FTL REPORT R84-3

SOLVING LARGE-SCALE DIAL-A-RIDE VEHICLE ROUTING AND SCHEDULING PROBLEMS

by

JANG-JEI JAW

June 1984

SOLVING LARGE-SCALE DIAL-A-RIDE VEHICLE ROUTING AND SCHEDULING PROBLEMS

by

JANG-JEI JAW

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

ABSTRACT

In this thesis we develop two heuristic algorithms for large-scale multi-vehicle advance-request version of the dial-a-ride problem. This problem is concerned with developing a set of routes for a fleet of vehicles serving customers who have to be picked up from specified origins and be delivered to specified destinations. In this thesis it is assumed that each customer has specified either a desired pick-up time or a desired delivery time and that customer requests for service as well as the number of available vehicles throughout the time period of interest are known well in advance of the time of actual vehicle dispatching.

The first heuristic approach consists of three successive and distinct steps: "grouping", "clustering" and "routing". Grouping divides customers into "time groups" on the basis of their desired pick-up and delivery times. Clustering separates customers in each time group into "clusters" and assigns vehicles to serve each cluster. Finally routing generates routes for each individual vehicle to serve every cluster in turn and for every time group.

The second algorithm, Advanced Dial-A-Ride with Time Windows (ADARTW), treats customers' desired service times as strict constraints and can guarantee prespecified standards of service quality. The service quality constraints refer to guarantees that (i) customer's ride time will not exceed a pre-specified maximum and (ii) the service time will not deviate from the most desired time by more than a pre-specified amount ("the time windows"). The algorithm builds up vehicle tours through sequential insertion of customers and uses a nonlinear objective function to guide such insertions. Variations of this basic approach are also discussed.

We have tested the two algorithms on many simulated cases using computer-generated data. Computational experience with a large-scale real-world dial-a-ride problem (2617 customers and 30 simultaneously operating vehicles) is also presented.

Thesis Supervisor : Amedeo R. Odoni

Title : Professor of Aeronautics and Astronautics
and of Civil Engineering

ACKNOWLEDGEMENTS

I wish to thank my thesis supervisor, Professor Odoni, who first introduced the topic of dial-a-ride problems to me during his class lectures and later directed my research work. His constant encouragements and support have made my years at M.I.T. more memorable. His attitude towards academic research as well as towards people has also set a model for me to follow in the years to come.

I would also like to express my gratitude to other members in my thesis committee, Professors Harilaos N. Psaraftis, Nigel H.M. Wilson and Robert W. Simpson. Professor Psaraftis has sat through many discussion periods with me and provided valuable assistance. His comments on the contents of the thesis have made it a more complete piece of work. Professor Wilson is instrumental in the development of the second algorithm in this thesis. His comments and suggestions have made every meeting with him an informative one. Professor Simpson has helped me go through difficult times when I first arrived at M.I.T. and has always made himself available when I call for help.

This research work has been supported by the Urban Mass Transportation Administration, U.S. Department of Transportation. This support is gratefully acknowledged. I also wish to acknowledge Mr. Edward G. Neigut, Paratransit Integration Program Manager, Office of Methods and Support, UMTA, for his many suggestions regarding the scenario and "ground rules" for the operating environment of the ADARTW algorithm which is described in Chapter 4. The definition of the time windows for service acceptability used in Chapter 4 was also suggested by Mr. Neigut.

I would like to dedicate this thesis to my wife Li-Giaung. Without her standing by me and taking care of our son, Darcy, all this time, my study at M.I.T. would not have been possible. Finally, I am grateful to my parents for providing me with excellent education since my childhood.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Description of Dial-A-Ride Systems

Dial-a-ride is a form of mass transit system which would serve areas of low travel demand and/or a population with special needs. Such a system is run with a fleet of vehicles operating on flexible routes without fixed schedules. Customers call the dial-a-ride agency requesting to be carried from specified points to other specified points. The agency, in turn, is responsible for dispatching this fleet of vehicles to meet such demands during a typical day of operation. The aim of the dial-a-ride system is to provide a quality of service similar to that of a taxi at a cost much lower than that of a taxi. It also can serve as an alternate means of transportation for handicapped people and senior citizens.

Such demand-responsive transportation systems have been in operation in several metropolitan and suburban areas of the U.S. (e.g. Rochester, New York and Dade County, Florida). Several cities abroad (e.g. Montreal, Canada and Tokyo, Japan) also provide similar services to their residents.

The concept of dial-a-ride service is simple; the difficult part is to operate such a system efficiently. There has been increasing concern, about the high operating costs experienced by many dial-a-ride systems. Many studies have investigated various facets of system operations Much research, in particular, has concentrated on ways to utilize efficiently

the resources and fleet of vehicles operated by dial-a-ride agencies. This thesis is devoted to the development of a set of tools that can help to operate such vehicles in an efficient manner. The focus of the work is on designing computerized algorithms which will either serve as planning aids for operators or will actually produce a (preliminary) set of vehicle routes and schedules which can be followed by the operators. In the following section, we will discuss various types of dial-a-ride problems that might arise in this context. In Section 1.3 we explain the purposes of this thesis and present an outline of this work.

1.2 Dial-A-Ride Problems

The successful implementation of a dial-a-ride system depends on the efficient utilization of its vehicles while providing satisfactory quality of service to its customers. The problem of developing efficient vehicle routes and schedules for a dial-a-ride system to satisfy customer travel demands has become known as the Dial-A-Ride Problem (DARP). Several versions of DARP exist depending on the type of service being offerred by the system. In [24], Psaraftis classified different types of dial-a-ride services as follows:

(1) Many-to-many vs many-to-one

"Many-to-one" refers to a service in which customers are picked up from distinct points but delivered to one common destination. Operation of "feeder" services, carrying passengers to a bus terminal, a train station or an airport are typical examples of a

"many-to-one" configuration. The reverse situation, "one-to-many", is clearly a similar problem. In the "many-to-many" problem customers can have distinct origins and distinct destinations.

(2) Immediate-request vs advance-request

For "immediate-request" service, customers call the dial-a-ride agency and request to be serviced as soon as possible. For "advance-request" service, customers are required to inform the agency of their intended trips before a closing time, e.g. no later than 5 p.m. the day before the intended trips. No request is accepted after this deadline. It is of course possible that a dial-a-ride system may serve a mix of these two types of customers.

The "many-to-one" dial-a-ride problem can be considered as a special case of the "many-to-many" problem and thus it can be at most as difficult as the latter. Most research efforts to date have been directed toward solving the more general many-to-many version of the problem. In this thesis all the dial-a-ride problems discussed are assumed to be many-to-many problems unless otherwise specified.

The dial-a-ride problem can be viewed as a more complicated extension of the vehicle routing problem (or truck dispatching problem) which was first formulated by Dantzig and Ramser [8] in 1959. The basic vehicle routing problem (VRP) involves a set of customers, each with a known location and a known requirement for some commodity, and a fleet of

delivery vehicles of known capacity located at a central depot. The problem is to design the delivery routes for these vehicles such that :

(a) the requirements of all the customers are met.

(b) the vehicles' capacity is not violated.

(c) the total distance for each vehicle to complete its tour does not exceed some predetermined level.

The objective of a solution can be stated in general terms as that of minimizing the total cost of delivery, namely the sum of costs associated with the fleet size and the costs of completing the delivery tours.

In the above formulation, if the fleet is a single vehicle having sufficiently large capacity, and if constraint (c) is ignored, then the vehicle routing problem becomes the well-known travelling salesman problem (TSP). When the vehicle routing problem has additional constraints on delivery times and on the sequence of demand points to be visited, a combined vehicle routing and scheduling problem (VRSP) arises. Such problems are abundant in practice and are representative of many real-world applications. The DARP can be considered as a combined vehicle routing and scheduling problem in which a customer's pick-up point must be visited before his(her) delivery point and the customer pick-up or delivery must be made within certain time intervals so that some service quality criteria are met.

Vehicle routing problems have been intensively studied during the past decade. An overview of the major advances in this area is

instrumental in our study of the dial-a-ride problem. A useful taxonomy for various vehicle routing and scheduling problems can be found in Bodin and Golden [2]. Solution procedures are also classified in [2] as following one of the approaches listed below:

1. Cluster first, route second.
2. Route first, cluster second.
3. Savings/insertion.
4. Improvement/exchange.
5. Mathematical-programming-based heuristics.
6. Interactive optimization.
7. Exact procedures.

