Consequently, an essential factor in receiving acceptance of the system is that it be easy to use and understand for persons not well versed in computer technology. The factors in Sections 5.2 and 5.4 that led to the success of the enhanced RUCUS systems fall into this category. An interactive user interface and efficient data handling routines greatly increased the ability of users to handle the computer system.

(c) *Gaining acceptance by impacted workers.* Nearly all routing and scheduling studies will impact crews even when crew scheduling is not explicitly considered. In the case of crew scheduling studies, the objective function itself is usually crew costs. There will be obvious reaction from unions if total crew wages are reduced as schedules become more efficient or if more work is being asked of crews with no increase in pay. In addition, even though a computer generated set of crew schedules does not explicitly violate union rules, the union may find it unacceptable simply because it contains more "undesirable" schedules. There is no set solution to this problem, however, we offer a few methods we have encountered in our experience:

(1) A portion of the cost reduction produced by more efficient schedules could be passed on to the workers, e.g. if the computer generated solution reduced costs by 2%, then union wages could be raised by 1%, thus, the workers get 1% and the transit agency gets 1%.

(2) New scheduling procedures could be implemented at the same time that changes in service are made, e.g. computer-generated crew schedules could be implemented at the same time as the addition of new lines to the system. Thus total crew costs might remain the same or increase slightly even though substantial savings were being realized.

(3) Constraints or objective function components could be added to the model to ensure a certain number of desirable crew schedules e.g., if the workers preferred schedules that started at 8 AM and had a short lunch break, a constraint could be added to the model guaranteeing a certain percentage of these types of schedules. Thus, a slight deviation from the minimum cost solution might be well worthwhile if it ensures the acceptance of the results.

(d) *Assessing an organization's true objectives.* The typical operations research viewpoint is to identify a cost function and then make the objective of the study the minimization of this function. In many cases the true objective might be much more subtle. For example, the primary objective of much of the RUCUS[69] and UCOST[92] work in transit system planning is to simplify and streamline a rather cumbersome manual scheduling and transit planning process as well as to produce lower cost crew and vehicle schedules. Thus, the costs of the scheduling and planning departments will decrease and much more time will be available to plan for a wider variety of services. The fact that a primary objective relates to how quickly and efficiently the system can be used suggests that a great deal of effort should be put into designing an efficient usable computer system.

In Chapters 1 and 4, we discuss situations involving school bus routing and street sweeper routing where safety and convenience objectives were explicitly considered. Such objectives can be hard to quantify and even harder to trade off with cost objectives.

(e) *Identifying all of a solution's benefits.* It may be difficult to justify a computer system based only on the direct cost savings it generates, e.g. old crew costs minus new crew costs. However, as the example cited above suggests there may be a variety of other costs and benefits. Every effort should be made to identify these costs and benefits and where possible to quantify them.

5.6. CHOOSING THE RIGHT MODEL

In the preceding chapters, we have presented a variety of models and algorithms. We have attempted to describe the assumptions underlying each model and the factors guiding each algorithm. In a particular situation, the user has to decide which model and/or algorithm is most appropriate. This choice, in many cases, is not obvious. If the wrong model or algorithm is selected, then either the results will be unusable or the algorithms will be too time consuming for all practical purposes.

To illustrate this point, consider the example of estimating the cost of a mass transit system (see [92] for further details). The nominal approach in attempting to estimate the cost of a mass transit system is by the use of a macromodel. The nominal form of a unit cost model is the following:

$$\text{Operating cost} = \sum_{i=1}^{n} A(i)\ [\text{level of causal factor } i].$$

The causal factor is typically some physical characteristic of the proposed system such as vehicle miles or number of passengers. $A(i)$ is the cost per unit of causal factor i. The relations between costs and causal factors are often based on regression models.

Examples of unit cost models which have been used are the following:

(a) UTPS model developed by U.S. Department of Transportation[658]:

$$\text{operating cost} = \frac{\text{total cost}}{\text{in-service vehicle mile}} \times (\text{No. of in-service miles}).$$

(b) Roess, Huss and Kivicklis[567] (also, the Gilman and Voorhees[275] model): operating cost = $A \times$ (No. of car miles) + $B \times$ (No. of car hours) + $C \times$ (No. of cars in operation during peak period) + $D \times$ (No. of passengers for the system).

(c) Gavin and Roark for the Washington Metropolitan Transit Authority[252]: repairs and services of equipment = \$168.07 × (vehicle miles in thousands); tires and tubes = \$15.5 × (vehicle miles in thousands); maintenance of service facilities = \$363.07 × (No. of vehicles); supervision of shops and garages = \$135.37 × (No. of service employees); other maintenance and service expenses = \$20.74 × (vehicle miles in thousands); operator supervision = \$1026.00 × (number of operators); fuel and oil = \$39.52 × (vehicle miles in thousands); solicitation and advertising = \$5.43 × (No. of passengers in thousands); insurance and safety = \$49.57 × (vehicle hours in thousands); operator wages and bonuses = \$4132.00 × (vehicle hours in thousands); total operating cost = sum of costs given above.

Although the above cost models may give accurate cost estimates in a particular situation, they do not take into account enough of the significant characteristics of the system in estimating system operating cost to be used as general cost models. As an example of a potential problem area, these cost models (with the exception of supervisory cost for the WMATA study[252]) never utilize the number of operators in their cost computations, and the wages paid to the operators make up a significant portion of the cost of any transportation system. Because the scheduling of crews over the day is complex, the planner is forced to use a surrogate term (such as in-service hours) to estimate crew costs. Since the relationships between operator wages and proposed surrogates are usually tenuous, such a cost estimate may be inaccurate.

As a second example of a problem area, estimation of the cost for maintenance and fuel and tires requires accurate estimates of in-service and deadhead miles and total number of vehicles.

However, due to layover time, deadheading and other complications of scheduling, accurate accounts of deadhead vehicle miles and peak vehicles cannot be satisfactorily determined without vehicle schedules.

Thus, existing unit cost models can be criticized because they do not utilize the number of operators and the vehicle schedules over the day in deriving their cost estimates and that they may lack specificity. The cost model in [92] overcame these criticisms by more closely analyzing the actual operations of the proposed transit system. The unit cost procedures are called MACRO models while the cost model in [92] is called a micro cost model.

In this setting a micro cost model is costly in terms of computational requirements, however, it gives an estimate of crew requirements (without actually finding actual crew schedules). Moreover, a cost model which requires actual crew schedules (using the approaches described in Section 3.5) is even more costly in terms of computer time.

How does an organization choose between these models? The macro model is the most flexible and most efficient computationally but the least accurate. Consequently, it would be most appropriate for long range planning where a wide variety of possibilities had to be considered and where the input data itself was not too accurate. For shorter range planning, fewer configurations are considered and input data is more accurate. In this case the micro model would be the most appropriate.

Accuracy and computational efficiency are only two dimensions of a model. In general, there are a variety of other tradeoffs a user must make in choosing the best model.

CHAPTER 6
CONCLUSIONS AND FUTURE DIRECTIONS

In the previous chapters, we have discussed a large number of routing, scheduling, and routing and scheduling problems. The origin of some of these problems was theoretical while other problems grew out of applications. Some of these problems were amenable to elegant mathematical formulations while others were so confounding that no clean and concise mathematical formulations have been developed. Effective algorithms already exist for some of these problems while for others the current algorithmic capabilities leave much room for improvement. Moreover, if one examines the taxonomy of routing and scheduling problems in Chapter 1, one would note that there is a large number of important routing and scheduling problems which have not been explored. Thus, we believe that this area is maturing but still remains in the formative stage. Work by both theoretical and applied analysts will continue in the future and more elegant, sophisticated and comprehensive formulations and algorithms will evolve.

In what follows, we first discuss the prospects for using routing and scheduling models within real-world settings and then list possible directions of future research in the area of routing and scheduling.

6.1. FUTURE DEVELOPMENT AND USE OF ROUTING AND SCHEDULING SYSTEMS

We believe that at this time, organizations are just starting to realize the potential for savings resulting from the automation of their routing and scheduling activities. As the success of many routing and scheduling systems becomes better known, investments in new systems should increase significantly.

The decrease in the cost of computing in general, and of micro- and mini-computers in particular, will make automated systems particularly economical. One may currently purchase a 64 K micro-computer with 5–10 megabytes of hard disk storage, printer terminal and all systems software for under $20,000. If compatible routing and scheduling software existed, even small organizations would be able to automate their routing and scheduling activities.

Unfortunately, one of the major weaknesses in the utilization of routing and scheduling models is the lack of effective general purpose software for these problems. At this point in time, little has been done in the development of general software for solving these problems. The IBM Vehicle Scheduling System [10] for solving the single depot vehicle routing problem was one such package. This software had mixed success because of its inability to handle secondary constraints and its relatively cumbersome data requirements. The RUCUS system [69] for scheduling mass transit drivers is another general purpose system. Although its earliest versions were plagued with difficulties, later enhanced versions of the RUCUS system have proved to be quite successful. For air crew scheduling, computer packages have been implemented and documented in Rubin [570] and CAPS [269]. These packages are very expensive to use (they can require several hours of CPU time on a large pairings problem) especially when additional constraints are added to the problem. This was the case with Federal Express Corporation's crew scheduling problem when run on the CAPS system (Baker [20]). Christofides *et al.* [110] have developed a fairly general computer package for solving some vehicle routing and scheduling problems. However, systems such as these are rare and as we have described, typically they are inflexible and require a high degree of "on-site fine tuning".

Two main shortcomings of most general purpose systems to date have been: (i) the inability to handle particular problem settings and additional side constraints, and (ii) the inadequacy of the data-base system especially in its interface with the user. The second point is a constant source of difficulties. Nothing can discourage the user more than the inability to obtain and

manipulate data that is supposedly "in the computer". This lack of emphasis on the data-handling capabilities of routing and scheduling systems has led to the demise of many such systems. This oversight may be traced in part to the tendency in certain developers to pay much more attention to the algorithmic aspects of the system (a common vice of academics to which the present authors have also succumbed!). To remedy this, general software systems of the future should incorporate the following vital ingredients: (i) a flexible interactive user interface; (ii) a data-base system that is easy to understand and manipulate.

The preceding characteristics were particularly emphasized in Chapter 5 where a strong case for interactive capabilities was made. The use of man/machine interaction in the data input stage is vital to the correct preparation and validation of the data. On the other end, an interactive capability will facilitate the incorporation of problem-dependent parameters and side constraints and could be used to guide the algorithmic procedure itself. In this way some of the inevitable insensitivity of general purpose algorithms to specialized problem environments may be alleviated.

The next few years, in our view, will witness the emergence of efficient general purpose software or software/hardware combinations for solving some standard routing and scheduling problems. Although the development of such software will require a considerable investment, the right package for an appropriate set of applications, if correctly priced and marketed, would do well financially and further the acceptability of software systems as a superior alternative to manual routing and scheduling.

In addition to the preceding considerations, the algorithmic effectiveness of routing and scheduling software systems will be a major determinant of their success and widespread use. Consequently, the evolution of routing and scheduling systems is intimately related to future algorithmic development. The next section focuses on future directions of research in this area.

6.2. FUTURE RESEARCH DIRECTIONS IN MODELING AND ALGORITHMIC DEVELOPMENT

Organizations engaged in distribution activities are constantly encountering new problems that differ in their set of constraints, cost structures, and performance criteria. These problems will necessitate the development of new mathematical formulations, solution methodologies, and computer implementations. On the other hand, new developments in mathematical programming constantly suggest and lead to more refined solution techniques for even the most well-studied problems, for instance the traveling salesman problem. It is this combined input from both applications and theory that keeps the research in routing and scheduling alive and vibrant.

The development of solution methodologies for the standard single-depot vehicle routing problem (VRP) should illustrate this point: In 1973, only two general approaches were available for the VRP—"the route first-cluster second" approach and the "cluster first-route second" approach. Seven years later, we have at least five additional approaches (see Chapter 2). Moreover, these approaches represent significant advances in our ability to solve the problem accurately and efficiently and to justify the goodness of the solution.

The area of routing and scheduling is certainly at an early stage of its development. Consequently, a detailed list of open problem areas could go on for hundreds of pages. In the recent NSF symposium on routing and scheduling, a number of discussion sessions focused on future research directions. The resulting papers [39, 436, 455, 585] provide an extensive collection of issues that may serve as points of departure for future research. In the remainder of this chapter, we offer our own thoughts on future research directions in summary form.

Modeling strategies. We feel that the comparison of alternative modeling approaches and the question of model validation are two important and largely neglected issues in routing and scheduling. In Chapter 1, we argued that since all routing and scheduling problems have essentially the same output, the differentiation between problems lies in the modeling assumptions and the nature of required inputs. The logic behind these modeling assumptions and their role in converting a complex real-world problem into a structured model are therefore as important as the algorithmic strategies adopted to solve the problem. To our knowledge, no formal approach for studying these issues is available. The following examples illustrate the type of questions that may emerge in this context.

(1) Given a particular real-world distribution system, how important are timing con-

siderations? In particular, when can all timing constraints be ignored so that the simplest form of the vehicle routing model can be used? When can timing constraints be handled by a single constraint on total route time? When must timing constraints be explicitly considered so that a routing and scheduling model must be used?

(2) Three types of schedules can be associated with any airline or mass transit system: a timetable (passenger schedule), a crew schedule, and a vehicle schedule. What should be the relationship between procedures for generating these schedules? Should these problems be posed and solved independently? That is, generate timetables first, then vehicle schedules, and finally crew schedules. Should pairs of these problems be solved simultaneously, e.g. generate timetables and vehicle schedules simultaneously and then crew schedules as is done in the airline industry? Could some sort of feedback loop be imposed? What are the potential errors associated with these different approaches?

These are just some examples of a wide variety of modeling questions that could be posed. We feel that exploration of such issues is important and deserves research attention.

Categorization, formulation and characterization. A comparison of Chapters 2 and 4 suggests that the level of mathematical structure decreases as one passes from routing problems to scheduling problems and then on to combined routing and scheduling problems. Each of the following factors have had a part in leading to this fact concerning the structure of routing and scheduling problems:

(a) It may be argued that routing problems intrinsically admit more well-defined structures than scheduling problems which, in turn, are simpler than combined routing and scheduling problems. Generally speaking, scheduling problems are more constrained than routing problems and the incorporation of all relevant constraints tends to destroy the simpler structures of the former class of problems.

(b) Routing problems have been studied and analyzed more extensively than either scheduling or combined routing and scheduling problems. Few would disagree, for example, that the first routing problem we discussed—the traveling salesman problem—is one of the most celebrated and well-studied problems of mathematical programming. The greater attention paid to routing problems in the form of abstract models has led to much more satisfying categorizations and characterizations. The problems of Chapters 3 and 4, however, have scarcely received the same kind of attention and still await a full categorization.

(c) Most crew scheduling and combined routing and scheduling problems have emerged out of particular applications and have been studied by researchers who have mainly focused on solving particular problems as opposed to establishing general structures. This historical factor has undoubtedly contributed to the distinction raised in (b) above.

One may readily observe that the development of effective algorithms (both exact and heuristic) for routing and scheduling problems is closely related to the degree of categorization and characterization attained for such problems. The close relationship between good characterizations and effective algorithms is a recurring theme in combinatorial optimization. In routing and scheduling, too, some of the most successful heuristics developed recently have grown out of insightful formulations and related relaxations[219, 437]. Magnanti's thoughtful paper[455] gives particular attention to elucidating the relationship between the structure of formulations and the nature of resulting algorithms.

These comments suggest that the development of new mathematical characterizations and formulations is crucial to the maturation of this field. This development should aim at not only new algorithmic strategies, but also at an integrative effort that would bring out the close relationships among seemingly different applied problems. Two important directions for such development, therefore, come to mind:

(i) The expansion of formulations to include wider classes of problems including, for example, some currently ill-defined scheduling and routing problems and certain generalizations of routing problems[585].

(ii) The search for characterizations and formulations to exhibit problem structures that may be readily utilized to motivate and propel algorithm strategies.