Categories 1 through 6 in the above classification are considered to be heuristics which solve the problem approximately. Exact procedures in category 7 can solve the problem optimally by branch and bound techniques, but are viable only for very small problems. The computational complexity of vehicle routing problems has been shown by Lenstra and Rinnooy Kan [18] to be NP-hard which suggests that algorithmic possibilities for optimization methods are limited. (To our knowledge, the largest vehicle routing problem that has been solved exactly involved only 31 customers [4].) This suggests that it is unlikely to find an exact procedure which is polynomial-time bounded with respect to the input size for the dial-a-ride problem.

In view of the computational complexity involved, we have chosen to put the emphasis of this thesis on heuristics as the solution to dial-

a-ride problems. For the large-scale DARPs which we attempt to solve (30 vehicles and 2000 customers) only efficient heuristics are practical in terms of computation effort involved. In this section we examine algorithmic approaches that have been developed for vehicle routing problems and have direct applications to dial-a-ride problems. Based on the Bodin and Golden classification, we divide the various approaches for VRP into one of four generic types:

(i) tour-building heuristics,

(ii) tour-improvement heuristics,

(iii) two-phase methods, and

(iv) exact procedures.

The four types are not mutually exclusive. For example, type (iii) might use both type (i) and (ii) approaches. We now discuss each type of approach separately:

(i) Tour-building heuristics

Tour-building heuristics build up tours by adding one customer at a time to some tour until all customers have been assigned to some tour. Some measure of cost savings guides the choice of the next customer to be included in a tour. The Clarke and Wright method [5] is one example of this type of heuristic. Modifications to the basic Clarke and Wright method have been suggested by numerous authors, including Gaskell [12] and Yellow [44]. Golden et al. [14] have used computer science techniques to reduce the running time of this method.

(ii) Tour-improvement heuristics

Tour-improvement heuristics begin with a feasible vehicle route. At each step, some combination of links is exchanged for another with a reduced overall cost while maintaining feasibility of the route. This prodedure continues until no additional cost reductions are possible. All heuristics of this type are based on the Lin [19] and Lin and Kernighan [20] heuristic for the travelling salesman problem. Christofides and Eilon [3] and Russell [31] have extended this approach to the multiple vehicle routing problems. Cook and Russell [6] later applied the approach in [31] to problems with time window constraints.

(iii) Two-phase method

This category includes the sweep algorithm by Gillett and Miller [13], and those work of Christofides et al. [3] and Tyagi [39]. Usually, phase 1 involves the assigment of customers to vehicles and the establishment of initial routes. Phase 2 uses the tour-improvement heuristic to obtain tours for each vehicle. Recently, Fisher and Jaikumar [10] used Lagrangian relaxation to solve the generalized assigment problem in the first phase and used an exact procedure by Miliotis [21,22] to solve the travelling salesman problem in the second phase.

(iv) Exact procedures

Christofides and Eilon [3] have used the branch and bound approach to solve the vehicle routing problem. The lower bound used in the branch

and bound method was based on the length of the minimum spanning tree. Pierce [23] developed a similar approach. Balinski and Quandt [1] and Foster and Ryan [11] employed the cutting plane technique in solving the integer programming formulation of the problem. Recent work by Christofides et al. [4] is based on spanning tree and shortest path relaxasions.

A comprehensive survey of the state of the art in solving vehicle routing and scheduling problems can be found in Bodin et al.[2].

Applying approaches designed for VRP to solve DARP directly, one may encounter the following problems: (a) DARP is a two-end problem in which a customer's pick-up point must be visited before the delivery point and they must be served by the same vehicle. (b) A Dial-a-ride vehicle's passenger load is difficult to model mathematically. (Vehicle load can increase or decrease at each stop.) (c) DARP is likely to include some forms of time constraints. These complications have motivated researchers to seek specialized algorithms for DARP (one exception might be those tour-improvement heuristics which are flexible enough to deal with complicated constraints). We will discuss some of these approaches designed for DARP in Chapter 2.

1.3 Purpose of This Thesis

The goal of this thesis is to develop computerized algorithms for solving large-scale advance-request dial-a-ride problems. The capability to solve large-scale problems is important since manual scheduling in

such an environment seems impossible or, at best, inefficient. Computerized scheduling offers great savings in time and with sufficient sophistication it is likely to produce better schedules than the manual process.

Past work in the area of dial-a-ride problems has concentrated on the immediate-request version of the problem. Little has been known about the advance-request problem. Recently, several simultaneous efforts, including the work described in this thesis, have focused on the advance-request version of the problem.

Another important concern in designing heuristics for the dial-a-ride problem is that some form of service quality should be considered. Two fundamentally different scenarios have been suggested by various researchers during the past decade or so. One scenario has been that desired pick-up or delivery times should be treated as "hard" time constraints and that service is acceptable only if the actual pick-up or delivery times fall within a so-called "time window" of specified duration. A delivery time window of (say) 20 minutes would allow delivery of a customer within up to 20 miuntes prior to that customer's desired delivery time and would consider any late delivery infeasible. Another scenario is that desired pick-up or delivery times are not "hard" constraints and that the service operator will try to bring a customer's actual pick-up or delivery time as close as possible to the desired ones, without providing guarantees.

Both scenarios are plausible and each of them may be applicable to certain classes of system users. We will examine both scenarios in this thesis. For each scenario, one specialized algorithm was developed and extensive computational experience with each algorithm was obtained over a wide range of problem sizes.

Chapter 2 of this thesis gives an overview of past work on various versions of the dial-a-ride problem. Chapter 3 describes the first algorithm which treats the time constraints as "soft" constraints. Chapter 4 introduces the second algorithm which guarantees a pre-specified level of service quality. Chapter 5 provides computational results with the second algorithm. Chapter 6 suggests strategies which may improve the algorithms and concludes this work by suggesting possible areas for future research.

CHAPTER 2 REVIEW OF ALGORITHMS FOR DIAL-A-RIDE PROBLEMS

2.1 Introduction

This chapter reviews earlier work that has investigated the dial-a-ride problem. The discussion consists of four parts. In Section 2.2, the immediate-request dial-a-ride problem is discussed followed by the advance-request version in Section 2.3. Section 2.4 discusses a probabilistic analysis of the problem and finally Section 2.5 outlines two algorithmic approaches taken in this thesis for solving the multi-vehicle advance-request dial-a-ride problem.

2.2 Immediate-Request Dial-A-Ride Problem

The immediate-request dial-a-ride problem involves diapatching a fleet of vehicles in response to customer demands that require immediate service. When a dial-a-ride agency receives a new customer request, its vehicles are either in an idle state, i.e. waiting for further assignments, or are busy executing tasks that have been assigned to them previously. Similarly, customers in the system are either on board some vehicle on the way to their destination or are still waiting to be picked up. With the current status of vehicles and customers known, the problem is to determine which vehicle should serve the new customer and to develop a new route and schedule for the responsible vehicle.

Wilson and others [40,41,42,43] have examined this problem in their pioneering work associated with the Rochester, New York Dial-A-Ride

Demonstration Project. In [26], Psaraftis introduced an algorithm which can solve the single-vehicle immediate-request problem optimally using dynamic programming. Psaraftis and Tharakan [25] later extended this approach to the multi-vehicle case and in [29] Psaraftis enhance the dynamic programming algorithm by including the capability of handling time window constraints. Moreover, Tharakan and Psaraftis [34] have shown that this dynamic programming approach can be extended to a similar problem with an exponential disutility function. We now discuss these approaches in detail.

2.2.1 An Assignment Algorithm (Wilson et al. [42,43])

This algorithm selects the best way to incorporate the new customer into the existing vehicle route which is called "the provisional route". All possible combinations of insertions of the new customer into the provisional route are examined and preferences are determined by using a cost criterion which is a weighted sum of customer disutility and a vehicle resource function.

Customers are divided into classes depending upon the type of service demanded. Different classes of customers have different disutility functions. Let D_{ij} be the disutility for the ith customer in class j. Then D_{ij} is defined as:

$$D_{ij} = a_j W_i^2 + b_j R_i^2 + c_j P_i^2 \qquad (2-1)$$

where a_j, b_j and c_j are constants which reflect the particular service preferences of customers in class j.

W_i is the waiting time between the instant when a request was made

and the actual pick-up time.

R_i is the time between a passenger's pick-up and delivery, i.e. the ride time.

P_i is the time between the promised pick-up time and the actual pick-up time.

The objective function, Z, that determines the relative merits of different insertions for a new customer p (in class q) is the sum of three parts:

1) The new customer p's own disutility: D_{pq}
2) The marginal disutility of other customers caused by the insertion of the new customer into the provisional route:

$$\sum_i \sum_j (D'_{ij} - D_{ij})$$

where D'_{ij} is the disutility for customer i in class j after the insertion of customer p into the route.