Computational complexity. Most of the well-known "core" problems within routing and scheduling have been classified as either *NP*-hard or polynomial[436]. Of course, many problems have yet to be defined without ambiguity so that questions concerning their com-

putational complexity can be posed. It can be rightfully argued that, since most routing and scheduling problems are known to be *NP*-hard, any of the yet to be defined more complex generalizations of these problems would be *NP*-hard by definition. We feel, nevertheless, that this argument should not bar further investigation into the complexity of structured versions of routing and scheduling problems whose most general form is known to be *NP*-hard. This research could identify tractable subproblems or reasonably good pseudopolynomial algorithms for hard problems. For example, the three generalizations of the vehicle scheduling problem discussed in Chapter 3 are known to be *NP*-hard if no restrictions are placed on the acyclic network over which they are posed. A realistic special case for all three problems is obtained by specifying a path minimization objective function and by not excluding any arcs based on a maximum time weight. The computational complexity of these three vehicle scheduling problems for this special case is an open question whose resolution would be very valuable to algorithmic researchers.

Solution strategies. (a) *Exact algorithms.* Except for the traveling salesman problem, the effort in developing exact algorithms for *NP*-hard routing and scheduling problems has been very limited. Exceptions include set partitioning approaches to crew scheduling and exact vehicle routing algorithms (the latter developed by Christofides *et al.*[110, 29]). Even for vehicle routing problems where substantial structure is present, the existing exact algorithms are still largely inferior to those for the traveling salesman problem. One may safely conclude that the area of exact solution techniques for routing and scheduling is still in its early infancy.

Since, even in the presence of effective heuristics, the inherent appeal of optimal solutions is evident, we feel that exact algorithms for all hard routing and scheduling problems should be developed and investigated. Possible directions of such development may be found in Magnanti's paper[455] where a strong case for further study of exact algorithms is presented.

(b) *Heuristic algorithms.* Most algorithms presented in this report are heuristic in nature and cannot guarantee optimality. Given the inherent complexity of routing and scheduling problems and the large size of the problems encountered in practice, heuristics may well continue to be the dominant algorithmic approach to such problems. Ball and Magazine[39] note the significance of further research on routing and scheduling heuristics. Some of the promising research directions are listed below:

(i) The design of mathematical programming-based heuristics. These heuristics would use the structure revealed by new formulations, relaxations, and related exact algorithms. In fact, some novel recent heuristic approaches are of this variety[219, 437, 629]. The mathematical programming structure of such algorithms provide them with capabilities usually absent in heuristics, e.g. the ability to bound the deviation between the heuristic and optimal solutions as well as the potential for some sensitivity analysis.

(ii) Research in the analysis of heuristics. The performance of heuristics may be analyzed through worst-case, probabilistic, and statistical analysis in addition to the usual empirical studies.

(iii) Comparison of heuristics on data bases arising from particular applications. Many studies in routing and scheduling have compared different heuristics (among each other or against manually-obtained solutions) on databases of particular real-world problems. We may ask if this comparison process could be more formalized. What, in particular, can be said of the performance of heuristics on data-sets arising from the same problem setting? The formalization of the appropriate questions in this context and the means by which they can be answered is of special interest in view of the applied nature of the routing and scheduling area.

(iv) Investigation of the role of man/machine interaction within heuristics. This is one of the many issues related to the use of interactive systems discussed in Chapter 5. The precise role of the human element and the nature of his inputs within such systems has yet to be defined.

In conclusion, we feel that research in the area of routing and scheduling will continue to flourish due to both the vital importance of the area in distribution activities and its academically challenging nature. This highly attractive combination of methodological and applications-oriented interest will, hopefully, help routing and scheduling to evolve into one of the most successful areas of applied operations research.

Integration of routing and scheduling with other functions. In Chapter 1, we opened this report with an overview of the distribution functions of a firm. At some point of the planning

process, effective distribution management was shown to involve a detailed study of routing and scheduling. Correspondingly, the results of such a study can be vertically integrated into the planning process by studying the effect of proposed routes and schedules on the distribution activities and the required system design. Issues such as facility location, centralization versus decentralization, districting, etc. could benefit from the use of routing and scheduling "micro" models.

The full integration of the results of routing and scheduling models with "higher-level" questions such as those mentioned above constitutes a new direction of research that may prove to have a great impact on the future use of routing and scheduling models. In the area of distribution management, the work of Christofides[135] and Fisher *et al.*[221] indicates that routing models have come of age to respond to inventory management and customer service level considerations. A more complete integration of routing and inventory components of a distribution system now constitutes a realizable goal. In the same way, for other systems relying on routing and scheduling on the operational level, one may now start to pose questions of tactical or strategic scope and hope to answer them through a creative use of routing and scheduling subsystems. Numerous new problem areas may emerge in the process of devising and using hierarchical models of systems that rely on vehicles and crews to function effectively.

ACKNOWLEDEGEMENTS

We wish to thank our colleagues Nicos Christofides, Jim Bookbinder, Jacques Ferland, Ed Baker, Hoon-Liong Ong, Bob Dial, Gilbert Laporte, Jean-Marc Rousseau, Harry Psaraftis, and Marshall Fisher for reading an earlier draft of this manuscript and for providing us with some insightful comments. In addition, we express gratitude to Ron Fisher, Nat Jasper, and Cara Gossard of the Urban Mass Transportation Administration (UMTA) of the U.S. Department of Transportation for their guidance throughout this project. We are grateful to Mrs. Dawn Abatemarco for assisting in the management of this undertaking. Finally, we thank Dean Rudolph P. Lamone who has been and continues to be uncommonly supportive and helpful. This project was sponsored in part by UMTA's Office of Policy Research under contract number MD-11-0004.

REFERENCES

1. N. Agin and D. Cullen, An algorithm for transportation routing and vehicle loading. *Logistics* (Edited by M. Geisler), pp. 1–20. North Holland, Amsterdam (1975).
2. A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass. (1974).
3. S. Akl, An analysis of various aspects of the traveling salesman problem. Ph.D. Thesis, McGill University (1978).
4. S. Akl, The minimal directed spanning graph for combinatorial optimization. *Austral. Comput. J.* **12**(4), 132–136 (1980).
5. S. Akl, On the expected number of optimal and near-optimal solutions to the Euclidean traveling salesman problem. *J. Comput. Appl. Math.* **7**(4), 287–288 (1981).
6. A. Ali and J. Kennington, *The M-travelling salesman problem: a duality based branch and bound algorithm.* Technical Report No. OR80018. Department of Operations Research, Southern Methodist University, Dallas, Texas (March 1981).
7. S. Altman, E. Beltrami, S. Rappaport and G. Schoepfle, A nonlinear programming model for household refuse collection. *IEEE Trans. Systems, Man. Cybernetics* **SMC-1**., 289–291 (July 1971).
8. R. Angel, W. Caudle, R. Noonan and A. Whinston, Computer Assisted School Bus Scheduling. *Management Sci.* **18**(6), 279–288 (1972).
9. S. Ankolekar, N. Patel and J. Saha, Optimization of vehicle schedules for a road transit corporation. University of Montreal Transportation Research Center Publication No. 213 (1981).
10. Anonymous, *Vehicle Scheduling Program, Application Description Manual,* H20-0464. IBM Corporation, White Plains, New York (1968).
11. J. Arabeyre, J. Fearnley, F. Steiger and W. Teather, The airline crew scheduling problem: a survey. *Transportation Sci.* **3**(2), 140–163 (May 1969).
12. S. Arisawa and S. Elmaghraby, The "hub" and "wheel" scheduling problems, Parts I and II. *Transportation Sci.* **2**, 124–165 (1977).
13. R. Armstrong, W. Cook and G. Martin, Decomposition and sensitivity analysis in an aircraft tasking problem. *INFOR* **15**(3), 332–343 (1977).
14. K. Asghrazadeh and G. Newell, Optimal dispatching strategies for vehicles having exponentially distributed trip times. *Naval Res. Logistics Quart.* **25**(3), 489–509 (1978).
15. A. Assad, Analytic models in rail transportation: an annotated bibliography. *INFOR* **19**(1), 59–80 (February 1981).
16. A. Assad, Models for rail transportation, *Transportation Res.* **14A**, 205–220 (1980).
17. A. Assad, Modeling rail freight management. Ph.D. Thesis, Sloan School of Management, MIT (September 1978).
18. A. Assad, M. Ball, L. Bodin and B. Golden, Combined distribution routing and scheduling in a large commercial firm. *Proc.* 1981 *Northeast AIDS Conf.*, (Edited by R. Pavan and P. Anderson), pp. 99–102 Boston (1981).
19. A. Assad, B. Golden, R. Dahl and M. Dror, Design of an inventory/routing system for a large propane-distribution firm. *Proc. of* 1982 *Southeast TIMS Conference* (C. Gooding, ed.), 315–320 (1982).
20. E. Baker, Efficient heuristic solutions for the airline crew scheduling problem. D.B.A. Thesis, University of Maryland, College Park, Maryland (1979).
21. E. Baker, *Time oriented vehicle routing and the traveling salesman problem.* School of Business, University of Miami, Florida (August 1980).
22. E. Baker, *An algorithm for vehicle routing with time window constraints.* Management Science Department Report, University of Miami (1981).
23. E. Baker, L. Bodin, W. Finnegan and R. Ponder, Efficient heuristic solutions to an airline crew scheduling problem. *AIIE Trans.* **11**(2), 79–85 (1979).
24. E. Baker, L. Bodin and M. Fisher, *The development and implementation of a heuristic set covering based system for air crew scheduling.* Working Paper #80-015, University of Maryland (December 1980).
25. E. Baker and M. Fisher, Computational results for very large air crew scheduling problems. *Omega* **9**, 613–618 (1981).
26. F. Baker, T. Crabill and M. Magazine, An optimal procedure for allocating manpower with cyclic requirements. *AIIE Trans.* **5**, 119–126 (1973).
27. K. Baker, Scheduling a full-time workforce to meet cyclic staffing requirements. *Management Sci.* **20**, 1561–1568 (1974).
28. E. Balas and N. Christofides, A new penalty method for the travelling salesman problem. Presented at the *Ninth Math. Prog. Symp.*, Budapest (1976).
29. E. Balas and N. Christofides, A restricted Lagrangean approach to the travelling salesman problem. *Math. Prog.* **21**, 19–46 (1981).
30. E. Balas and A. Ho, "Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study," *Mathematical Programming Study,* **12**, 37–60 (1980).
31. E. Balas and M. Padberg, Set partitioning: a survey. *SIAM Rev.* **18**(4), 710–760 (October 1976).
32. M. Balinski and R. Quandt, On an integer program for a delivery problem. *Op. Res.* **12**, 300–304 (1964).
33. M. Ball, *A comparison of relaxations and heuristics for certain crew and vehicle scheduling problems.* Presented at National ORSA/TIMS Meeting, Washington, D.C. (1980).
34. M. Ball, A. Assad, L. Bodin, B. Golden and F. Spielberg, *Garage Location for an Urban Mass Transit System,* Management Science & Statistics Working Paper No. 81-039, University of Maryland at College Park (1981).
35. M. Ball, L. Bodin and R. Dial, A matching based heuristic for scheduling mass transit crews and vehicles. *Transportation Science,* **17**(1), 4–31 (1983).
36. M. Ball, L. Bodin and R. Dial, Experimentation with a computerized system for scheduling mass transit vehicles and crews. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 313–336. North Holland, Amsterdam (1981).
37. M. Ball, L. Bodin and R. Dial, Scheduling of drivers for mass transit systems using interactive optimization. *World Conf. on Transport Res.*, London, England (April 1980).

38. M. Ball, B. Golden, A. Assad and L. Bodin, Planning for truck fleet size in the presence of a common carrier option. *Decision Sciences*, **14**(1), 103–120 (1983).
39. M. Ball and M. Magazine, The design and analysis of heuristics. *Networks* **11**, 215–219 (1981).
40. L. Barachet, Graphical solution of the traveling-salesman problem. *Ops Res.* **5**, 841–845 (1957).
41. R. Barr, F. Glover and D. Klingman, An improved version of the out-of-kilter method and a comparative study of computer codes. *Math. Prog.* **7**(1), 60–87 (1974).
42. R. Barr, F. Glover and D. Klingman, The alternating basis algorithm for assignment problems. *Math. Prog.* **13**, 1–13 (1977).
43. R. Barr, F. Glover and D. Klingman, The generalized alternating path algorithm for transportation problems. *Eur. J. Ops Res.* **2**, 137–144 (1978).
44. J. Bartholdi, A guaranteed accuracy round-off algorithm for cyclic scheduling and set covering. *Ops Res.* **29**, 501–510 (1981).
45. J. Bartholdi, J. Orlin and H. D. Ratliff, Cyclic scheduling via integer programs with circular ones. *Ops Res.* **28**, 1074–1085 (1980).
46. T. Bartlett, An algorithm for the minimum number of transport units to maintain a fixed schedule. *Naval Res. Logistics Quart.* **4**, 139–149 (1957).
47. T. Bartlett and A. Charnes, Cyclic scheduling and combinatorial topology: assignment of routing and motive power to meet scheduling and maintenance requirements: Part II. Generalizations and analysis. *Naval Res. Logistics Quart.* **4**, 207–220 (1957).
48. R. Barton and R. Gumaer, The optimum routing for an air cargo carrier's mixed fleet. *Transportation—A Service*, pp. 549–561. New York Academy of Science, New York (1968).
49. M. Bazaraa and J. Goode, The traveling salesman: a duality approach. *Math. Prog.* **13**, 221–237 (1977).
50. J. Beardwood, H. Halton and J. Hammersley, The shortest path through many points. *Proc. Cam. Phil. Soc.* (*Math. and Phys. Sci.*) **55**, 299–327 (1959).
51. J. Beasley, Adapting the savings algorithm for varying inter-customer travel times. *Omega* **9**, 658–659 (1981).
52. M. Beckman, G. McGuire and C. Winsten, *Studies in the Economics of Transportation.* Yale University Press, New Haven, Connecticut (1955).
53. R. Bellman, *Dynamic Programming.* Princeton University Press, Princeton, New Jersey (1957).
54. R. Bellman, On a routing problem. *Quart. Appl. Math.* **16**, 87–90, (1958).
55. R. Bellman, Dynamic programming treatment of the traveling salesman problem. *J. ACM* **9**, 61–63 (1962).
56. R. Bellman and K. Cooke, The Konigsberg Bridges problem generalized. *J. Math. Anal. Appl.* **25**, 1–7 (1969).
57. M. Bellmore, Dynamic programming treatment of the travelling salesman problem. *J. ACM* **9**, 61–63 (1962).
58. M. Bellmore, G. Bennington and S. Lubore, A multivehicle tanker scheduling problem. *Transportation Sci.* **5**(1), 36–74 (February 1971).
59. A. Bellmore and S. Hong, Transformation of multi-salesman problem to the standard traveling salesman problem. *J. ACM* **21**, 500–504 (1974).
60. M. Bellmore and J. Malone, Pathology of traveling-salesman subtour-elimination algorithms. *Ops Res.* **19**, 278–307 (1971).
61. M. Bellmore and G. Nemhauser, The traveling salesman problem: a survey. *Ops Res.* **16**, 538–558 (1974).
62. E. Beltrami, *Models for Public Systems Analysis.* Academic Press, New York (1977).
63. E. Beltrami and L. Bodin, Networks and vehicle routing for municipal waste collection. *Networks* **4**, 65–94 (1974).
64. J. Benders, Partitioning procedures for solving mixed variable programming problems. *Num. Math.* **4** (1962).
65. B. Bennett, *Optimization of bus crew rosters: an application of combinatorial mathematics.* Ph.D. Thesis, Department of Mathematics, University of Adelaide, Adelaide, Australia (May 1967).
66. B. Bennett and D. Gazis, School bus routing by computer. *Transportation Res.* **6**(4), 317–325 (December 1972).
67. B. Bennett and R. Potts, Rotating roster for a transit system. *Transportation Sci.* **2**(1), 14–34 (February 1968).
68. G. Bennington and S. Lubore, Resource allocation for transportation. *Naval Res. Logistics Quart.* **17**(4), 471–484 (December 1970).
69. G. Bennington and K. Rebibo, Overview of RUCUS vehicle scheduling program (BLOCKS). *Preprints: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by D. Bergmann and L. Bodin) (1975).
70. J. Bentley and J. Saxe, An analysis of two heuristics for the Euclidean traveling salesman problem. *Proc. of the 18th Annual Allerton Conference on Communication, Control, and Computing*, pp. 41–49 (1980).
71. X. Berenguer, A characterization of linear admissible transformations for the *M*-traveling salesman problem. *European J. Op. Res.* **3**(3), 232–238 (1979).
72. C. Berge and A. Ghouila-Houri, *Programming, Games and Transportation Networks.* Methuen, London (1965).
73. D. Bergmann, Minimal Cost allocations of bus driving assignments between split runs and trippers. *Preprints: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by D. Bergmann and L. Bodin). Chicago, Illinois (1975).
74. D. Bergmann and L. Bodin (Eds.), *Preprints: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services*, Chicago, Ill. (1975).
75. W. Biles and J. Bradford, *A heuristic approach to vehicle scheduling with due-date constraints.* Presented at the 1975 Chicago ORSA meeting (May 1975).
76. J.-Y. Blais and J. Rousseau, *HASTUS: an evaluation model for drivers union negotiation in transit companies.* Publication 163, Centre de Recherche sur les Transports, Universite de Montreal (1980), forthcoming in *INFOR.*
77. J.-Y. Blais and J. Rousseau, *HASTUS for Bus operator scheduling: the man–machine interface.* University of Montreal Transportation Research Center Publication No. 231 (1981).
78. L. Bodin, *Specifications for a subscriber dial-a-ride system.* Unpublished report (1980).
79. L. Bodin, A taxonomic structure for vehicle routing and scheduling problems. *Comput. Urban Soc.* **1**, 11–29 (1975).
80. L. Bodin, A transit operating cost model based on direct systems characteristics. *Transportation Res. Record* No. 654, 28–30 (1977).
81. L. Bodin, Towards a general model for manpower scheduling—Parts 1 and 2. *J. Urban Anal.* 210–245 (January 1973).
82. L. Bodin and L. Berman, Routing and scheduling of school buses by computer. *Transportation Sci.* **13**(2), 113–129 (1979).

83. L. Bodin and R. Dial, Hierarchical procedures for determining vehicle and crew requirements for mass transit systems. *Transportation Res. Record*, **746**, 58–64 (1980).
84. L. Bodin and A. Friedman, Scheduling of committees for the New York State Assembly. *Proc. ACM Urban Symp.* New York City (October 1971).
85. L. Bodin and B. Golden, Classification in vehicle routing and scheduling. *Networks* **11**(2), 97–108 (1981).
86. L. Bodin, B. Golden, A. Schuster and W. Romig, A model for the blocking of trains. *Transportation Res.* **14B**, 115–120 (1980).
87. L. Bodin and S. Kursh, A detailed description of a street sweeper routing and scheduling system. *Comput. Ops Res.* **6**, 181–198 (1979).
88. L. Bodin and S. Kursh, A computer-assisted system for the routing and scheduling of street sweepers. *Ops Res.* **26**(4), 525–537 (1978).
89. L. Bodin, A. Kydes and D. Rosenfield, Approximation Techniques for automated Manpower Scheduling, *Preprints: Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by D. Bergmann and L. Bodin). Chicago, Illinois (1975).
90. L. Bodin and D. Rosenfield, *Estimation of the operating cost of mass transit systems.* Report NO. WAHCUPS-UMTA-1-76, College of Urban and Policy Sciences, State University of New York, Stony Brook, N.Y. (Available through the National Technical Information Service (NTIS)), (September 1976).
91. L. Bodin, D. Rosenfield and A. Kydes, Scheduling and estimation techniques for transportation planning. *Comput. Ops Res.* **8**, 25–38 (1981).
92. L. Bodin, D. Rosenfield and A. Kydes, UCOST. A micro approach to a transit planning problem. *J. Urban Anal.* **5**(1), 47–69 (1978).
93. L. Bodin and T. Sexton, *The subscriber dial-a-ride problem: the Baltimore Maryland Benchmark.* ORSA/TIMS Meeting, Colorado Springs (November, 1980).
94. L. Bodin and T. Sexton, *The subscriber dial-a-ride problem.* College of Business and Management, University of Maryland, College Park, Maryland, Report No. UMCP-UMTA-1-79 (1979).
95. L. Bodin and T. Sexton, *The multi-vehicle subscriber dial-a-ride problem.* Management Science & Statistics Working Paper No. 82-005, University of Maryland at College Park (1982).
96. R. Bodner, E. Cassell and P. Andros, Optimal routing of refuse collection vehicles. *J. Stationary Engng Div. ASCE*, **96**(SA4), Proc. Paper 7451, 893–904 (1970).
97. D. Bornemann, A crew planning and scheduling system. *Proc. of the Tenth AGIFORS Symp.* American Airlines, New York (November 1970).
98. J. Borret and A. Roes, Crew scheduling by computer: a test on the possibility of designing duties for a certain busline. *Computer scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 237–254. North-Holland, Amsterdam (1981).
99. M. Bourque, J. Ferland and J. Rousseau, Itineraries generator for a linear network. *INFOR* **15**(3), 393–398 (1977).
100. G. Bradley, G. Brown and G. Graves, Design and implementation of large scale primal transshipment algorithms. *Management Sci.* **24**(1), 1–34 (1977).
101. V. Brandani, G. Cataoli, G. Orsi and P. Toni, Bus scheduling program development for A.T.A.F. Florence, *Computer Scheduling of Public Transportation: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 71–84. North-Holland, Amsterdam (1981).
102. G. Brown and G. Graves, Real time dispatch of petroleum tank trucks. *Management Sci.* **27**, 19–32 (1981).
103. P. Bunt, P. Miller and R. Wolff, *Wheel trans: feasibility of computer-aided reservations, scheduling and dispatching.* The University of Toronto and York University Joint Program in Transportation (August 1980).
104. R. Burness and J. White, The traveling salesman location problem. *Transportation Sci.* **10**(4), 348–360 (November 1976).
105. P. Burger and D. Rice, *Integer programming models of transportation systems: an airline system example.* School of Industrial Administration, Report No. 133, Purdue University, Lafayette, Indiana (April 1966).
106. R. Burkard, Traveling salesman and assignment problems: a survey. *Ann. Discrete Math.* **4**, 193–215 (1979).
107. R. Burkard and U. Derigs, Assignment and matching problems: solution methods with FORTRAN programs. *Lecture Notes in Economics and Mathematical Systems*, No. 184. Springer-Verlag, Berlin (1980).
108. R. Busacker and T. Saaty, *Finite Graphs and Networks: an Introduction with Applications.* McGraw-Hill, New York (1965).
109. G. Buxey, The vehicle scheduling problem and Monte Carlo simulation. *J. Op. Res. Soc.* **30**, 563–573 (1979).
110. A. Cabot and A. Hurter, The optimal use of truck fleets. *The Cost of Trucking: Econometric Analysis*, pp. 47–105. Brown, Dubuque, Iowa (1965).
111. G. Carpaneto and P. Toth, Some new branching and bounding criteria for the asymmetric travelling salesman problem. *Management Sci.* **26**(7), 736–743 (July 1980).
112. P. Carraresi and G. Gallo, *A multilevel bottleneck assignment approach to the bus drivers' rostering problem.* University of Montreal Transportation Research Center Publication No. 180 (1980).
113. P. Carraresi, G. Gallo and J. Rousseau, *Decomposition approaches to large scale bus driver scheduling problems.* University of Montreal Transportation Research Center Publication No. 172 (1980).
114. P. Cassidy and H. Bennett, TRAMP—a multi-depot vehicle scheduling system. *Op. Res. Quart.* **23**(2), 151–163 (1972).
115. A. Ceder and H. Stern, Deficit function bus scheduling with deadheading trip insertions for fleet size reduction. *Transportation Sci.* **15**, 338–363 (1981).
116. Center for Urban Analysis, Santa Clara County, Personal Communication (1976).
117. K. Chandy and T. Lo, The capacitated minimum spanning tree. *Networks* **3**(2), 173–182 (1973).
118. L. Chapleau, J. Desrosiers, J. Ferland, G. Lapalme and J. Rousseau, *TRANSCOL: un système de transport intégré.* University of Montreal Transportation Research Center Publication No. 156 (1980).
119. L. Chapleau, J. Ferland, G. Lapalme and J. Rousseau, *A parallel insert method for the capacitated arc routing problem.* University of Montreal Transportation Research Center Publication No. 234 (1982).
120. L. Chapleau, J. Ferland and J. M. Rousseau, *Clustering for routing in dense area*, University of Montreal Transportation Research Center Publication No. 206 (1981).
121. R. Chard, An example of an integrated man-machine system for truck scheduling. *Op. Res. Quart.* **19**, 108 (1968).

122. T-Y. Cheung, Computational comparison of eight methods for the maximum network flow problem. *ACM Trans. Math. Software* **6**(1), 1–16 (1980).
123. J. Chisman, The clustered traveling salesman problem. *Comput. Ops Res.* **2**(2), 115–119 (1975).
124. Y. Choi and T. Sexton, *The single vehicle many to many routing and scheduling problem with customer-dependent objective function coefficients.* Working paper, SUNY at Stony Brook (1981).
125. V. Chvatal, Edmonds' polytopes and weakly Hamiltonian graphs. *Math. Prog.* **5**, 29–40 (1973).
126. N. Christofides, The shortest Hamiltonian chain of a graph. *SIAM J. Applied Math.* **19**, 689–696 (1970).
127. N. Christofides, Bounds for the traveling salesman problem. *Ops Res.* **20**, 1044–1056 (1972).
128. N. Christofides, The optimum traversal of a graph. *OMEGA* **1**(6), 719–732 (1973).
129. N. Christofides, *The vehicle routing problem.* Presented at *NATO Conf. on Combinatorial Optimization*, Paris (July, 1974).
130. N. Christofides, *Graph Theory: An Algorithmic Approach.* Academic Press, New York (1975).
131. N. Christofides, The vehicle routing problem, *Rev. Frans. Res. Op.* **10**, 55–70 (1976).
132. N. Christofides, *Worst-case analysis of a new heuristic for the traveling salesman problem.* Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (February 1976).
133. N. Christofides, Trees, cacti, and the traveling salesman problem. Presented at the *Miami ORSA/TIMS Meeting*, November 3–5, 1976.
134. N. Christofides, The traveling salesman problem. *Combinatorial Optimization* (Edited by N. Christofides, R. Mingozzi, P. Toth and C. Sandi), Wiley, New York (1979).
135. N. Christofides, Uses of a vehicle routing and scheduling system in strategic distribution planning. *Scand. J. Mat. Admin.* **7**(2), 39–55 (1981).
136. N. Christofides and J. Beasley, *The period routing problem.* Department of Management Science, Imperial College, England (1979).
137. N. Christofides and S. Eilon, Expected distances in distribution problems. *Op. Res. Quart.* **20**, 437–443 (1969).
138. N. Christofides and S. Eilon, An algorithm for the vehicle dispatching problem. *Op. Res. Quart.* **20**, 309–318 (1969).
139. N. Christofides and S. Eilon, Algorithms for large-scale traveling salesman problems. *Op. Res. Quart.* **23**, 511–518 (1972).
140. N. Christofides, A shortest path algorithm for generalized weighted matchings in graphs. Presented at *Fall ORSA/TIMS Meeting*, Houston (1981).
141. N. Christofides, A. Mingozzi and P. Toth, The vehicle routing problem. *Combinatorial Optimization*, Chap. 11. Wiley, New York (1979).
142. N. Christofides, A. Mingozzi and P. Toth, Exact algorithms for the TSP with additional constraints. *Euro-IV Cong.*, Cambridge, England (July 1980).
143. N. Christofides, A. Mingozzi and P. Toth, Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Math. Prog.* **20**, 255–282 (1981).
144. N. Christofides, A. Mingozzi and P. Toth, State space relaxation procedures for the computation of bounds to routing problems. *Networks* **11**(2), 145–164 (1981).
145. G. Clarke and J. Wright, Scheduling of vehicles from a central depot to a number of delivery points. *Ops. Res.* **12**, 568–581 (1964).
146. J. Cloonan, A heuristic approach to some sales territory problems. *Proc. Fourth Int. Conf. Ops Res.* (Edited by J. D. C. Little), pp. 81–84. M.I.T., Cambridge, Mass. (1966).
147. H. Cochran, *Optimization of a carrier routing problem*, unpublished M.S. Thesis, Industrial Engineering Dept., Kansas State University (1967).
148. R. Conway, W. Maxwell and L. Miller, *Theory of Scheduling.* Addison-Wesley, Reading, Mass. (1967).
149. T. Cook and R. Russell, A simulation and statistical analysis of stochastic vehicle routing with timing constraints. *Decision Sci.* **9**, 673–687 (1978).
150. G. Cornuejols, M. Fisher and G. Nemhauser, Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Sci.* **23**, 789–810 (1977).
151. G. Cornuejols and G. Nemhauser, Tight bounds on Christofides' traveling salesman heuristic. *Math. Prog.* **14**, 116–121 (1978).
152. G. Cornuejols and W. Pulleyblank, *The travelling salesman polytope and {0, 2}-matchings.* Management Science Research Report No. 470, Graduate School of Industrial Administration, Carnegie-Mellon University (February 1981).
153. T. Crainic, *Le probleme des horaires d'equipage d'une compagnie d'aviation.* University of Montreal Transportation Research Center Publication No. 122 (1978).
154. G. Croes, A method for solving traveling salesman problems. *Ops Res.* **6**, 791–812 (1958).
155. H. Crowder and M. Padberg, Solving large-scale symmetric travelling salesman problems to optimality. *Management Sci.* **26**(5), 495–509 (May 1980).
156. F. Cullen, J. Jarvis and H. Ratliff, Set partitioning based heuristics for interactive routing. *Networks* **11**(2), 125–144 (1981).
157. W. Cunningham and A. Marsh, A primal algorithm for optimum matching. *Math. Prog. Study No. 8: Polyhedral Combinatorics*, 50–72 (July 1978).
158. E. Cunto, Scheduling boats to sample oil wells in Lake Maracaibo, *Ops Res.* **26**(1), 183–196 (1978).
159. M. Cutler, Efficient special case algorithms for the N-line planar travelling salesman problem," *Networks* **10**, 183–195 (1980).
160. C. Daganzo, An approximate analytic model of many-to-many demand responsive transportation systems. *Transportation Res.* **12**, 325–333 (1978).
161. G. Dantzig, *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey (1963).
162. G. Dantzig, On the shortest route through a network. *Management Sci.* **6**(2), (1960).
163. G. Dantzig, W. Blattner and M. Rao, Finding a cycle in a graph with minimum cost to time ratio with applications to a ship routing problem. *Theory of Graphs* (Edited by P. Rosenstiehl), pp. 77–84. Dunrod, Paris, and Gordon & Breach, New York (1967).
164. G. Dantzig and D. Fulkerson, Minimizing the number of tankers to meet a fixed schedule. *Naval Res. Logistics Quart.* **1**, 217–222 (1954).