3) The cost in system resources:

$$(TL'_v - TL_v) (d \cdot \overline{TL} + e \cdot TL_v)$$

where d and e are parameters.

TL_v and TL'_v are the tour length for vehicle v before and after the insertion of customer p.

$\overline{TL}$ is the mean tour length for all active vehicles.

The system resource function is designed to conserve system resources, i.e. vehicle time, in relative response to the changing pattern of demands and to equalize workload among all vehicles.

Thus, the total objective function used in determining the desirability of an insertion is:

$$Z = D_{pq} + \sum_i \sum_j (D'_{ij} - D_{ij}) + (TL'_v - TL_v) (d \cdot \overline{TL} + e \cdot TL_v)$$

After Z values are evaluated for all possible insertions, the algorithm selects the one which has the minimal Z value as the best insertion. The new customer is then incorporated into the schedule according to the selected insertion.

Wilson also included a reassignment capability in the algorithm which reconsiders previous assignments if required to do so. For example, if a vehicle is considerably late in carrying out its route, all customers waiting to be served by that vehicle are considered for reassignment. In [43], Wilson and Miller described an alternative approach, "deferred assignment", which defers the decision on assignment until as late as possible in the hope that better routes can be developed as more information on future customers is available. However, the effect of such a strategy was not reported.

2.2.2 Dynamic Programming (DP) Approach by Psaraftis [26]

Psaraftis has developed an algorithm which can solve the single-vehicle immediate-request problem with a linear objective function to optimality using dynamic programming. In [25], he further extended the approach to solve the multi-vehicle immediate-request problem. In this section, we will describe in detail the single-vehicle DP formulation and its extension to the multi-vehicle case, both within the immediate-request environment.

(i) Single-vehicle case

Assume that a dial-a-ride agency receives a customer request at time t=0 and the vehicle is located at A at that time with none or some customers on board. The problem is to develop a new vehicle route which includes the new customer and minimizes a generalized objective function. The generalized objective function is expressed as follows:

$$\text{minimize} \quad w_1 \cdot \sum_j T_j + w_2 \cdot \sum_i (\alpha \cdot WT_i + (2-\alpha) \cdot RT_i) \qquad (2\text{-}2)$$

where T_j is the duration of the jth leg of the trip.

WT_i is the waiting time of customer i.

RT_i is the ride time of customer i.

w_1, w_2, α are weights associated with each term which can be specified by the user.

$0 \leq \alpha \leq 2$

The term, $\sum_j T_j$, represents the time to serve all N customers (including the new one). The other term, $\sum_i (\alpha \cdot WT_i + (2-\alpha) \cdot RT_i)$, represents the total customer disutility which is a linear combination of the time each passenger waits for the vehicle and of the time he spends inside the vehicle until his delivery.

Constraints on the problem include: (a) vehicle capacity constraints (b) the maximum possible position shift from the customer's original standing in the call list. This second constraint acts as a "service guarantee" for each customer.

States in the DP formulation are represented by the vector $(L,K_1,\ldots,K_N)$ where

a) L: Point the vehicle is currently visiting. It is assumed that:

L = Between 1 and N : Vehicle is at origin of customer L
Between N+1 and 2N : Vehicle is at destination of customer L−N
2N+1 : Vehicle is at starting point A.

b) K_j: "status" of customer j (j=1,.....,N). It is assumed that:

K_j = 3 : Customer j has not been picked up so far
2 : Customer j is in the vehicle
1 : Customer j has been delivered.

Stage variable n is zero when the vehicle is at the starting point A, one at the first pick-up stop and so on, until n = 2N at the last delivery stop.

Based on the above formulation, the DP algorithm can solve the single-vehicle problem to optimality. Due to the exponential growth in computation time with respect to the number of customers involved, the algorithm can solve only small problems, i.e. 8-10 customers, within reasonable computer time. For example, it takes 2.7 seconds of CPU time (VAX 11/782) for the case of N = 5, 46.8 seconds for N = 7 and 591.4 seconds for N = 9.

(ii) multi-vehicle case

In the multi-vehicle problem, Psaraftis and Tharakan [25] have

designed an assignment procedure which determines which vehicle should serve the new customer. After such an assignment is made, the untravelled portion of the chosen vehicle's route is reoptimized with the new customer included using the single-vehicle algorithm. The assignment procedure employs an objective function which determines the desirability of assigning the new customer to a vehicle. The objective function can take two forms:

(a) MINIMAX: The objective here is to minimize the latest time for all vehicles to complete their assigned route.

$$\text{minimize} \quad \max_{j} \left[\sum_{j} T_j^k \right]$$

where m is the number of vehicles, N^k and T_j^k are as defined in (2-2) for the kth vehicle

(b) MINISUM: The objective here is to minimize the sum, evaluated over all vehicles, of expressions similar to (2-2). It can be written as follows:

$$\text{minimize} \sum_{k} V_k$$

where V_k represents the expression in (2-2) for vehicle k.

The complete multi-vehicle algorithm can be outlined as follows: Let Z denote the adopted form of the objective function.

Step 1 : For all vehicles j (j=1,....,m), compute V_j, the objective function for the jth vehicle. If Z = MINIMAX, set $Q = \max_j V_j$. This is an initialization step which does not involve the new customer.

Step 2 : Tentatively assign the new customer to each vehicle j and evaluate the prospective V_j'. After V_j' is obtained, do the following: If Z = MINIMAX and $V_j' \leq Q$, then the new customer is

assigned to vehicle j. The assignment procedure terminates. Otherwise, obtain V_j' for another vehicle.

Step 3 : If Z = MINISUM, then the new customer is assigned to that vehicle j for which $(V_j' - V_j)$ in minimized. If Z = MINIMAX, then the new customer is assigned to vehicle j for which $\max(V_j', Q)$ is minimized.

This assignment procedure is optimal if there is only one new customer and changing previously committed assignments is not allowed. Like the single-vehicle algorithm, this multi-vehicle algorithm is limited by the exponential growth in computation time with respect to the number of points unvisited by each vehicle at the time of the new request. For large problems, the algorithm can be used under a constraint which limits the number of points to be considered for resequencing at the time of appearance of the new customer.

2.2.3 Two Extensions of the DP Algorithm [29,38]

(i) With exponential disutility function:

In [38], Tharakan and Psaraftis show that the DP formulation used for solving the immediate-request problem with a linear objective function (see (2-2)) can be extended to the problem with an exponential disutility function:

$$\text{minimize} \sum_{i=1}^{N} a_i \exp\left(\lambda \sum t_n^i(s)\right) \qquad (2\text{-}3)$$

where S = the set of all feasible sequences of stops.

s = an element of S.

a_i = constant for customer i (i=1,....,N).

λ = constant.

$t_n^i(s)$ = 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+1st stops of sequence s.

(ii) With time window constraints:

In [29], Psaraftis shows that the same DP formulation can be modified to consider time window constraints on customers' pick-up times and delivery times. The backward recursion used in [26] has to be replaced by a forward recursion so that the temporal aspect of the problem can be tracked. The computation effort of this modified version is the same as in the original algorithm.

2.3 Algorithms for the Advance-Request Dial-A-Ride Problem

Algorithms for the advance-request dial-a-ride problem can be divided into two categories: single-vehicle or multi-vehicle algorithms. Since the single-vehicle problem is a subproblem of the multi-vehicle case, its algorithms are mostly used as "subroutines" in the multi-vehicle problem. In this section, we first introduce three heuristic single-vehicle algorithms, followed by three multi-vehicle algorithms.

2.3.1 Single-Vehicle Algorithms

The single-vehicle advance-request dial-a-ride problem can be

defined as follows: Given N pairs of pick-up and delivery points, find the shortest route covering all the points with a customer's pick-up point visited first before visiting his delivery point. This problem is similar to the travelling salesman problem with precedence constraints. The objective function can vary from the basic shortest route length to include some measurement of customers' disutility. Time constraints can also be added to the basic problem so that some quality of service can be guaranteed. We first discuss two heuristic algorithms by Psaraftis [27,28] for this version of DARP followed by a Bender's Decomposition method by Sexton and Bodin [32].

(i) Two polynomial-time heuristics

Psaraftis [27,28] has developed two polynomial-time heuristics for the single vehicle dial-a-ride problem with the objective of minimizing route length. Time constraints and customer disutility are not considered in this case. The first heuristic is rather simple: It is based on the Minimum Spanning Tree (MST) that is defined on the N origins and the N destinations of the problem. From the MST in question, an initial travelling salesman tour T_0 through the above 2N points can be constructed. Choose any customer origin on T_0 as the first pick-up point P_1 on the dial-a-ride route. From this point, move on T_0 clockwise until all points are visited and then return to A (the initial vehicle location). While doing this , do not visit any point that has been previously visited, or any destination whose origin has not been previously visited. Call this dial-a-ride tour T_1.