165. G. Dantzig, D. Fulkerson and S. Johnson, Solution of a large-scale traveling salesman problem. *Ops Res.* **2**(4), 393–410 (1954).
166. G. Dantzig, D. Fulkerson and S. Johnson, On a linear programming, combinatorial approach to the traveling salesman problem. *Ops Res.* **7**(1), 58–66 (1959).
167. G. Dantzig and J. Ramser, The truck dispatching problem. *Management Sci.* **6**, 81–91 (1959).
168. C. Date, *An Introduction to Database Systems.* Addison-Wesley, Reading, Mass. (1981).
169. G. d'Atri, A note on heuristics for the traveling salesman problem. *Math. Prog.* **19**(1), 111–114 (1980).
170. R. D. Davis, *On the delivery problem and some related topics.* Doctoral Dissertation, Northwestern University (1968).
171. R. Davies and D. Williams, Service optimisation and route costing and associated computer programs. *Computer Scheduling of Public Transportation: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 23–34. North-Holland, Amsterdam (1981).
172. P. Davis and T. Ray, A branch-bound algorithm for the capacitated facilities location problem. *Naval Res. Logistics Quart.* **16**, 331–344 (1969).
173. E. Denardo and B. Fox, Shortest-route methods: 1. Reaching, pruning and buckets. *Ops. Res.* **27**(1), 161–186 (1979).
174. U. Derigs, A shortest augmenting path method for solving minimal perfect matching problems. *Networks* **11**, 379–390 (1981).
175. U. Derigs, *Another composite heuristic for solving Euclidean traveling salesman problems.* Management Science & Statistics Working Paper No. 81-043, University of Maryland at College Park (1981).
176. U. Derigs, *Matching code theory Part I. Combinatorial structures and the cardinality matching problem.* Management Science & Statistics Working Paper No. 81-041, University of Maryland at College Park (1981).
177. U. Derigs and G. Kazakidis, *On two methods for solving minimal perfect matching problems.* Technical Report, University of Cologne, Cologne, West Germany (May 1979).
178. U. Derigs and U. Zimmerman, An augmenting path method for solving linear bottleneck assignment problems. *Computing*, **19**, 285–295 (1978).
179. C. Derman and M. Klein, Surveillance of multicomponent systems: a stochastic traveling salesman's problem. *Naval Res. Logistics Quart.* **13**, 103–112 (1966).
180. J. Desrosiers, J. Ferland, J. Rousseau, G. Lapalme and L. Chapleau, *A school busing system.* Publication No. 164, Centre de Recherche sur Les Transports, Montreal, University of Montreal (April 1980).
181. M. Devine and F. Glover, *Computational Study of the Symmetric assignment problem.* Working Paper, School of Industrial Engineering, Univ. of Oklahoma at Norman (1972).
182. D. DeWerra, On some combinatorial problems arising in scheduling. *J. CORS* **8**, 165–175 (1970).
183. D. DeWerra, Construction of school timetables by flow methods. *INFOR* **9**(1), 12–22 (March 1971).
184. K. Dexter, Scheduling an urban railway. *Computer Scheduling of Public Transportation: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 147–182. North-Holland, Amsterdam (1981).
185. R. Dial, Algorithm 360: shortest path forest with topological ordering. *Commun. ACM* **12**, 632–633 (1969).
186. R. Dial, F. Glover, D. Karney and D. Klingman, A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. *Networks* **9**(3), 215–248 (1979).
187. R. Dickinson, W. Drynan and P. Manington, The role of the systems department and the role of operations management in introducing computer assistance to bus scheduling. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 53–60. North-Holland, Amsterdam (1981).
188. E. Dijkstra, A note on two problems in connection with graphs. *Numer. Math.* **1**, 269–271 (1959).
189. J. Dinkel, G. Kleindorfer, G. Kochenberger and S. Wong, Environmental inspection routes and the constrained traveling salesman problem. *Comput. Ops. Res.* **3**(4), 269–282 (1976).
190. J. Discenza, A more compact formulation of the symmetric multiple traveling salesman problem with fixed charges. *Networks*, **11**(1), 73–75 (Spring 1981).
191. S. Dreyfus, An appraisal of some shortest-path algorithms. *Ops Res.* **17**, 395–412 (1969).
192. G. Dulac, J. Ferland and P. Forgues, School bus routes generator in urban surroundings. *Comput. Ops. Res.* **7**, 199–213 (1980).
193. J. Edmonds, Paths, trees and flowers. *Can. J. Math.* **17**, 449–467 (1965).
194. J. Edmonds, *The Chinese Postman's Problem.* National Bureau of Standards, Unpublished paper (1965).
195. J. Edmonds, Matroid partition. *Mathematics of the Decision Sciences* (Edited by G. B. Dantzig and A. Veinott, Jr.), AMS, Providence, Rhode Island (1968).
196. J. Edmonds and E. Johnson, Matching: a well-solved class of integer linear programs. *Combinatorial structures and their Applications*, pp. 89–92. Gordon & Breach, New York (1970).
197. J. Edmonds and E. Johnson, Matching, Euler tours, and the Chinese postman. *Math. Prog.* **5**, 88–124 (1973).
198. J. Edmonds and R. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* **19**, 248–264 (1972).
199. M. Effroymson and T. Ray, A branch and bound algorithm for plant location. *Ops Res.* **14**, 361–368 (1966).
200. S. Eilon and N. Christofides, The loading problem. *Management Sci.* **17**, 259–268 (1971).
201. S. Eilon, C. Watson-Gandy and N. Christofides, *Distribution Management: Mathematical Modeling and Practical Analysis.* Hafner, New York (1971).
202. J. Elam, F. Glover and D. Klingman, A strongly convergent primal algorithm for generalized networks. *Math. Ops Res.* **4**(1), 39–59 (1979).
203. S. Elias, *A mathematical model for optimizing the assignment of man and machine in public transit "run cutting".* West Virginia University, Engineering Experiment Station, Bulletin No. 81, Morgantown, West Virginia (September 1966).
204. S. Elmaghraby, The theory of networks and management science, Part I. *Management Sci.* **17**, 1–34 (1970).
205. S. Elmaghraby, The theory of networks and management science, Part II. *Management Sci.* **17**, 54–71 (1970).
206. L. Euler, The Konigsberg bridges. *Scientific American* **189**, 66–70 (1953).
207. H. Everett, Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Ops Res.* **11**, 399–417 (1963).
208. A. Federgruen and P. Zipkin, *A combined vehicle routing and inventory allocation problem.* Research Working Paper No. 345A, Graduate School of Business, Columbia University (June 1980).

209. W. Felts, *Solution Techniques for Stationary and Time Varying Traveling Salesman Problems.* Vanderbilt University Ph.D. dissertation 1970.
210. J. Ferebee, Controlling fixed-route operations. *Ind. Engng* **6**(10), 28–31 (October 1974).
211. A. Ferguson and G. Dantzig, The allocation of aircraft to routes—an example of linear programming under uncertain demand. *Management Sci.* **3**(1), 45–73 (October 1956).
212. J. Ferland and M. Florian, A sub-optimal algorithm to solve a large scale 0–1 programming problem. presented at the *IX Int. Symp. Math. Prog.* (1979).
213. J. Ferland and S. Pelland, *A simplex-type approach to solve the engine scheduling problem.* University of Montreal Transportation Research Center Publication No. 157 (1980).
214. W. Finnegan, A network model for bidline generation. *FEC Technical Report*, Federal Express Corporation, Memphis, Tennessee (1977).
215. M. Fisher, Worst-case analysis of heuristic algorithms. *Management Sci.* **26**(1), 1–17 (January 1980).
216. M. Fisher, The Lagrangian relaxation method for solving integer programming problems. *Management Sci.* **27**, 1–12 (1981).
217. M. Fisher and R. Jaikumar, An algorithm for the space-shuttle scheduling problem. *Ops Res.* **26**(1), 166–182 (1978).
218. M. Fisher and R. Jaikumar, *A decomposition algorithm for large-scale vehicle routing.* Working Paper No. 78-11-05, Dept. of Decision Sciences, University of Pennsylvania (July 1978).
219. M. Fisher and R. Jaikumar, A generalized assignment heuristic for vehicle routing. *Networks*, **11**(2), 109–124 (1981).
220. M. Fisher and R. Jaikumar, Private Communication (1980).
221. M. Fisher, R. Jaikumar and W. Bell, The impact of advanced technologies in inventory management. *Proc. National Council of Physical Distribution Management*, pp. 645–668 (1981).
222. M. Fisher, G. Nemhauser and L. Wolsey, An analysis of approximations for finding a maximum weight Hamiltonian circuit. *Ops Res.* **27**(4), 799–809 (1979).
223. M. Fisher and J. Shapiro, Constructive duality in integer programming. *SIAM J. Appl. Math.* **27**, 31–52 (1974).
224. M. Flood, The traveling salesman problem. *Ops Res.* **4**, 61–75 (1956).
225. M. Florian, G. Guerin and G. Bushel, The engine scheduling problem in a railway network. *INFOR J.* **14**, 121–138 (1976).
226. R. Floyd, Algorithm 245: Treesort 3. *Commun. ACM* **7**(12), 701 (1964).
227. L. Ford and D. Fulkerson, *Flows in Networks.* Princeton University Press, Princeton, New Jersey (1962).
228. B. Foster and D. Ryan, An integer programming approach to the vehicle scheduling problem. *Ops Res. Quart.* **27**, 367–384 (1976).
229. B. Fox, Finding a minimal cost to time ratio circuit. *Ops Res.* **17**, 546 (1969).
230. B. Fox, Data structures and computer science techniques in operations research. *Ops Res.* **26**, 686–717 (1978).
231. K. Fox, B. Gavish and S. Graves, An n-constraint formulation of the (time dependent) traveling salesman problem. *Ops Res.* **28**(4), 1018–1021 (1980).
232. G. Frederickson, Approximation algorithms for some postman problems. *J. ACM* **26**, 538–554 (1979).
233. G. Frederickson, M. Hecht and C. Kim, Approximation algorithms for some routing problems. *SIAM J. Computing*, **7**(2), 178–193 (1978).
234. A. Frieze, Worst-case analysis of algorithms for travelling salesman problems. *Ops Res. Verfahren* **32**, 94–112 (1978).
235. A. Frieze, *An extension of Christofides heuristic to the k-person travelling salesman problem.* Working paper, Dept. of Computer Sci., and Statistics, Queen Mary College, University of London, London (November 1980).
236. A. Frieze, G. Galbiati and F. Maffioli, On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks* **12**(1), 23–39 (1982).
237. D. Fulkerson, An out-of-kilter method for minimal cost flow problems. *SIAM Journal on Applied Mathematics*, **9**, 18–27 (1961).
238. H. Gabbay, *An overview of vehicular scheduling problems.* M.I.T. Operations Research Center Technical Report, No. 103 (September 1974).
239. H. Gabow, An efficient implementation of Edmond's algorithm for maximum matching on graphs. *J. ACM* **23**(2), 221–234 (1976).
240. H. Gabow, A good algorithm for smallest spanning trees with a degree constraint. *Networks* **8**, 201–208 (1978).
241. M. Garey, R. Graham and D. Johnson, Some *NP*-complete geometric problems. *Proc. 8th SIGACT Symp. on the Theory of Computing*, pp. 10–22 (1976).
242. M. Garey and D. Johnson, *Computer and Intractibility: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco (1979).
243. R. Garfinkel, On partitioning the feasible set in a branch-and-bound algorithm for the asymmetric traveling salesman problem. *Ops Res* **2**(1), 340–342 (1973).
244. R. Garfinkel, Minimizing wallpaper waste, Part 1. A class of traveling salesman problems. *Ops Res.* **25**(5), 741–751 (1977).
245. R. Garfinkel and K. Gilbert, The bottleneck traveling salesman problem: algorithms and probabilistic analysis. *J. ACM* **25**(3), 435–448 (1978).
246. R. Garfinkel and G. Nemhauser, *Integer Programming.* Wiley, New York (1972).
247. R. Garfinkel and G. Nemhauser, Optimal set covering: a survey. *Perspectives on Optimization* (Edited by A. Geoffrion). Addison–Wesley, Reading, Mass. (1972).
248. R. Garfinkel and G. Nemhauser, The set partitioning problem: set covering with equality constraints. *Ops Res.* **17**, 848–856 (1969).
249. N. Gartner, B. Golden and R. Wong, Modelling and optimization for transportation systems planning and operations. *Proc. Int. Symp. on Large Engineering Systems.* Winnipeg, Canada (August 1976).
250. W. Garvin, H. Crandall, J. John and R. Spellman, Applications of vehicle routing in the oil industry. *Management Sci.* **3**, 407–430 (1957).
251. T. Gaskell, Bases for vehicle fleet scheduling. *Op. Res. Quart.* **18**, 281–295 (1967).
252. W. Gavin and A. Roark, *WMATA bus operating cost model.* Memorandum Report H20, Subtask 2.1.4, Transit Technical Studies, Wilbur Smith and Associates, Washington, D.C. (1974).
253. B. Gavish, A note on the formulation of the M-salesman traveling salesman problem, *Management Sci.* **22**, 704–705 (1976).

254. B. Gavish, On obtaining the 'best' multipliers for a Lagrangean relaxation for integer programming. *Comput. Ops Res.* **5**, 55–71 (1978).
255. B. Gavish, *Formulations and algorithms for the capacitated minimal directed tree problem.* Working Paper, Graduate School of Management, University of Rochester (1980).
256. B. Gavish, Topological design of centralized computer networks-formulations and algorithms. *Networks* (to appear).
257. B. Gavish and S. Graves, *The travelling salesman problem and related problems.* Working Paper, Graduate School of Management, University of Rochester (May 1978).
258. B. Gavish and P. Schweitzer, An algorithm for combining truck trips. *Transportation Sci.* **8**, 13–23 (1974).
259. B. Gavish, P. Schweitzer and E. Shlifer, Assigning buses to schedules in a metropolitan area. *Comput. Ops Res.* **5**, 129–138 (1978).
260. B. Gavish and E. Shlifer, An Approach for solving a class of transportation scheduling problems. *European J. Op. Res.* **3**, 122–134 (1978).
261. B. Gavish and K. Srikanth, *Mathematical formulations for the dial-a-ride problem.* Graduate School of Management, University of Rochester (March 1979).
262. B. Gavish and K. Srikanth, $0(N^2)$ *Algorithms for sensitivity analysis of minimal spanning trees and related subgraphs.* Working Paper, Graduate School of Management, University of Rochester (1979).
263. B. Gavish and K. Srikanth, *An optimal method for the multiple travelling salesman problem.* University of Rochester, Graduate School of Management, Report No. 8027, Rochester, N.Y., (November 1980).
264. D. Gensch, An industrial application of the traveling salesman's subtour problem. *AIIE Trans.* **10**(4), 362–370 (1978).
265. A. Geoffrion, Lagrangian relaxation for integer programming. *Math. Prog. Study* **2**, 82–114 (1974).
266. A. Geoffrion, A guide to computer-assisted methods for distribution system planning. *Sloan Management Rev.* **16**(2), (1975).
267. A. Geoffrion, Making better use of optimization capability in distribution systems planning. *AIIE Trans.* **11**(2), 96–108 (1979).
268. A. Geoffrion and G. Graves, Multicommodity distribution system design by Benders decomposition. *Management Sci.* **20**, 822–844 (1974).
269. J. Gerbracht, A new algorithm for very large crew pairing problems. Presented at the *18th AGIFORS Symposium*, Vancouver, Canada, 1978.
270. I. Gertsbach and Y. Gurevich, Constructing an optimal fleet for a transportation schedule. *Transportation Sci.* **11**, 20–36 (1977).
271. I. Gertsbach and H. Stern, Minimal resources for fixed and variable job schedules. *Ops Res.* **26**, 68–85 (1978).
272. F. Gheysens, *A more compact transformation of the symmetric multiple traveling salesman problem with fixed charges.* Management Science & Statistics Working Paper No. 82-015, University of Maryland at College Park (1982).
273. B. Gillett and J. Johnson, Multi-terminal vehicle-dispatch algorithm. *Omega* **4**, 711–718 (1976).
274. B. Gillett and L. Miller, A heuristic algorithm for the vehicle dispatch problem. *Ops Res.* **22**, 340–349 (1974).
275. W. Gilman and Company and A. Voorhees and Associates, *Revenues and operating costs.* Prepared for Washington Metropolitan Transit Authority (1971).
276. P. Gilmore and R. Gomory, Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Ops Res.* **12**(5), 655–679 (September 1964).
277. J. Gilsinn and C. Witzgall, *A performance comparison of labeling algorithms for calculating shortest path trees.* NBS Technical Note 777, National Bureau of Standards, Washington, DC (1973).
278. F. Glover, *Finding an optimal edge-covering tour of a connected graph.* ORC 67-13, Operations Research Center, University of California, Berkeley, California (1967).
279. F. Glover, *Minimum complete matchings.* ORC 67-15, Operations Research Center, University of California, Berkeley (1967).
280. F. Glover, D. Karney and D. Klingman, Implementation and computational comparisons of primal, dual, and primal–dual computer codes for minimum cost network flow problems. *Networks* **4**(3), 191–212 (1974).
281. F. Glover and D. Klingman, A note on computational simplifications in solving generalized transportation problems, *Transportation Sci.* **7**(4), 351–361 (1973).
282. F. Glover and D. Klingman, Finding minimum spanning trees with fixed number of links at a node. *Combinatorial Programming: Methods and Applications* (Edited by B. Roy), pp. 191–201. Reidel, Dordrecht-Holland (1975).
283. F. Glover, D. Klingman and J. Stutz, Augmented threaded index method for network optimization. *INFOR* **12**(3), 293–298 (1974).
284. F. Glover, D. Klingman and J. Stutz, Extensions of the augmented predecessor index method to generalized network problems. *Transportation Sci.* **7**(4), 377–389 (1974).
285. D. Goeddel, An examination of the run cutting and scheduling (RUCUS) system—a case analysis. *Preprints, Workshop on Automated techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin & D. Bergmann). Chicago, Illinois (1975).
286. B. Golden, *Vehicle routing problems: formulations and heuristic solution techniques.* Technical Report No. 113, Massachusetts Institute of Technology (1975).
287. B. Golden, *Large scale vehicle routing and related combinatorial problems.* Ph.D. Thesis, Operations Research Center, MIT (June 1976).
288. B. Golden, Shortest path algorithms: a comparison. *Ops Res.* **24**(6), 1164–1168 (1976).
289. B. Golden, Evaluating a sequential vehicle routing algorithm. *AIIE Trans.* **9**, 204–208 (1977).
290. B. Golden, Statistical Estimation for combinatorial decision problems. *Proc. of AIDS 1978 Annual Convention* (Edited by R. Ebert, R. Monroe and K. Roering), Vol. 1, pp. 255–257. St. Louis (1978).
291. B. Golden, A statistical approach to the TSP. *Networks* **7**, 209–225 (1977).
292. B. Golden, Recent developments in vehicle routing. *Computers and Mathematical Programming* (Edited by W. White), National Bureau of Standards Special Publication 502, Washington, D.C. pp. 233–240. (1978).
293. B. Golden, Point estimation of a global optimum for large combinatorial problems. *Commun. Statistics* **B7**(4), 361–367 (1978).
294. B. Golden and F. Alt, Interval estimation of a global optimum for large combinatorial problems. *Naval Res. Logistics Quart.* **26**(1), 69–77 (1977).
295. B. Golden, A. Assad, L. Bodin, M. Ball and R. Dahl, *Listings and documentation for selected network optimization*