The procedure described above is a generic version of the heuristic. Several optional steps are possible: (a) Instead of moving clockwise on T_0, we can choose to move counterclockwise and compare with the previous result. (b) Each time choose a different customer origin as P_1. (c) Improve upon T_1 by performing a sequence of local interchanges (see details below).

The computational complexity of this heuristic in $O(N^2)$. A pathological case is constructed which shows that the relative error of the heuristic can potentially go as high as 300%. However, the average performance of the heuristic is projected to be 13% off the optimum by comparing simulation results with the asymptotic value obtained by Stein [37].

In the optional step (c) of the MST-based heuristic, it is indicated that T_1 can be improved by applying a sequence of local interchanges. As the second polynomial-time heuristic, Psaraftis [28] has designed a k-interchange procedure which draws from the k-interchange procedure introduced by Lin [19] and Lin and Kernighan [20] for the TSP. As in the TSP, a k-interchange in the dial-a-ride problem is a substitution of k of the links of a dial-a-ride tour with k other links. A dial-a-ride tour in said to be k-optimal (or k-opt) if it is impossible to obtain another dial-a-ride tour of shorter length by replacing any K of its links by any other set of K links.

In [28], Psaraftis was able to develop a method that finds the best k-interchange that can be produced from an initial feasible dial-a-ride

tour in $O(N^k)$ time, the same order of magnitude as in Lin's heuristic for the TSP. Psaraftis has described in detail the k-interchange procedure for the cases of k=2 and k=3. Computational experimentation shows that the 3-opt procedures outperform the corresponding 2-opt algorithms by producing tours that are about 30% shorter on the average if the initial tour is random and about 6% shorter on the average if the initial tour is MST-generated.

(ii) Bender's Decomposition

In [32], Sexton and Bodin introduce a heuristic algorithm aimed at minimizing total inconvenience for all N customers. The total inconvenience D_i of customer i is defined to be a weighted sum of his excess ride time, ERT_i, and his delivery time deviation, DV_i (all customers specify a desired delivery time).

$$D_i = A \cdot ERT_i + B \cdot DV_i \qquad A,B \text{ are given constants.}$$

The problem is to find a schedule that minimizes $\sum_{i=1}^{N} D_i$.

Constraints of the problem include that vehicles have limited capacity and customers cannot be delivered later than their desired delivery times.

The algorithm can be outlined as follows:

Step 1 Use a space-time heuristic to form an initial feasible route.

Step 2 Find the optimal schedule for the given route by converting the problem to a maximum profit network flow problem which can

be solved without substantial computational effort.

Step 3 Improve upon the solution at hand by changing the objective function coefficients of the routing problem and see whether the new route and schedule produced yields better results in terms of total customer inconvenience.

This algorithm was coded in FORTRAN and implemented on a UNIVAC 1100/70 computer. Test data were obtained from operating sites with problem size ranging from 7 - 20 customers per vehicle. Computation time averaged roughly 18 seconds of CPU time per vehicle. This algorithm is later used as a core component in a multi-vehicle algorithm by the same author.

2.3.2 Multi-Vehicle Algorithms

The multi-vehicle advance-request dial-a-ride problem has attracted much interest lately among several groups of researchers. The main purpose of this thesis is to develop algorithms for this particular problem. We will examine other works on this problem in this section:

(i) Bender's Decomposition

Bodin and Sexton proposed a heuristic algorithm which employs the single vehicle dial-a-ride algorithm described in Section 2.2 as the core component for solving the multi-vehicle case. It first partitions customers into several clusters. Within each cluster, the single-vehicle algorithm is used to develop the vehicle route and schedule which

minimizes customer inconvenience. After initial schedules are obtained for each vehicle, a route improvement procedure, the "swapper" algorithm, is used to swap customers one at a time between vehicles so that the total customer inconvenience is reduced. This improvement procedure is repeated until no further improvement can be found.

This algorithm was applied to one small data set consisting of about 85 afternoon customers of the dial-a-ride system in Baltimore, MD. It took about 2-3 minutes CPU time on Univac 1108 to complete a single run. Results compared favorably with the manual solution.

(ii) NEIGUT/NBS algorithm [15]

The NEIGUT/NBS algorithm is aimed at maximizing vehicle productivity by determining the minimum number of vehicles necessary to provide dial-a-ride service. The algorithm uses the concept of a "base-trip" which is defined as a vehicle route starting from a customer's origin and ending at his destination. The base-trip customer serves as a seed in attracting other customers to share the ride with him. By packing as many customers as possible into a base trip, the algorithm forms a vehicle sub-tour (partial schedule) which starts with the first pick-up of a customer when the vehicle is empty and ends after delivering the last customer on board in the expanded base-trip. A vehicle's whole-day schedule is then constructed by linking different sub-tours together. Not all customers have to be served since the algorithm assumes that a supplemental service ,e.g. taxi, is available and might be less expensive than using the dial-a-ride vehicles.

The algorithm also seeks to meet a specified quality of service for all customers. Customers are guaranteed to be served within given time intervals and their maximum on-board times cannot exceed a prespecified limit. No customer disutility or inconvenience is assumed in the model.

The algorithm starts by treating every customer as a base-trip and for each base-trip, it evaluates the possibility of insertion of another customer into the base-trip. For example, assume that the base-trip customer is customer k and p_k and d_k denote the pick-up and delivery of customer k. To insert another customer, say , i to this base-trip, two possible permutations are examined: p_k, p_i, d_i, d_k, and p_k, p_i, d_k, d_i. Of all possible permutations (there might be more than two if the base-trip has been expanded already), the permutation selected is the one that is feasible in term of service time constraints and maximum on-board time constraints and that minimizes the travel time from p_k to d_k. This procedure is repeated until no more customers can be added to the base-trip. The partial schedule formed is then assigned to a vehicle by linking it with other partial schedules already assigned to that vehicle.

The process of constructing partial schedules is repeated over the remaining set of unassigned customers until all customers are included in the vehicle schedules. It is to be expected that the routes generated first are the most productive and the routes generated last tend to be inferior in this respect. As a result, vehicle workloads tend to be unevenly distributed.

(iii) Parallel insertion [30]

Roy et al. [30] develop an insertion-oriented algorithm for dial--a-ride systems in Canada which provide transportation service to the handicapped. The algorithm guarantees a quality of service similar to that of the NEIGUT/NBS approach. Each customer is allowed to specify either a desired departure time or desired delivery time. Deviation from the desired service time is kept below a specified limit. The algorithm also places an upper bound on a customer's excess ride time. The customer disutility function is taken to be a linear weighted sum of service time deviation and excess ride time. Without considering customer disutility at first, the algorithm constructs an initial set of vehicle routes and schedules by sequentially inserting customers to the routes that have been built up from previous insertions. After all customers have been inserted into the initial set of routes, the algorithm then considers exchanging customers between vehicles so that customers' disutility can be reduced.

Before considering the insertions of customers, a concept of "neighbor" is introduced to group together neighboring customers who are likely to be served by one vehicle. For example, customer k is considered to be a neighbor of customer i if

(a) $LPT_k - EDT_i \geq d(-i,+k)$ and $d(-i,+k) \leq MAXSEP$,

where MAXSEP is a pre-determined constant

or

(b) $LDT_k - EDT_i \geq d(-i,-k)$ and $d(-i,-k) \leq MAXSEP$

and $d(+i,+k) + d(+k,-i) < 1.4 \cdot d(+i,-i)$

The constant 1.4 is a parameter which approximately defines an ellipse

around the stops. A "time-horizon" concept is also used to define the set of customers considered simutaneously as the candidates for the next insertion. The time-horizon is defined by a time window and a maximum number of customers allowed in the horizon.

The algorithm starts by sorting the customers in ascending order with respect to their desired service time. Initialization occurs by identifying those customers whose desired service time falls within the specified horizon. For each customer in the horizon, find his neighbors satisfying the relationship (a) or (b) above. Initialize one or several routes with groups of neighboring customers (details of which are not clear in [30]). Then, repeat the following procedure:

Step 1 Update the horizon, i.e. move the time window defining the horizon later in time and introduce more customers into the horizon.

Step 2 Attempt to find the best route for a chosen customer d in the horizon. If it is infeasible to insert customer d, then relax the time windows of customer d and try again. If still not feasible, initialize a new route for customer d.

Step 3 Attempt to insert the neighbors of customer d in the horizon into the same route to which customer d is inserted.