computer codes. Management Science & Statistics Working Paper No. 81-003, University of Maryland at College Park (1981).
296. B. Golden, A. Assad, L. Levy and F. Gheysens, *The fleet size and mix vehicle routing problem.* Management Science & Statistics Working Paper No. 82-020, University of Maryland at College Park (1982).
297. B. Golden and M. Ball, Shortest paths with Euclidean distances: an explanatory model. *Networks* **8**(4), 297–314 (1978).
298. B. Golden, M. Ball and L. Bodin, Current and future research directions in network optimization. *Comput. Ops Res.* **8**, 71–81 (1981).
299. B. Golden and L. Bodin, Solving large distribution-routing problems efficiently. *Proc. of* 1978 *Transportation and Logistics Educators Conference* (Edited by R. House), pp. 11–14. Chicago (1978).
300. B. Golden and L. Bodin, *Network analysis.* Working Paper No. 80-007, University of Maryland at College Park (August 1980).
301. B. Golden, L. Bodin and J. DeArmon, Solving large distribution-routing problems efficiently, Part II. *Proc. of* 1979 *Northeast AIDS Conference* (Edited by R. Cerveny and C. Lewis), pp. 71–73. Boston, Mass. (1979).
302. B. Golden, L. Bodin, T. Doyle and W. Stewart, Approximate traveling salesman algorithms. *Ops Res.* **28**(3), 694–711 (May–June 1980).
303. B. Golden, J. DeArmon and E. Baker, Computational experiments with algorithms for a class of routing problems. Management Science & Statistics Working Paper No. 81-033, University of Maryland at College Park (1981), forthcoming in *Computers & Operations Research.*
304. B. Golden, L. Levy and R. Dahl, Two generalizations of the traveling salesman problem. *Omega* **9**(4), 439–441 (1981).
305. B. Golden and T. Magnanti, Deterministic network optimization—a bibliography, *Networks* **7**, 149–183 (1977).
306. B. Golden, T. Magnanti and H. Nguyen, Implementing vehicle routing algorithms. *Networks* **7**, 113–148 (1977).
307. B. Golden and W. Stewart, Vehicle routing with probabilistic demands. *Computer Science and Statistics: Tenth Annual Symposium on the Interface* (Edited by D. Hogben and D. Fife), pp. 252–259. NBS Special Publication 503 (1978).
308. B. Golden and W. Stewart, *The empirical analysis of* TSP *heuristics.* Management Science & Statistics Working Paper No. 81-040, University of Maryland at College Park (1981).
309. B. Golden and E. Wasil, A curve-fitting experiment in estimating optimal solution values to traveling salesman problems. *Proc. of AIDS* 1981 *Annual Convention* (Edited by R. Markland and T. Rakes), Vol. 2, pp. 276–278. Boston (1981).
310. B. Golden and R. Wong, Capacitated arc routing problems. *Networks* **11**(3), 305–315 (1981).
311. B. Golden and J. Yee, A framework of probabilistic vehicle routing. *AIEE Trans.* **11**(2), 109–112 (1979).
312. D. Goren, Schneider Transport, Green Bay, Wisconsin, Personal communication (1980).
313. B. Gorenstein, S. Poley and W. White, *On the scheduling of railroad freight operations.* Technical Report No. 320-2999, IBM Philadelphia Scientific Center (January 1971).
314. R. Graham, E. Lawler, J. Lenstra and A. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.* **5**, 287–326 (1979).
315. S. Graves and T. Magnanti, An optimization-based approach to vehicle routing. *ORSA Bulletin*, Vol. 10. National Meeting Colorado Springs (November 1980).
316. P. Greenwood, *Scheduling and dispatching policies in transportation systems.* Ph.D. Thesis, Department of Industrial Engineering, Stanford University, Palo Alto, California (December 1967).
317. M. Grötschel, On the symmetric traveling salesman problem: solution of 120-city problem. *Math. Prog. Study* **12**, 61–77 (1980).
318. M. Grötschel, On the monotone symmetric travelling salesman problem: hypohamiltonian/hypotraceable graphs and facets. *Math. Ops Res.* **5**, 285–292 (1980).
319. M. Grötschel and M. Padberg, On the symmetric traveling salesman problem I: inequalities. *Math. Prog.* **16**(3), 265–280 (1979).
320. M. Grötschel and M. Padberg, On the symmetric traveling salesman problem II: Lifting theorems and facets. *Math. Prog.* **16**(3), 281–302 (1979).
321. D. Guha and J. Browne, Optimal scheduling of tours and days off. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
322. J. Gupta, A search algorithm for the traveling salesman problem. *Comput. Ops Res.* **5**(4), 243–250 (1978).
323. S. Hakimi, Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Ops Res.* **13**(3), 462–475 (1965).
324. S. Hakimi, Optimum location of switching centers and the absolute centers and medians of a graph. *Ops Res.* **12**, 450–459 (1964).
325. J. Halton and R. Terada, A fast algorithm for the Euclidean traveling salesman problem, optimal with probability one. *SIAM J. Computing* **11**(1), 28–46 (1982).
326. P. Hammer, Time-minimizing transportation problems. *Naval Res. Logistics Quart.* **16**, 345–357 (1969).
327. K. Hansen and J. Krarup, Improvements of the held-karp algorithm for the symmetric traveling salesman problem. *Math. Prog.* **7**, 87–96 (1974).
328. F. Harary, *Graph Theory.* Addison Wesley, Reading, Massachusetts (1969).
329. W. Hardgrave and G. Nemhauser, On the relation between the traveling salesman problem and the longest path problem. *Ops Res.* **10**, 647–657 (1962).
330. H. Harrison, A planning system for facilities and resources in distribution networks. *Interfaces* **9**, 6–22 (1979).
331. H. O. Hartley and M. McKay, Computerized Scheduling of Sea-Going Tankers. *Proc. NATO Conf. on the Application of Operational Research to Transport Problems.* Sandefjord, Norway (August 1972).
332. T. Hartley, A glossary of terms in bus and crew scheduling. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 353–359. North-Holland, Amsterdam (1981).
333. E. Hauer, Fleet selection for public transportation routes. *Transportation Sci.* **5**(1), 1–21 (February 1971).
334. W. Hausman and P. Gilmour, A multi-period truck delivery problem. *Transportation Res.* **1**(4), 349–357 (December 1967).
335. R. Hays, *The delivery problem.* Carnegie Institute of Technology, Management Science Research Report No. 106 (1967).

336. M. Held and R. Karp, A dynamic programming approach to sequencing problems. *J. SIAM* **10**, 196–210 (1962).
337. M. Held and R. Karp, The traveling salesman problem and minimum spanning trees. *Ops Res.* **18**, 1138–1162 (1970).
338. M. Held and R. Karp, The traveling salesman problem and minimum spanning trees Part II. *Math. Prog.* **1**, 6–25 (1971).
339. M. Held, P. Wolfe and H. Crowder, Validation of subgradient optimization. *Math. Prog.* **6**, 62–88 (1974).
340. N. Heller, *Proportional rotating schedules.* Unpublished doctoral dissertation, University of Pennsylvania (1969).
341. N. Heller and W. Stenzel, A system for computer design of proportional rotating work schedules. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
342. W. Henderson, Relationships between the scheduling of telephone operators and public transportation vehicle drivers. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
343. R. Hering, *Evaluation of some heuristic look ahead rules for multiple terminal delivery problems.* Master's Thesis, Kansas State University (1970).
344. E. Heurgon, Preparing duty rosters for bus routes by computer. *Preprints, Workshop on Automated Techniques for Scheduling of vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
345. P. Hildyard and N. Wallis, Advances in computer assisted runcutting in North America. *Computer Scheduling of Public Transport: Urban Passenger and Crew Scheduling* (Edited by A. Wren), pp. 183–192. North-Holland, Amsterdam (1981).
346. A. Hill, *Structuring and solving the bank messenger vehicle scheduling problem.* Ph.D. Thesis, Purdue University (1977).
347. A. Hill and J. McKenzie, Computer algorithm for messenger vehicle scheduling at Ohio National Bank. *Ind. Engng* **9**(12), 60 (1977).
348. A. Hill, A comparison of human decision makers and a computer algorithm for the traveling salesman problem. *Proc. 1979 AIDS Conf.* **1**, 368–370 (1979).
349. D. Hinds, RUCUS: a comparative status report and assessment. *Transit J.* 17–34 (Winter 1979).
350. J. Hinson and S. Mulherkar, *Improvements to the Clarke and Wright algorithm as applied to an airline scheduling problem.* Technical Report, Federal Express Corp. (1975).
351. A. Ho, Worst case analysis of a class of set covering heuristics. *Math. Prog.* **23**, 170–180 (1982).
352. J. Hoffstadt, Computerized vehicle and driver scheduling for the Hamburger Hochbahn Aktiengesellschaft. *Computer Scheduling of Public Transport: Urban Passenger and Crew Scheduling* (Edited by A. Wren), pp. 35–52. North-Holland, Amsterdam (1981).
353. N. Holden, *The school transportation problem.* Doctoral Thesis in Business Administration, University of Indiana (1967).
354. R. Holmes and R. Parker, A vehicle scheduling procedure based upon savings and a solution perturbation scheme. *Op. Res. Quart.* **27**(1), 83–92 (1976).
355. S. Hong and M. Padberg, A note on the symmetric multiple traveling salesman problem with fixed charges. *Ops Res.* **25**(5), 871–874 (1977).
356. D. Houck, J. Picard, M. Queyranne and R. Vemuganti, The travelling salesman problem and shortest n-paths. *Opsearch* **17**, 93–109 (1980).
357. J. Hudson, D. Grossman and D. Marks, *Analysis models for solid waste collection.* M.I.T. Civil Engineering Report (September 1973).
358. W. Hyman and L. Gordon, Commercial airline scheduling technique. *Transportation Res.* **2**(1), 23–30 (March 1968).
359. A. Isaac and E. Turban, Some comments on the traveling salesman problem. *Ops Res.* **17**(3), 543–546 (1969).
360. D. Isleb, *Integer linear programming as an aid in solving the airline crew schedule planning problem.* M.S. Thesis and Transportation Center Research Report, Northwestern University, Evanston, Ill. (June 1970).
361. J. Jachnik, Attendance and rostering system. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Editing by A. Wren), pp. 337–344 North-Holland, Amsterdam (1981).
362. S. Jacobsen and O. Madsen, *A comparative study of methods for solving a real life two-level distribution problem,* presented at the Third European Congress on Operational Research, Amsterdam, Netherlands (1979).
363. S. Jacobsen and O. Madsen, *On the location of transfer points in a two-level newspaper delivery system—a case study.* presented at the International Symposium on Location Decisions, Banff, Alberta, Canada (1978).
364. J. Jaw, A. Odoni, H. Psaraftis and N. Wilson, *A heuristic algorithm for the multi vehicle many to many advanced request dial-a-ride problem, Version* 2, presented at the ORSA/TIMS Meeting, Houston, Texas (1981).
365. R. Jenkins, An automated technique for scheduling motormen and conductors for the New York City Subways. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann) Chicago, Illinois (April 1975).
366. R. Jonker, R. Kaas and T. Volgenant, Data-dependent bounds for heuristics to find a minimum weight Hamiltonian circuit, *Ops Res.* **28**, 1219–1222 (1980).
367. D. Johnson, Approximate Algorithms for Combinatorial Problems, *J. Comput. System Sci.* **9**, 256–278 (1974).
368. E. Johnson, Networks and basic solutions. *Ops Res.* **14**, 619–623 (1966).
369. R. Jonker, G. DeLeve, J. Van Der Velde and A. Volgenant, Bounding symmetric traveling salesman problems with an asymmetric assignment problem. *Ops Res.* **28**, 623–627 (1980).
370. P. Kanellakis and C. Papadimitriou, Local search for the asymmetric traveling salesman problem. *Ops Res.* **28**(5), 1086–1099 (1980).
371. E. Kao, A preference order dynamic program for a stochastic traveling salesman problem. *Ops Res.* **26**(6), 1033–1045 (1978).
372. L. Karg and G. Thompson, A heuristic approach to solving traveling salesman problems. *Management Sci.* **10**(2), 225–248 (1964).
373. R. Karp A patching algorithm for the nonsymmetric traveling salesman problem. *SIAM J. Comp.* **8**, 561–573 (1979).
374. R. Karp, On the computational complexity of combinatorial problems. *Networks* **5**, 45–68 (1975).
375. R. Karp, The probabilistic analysis of some combinatorial search algorithms, *Algorithms and Complexity* (Edited by J. Traub,), pp. 1–19 Academic Press, New York (1976).

376. R. Karp, Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. *Math. Ops Res.* **2**, 209–224 (1977).
377. R. Karp, Reducibility among combinatorial problems. *Complexity of Computer Computations* (Edited by R. Miller and J. Thatcher) pp. 85–104. Plenum Press, New York (1972).
378. G. Kazakidis, *Verfahren zur Losung minimaler perfekter Matching-probleme mittels kurzester erweiternder Pfade.* Diploma Thesis, Mathematisches Institut der Universitat zu Koln (1979).
379. A Kearney, *Improving productivity in physical distribution.* Report undertaken for CPDM, London (1980).
380. I. Keaveny and S. Burbeck, Automating trip scheduling and optimal vehicle assignments. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, (Edited by A. Wren) pp. 125–146 North-Holland, Amsterdam (1981).
381. J. Kelly and J. Browne, Practical considerations in developing effective manpower and equipment schedules. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann) Chicago, Illinois (April 1975).
382. W. Kelton and A. Law, A mean-time comparison of algorithms for the all-pairs shortest path problem with arbitrary arc lengths. *Networks* **8**, 97–106 (1978).
383. J. Kennington and R. Helgason, *Algorithms for Network Programming.* Wiley, New York (1980).
384. A. Kershenbaum, Computing capacitated minimal spanning trees efficiently. *Networks* **4**(4), 299–310 (1974).
385. A. Kershenbaum and R. Van Slyke, Computing minimum spanning trees efficiently. *Proc. ACM Ann. Conf.*, pp. 518–527. Boston, Mass. (1972).
386. A. Kershenbaum and R. Van Slyke, An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. J.* **49**(2), 291–308 (February 1970).
387. C. Kim, *A minimal spanning tree and approximate tours for a traveling salesman.* Comp. Sci. Tech. Report, University of Maryland (1975).
388. R. Kirby and J. MacDonald, The savings method for vehicle scheduling. *Op. Res. Quart.* **24**(2), 305 (June 1973).
389. S. Kirkpatrick, C. Gelatt, Jr. and M. Vecchi, *Optimization by simulated annealing.* IBM Thomas J. Watson Research Center Research Report, Yorktown Heights, New York (1982).
390. V. Klee, Combinatorial optimization: what is the state of the art? *Math. Ops Res.* **5**, 1–26 (1980).
391. J. Klincewicz, *The Tyagi algorithm for truck dispatching.* UROP Final Project Report, MIT (1975).
392. K. Knight and J. Hafer, Vehicle scheduling with timed and connected calls: a case study. *Op. Res. Quart.* **19**(3), 299 (*September* 1968).
393. K. Knowles, *The use of a heuristic tree search algorithm for vehicle routing and scheduling.* Lecture at OR Conference, Exeter, England (1967).
394. D. Knuth, *The Art of Computer Programming*, Vol. 1. Addison-Wesley, Reading, Mass. (1973).
395. D. Knuth, *The Art of Computer Programming. Sorting and Searching*, Vol. 3. Addison-Wesley, Reading, Mass. (1973).
396. S. Koljonen and M. Tamminen, *Bus crew scheduling at Helsinki City Transport.* presented at the Nordic Operations Analysis Conference (1977).
397. H. Kregeloh and M. Mojsilovic, Automated formation of staff schedules and duty rosters. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
398. P. Krolak, *The method of ellipses (MOE): A Vehicle selection heuristic for the GRASP system.* DOT/TSC Report, U.S. Dept. of Transportation, Transportation Systems Center, Cambridge, Mass. (1980).
399. P. Krolak, *'Taxi', a realistic advance reservation dial-a-ride algorithms.* U.S. Department of Transportation, Transportation Systems Center, Cambridge, Mass. (1981).
400. P. Krolak, Three scheduling heuristics for 24 hour advance reservation paratransit systems. *Proc. AIDS National Conf.* (1981).
401. P. Krolak, W. Felts and G. Marble, *Efficient heuristics for solving large traveling salesman problems.* Presented at the 7th Mathematical Symposium, Hague, Netherlands (1970).
402. P. Krolak, W. Felts and G. Marble, A man-machine approach toward solving the traveling saleman problem. *Commun. ACM*, **14**, 327–334 (1971).
403. P. Krolak, W. Felts and J. Nelson, A man–machine approach toward solving the generalized truck dispatching problem. *Transportation Sci.* **6**, 149–170 (1972).
404. P. Krolak and J. Nelson, A family of truck load clustering heuristics for solving vehicle scheduling problems. Working Paper, Vanderbilt University (1978).
405. M. Krone, *Heuristic programming applied to scheduling problems.* unpublished doctoral dissertation, Princeton University (1970).
406. J. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc Am. Math. Soc.* **2**, 48–50 (1956).
407. A. Kuehn and M. Hamburger, A heuristic program for locating warehouses. *Management Sci.* **9**, 643–666 (1963).
408. H. Kuhn, The traveling salesman problem. *The Proc. Sixth Symposium in Applied Mathematics of the American Mathematical Society.* McGraw-Hill, New York (to appear).
409. S. Kursh, *Implementing operations research: a strategy based on case studies.* Ph.D. Thesis, George Washington University (December 1973).
410. S. Ladany and M. Hersh, Non-stop vs one stop flights. *Transportation Res.* **11**(3), 155–159 (1977).
411. B. Lalonde and P. Zinszer, *Customer service: meaning and measurement.* Presented at NCPDM Conference, Chicago (1976).
412. T. Lam, Comments on a heuristic algorithm for the multiple terminal delivery problem. *Transportation Sci.* **4**(4), 403 (November 1970).
413. R. Landis, A perspective on automated bus operator scheduling: five years experience in Portland, Oregon. *Computer Scheduling of Public Transport Urban Passenger Vehicle and crew Scheduling* (Edited by A. Wren), pp. 61–70. North Holland, Amsterdam (1981).
414. G. Laporte and Y. Nobert, A cutting planes algorithm for the *m*-saleman problem. *J. Op. Res. Soc.* **31**, 1017–1023 (1980).
415. G. Laporte and Y. Nobert, *Subtour elimination algorithms for the symmetrical traveling salesman problem and its variants.* University of Montreal Transportation Research Center Publication No. 161 (1980).

416. G. Laporte and Y. Nobert, An exact algorithm for minimizing routing and operating costs in depot location. *European J. Op. Res.* **6**(2), 224–226 (1981).
417. G. Laporte, Y. Nobert and H. Mercure, *The multi-depot traveling salesman problem.* University of Montreal Transportation Research Center Publication No. 182 (1980).
418. G. Laporte, Y. Nobert and P. Pelletier, *Generalized traveling salesman problem through* n *sets of nodes: an integer programming approach.* University of Montreal Transportation Research Center Publication No. 197 (1980).
419. R. Larson, A dynamic programming approach to airline scheduling. *Proc. Fifth AGIFORS Symp. Operations Research Division.* pp. 137–147. American Airlines, New York, (October 1965).
420. R. Larson and V. Li, Finding minimum rectilinear distance paths in the presence of barriers. *Networks* **11**(3), 285–304 (1981).
421. R. Larson and A. Odoni, *Urban Operations Research.* Prentice Hall, Englewood Cliffs, New Jersey (1980).
422. L. Lasdon, *Optimization Theory for Large Systems.* Macmillan, London (1970).
423. E. Lawler, A solvable case of the traveling salesman problem. *Math. Prog.* **1**, 267–269 (1971).
424. E. Lawer, Optimal cycles in graphs and the minimal cost-to-time ratio problem. *Proc. Conf. on Periodic Optimization.* CISM, Udine, Italy (1973).
425. E. Lawler, *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York (1976).
426. E. Lawler, Shortest path and network flow algorithms. *Ann. Discrete Math.* **4**, 251–263 (1979).
427. J. Lawrence, *Interactive vehicle dispatching: a Hybrid approach.* Ph.D. Dissertation, University of Missouri-Rolla (1980).
428. J. Lawrence and K. Steiglitz, Randomized pattern search. *IEEE Trans.Computers* (to appear).
429. C. Leddon and E. Wrathall, Scheduling empty freight car fleets on the Louisville and Nashville Railroad. *Proc. Second Int. Symp. on the Use of Cybernetics on the Railroads,* pp. 154–158 (1967).
430. T. Leibling, Graphen Theorie in Planungs—und Touren Problemen. *Lecture notes in Operations Research,* **21**, Springer-Verlag, Berlin (1970).
431. T. Leipala, On the solutions of stochastic traveling salesman problems. *European Journal of Operational Research,* **2**(4), 291–297 (1978).
432. J. Lenstra and A. Rinnooy Kan, On general routing Problems. *Networks* **6**(3), 273–280 (1976).
433. J. Lenstra and A. Rinnooy Kan, Some simple applications of the traveling salesman problem. *Op. Res. Quart.* **24**(4), 717–733 (1975).
434. J. Lenstra and A. Rinnooy Kan, Computational complexity of discrete optimization problems. *Ann. Discrete Math.* **4**, 121–140 (1979).
435. J. Lenstra and A. Rinnooy Kan, A characterization of linear admissible transformations for the *m*-traveling salesman problem: a result of Berenguer. *European J. Op. Res.* **3**(3), 250–252 (1979).
436. J. Lenstra and A. Rinnooy Kan, Complexity of vehicle routing and scheduling problems. *Networks* **11**(2), 221–227 (1981).
437. R. Lessard, J. Rousseau and D. DuPuis, Hastus I: a mathematical programming approach to the bus driver scheduling problem. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 259–268. North Holland, Amsterdam (1981).
438. R. Levary, Heuristic vehicle scheduling. *Omega* **9**, 660–663 (1981).
439. A. Levin, *Crew scheduling on a transportation network.* M.S. Thesis, Department of Aeronautics and Astronautics, M.I.T., Cambridge, Mass., (June 1966).
440. A. Levin, *Some fleet routing and scheduling problems for air transportation systems.* Flight Transportation Laboratory, FTL Report R-68-5, M.I.T., Cambridge, Mass. (January 1969).
441. A. Levin, Scheduling and fleet routing models for transportation systems. *Transportation Sci.* **5**, 232–255 (1971).
442. L. Levy, *The fleet mix vehicle routing problem.* Masters Thesis, Applied Mathematics, University of Maryland at College Park (1980).
443. H. Lewis and C. Papadimitrio, *Elements of the Theory of Computation.* Prentice-Hall, Englewood Cliffs, New Jersey (1981).
444. S. Lin, Computer solutions of the traveling salesman problem. *Bell System Tech. J.* **44**, 2245–2269 (1965).
445. S. Lin, Heuristic programming as an aid to network design, *Networks* **5**, 33–44 (1975).
446. S. Lin and B. Kernighan, An effective heuristic algorithm for the traveling salesman problem. *Op. Res.* **21**, 498–516 (1973).
447. J. Little, K. Murty, D. Sweeney and C. Karel, *An algorithm for the traveling salesman problem.* Ops. Res. **11**(6), 972–989 (1963).
448. F. Lokin, Procedures for traveling salseman problems with additional constraints. *European J. Op. Res.* **3**, 135–141 (1978).
449. J. Loo, B. O'Donald and I. Whiteman, CREATION—a heuristic program for crew allocation. *Proc. Tenth AGIFORS Symp.* Operations Research Division, American Airlines, New York (November 1970).
450. S. Lubore, *Extensions of the tanker scheduling problem.* Ph.D. Thesis, The Johns Hopkins University, Baltimore (1969).
451. J. MacDonald, Vehicle scheduling—a case study. *Op. Res. Quart.* **23**(4), 433–444 (1972).
452. R. MacDonald-Taylor, Scheduling Vehicles in a Transportation Network, Operations Research Group, Technical Memorandum No. 76, Case Institute of Technology, Cleveland, Ohio (January 1967).
453. R. Maffei, "Modern Methods for Local Delivery Route Design," *Journal of Marketing,* **29**, 13–18 (April 1965).
454. T. Magnanti, "Optimization For Sparse Systems," in *Sparse Matrix Computations,* (J. R. Bunch and D. J. Rose, eds), Academic Press, 147–176 (1976).
455. T. Magnanti, "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects," *Networks,* **11**(2), 179–214 (Summer 1981).
456. T. Magnanti and B. Golden, "Transportation Planning: Network Models and Their Implementation," Ch. 16 in *Studies in Operations Management,* (A. C. Hax, ed.), North-Holland, Amsterdam (1978).
457. T. Magnanti, J. Shapiro and M. Wagner, "Generalized Linear Programming Solves the Dual," *Management Science,* **11**, 1195–1203 (1976).
458. T. Magnanti and R. Wong, "Accelerating Benders Decomposition: Algorithmic Enhancements and Model Selection Criteria," *Ops Res.* **29**, 464–484 (1981).

459. T. Mairs, G. Wakefield, E. Johnson and K. Spielberg, "On a Production Allocation and Distribution Problem," *Management Science*, **24**, 1622–1630 (1978).
460. J. Male, *A heuristic solution to the M-postman's problem.*" Ph.D. dissertation, The Johns Hopkins University, Baltimore (1973).
461. J. Male, J. Liebman and C. Orloff, An improvement of Orloff's general routing problem. *Networks* 7(1), 89–92 (1977).
462. J. Malone, *Approximate solution of the traveling salesman problem by nonlocal, non-iterative methods.* Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, New York (1978).
463. B. Manington and A. Wren, A general computer method for bus crew scheduling. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
464. P. Manington and A. Wren, Experiences with a bus scheduling algorithm which saves vehicles. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Service* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
465. D. Marks, J. Cohen, H. Moore and R. Stricker, "Routing for Muncipal Services," *Proc. of the 6th Ann. ACM Urban Symp.* New York City (1971).
466. D. Marks and R. Stricker, Routing for public service vehicles. *J. Urban Planning Dev. Div.—ASCE* **97**, 165–178 (December 1971).
467. D. Marks and J. Liebman. Locational models: solid waste collection example. *J. Urban Planning Dev. Div.—ASCE* **97**, Proc. Paper 8027, 15–30 (April 1971).
468. J. Marquez Diez-Canedo and O. Escalante, A network solution to a general vehicle scheduling problem. *European J. Op. Res.* **1**(4), 255–261 (1977).
469. R. Marsten, An algorithm for large set partitioning problems. *Management Sci.* **20**, 774–787 (1974).
470. R. Marsten, M. Muller and C. Killion, Crew planning at flying tiger: a successful application of integer programming. *Management Sci.* **25**, 1175–1183 (1979).
471. R. Marsten and F. Shepardson, Exact solution of crew scheduling problems using the set partitioning model: recent successful applications. *Networks* **11**(2), 167–177 (Summer 1981).
472. A. Martin-Lof, *An L. P. algorithm for scheduling vehicles in a transportation network.* Operations Research Center, Research Report No. 5, MIT, Cambridge, Mass. (January 1969).
473. A. Martin-Lof, A branch-and-bound algorithm for determining the minimal fleet size of a transportation system. *Transportation Sci.* **4**(2), 159–163 (May 1970).
474. R. McBride, Controlling left and U-turns in the routing of refuse collection vehicles. *Comput. & Ops Res.* **9**(2), 145–152 (1982).
475. N. Megiddo, Combinatorial optimization with rational objective functions. *Math. Ops Res.* **4**, 414–424 (1979).
476. K. Mei-Ko, "Improvement on Graphic Programming," *Chinese Mathematics*, **1**, 278–287 (1962).
477. K. Mei-Ko, Graphic programming using odd or even points. *Chinese Math.* **1**, 273–277 (1962).
478. A. Mercer, M. Cantley and G. Rand, *Operational Distribution Research.* Taylor & Francis, London (1978).
479. G. Michael, A review of heuristic programming. *Decision Sci.* **3**, 74–100 (1972).
480. P. Miliotis, Interger programming approaches to the traveling salesman problem. *Math. Prog.* **10**, 367–378 (1976).
481. P. Miliotis, Using cutting planes to solve the symmetric traveling salesman problem. *Math. Prog.* **15**, 177–188 (1978).
482. P. Miliotis, G. Laporte and Y. Nobert, *Computational comparison of two methods for finding the shortest complete cycle or circuit in a graph.* University of Montreal Transportation Research Center Publication No. 175 (1980), forthcoming in *RAIRO.*
483. C. Miller, A. Tucker and R. Zemlin, Integer programming formulation of travelling salesman problems. *J. Ass. Comput. Mach.* **7**, 326–332 (1960).
484. H. Miller, Relationships between automated scheduling techniques for nurses and public transportation vehicle operators. *Preprints, Workshop on Automatic Techniques for Scheduling of Vehicle operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago. Illinois (April 1975).
485. J. Miller, A time-of-day model for aircraft scheduling. *Transportation Sci.* **6**, 221–246 (1972).
486. J. Minas and L. Mitten, The hub operations scheduling problem. *Ops Res.* **6**, 329–345 (1958).
487. E. Minieka, The Chinese postman problem for mixed networks. *Management Sci.* **25**, 643–648 (1979).
488. M.I.T., *Dial-a-bus: a prototype analysis.* Final Report to U.S. Department of Housing and Urban Development (September 1969).
489. M.I.T., *Scheduling algorithms for a dial-a-bus system.* Final Report to U.S. Department of Transportation (December 1970).
490. G. Mitra and A. Welsh, A computer based crew scheduling system using a mathematical programming approach. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 281–296. North-Holland, Amsterdam (1981).
491. The Mitre Corporation, *Vehicle scheduling and driver run cutting (RUCUS) package user documentation.* MTR-6678, McLean, Virginia (May 1974).
492. R. Mole, A survey of local delivery vehicle routing methodology. *J. Op. Res. Soc.* **30**, 245–252 (1979).
493. R. Mole and S. Jameson, A sequential route-building algorithm employing a generalized savings criterion. *Op. Res. Quart.* **27**, 503–511 (1976).
494. G. Monroe, Scheduling manpower for service operations. *Ind. Engng* 10–17 (August 1970).
495. J. Moon, The urban postman. *Omega* **10**, 334–337 (1982).
496. J. Moreland, *Scheduling of airline flight crews.* M.S. Thesis, Department of Aeronautics and Astronautics, M.I.T., Cambridge, Mass. (September 1966).
497. E. Morlok, W. Pierskalla and W. Vandersypen, Schedule planning and timetable construction for suburban railways. *Proc. Int. Conf. on Trans. Res.* pp. 757–767. Bruges, Belgium (1974).
498. J. Mulvey, Testing of large-scale network optimization program. *Math. Prog.* **15**(3), 291–314 (1978).
499. K. Murty, *The symmetric assignment problem.* ORC 67–12, Operations Research Center, University of California at Berkeley (1967).
500. G. Nemhauser, Scheduling local and express trains. *Transportation Sci.* **3**, 164–175 (1969).
501. G. Nemhauser, L. Trotter and R. Nauss, Set partitioning and chain decomposition. *Management Sci.* **20**, 1413–1423 (1974).