Step 4 If all customers are assigned to routes, go to Step 5. Otherwise, go to Step 1.

Step 5 Attempt to reduce customer disutility by exchanging customers between vehicles. Start with the customer who has the highest disutility and repeat the procedure until no improvement can be realized.

In [30], it is not clear that the algorithm will eventually provide a feasible solution in terms of satisfying service quality guarantees when time window constraints are relaxed in Step 2 above. In other words, the exchange procedure in Step 5 does not guarantee reestablishment of those relaxed constraints after exchanging customers between vehicles.

Test results of the algorithm are obtained for actual problems in Montreal and Sherbrooke in Canada. The results show that the algorithm can provide smaller cost schedules and better service quality when compared with the schedules produced by the dial-a-ride agencies. A case of 451 customer requests and 31 vehicles required 472.3 seconds of CPU time(type of computer used is not known).

2.4 Probabilistic Analysis

Stein [37] conducted a probabilistic analysis of the dial-a-ride problem. The approach taken is one of preplanning at a global level. It is assumed that in a large system where there is a large number of passengers, it should be possible to predict quite closely the behavior patterns of the set of all customers, and to design the system accordingly. With such an analysis, any proposed schemes in system design can be evaluated analytically, without the need to resort, at this basic level, to simulation.

Assume that there are n customers demanding service. Customers' origins and destinations are independently and uniformly distributed over a region R with area A. In [37], Stein proves the following theorem:

THEOREM 1. Let Y_N^* be the length of an optimal bus tour through N random demand pairs in a region of area A. If N is large, then with high probality

$$Y_N^* \approx 1.89\ b\ \sqrt{A}\ \sqrt{N}$$

The constant b has been estimated by Monte Carlo experiments to be 0.765.

For a multi-vehicle fleet problem, Stein states that to minimize total travel time without transfers, one tour would suffice. If transfers are permitted and the objective is to minimize the time-to-completion, 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

$$Y_N^k \approx 4/3\ \sqrt{(2)}\ b/k\ \sqrt{A}\ \sqrt{N}$$

Stein also evaluates asymptotically the performance of the design scheme for the dial-a-ride system in Ann Arbor, Michigan. Models and algorithms for the immediate-request problem are also discussed.

2.5 Two Algorithmic Approaches Proposed in This Thesis

The two algorithms proposed for the multi-vehicle advance-request dial-a-ride problem are heuristic in nature which can solve large-scale problems efficiently. The first algorithm, to be described in Chapter 3, assumes that the time constraints on customers' desired service time are "soft". In other words, the algorithm will try to serve customers close to their desired times, but does not guarantee it. The second algorithm treats service quality constraints as "hard" constraints. Furthermore, it

assumes the presence of customer disutility with respect to the quality of service offered. We will describe this algorithm in Chapter 4.

CHAPTER 3 A HEURISTIC ALGORITHM FOR DARP WITH SOFT TIME CONSTRAINTS

3.1 Introduction

This chapter describes a heuristic algorithm developed for the multi-vehicle advance-request dial-a-ride problem without "hard" time constraints. The heuristic approach has been designed with three basic objectives in mind. The first, motivated by increasing concern about the exceptionally high costs of many existing dial-a-ride systems, is to achieve high "vehicle productivity". Vehicle productivity is defined here as the "number of customers served per vehicle hour".

A second objective is to design routes and vehicle schedules that result in an acceptable level of service to the system's customers. We measure this level of service in two ways: (i) through the "circuity" of each customer's ride, i.e. the ratio of the distance actually ridden to the direct distance between the customer's origin and destination; and (ii) through the absolute value ("deviation") of the difference between each customer's actual and desired pick-up (or delivery, as appropriate) times. Clearly the minimum values of these two measures are 1 and 0, respectively.

The third major objective is to develop a solution procedure which can be used to solve large-scale problems efficiently. The target size is systems with approximately 60 simultaneously active vehicles serving as many as 500 customers per hour. (Problem size is best measured by hourly demand rather than by the total number of subscribers as in most vehicle

routing problems because of the way this algorithm treats the time dimension of the problem - see also Section 3.2). To our knowledge, no existing algorithm has been applied to date to problems of comparable size.

Since the nature of the problem does not require strict time constraints be met, it is assumed that the service operator will try to bring a customer's actual pick-up or delivery time as close as possible to the desired ones, without actually guaranteeing that. Thus, as will be seen in Section 3.2, if a customer has specified a desired delivery time of (say) 10:23 a.m., the proposed algorithm will attempt to deliver that customer within a specified "time group", say, between 10:00 a.m. and 10:30 a.m. The projected actual delivery time will be recorded and accounted for in the algorithm's overall performance. Moreover,in Section 3.4 we shall see that the routing part of the algorithm provides a capability of satisfying "hard" time window constraints within each individual time group, or minimizing the number of late deliveries or a related tardiness measure. Those customers whose deviations, despite all efforts, are excessively long can either refuse service, or be denied service, or finally serviced by a back-up fleet of vehicles (e.g. taxicabs) at the option of the dispatcher.

The algorithm that has been developed as a result of the above considerations consists of three successive and distinct steps, namely grouping, clustering, and routing. Figure 3.1 shows the algorithm's basic structure. Grouping consists of dividing customers into "time groups" (or, simply, groups) on the basis of their desired pick-up and delivery times.

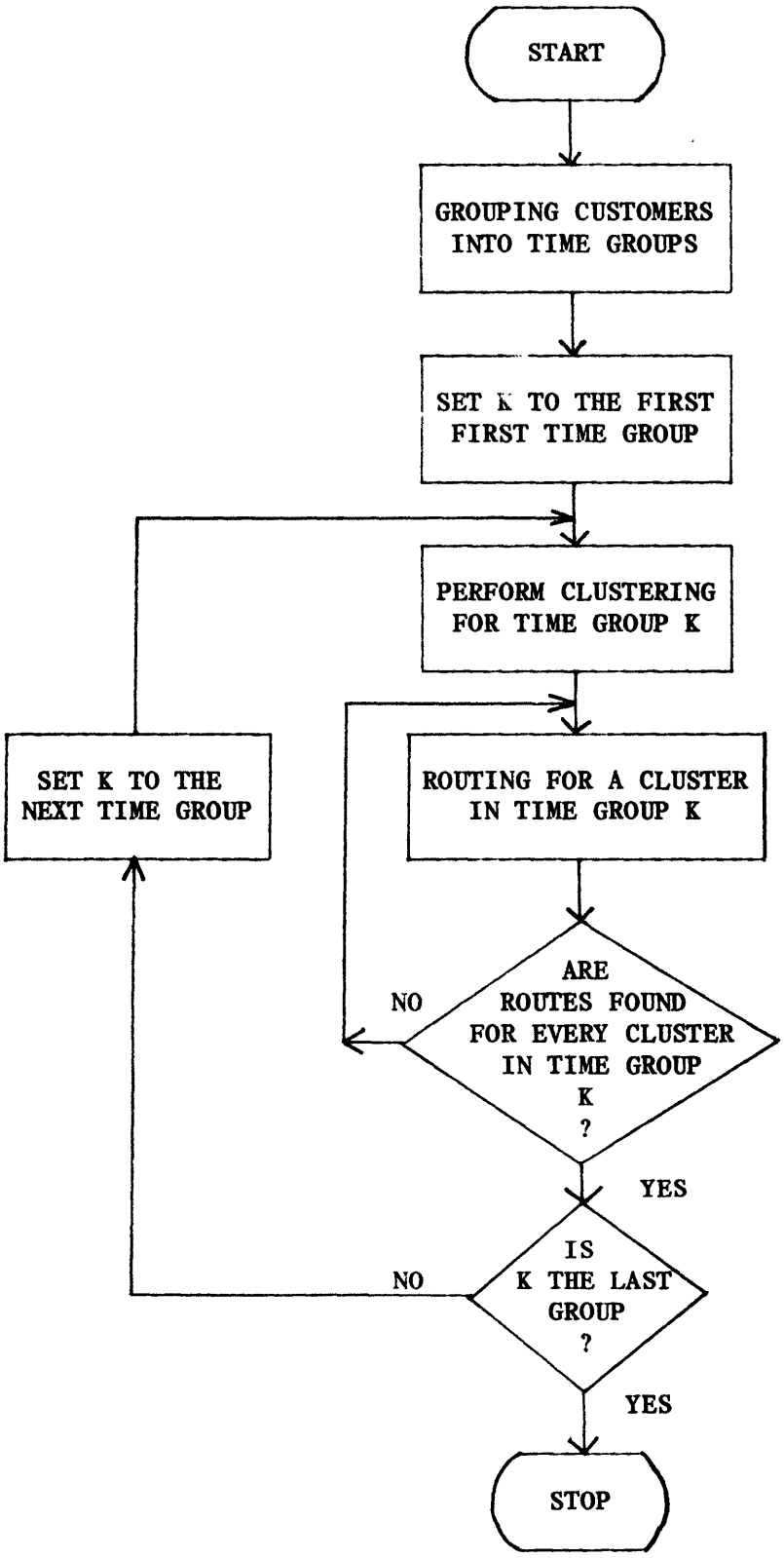


Figure 3.1 Structure of the GCR algorithm.