502. G. Nemhauser, L. Wolsey and M. Fisher, An analysis of approximations for maximizing submodular set functions—I *Math. Prog.* **14**, 265–294 (1978).
503. G. Newell, *Traffic Flow.* MIT Press, Cambridge, Mass. (1980).
504. R. Newton, *A school bus scheduling algorithm.* M.S. Thesis, State University of New York at Buffalo (1967).
505. R. Newton, *Bus routing in a multi-school system.* unpublished dissertation, State University of New York at Buffalo (1970).
506. R. Newton and W. Thomas, Design of school bus routes by computer. *Socio-Economic Planning Sci.* **3**, 75–85 (1969).
507. R. Newton and W. Thomas, *Developing a computer program for bus routing: final report.* State University of New York Research Foundation, Albany, New York (July 1970).
508. R. Newton and W. Thomas, Bus routing in a multi-school system. *Comput. & Ops Res.* **1**, 213–222 (1974).
509. H. Nguyen, *Multi-depot vehicle routing problems.* Masters Thesis, Sloan School of Management, MIT (1975).
510. T. Nicholson, A boundary method for planar traveling salesman problems. *Op. Res. Quart.* **19**, 444–452 (1968).
511. T. Nicholson, Optimization in industry, Vol. 1. *Optimization Techniques*, Chap. 10. Longman, London (1971).
512. A. Nijenhuis and H. Wilf, *Combinatorial Algorithms.* Academic Press, New York (1975).
513. R. Noonan and A. Whinston, An information system for vehicle scheduling. *Software Age* **3**(12), 8 (December 1969).
514. J. Norback and R. Love, Geometric approaches to solving the traveling salesman problem. *Management Sci.* **23**, 1208–1223 (1977).
515. J. Norback and R. Love, Heuristic for the Hamiltonian path problem in euclidean two space. *J. Op. Res. Soc.* **30**, 363–368 (1979).
516. A. Obruca, Spanning tree manipulation and the traveling salesman problem. *Comp. J.* **10**, 374–377 (1966).
517. H. Ong, Design and analysis of heuristics for some routing and packing problems. Ph.D. Thesis, University of Waterloo (1981).
518. I. Or, *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking.* Ph.D. Thesis, Dept. of Industrial Engineering and Management Sciences, Northwestern University (1976).
519. O. Ore, *Graphs and Their Uses.* Random House, New York (1963).
520. J. Orlin, *Minimizing the number of vehicles to meet a fixed periodic schedule: an application of periodic posets,* Working Paper, Sloan School of Management, MIT (July 1980).
521. J. Orlin, *Minimum convex-cost dynamic network flows.* Technical Report No. 172, Operations Research Center, MIT (March 1980).
522. C. Orloff, *Routing and scheduling a fleet of vehicles: the school bus problem.* unpublished dissertation, Cornell University, Ithaca, New York, 1973.
523. C. Orloff, A fundamental problem in vehicle routing. *Networks*, **4**, 35–64 (1974).
524. C. Orloff, Routing a fleet of M vehicles to/from a central facility. *Networks*, **4**, 147–162 (1974).
525. C. Orloff, On general routing problems: Comments. *Networks* **6**(3), 281–284 (1976).
526. C. Orloff, Route constrained fleet scheduling. *Transportation Sci.* **10**, 149–168 (1976).
527. C. Orloff and D. Caprera, Reduction and solution of large scale vehicle routing problems. *Transportation Sci.* **10**(4), 361–373 (November 1976).
528. M. Padberg and S. Hong, On the symmetric travelling salesman problem: a computational study. *Mathematical Programming Study* 12, 78–107 (1980).
529. C. Papadimitriou, The complexity of the capacitated tree problem. *Networks* **8**, 217–230 (1978).
530. C. Papadimitriou, *On the complexity of the Chinese postman problem.* Technical Report, Department of Electrical Engineering, Princeton University (August 1975).
531. C. Papadimitriou, On the complexity of edge traversing. *J. ACM* **23**(3), 544–554 (July 1976).
532. C. Papadimitriou, The Euclidean traveling salesman problem is *NP*-complete. *Theor. Comp. Sci.* 4(3), 237–244 (1977).
533. C. Papadimitriou, The adjacency relation on the traveling salesman polytope is *NP*-complete. *Math. Prog.* **14**(3), 312–324 (1978).
534. C. Papadimitriou and K. Steiglitz, The complexity of local search for the traveling salesman problem. *SIAM J. Comput.* **6**(1), 76–83 (1977).
535. C. Papadimitriou and K. Steiglitz, Some examples of difficult traveling salesman problems. *Ops Res.* **26**, 434–443 (1978).
536. C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization.* Prentice-Hall, Englewood Cliffs, New Jersey (1982).
537. C. Papadimitriou and M. Yannakakis, The complexity of restricted spanning tree problems. *J. Assoc. Comp. Mach.* **29**(2), 285–309 (1982).
538. U. Pape, Implementation and efficiency of Moore-algorithms for the shortest route problem. *Math. Prog.* **1**, 212–222 (1974).
539. M. Parker and B. Smith, Two approaches to computer crew scheduling. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling,* (Edited by A. Wren), pp. 193–222. North-Holland, Amsterdam (1981).
540. R. Parker, R. Deane and R. Holmes, On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent change over costs. *AIIE Trans.* **9**(2), 155–160 (1977).
541. R. Parker and R. Rardin, The traveling salesman problem: An update of research. submitted for publication (1981).
542. J. Picard and M. Queyranne The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Ops Res.* **26**(1), 86–110 (1978).
543. C. Piccione, A. Cherici, M. Bielli and A. LaBella, Practical aspects in automatic crew scheduling. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 223–236. North-Holland, Amsterdam (1981).
544. J. Pierce, Application of combinatorial programming algorithms for a class of all zero-one integer programming problems. *Management Sci.* **15**, 191–209 (1968).
545. J. Pierce, Direct search algorithms for truck-dispatching problems, Part I. *Transportation Res.* **3**, 1–42 (1969).
546. L. Platzman and J. Bartholdi III, *An 0 (N log N) planar traveling salesman heuristic based on spacefilling curves.* Georgia Institute of Technology PDRC Report Series 82-07 (1982).
547. M. Pollack, Some elements of the airline fleet planning problem. *Transport Res.* **11**, 301–310 (1977).
548. F. Preparata and S. Hong, Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM* **20**, 87–92 (1977).

549. H. Psaraftis, *Dynamic programming algorithms for specially structured sequencing and routing problems in transportation.* Ph.D. Thesis, M.I.T., Cambridge, Mass. (1978).
550. H. Psaraftis, A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Sci.* **2**, 130–154 (1980).
551. H. Psaraftis, *K-Interchange procedures for local search in a precedence-constrained routing problem.* MIT Working Paper OE-OR-81-01, Massachusetts Institute of Technology (1981).
552. H. Psaraftis, Analysis of an $O(N^2)$ Heuristic for the single vehicle many-to-many Euclidean dial-a-ride problem. *Transportation Res.* (to appear).
553. H. Psaraftis and G. Tharakan, *A dynamic programming approach to the dial-a-ride problem: an extension to the multi-vehicle case.* Report R79-39, Dept. of Civil Engineering, MIT (August 1979).
554. O. Raft, *Delivery planning by molar algorithms.* Ph.D. Thesis, IMSOR, Technical University of Denmark at Lyngby (1981).
555. T. Ramesh, *Traveling purchaser problem.* unpublished manuscript (1978).
556. M. Rao, A note on the multiple traveling salesman problem. *Ops Res.* **28**(3), 628–632 (1980).
557. T. Raymond, Heuristic algorithm for the traveling salesman problem. *IBM J. Res. Dev.* **13**(4), 400–407 (1969).
558. C. Revelle, D. Marks and J. Liebman, An analysis of private and public sector location models. *Management Sci. Theory* **16**, 692–707 (1970).
559. C. Revelle and R. Swain, Central facilities location. *Geographical Anal.* **2**, 30–42 (1970).
560. H. Richter, Experience with the aircraft rotation model. *Proc. Tenth AGIFORS Symp.* Operations Research Division, American Airlines, New York (November 1970).
561. R. Richardson, An optimization approach to routing aircraft. *Transportation Sci.* **10**, 52–71 (1976).
562. G. Rivard, *Construction des parcours des vehicules et des horaires des chauffeurs pour le transport des personnes handicapées.* University of Montreal Transportation Research Center Publication No. 240 (1981).
563. A. Roark and J. Payet, Scheduling of postal truck drivers. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Sciences* (Edited by L. Bodin and D. Bergmann) Chicago, Illinois (1975).
564. J. Robbins, *A program for solution of large scale vehicle routing problems.* Masters Thesis, Dept. of Industrial Engineering, Oklahoma State University (1976).
565. S. Roberts and B. Flores, An engineering approach to the traveling salesman problem. *Management Sci.* **13**(3), 269–288 (1966).
566. R. Roess, Operating cost models for urban public transportation and their use in analysis. *Transportation Res. Board* (1974).
567. R. Roess, M. Huss and C. Kivicklis, Predicting operating and maintenance costs for rail rapid transit. Technical Report, Department of Transportation, Planning and Engineering, Polytechnic Institute of New York, New York 1975.
568. D. Rosenkrantz, R. Sterns and P. Lewis, An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comp.* **6**, 563–581 (1977).
569. J. Rousseau, J. Desrosiers and J. Ferland, *A school bus scheduling system.* presented at National ORSA/TIMS Meeting, Washington, D.C. (1980).
570. J. Rubin, A technique for the solution of massive set covering problems with application to airline crew scheduling. *Transportation Sci.* **7**, 34–48 (1973).
571. R. Russell, *Efficient truck routing for industrial refuse collection.* Presented at the ORSA/TIMS Meeting San Juan, Puerto Rico (October 1974).
572. R. Russell, An effective heuristic for the M-tour traveling salesman problem with some side conditions. *Operations Res.* **25**(3), 517–524 (1977).
573. R. Russell and W. Igo, An assignment routing problem. *Networks* **9**(1), 1–17 (1979).
574. D. Ryan and B. Foster, An integer programming approach to scheduling. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 269–280. North-Holland, Amsterdam (1981).
575. J. Saha, An algorithm for bus scheduling problems. *Op. Res. Quart.* **21**, 463–474 (1970).
576. S. Sahni, General techniques for combinatorial approximation. *Op. Res.* **25**(6), 920–936 (1977).
577. H. Salkin and C. DeKluyver, The knapsack problem: a survey. *Naval Res. Logistics Quart.* **22**, 127–144 (1975).
578. F. Salzborn, Timetables for a suburban rail transit system. *Transportation Sci.* **3**(4), 297–316 (1969).
579. F. Salzborn, The minimum fleet size for a suburban railways system. *Transportation Sci.* **4**, 383–402 (1970).
580. F. Salzborn, A note on fleet routing models for transportation systems, *Transportation Science*, **6**, 335–337 (1972).
581. F. Salzborn, Minimum fleet size models for transportation systems. *Transportation and Traffic Theory* (Edited by D. Buckley), pp. 607–624. Reed, Sydney (1974).
582. J. Schmidt and R. Knight, The status of computer-aided scheduling in North America. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 17–22. North-Holland, Amsterdam (1981).
583. J. Schmidt and R. Fennessy Automating extraboard assignments and coach operator timekeeping. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 345–352. North-Holland, Amsterdam (1981).
584. A. Schnitt and L. Bush, *The feasibility study of the employment center bus service concept.* U.S. Department of Transportation Urban Mass Transportation Administration (1976).
585. L. Schrage, Formulation and structure of more complex/realistic routing and scheduling problems. *Networks* **11**(2), 229–232 (1981).
586. H. Schultz, A practical method for vehicle scheduling. *Interfaces* **9**(3), 13–19 (1979).
587. K. Schuster and D. Schur, *Heuristic routing for solid waste collection vehicles.* SW-113, U.S. Environmental Protection Agency, Cincinnati, Ohio (1974).
588. A. Scott, *Combinational Programming, Spatial Analysis and Planning,* Methuen, London (1971).
589. M. Segal, The operator-sceduling problem: a network-flow approach. *Op. Res.* **22**, 808–823 (1974).
590. T. Sexton, *The single vehicle many to many routing and scheduling problem.* Ph.D. Thesis, State University of New York at Stony Brook (1979).