Customers in each time group must be picked up or delivered (or both) within the corresponding time interval. After the groups are formed clustering is performed, starting with the first group (associated with the earliest time interval under consideration) and proceding sequentially to the last one. The aim of clustering is to separate customer groups into smaller "clusters" so that each vehicle can serve the cluster to which it is aasigned within the bounds of the corresponding time interval. Finally, a routing algorithm is called to generate routes for each individual vehicle, the routing algorithm being applied to every cluster in turn and for every time group. We shall describe each of the three basic steps of the algorithm in Section 3.2, 3.3 and 3.4, respectively. Section 3.5 provides an example of clustering and Section 3.6 describes our computational experience which includes runs using both simulated data and a real data base of 2,617 customers covering approximately 16 hours of service of the dial-a-ride system operating in the city of Friedrichshafen, West Germany. Section 3.7 discusses the computational complexity of the algorithm.

3.2 Grouping

During the grouping process we divide the time horizon into equal and consecutive time intervals and then assign customers to groups, according to the time interval into which the customers' pick-up and delivery times fall. To perform customer assignment to groups we associate a desired pick-up time (DPT) and a desired delivery time (DDT) with each and every customer. Specifically, for a customer j who has specified a desired delivery time, we estimate a desired pick-up time by using:

$$DPT_j = DDT_j - CF \cdot DRT_j \qquad (3\text{-}1)$$

where CF is a conversion factor ($CF \geq 1$) and DRT_j is the direct trip time of customer j.

Similarly, if a desired pick-up time has been specified by the customer, we set:

$$DDT_j = DPT_j + CF \cdot DRT_j \qquad (3\text{-}2)$$

The conversion factor CF is a user input. Indeed, CF is used as a "calibration" parameter in our algorithm. A small value of CF (≈ 1.0) would force the algorithm to design relatively non-circuitous routes. Larger values of CF would likely result in longer ride times, but, at the same time, lead to higher productivity.

For any time interval, we identify three categories of customers:

Category 1: Those customers whose DPT only falls within the time interval. Category 1 customers will only be picked up during the time inetrval in question, if this is feasible with available resources.

Category 2: Those customers whose DPT and DDT fall within the time interval. Category 2 customers will be picked up and delivered during the time interval in question, if this is feasible with available resources.

Category 3: Those customers whose DDT only falls within the time interval. Category 3 customers will only be delivered during the time interval in question (presumably they have been picked up at an earlier time) if this is feasible with available resources.

Example 3.1:

Assume that the time horizon has been sub-divided into 30-minute time intervals with the first interval beginning at 7:55 A.M. Then a customer with DPT = 8:08 and DDT = 8:36 is a category 1 customer with respect to group 1, i.e., the group associated with the first time interval (7:55-8:25) and a category 3 customer with respect to group 2 (8:25-8:55). A customer with DPT = 7:58 and DDT = 8:15 is a category 2 customer with respect to group 1. (End of Example 3.1)

The length DT of the time intervals on which grouping in based is also a calibration parameter that is specified by the algorithm's user. Note that the time interval serves as a time bracket during which the algorithm will try to schedule pick-ups and/or deliveries for all the customers in the corresponding group. Thus, the length of the time interval should be long enough so that the DPTs and DDTs associated with all customers assigned to the same group can be treated as if they were approximately the same.

Note also that if the above defined customer categories are to be collectively exhaustive for each time group, DT must be chosen such that all category 1 customers in a time group k are category 3 customers in the next time group k+1. It is straightforward to see that the lowest possible value of DT that can be chosen so that the above is valid is:

$$DT_{min} = 0.5 \cdot CF \cdot \operatorname{Max}_{j} DRT_j$$

(the maximum is taken over all customers of the problem at hand).

To avoid a possible deterioration of service quality, special attention should be paid to cases where DT_{min} proves to be higher than the minimum interval desired for assuring decent service: this may be caused, for instance, by some customers who have excessively long direct ride times relative to other customers. From a worst-case point of view, the larger the value of DT, the higher the risk of large pick-up or delivery deviations and/or excess ride times for all customers requesting service.

To reduce such a "worst-case" risk, some straightforward procedures can be implemented within the grouping part of the algorithm. Such procedures, collectively referred to as "enhanced grouping", will be discussed in Chapter 5 of this thesis. For the moment, we shall assume that DT, CF, the dimensions of the service region and the vehicle speed are such that every customer's DPT and DDT are guaranteed to fall either within the same or within consecutive time intervals. A sensitivity analysis on CF and DT will be reported in Section 3.6.

Even if its "enhanced" version is used, grouping is by far the simplest step in our algorithm. By sub-dividing customers into groups, we can subsquently deal with one time interval at a time, i.e., we decompose a large problem into many smaller ones. Of course, these smaller problems are not independent of each other, since a category 1 customer picked up by a vehicle during interval k will automatically become a category 3 customer on that same vehicle at the beginning of time interval k+1. We thus have to deal with each of the small problems individually as well as with the interface between these problems. The next section will describe the way this can be done.

3.3 Clustering

Through clustering each of the time groups of customers is sub-divided into subsets called "clusters". Each cluster is then assigned a vehicle which will service all points in that cluster, hopefully within the time bounds of the interval associated with the corresponding time group. More specifically, for a given time group , a cluster is a subset of customers which will be serviced (picked up or delivered or both) by the same vehicle during that time group.

The clustering part of the algorithm attempts to provide answers to the following questions:

(1) How do we divide customers into clusters?

(2) How do we assign vehicles to clusters?

(3) How do we link clusters belonging to adjacent time groups?

A cursory glance at the clustering component of our problem can reveal at least the following two facts:

(1) Clustering in a many-to-many environment is difficult. In fact, there is no guarantee that if a set of customers have origins close to one another (and hence potentially suitable to be serviced by the same vehicle), those same customers will have their destinations in roughly the same area too. This fact alone disqualifies the use of straightforward geometric procedures (such as the "sweep" algorithm of Gillett and Miller [13]) that have been proposed for the routing of a fleet of vehicles from a central depot to specified delivery points.

(2) Little or nothing on the problem of clustering of multi-vehicle many-to-many dial-a-ride systems has been reported in the literature to date. Sexton abd Bodin [33] address the problem of improving clusters by "swapping" customers among them but do not present a method for obtaining good initial clusters.

It was consequently decided to design a clustering procedure from scratch. In designing such a procedure the following cluster attributes and procedure features were considered important:

(1) Clusters should be "spread out" from one another as far as possible. Indeed, it would make little sense to have two clusters that end up producing similar directional travel patterns for two vehicles in the same region.

(2) Clusters should be such that resulting vehicle workloads are "about the same" for all vehicles. In other words, it would be unreasonable to have one or few vehicles "bear the brunt" of the demand load while the remainder of the fleet remain underutilized.

(3) Clusters should be such that resulting routes are as short and compact as possible. It is felt that such a feature would ultimately increase total vehicle productivity.

(4) There is a need to provide a good "predictor" of the route length associated with a cluster, <u>without</u> using a routing algorithm to do so. In this approach it is decided to use the length of the <u>spanning tree</u> associated with a cluster as such a predictor. Asymptotic analysis by Stein [36] has shown that the optimal dial-

a-rdie tour is proportional to the square root of the size of the area and the number of customers served (see Section 2.4). Similar relationship exists between the length of a minimum spanning tree and the number of vertices defining the graph and the size of the area . Since in the clustering step we only need the relative magnitude of workloads among vehicles, the length of a spanning tree provides reasonably good approximation for this purpose.

(5) There ia a need for "look-ahead" procedures to link the current time group with the next time group. In other words, the clusters at time group k should somehow take into account the destinations of category 1 customers that will be visited (as category 3 customers) at time group k+1.

(6) There is a need for each cluster to provide for good "directivity" in the resulting routes. A route has good "directivity" if it does not deviate too much from a simple directional pattern (e.g. east -to-west, north-to-south). An example of a cluster that provides good directivity is shown in Figure 3.2(a). The cluster of Figure 3.2(b) provides poor directivity, despite the similarity with the one in Figure 3.2(a). The procedures used should be able to discriminate between apparently similar clusters with different directivity characteristics.

In the approach chosen, clusters are formed using a tree-building process. In other words, at each iteration each cluster can be represented by a tree that connects the origins and destinations that comprise the cluster. We start with as many trees as the desired number of clusters by identifying an equal number of "seed customers" who will form a nucleus

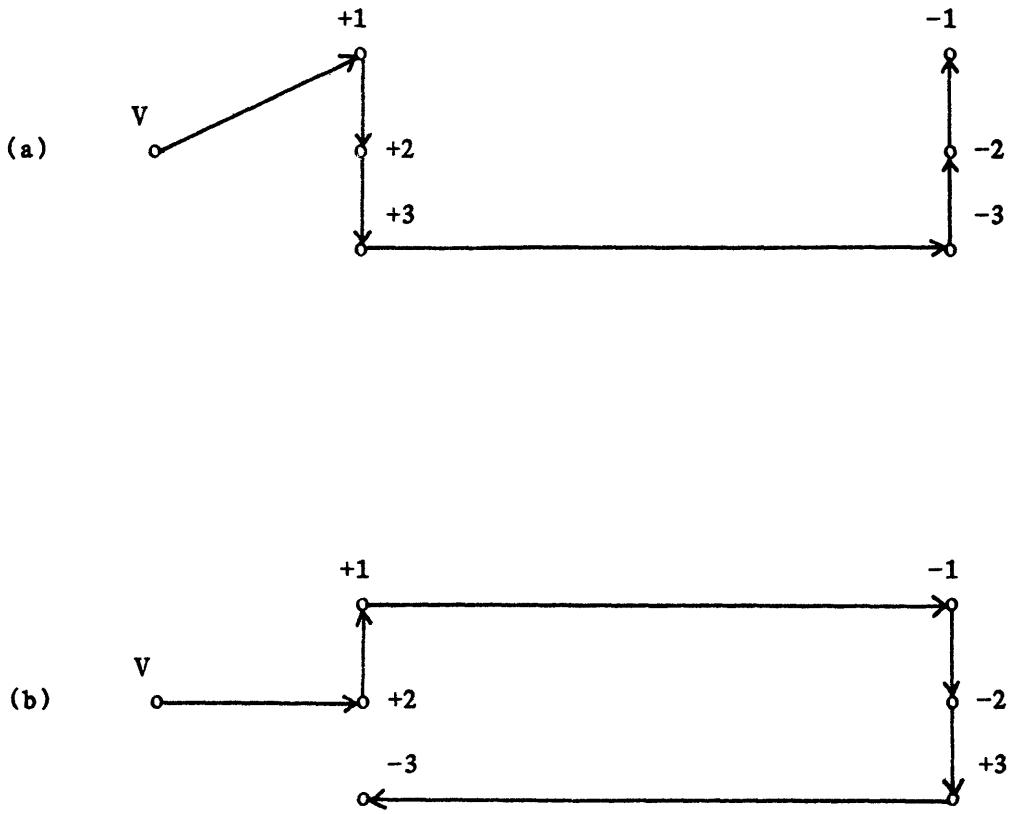


Figure 3.2 The consideration of directivity in forming clusters.

(a) A "good" cluster

(b) A "poor" cluster

for each tree. We then add customers to trees (clusters), one at a time, until all customers in each given time group have been assigned to a cluster.

The clustering algorithm processes each time group consecutively, starting with the group of the time period under consideration. The number of clusters into which each group is sub-divided is set equal to the number of vehicles available during the corresponding time interval (more on this point in Section 3.7). Let the required number of clusters for a given group be m. The clustering procedure can then be summarized by the following three steps:

Step 1: Identify the first customer for every cluster. Each of these m first customers serves as a "seed" for a cluster.

Step 2: Assign vehicles to clusters.

Step 3: Add other customers in the group to clusters one by one according to several geographical proximity rules until every customer belongs to one of the clusters.

Detailed descriptions of each of these three steps follow, along with explanations of the reasoning behind this approach. The list of symbols in Table 3.1 will be helpful to the discussion.

Step 1: Find seeds as a basis for the clusters.

As mentioned earlier the fundamental idea behind the selection of "seed customers" is that it is preferable to have these customers be as "spread out" (in the geographical sense) as possible within the region of interest. This will make it more likely that the clusters (and eventually

Table 3.1 List of Symbols for Clustering Step

i,j,n,t :	indices.
C_t:	the set of customers serving as cluster t's seed.
$d(j,C_t)$:	the distance between customer j and cluster t (see also Tables 3.6 and 3.7).
$d(j,CS_t)$:	the distance between customer j and cluster t's seed (see also Table 3.2).
$dir(x_1,x_2)$:	direct distance between point x_1 and point x_2.
m:	number of vehicles used in T^k.
+j:	origin point of customer j.
-j:	destination point of customer j.
ΔP_t^k:	slack or surplus of path length of vehicle serving cluster t at the end of T^k.
S_j:	the set of category j customers in T^k.
S_j':	the set of category j customers in T^k who have not yet been assigned to any clusters.
S_j^t:	the set of category j customers in cluster t.
$\|S\|$:	total number of members in set S.
T^k:	group k; T^k ia also used to represent the group in which clustering or routing is being performed.
V_j:	location of vehicle j when it becomes available in T^k.
W_t:	workload predictor for cluster t.
FW_t:	future workload (in the next time interval) of cluster t.
$\Delta W_{t,j}$:	incremental workload introduced by adding customer j to cluster t.
$\Delta FW_{t,j}$:	incremental future workload introduced by adding category 1 customer j to cluster t.

the routes) which will be developed around these seed customers will be compact and geographically distinct. In addition, customers who are geographically far from each other would be poor choices for inclusion in the same clusters (and routes) and, therefore, can justifiably be chosen as seeds for distinct clusters from the very beginning.

Category 3 customers in T^k are already on board some of the vehicles at the beginning of the current time interval k (these customers have been picked up, as category 1 customers, during time interval k-1). Thus vehicles with category 3 customers on board do not require a seed customer: the destination point(s) of the customer(s) on board will provide the seed(s) for the clusters associated with such vehicles. Call this set of vehicles v. Then, we have to find seed customers only for the remaining $m - |v|$ vehicles.

Our choices now lie in category 1 and 2 customers. Since category 2 customers, for whom both origin and destination must be visited in T^k, are more indicative of the likely area where a vehicle will travel, we prefer to choose them as seeds first. Only when there are no category 2 customers left can category 1 customers be considered as possible seed customers.

Selecting seed customers is an iterative procedure during which one seed customer is selected at a time. Each time we try to select a seed customer, one of the following four collectively exhaustive situations must apply:

(1) no cluster seeds present, and $S_2 \neq \phi$

(2) no cluster seeds present, and $S_2 = \phi$, $S_1 \neq \phi$

(3) some cluster seeds present, and $S_2' = \phi$

(4) some cluster seeds present, and $S_2' = \phi$, $S_1' = \phi$

(Note that it is possible that $S_j' = S_j$). In situations (1) and (2), we are trying to select the very first seed. The somewhat arbitrary rule that we have used is to choose the customer who has the most distant trip to make. In situations (3) and (4), we try to select a seed at a time when some other seeds have already been identified. The seed selection rules in these cases can be described as literally choosing the customer who has the maximum distance to the closest cluster. The distance between customer j and the seed t, $d(j,CS_t)$, is defined in Table 3.2. The mathematical expressions for the seed selection rules under different situations are given in Table 3.3.

Step 2: Assign vehicle to clusters.

The reason why assignment is performed now instead of at the end of clustering when we have complete information about clusters is that, associated with each vehicle i, we have some useful information, namely ΔP_i^{k-1} (to be defined later) which will be used in the subsequent clustering procedure.

It is desired that vehicle i be assigned to a cluster t as long as the location of vehicle i, V_i, is close to the position where the vehicle will start serving the cluster t. A good approximation for this starting point might be the origin point of the seed customer in cluster t. So, the cost of assigning vehicle i to cluster t is defined by $dir(V_i,+j)$, $j \in CS_t$, $|CS_t| = 1$. Our objective is to minimize the total cost of the assignment. Instead of using an optimizing assignment algorithm (e.g., the

Table 3.2 Definition of distance $d(j,CS_t)$ between customer j and seed t.