591. T. Sexton and L. Bodin, *The single vehicle many to many routing and scheduling problem with desired delivery times.* Working Paper No. 80–014, University of Maryland at College Park (1980).
592. T. Sexton and L. Bodin, *The single vehicle many-to-many routing and scheduling problem with customer-dependent dwell times.* working paper, SUNY at Stony Brook (1981).
593. D. Shapiro, *Algorithms for the solution of the optimal cost travelling salesman problem.* Sc. D. Thesis, Washington University, St. Louis (1966).
594. J. Shapiro, A survey of Lagrangian techniques for discrete optimization. *Annals of Discrete Mathematics—No.. 5 Discrete Optimization* (Edited by P. Hammer, E. Johnson and B. Korte). pp. 113–118, North Holland (1979).
595. G. Sharp, "Constants for Scheduling Operators for Urban Public Transit Systems". *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services,* L. Bodin and D. Bergmann, eds, Chicago, Illinois (April 1975).
596. W. Shepardson and R. Marsten, A Lagrangian relaxation algorithm for the two duty period scheduling problem, *Management Sci.* **26**, 274–281 (1980).
597. D. Shier, Iterative methods for determining the k shortest paths in a network **6**(3), 205–229 (1976).
598. D. Shier, On Algorithms for finding the K shortest paths in a network **9**(3), 195–214 (1979).
599. E. Silver, R. Vidal and D. DeWerra, A tutorial on heuristic methods. *European J. Op. Res.* **5**, 153–162 (1980).
600. R. Simpson, *Computerized schedule construction for an airline transportation system.* Flight Transportation Laboratory, Technical Report FT-66-3, M.I.T., Cambridge, Mass. (1966).
601. R. Simpson, *Scheduling and routing models for airline systems.* Flight Transportation Laboratory, Report FTL-R68-3, M.I.T., Cambridge, Mass. (December 1969).
602. R. Simpson, A review of scheduling and routing models for airline scheduling. *Proc. Ninth AGIFORS Symp.* Operations Research Division, American Airlines, New York (October 1969).
603. R. Simpson and M. Eglise, *A method for determination of optimum vehicle size and frequency of service for a short Haul V/STOL air transport system.* Flight Transportation Laboratory, FTL-R-68-1, M.I.T., Cambridge, Mass. (May 1968).
604. C. Skiscim and J. Sanborn, *Some new computational experiments with the TSP.* Management Science & Statistics Working Paper No. 82–018, University of Maryland at College Park (1982).
605. B. Smith and A. Wren, VAMPIRES and TASC: two successfully applied bus scheduling programs. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren) pp. 97–124. North-Holland, Amsterdam (1981).
606. T. Smith, A LIFO implicit enumeration algorithm for the asymmetric traveling salesman problem using a one-arborscence relaxation. *Mathematical Programming Study* 12, 108–114 (1980).
607. T. Smith, V. Srinivasan and G. Thompson, Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems. *Ann. Discrete Math.* **1**, 495–506 (1977).
608. T. Smith and G. Thompson, A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation. *Ann. Discrete Math.* **1**, 479–493 (1977).
609. F. Soumis, J. Ferland and J. Rousseau, A model for large scale aircraft routing and scheduling problems. *Transportation Res.* **14B**, 191–201 (1980).
610. M. Spitzer, Solution to the crew scheduling problem. *AGIFORS Symp.* (October 1961).
611. M. Spitzer, The computer art of schedule-making. *Datamation,* p. 84 (April 1969).
612. K. Srikanth, *An improved algorithm for computing degree-constrained minimal spanning trees.* Working Paper, Graduate School of Management, University of Rochester (1980).
613. K. Steiglitz and P. Weiner, Some improved algorithms for computer solution to the traveling salesman problem. *Proc. of the 6th Ann. Allerton Conf. on Circuit and Systems Theory* 814–821 (1968).
614. K. Steiglitz, P. Weiner and D. Kleitman, The design of minimum cost survivable networks. *IEEE Trans. Circ. Theory* **CT-16**, 455–460 (1969).
615. D. Stein, Scheduling dial-a-ride transportation systems, *Transportation Sci.* **12**, 232–249 (1978).
616. D. Stein, *Scheduling dial-a-ride transportation system.* Ph.D. Thesis, Harvard University (1977).
617. D. Stein, An asymptotic probabilistic analysis of a routing problem. *Math. Ops Res.* **3**, 89–101 (1978).
618. H. Stern, *Optimal service policies for solid waste treatment facilities.* ORC 69-3, Operations Research Center, University of California, Berkeley (1969).
619. H. Stern, Bus and crew scheduling (note). *Transportation Res.* **14A**, 154 (1980).
620. H. Stern and A. Ceder, A deficit function approach for bus scheduling. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren), pp. 97–124. North-Holland, Amsterdam (1981).
621. H. Stern and A. Ceder, *Heuristic reductions for bus fleet scheduling with large M numbers of terminals.* CORS/ORSA/TIMSMeeting, Toronto, Canada (1981).
622. H. Stern and A. Ceder, The garage constrained-balance vehicle schedule minimum fleet size problem. *Proc. 8th Int. Symp. Transportation and Traffic Theory,* University of Toronto (1981).
623. H. Stern and M. Dror, Routing electric meter readers. *Comput. Ops Res.* **6**, 209–223 (1979).
624. H. Stern and M. Hersh, Scheduling aircraft cleaning crews. *Transportation Sci.* **14**(3), 277–291 (August 1980).
625. W. Stewart, *The delivery truck routing problem with stochastic demands.* Paper presented at the ORSA/TIMS Joint National Meeting, Miami Beach, Florida (1976).
626. W. Stewart, A computationally efficient heuristic for the traveling salesman problem. *Proc. Thirteenth Ann. Meeting of Southeastern TIMS,* pp. 75–85. Myrtle Beach, South Carolina (1977).
627. W. Stewart, *New algorithms for deterministic and stochastic vehicle routing problems.* Working Paper No. 81–009, University of Maryland at College Park (March 1981).
628. W. Stewart and B. Golden, A vehicle routing algorithm based on generalized Lagrange multipliers. *Proc. of AIDS* 1979 *Annual Convention* (Edited by L. Moore, K. Monroe, B. Taylor) pp. 108–110. Vol. 2. New Orleans (1979).
629. W. Stewart and B. Golden, *A Lagrangean Relaxation Heuristic for Vehicle Routing.* Management Science & Statistics Working Paper No. 82–013, University of Maryland at College Park (1982).
630. W. Stewart and B. Golden, *Stochastic vehicle routing: a comprehensive approach.* Management Science & Statistics Working Paper No. 82–010, University of Maryland at College Park (1982).
631. W. Stewart and B. Golden, A chance-constrained approach to the stochastic vehicle routing problem. *Proc. N.E. AIDS,* pp. 33–35 Philadelphia (1980).

632. W. Stewart and B. Golden. The subscriber bus routing problem. *Proc. IEEE Int. Conf. on Circuits and Computers*, pp. 153–156. Port Chester, New York (1980).
633. W. Stewart and B. Golden, Computing effective subscriber bus routes. *Proc. of* 1980 *SE TIMS Conference* (Edited by P. Dearing, G. Worm) pp. 170–178. Virginia Beach (1981).
634. J. Stinson, *A heuristic algorithm for obtaining an initial solution for the traveling salesman problem.* Presented at the New York ORSA/TIMS meeting (May 1978).
635. R. Stricker, *Public sector vehicle routing: the Chinese postman problem.* Master's Thesis, MIT (September 1970).
636. R. Stricker and D. Marks, *Public sector routing models.* Presented at 39th Operations Research meeting in Dallas, Texas (May 1971).
637. J. Svestka, A continuous variable representation of the traveling salesman problem. *Math. Prog.* **15**(2), 211–213 (1978).
638. J. Svestka and V. Huckfeldt, Computational experience with and *M*-salesmen traveling salesman algorithm. *Management Sci.* **19**(7), 790–799 (1973).
639. A. Swersey and W. Ballard, *Scheduling school buses.* Yale School of Organization and Management Tech. Report, Yale University (1982).
640. M. Syslo, A new solvable case of the traveling salesman problem. *Math. Prog.* **4**, 347–348 (1973).
641. B. Szpigel, Optimal train scheduling on a single track railway. *Ops. Res.* 343–352 (1972).
642. R. Tarjan, Complexity of combinatorial algorithms. *SIAM Rev.* **20**, 457–491 (1978).
643. G. Tharakan and H. Psaraftis, An exact algorithm for the exponential disutility dial-a-ride problem. Submitted to *Transportation Sci.* (1981).
644. H. Thiriez, *Airline crew scheduling: a group theoretic approach.* Flight Transportation Laboratory, Report R69-I, Department of Aeronautics and Astronautics, MIT, Cambridge, Mass. (October 1969).
645. R. Tibrewala, D. Phillipe and J. Browne, Optimal scheduling of two consecutive idle periods. *Management Sci.* **10**, 71–75 (1972).
646. F. Tillman, The multiple terminal delivery problem with probabilistic demands. *Transportation Sci.* **3**, 192–204 (1969).
647. F. Tillman The author's reply to Uebe's *Transportation Sci.* **4**(2), 232 (May 1970).
648. F. Tillman and T. Cain, An upper bounding algorithm for the single and multiple terminal delivery problem. *Management Sci.* **18**, 664–682 (1972).
649. F. Tillman and H. Cochran, A heuristic approach for solving the delivery problem. *J. Ind. Engng* **19**, 354–358 (1968).
650. F. Tillman and R. Hering, A study of a look-ahead procedure for solving the multiterminal delivery problem. *Transortation Res.* **5**, 225–229 (1971).
651. C. Toregas, R. Swain, C. Revelle and L. Berman, The location of emergency service facilities. *Ops Res.* **19**, 1363–1373 (1971).
652. G. Tracz and M. Norman, *Computerized route planning for school buses: design and implementation.* Paper presented at the 36th National Meeting of the Operations Research Society of America, Miami Beach, November 10–12 (1969).
653. G. Tracz and M. Norman, *A computerized system for school bus routing.* Ontario Institute for Studies in Education, Toronto Technical Report 181, Ontario (1970).
654. W. Turner, P. Ghare and L. Fourds, Transportation routing problem—a survey. *AIIE Trans.* **6**, 288–301 (1974).
655. M. Tyagi, A practical method for the truck dispatching problem. *J. Ops Res. Soc. Japan*, **10**, 76–92 (1968).
656. G. Uebe, Comments on Tillman's paper. *Transportation Sci.* **4**(2), 226 (May 1970).
657. E. Unwin, Bases for vehicle fleet scheduling. *Op. Res. Quart.* **19**(2), 201 (June 1968).
658. Urban Mass Transportation Administration. *UTPS Reference Manual.* Urban Mass Transportation Administration, Washington, D. C. (1975).
659. P. Van der Cruyssen and M. Rijckaert, Heuristic for the asymmetric travelling salesman problem. *J. Op. Res. Soc.* **30**, 697–701 (1978).
660. H. Vandersypen and J. Mayne, *A Transportation scheduling model.* Unpublished paper, Transportation Center Nothwestern University, Evanston, Illinois (March 1971).
661. W. Verderber, Automated pupil transportation. *Comput. Ops Res.* **1**, (1974).
662. H. Voccaro, *Alternative techniques for modeling travel distances.* Masters Thesis, Civil Engineering, Massachusetts Institute of Technology (1979).
663. V. Vuchic and G. Newell, Rapid transit interstation spacings for minimum travel time. *Transportation Sci.* **2**, 303–339 (1968).
664. W. Walker, J. Chaiken and E. Ignall, *Rand Fire Project: Fire Department Deployment Analysis.* Elsevier–North-Holland, New York (1979).
665. R. Ward, P. Durant and A. Hallman, A Problem Decomposition Approach to Scheduling the Drivers and Crews of Mass Transit Systems. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling* (Edited by A. Wren) pp. 297–312. North-Holland, Amsterdam (1981).
666. C. Watson-Gandy and L. Foulds, The vehicle scheduling problem: a survey: unpublished manuscript.
667. M. Webb, Some methods of producing approximate solutions to traveling salesman problems with hundreds or thousands of cities. *Op. Res. Quart.* **22**(1), 49–66 (1971).
668. M. Webb, Relative performance of some sequential methods of planning multiple delivery journeys. *Op. Res. Quart.* **23**(3), 361–372 (1972).
669. M. Webb, The savings method for vehicle scheduling—a reply. *Op. Res. Quart.* **24**(2), 307 (June 1973).
670. P. Weiner, S. Savage and A. Bagchi, Neighborhood search algorithms for finding optimal traveling salesman tours must be inefficient. *5th SIGACT Proc.* pp. 207–213. (1973).
671. G. White and D. Sweeney, Using multidimensional scaling to solve traveling salesman and machine scheduling problems, *Comput. Ops Res.* **7**, 177–184 (1980).
672. W. White and A. Bomberault, A network algorithm for empty freight car allocation. *IBM Systems J.* **9**, 147–169 (1969).
673. E. Wilhelm, Overview of the RUCUS package driver run cutting program (RUNS). *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
674. J. Williams, Algorithm 232: Heapsort, *Commun. ACM* **7**, 347–348 (1974).
675. W. Williamson, *Computer programs for fleet and schedule planning. Proc. Seventh AGIFORS Symp.* pp. 11–122. Operations Research Division, American Airlines, New York (October 1967).
676. N. Wilson, *CARS—computer aided routing system.* Department of Civil Engineering Research report, R67-12, MIT (1967).

677. N. Wilson, *Dynamic routing: a study of assignment algorithms.* Ph.D. Thesis, Department of Civil Engineering, MIT, Cambridge, Mass. (October 1969).
678. N. Wilson, The effect of driver schedules on dial-a-ride performance. *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
679. N. Wilson and N. Colvin, *Computer control of the Rochester dial-a-ride system.* CTS Report Number 77-22, Cambridge, Mass., MIT, Center for Transportation Studies (July 1977).
680. N. Wilson and E. Miller, *Advanced dial-a-ride algorithms research project, Phase II. Interim report.* Report R77-31 Cambridge, Mass., MIT, Dept. of Civil Engng (July 1977).
681. N. Wilson and J. Sussman, *Implementation of computer algorithms for the dial-a-bus system.* 39th National Meeting of ORSA, Dallas, Texas (May 1971).
682. N. Wilson, J. Sussman, H. Wong and T. Higonnet, *Scheduling algorithms for a dial-a-ride system.* Report USL TR-70-13, Urban Systems Laboratory, MIT, (March 1971).
683. N. Wilson, R. Weissberg and J. Hauser, *Advanced dial-a-ride algorithms research project final report,* Report R76-20, Cambridge, Mass., MIT, Dept. of Civil Engng (March 1976).
684. N. Wilson, R. Weissberg, B. Higonnet and J. Hauser, *Advanced dial-a-ride algorithms. Interm report,* Report R75-27, Cambridge, Mass., MIT, Dept. of Civil Engng (July 1975).
685. J. Wiorkowski and K. McElvain, A rapid heuristic algorithm for the approximate solution of the traveling salesman problem. *Transportation Res.* **9**, 181–185 (1975).
686. S. Wirasinghe, V. Hurdle and G. Newell, Optimal parameters for a coordinated rail and bus transit system. *Transportation Sci.* **11**, 359–374 (1977).
687. J. Wolters, Minimizing the number of aircraft for a transportation network, *European J. Op. Res.* **3**(5), 394–402 (1979).
688. J. Wong, *Computer aided scheduling for wheel-trans service.* Project No. 3323, Transit Systems Office, Systems Research and Development Branch, Toronto: Ontario Ministry of Transportation and Communications (March 1978).
689. J. Wong, *Computer scheduling for door-to-door transit service—computer program specification.* Toronto: Ontario Ministry of Transportation and Communications (March 1979).
690. R. Wong, Interger programming formulations of the traveling salesman problem. *Proc. of IEEE International Conference on Circuits and Computers,* pp. 149–152 (1980).
691. W. Worcester, Techniques for vehicle operator scheduling at the Chicago Transit Authority (CTA). *Preprints, Workshop on Automated Techniques for Scheduling of Vehicle Operators for Urban Public Transportation Services* (Edited by L. Bodin and D. Bergmann). Chicago, Illinois (April 1975).
692. A. Wren (Ed.) *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling.* North-Holland, Amsterdam (1981).
693. A. Wren, General review of the use of computers in scheduling buses and their crews. *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling,* (Edited by A. Wren) pp. 3–16. North-Holland, Amsterdam (1981).
694. A. Wren, *Applications of computers to transport scheduling in the United Kingdom.* College of Engineering, West Virginia University, Engineering Experiment Station Bulletin, 91, Morgantown (March 1969).
695. A. Wren and A. Holliday, Computer scheduling of vehicles from one or more depots to a number of delivery points. *Op. Res. Quart.* **23**, 333–344 (1972).
696. C. E. Yang, *How good is the largest angle method.* Submitted for publication.
697. J. Yee and B. Golden, A note on determining operating strategies for probabilistic vehicle routing. *Naval Res. Logistics Quart.* **27**, 159–163 (1980).
698. P. Yellow, A computational modification to the savings method of vehicle scheduling. *Op. Res. Quart.* **21**, 281–283 (1970).
699. D. Young, Scheduling a fixed schedule, common carrier passenger transportation system. *Transportation Sci.* **4**, 243–269 (1970).