	$j \in S_2'$	$j \in S_1'$
$\lvert S_1^t \rvert = 1$	impossible case	$dir(+j,+n) + dir(-j,-n)$, $n \in S_1^t$
$\lvert S_2^t \rvert = 1$	$dir(+j,+n) + dir(-j,-n)$, $n \in S_2^t$	$\min_{n \in S_2^t} [dir(+j,+n), dir(+j,-n)]$
$\lvert S_3^t \rvert \geq 1$	$\min_{n \in S_3^t}[dir(+j,-n)] + \min_{n \in S_3^t}[dir(-j,-n)]$	$\min_{n \in S_3^t}[dir(+j,-n)]$

Table 3.3 Seed selection rules

Situations	Choose customer i who solves
(1)	$\max_{j \in S_2} [\, dir(+j,-j) \,]$
(2)	$\max_{j \in S_1} [\, dir(+j,-j) \,]$
(3)	$\max_{j \in S_2'} \{ \min_t [\, d(j,CS_t) \,] \}$
(4)	$\max_{j \in S_1'} \{ \min_t [\, d(j,CS_t) \,] \}$

Hungarian Method), we have chosen to use a simple and quick heuristic -- the greedy approach. This can be stated as follows:

> Set up the assignment matrix C with rows corresponding to vehicles and columns corresponding to clusters. Find the least cost element C^*_{ij} in C. Assign vehicle i to cluster j. Eliminate row i and column j from C. Repeat the process until every cluster has been assigned a vehicle.

Step 3: Add remaining customers to clusters.

The population of customers, T^k, is categorized into three sets, S_1, S_2, S_3. Since all the customers in S_3 (as well as those in S_2 and/or S_1 who have been chosen as cluster seed) already belong to clusters, we only have to consider the remaining population S_2' and S_1' to add to clusters. Our strategy for clustering is to use some geographical proximity rules to evaluate which customer should be added to which cluster so that high vehicle productivity can be achieved. As noted earlier we would also like to distribute "workload" evenly among vehicles so that ride circuity and time deviations are minimized. The "workload" is defined simply as the distance that a vehicle will have to travel in order to serve every customer in the cluster to which it has been assigned. For computational efficiency reasons, i.e., in order to avoid calling a routing algorithm every time a customer is being considered for addition to a cluster, we have devised predictors for workloads. Note that each addition of a category 1 customer to a cluster obligates the corresponding vehicle to visit that customer's origin point in T^k, as well as the customer's destination point in T^{k+1}. For each cluster, we thus need two

predictors, W_t and FW_t, which denote respectively the workload of cluster t in T^k and the future workload of cluster t in T^{k+1}. An addition of a category 1 customer to a cluster t increases both W_t and FW_t. Let the incremental workload to W_t and FW_t, due to adding customer j to cluster t be $\Delta W_{t,j}$ and $\Delta FW_{t,j}$. With knowledge of W_t, FW_t, $\Delta W_{t,j}$, $\Delta FW_{t,j}$, we can then prevent a cluster from growing out of proportion with respect to others by choosing t and j such that the resulting $W_t + \Delta W_{t,j}$ and/or $FW_t + \Delta FW_{t,j}$ are consistent with the objective of equalizing workloads.

We now proceed to present the details of this process which can be summarized in four iterative steps:

Step 3.1: Compute each cluster's workload following completion of the seeding procedure.

Step 3.2: For each cluster, find a candidate customer to join it and determine the incremental workload incurred. Category 2 customers are considered first.

Step 3.3: Choose one customer among all candidate customers such that if this customer is added to the corresponding cluster, the resulting maximum workload among all clusters is minimized. Add this customer to the corresponding cluster and update the cluster's workload.

Step 3.4: If all customers have been assigned to clusters, stop. If not, go to Step 3.2.

Let us consider the details of these four steps.

Step 3.1: The starting workload for cluster t (which only contains a seed

at this stage) is composed of three components:

Component (A): Distance between cluster t and the location of vehicle i responsible for the cluster. Table 3.4 gives this distance as a function of the seed customer's category.

Component (B): Workload introduced by seed customers in the cluster. See Table 3.5. FW_t^{k-1} in Table 3.5 indicates the workload required to fulfill the obligation of delivering category 3 customers on board (who were picked up in T^{k-1} and are thus acting as seed customers in time group T^k).

Component (C): Surplus or slack workload transferred from the previous time group, T^{k-1}.

If a vehicle serving cluster t cannot finish its assigned path in T^{k-1} and the unfinished path ΔP_t^{k-1} must be extended into T^k, we shall transfer ΔP_t^{k-1} to the workload of vehicle t in T^k. Note that ΔP_t^{k-1} can be (and often is) negative which indicates that the vehicle can finish its assigned path earlier than the end of T^{k-1}. Thus, we have: Component (C) = ΔP_t^{k-1}.

On completion of Step 3.1 we therefore have:

W_t = Component (A) + Component (B) + Component (C)

Step 3.2: In this step, we wish to choose one candidate customer for every cluster. We give priority in this selection process to category 2 customers for the same reasons that they were considered first as possible seed customers earlier. To formalize the decision rules for choosing these candidates, we need to introduce the quantity $d(j,C_t)$, which denotes the distance between customer j and cluster t. The value of $d(j,C_t)$ depends on

Table 3.4 Component (A): distance between cluster t and the location of vehicle i responsible for the cluster.

Seed customer j's category	1	2	3
Component (A)	$dir(V_i,+j)$	$dir(V_i,+j)$	$\min_{j \varepsilon CS_t} [\ dir(V_i,-j)\ ]$

Table 3.5 Component (B): workload introduced by seed customer in the cluster.

Seed customer j's category	1	2	3
Component (B)	zero	$dir(+j,-j)$	FW_t^{k-1}

customer j's category and the categories of customers in cluster t. Table 3.6 and 3.7 give the expression for $d(j,C_t)$ for all possible different cases.

The category 2 candidate customer for cluster t should be that customer i who solves:

$$\min_{j \varepsilon S_t} [d(j,C_t)]$$

The incremental workload then is: $\Delta W_{t,i} = d(i,C_t)$, $\Delta FW_{t,i} = 0$

If all category 2 customers have been exhausted, the category 1 candidate customer for cluster t should be the customer i who solves:

$$\min_{j \varepsilon S_t} [d(j,C_t)]$$

The incermental workload then is: $\Delta W_{t,i} = \Delta_i$, $\Delta FW_{t,i} = \delta_i$ where Δ_i and δ_i are as defined in Table 3.7.

Step 3.3: Let (t,i) denote the fact that customer i is the candidate for cluster t. Which candidate should actually be added to his corresponding cluster is determined by solving, for all (t,i):

$$\min_{\text{all } (t,i)} \{ \max_{j \neq t} [\max(W_t + \Delta W_{t,i}, FW_t + \Delta FW_{t,i}) , W_j] \}$$

Let (t^*,i^*) solve the above expression. (Should a tie arise, choose that (t^*,i^*) among the tied candidates which is associated with the smallest $\Delta W_{t,i}$). Then, add customer i^* to cluster t^* and update the workload of C_t^* as follows:

$$C_t^* + \{i^*\} \rightarrow C_t^*$$

$$W_t^* + \Delta W_{t^*,i^*} \rightarrow W_t^*$$

Table 3.6 Definition of distance $d(j,C_t)$ between customer j and cluster t $(j \varepsilon S_2')$

	$S_2^t = \phi$	$S_2^t \neq \phi$
$j \varepsilon S_2'$	$\min_{n \varepsilon S_3^t} [\ dir(+j,+n)\ ]$ + $\min_{n \varepsilon S_3^t} [\ dir(-j,-n)\ ]$	$\min_{n \varepsilon S_2^t} [\ dir(+j,+n)\ ]$ + $\min_{n \varepsilon S_2^t} [\ dir(-j,-n)\ ]$

Table 3.7 Definition of distance $d(j,C_t)$ between customer j and cluster t $(j \varepsilon S_1')$

$d(j,C_t)$:

	$S_1^t = \phi$	$S_1^t \neq \phi$
$j \varepsilon S_1'$	$d(j,C_t) = \Delta_j + \delta_j$ $\Delta_j = \min$ [dir(+j,x)] x is any point in C_t $\delta_j^* = 0.5 \cdot (A/N)^{1/2}$ A : the service area, N : the number of category 3 customers in T^k.	$d(j,C_t) = \Delta_j + \delta_j$ $\Delta_j = \min$ [dir(+j,x)] x is any point in C_t $\delta_j = \min_{n \varepsilon S_1^t}$ [dir(-j,-n)]

* See Larson and Odoni [17].

$$FW_t^* + \Delta FW_{t^*,i^*} \rightarrow FW_t^*$$

If $i^* \in S_2'$, then: $S_2' - \{i^*\} \rightarrow S_2'$. If $i^* \in S_1'$, then: $S_1' - \{i^*\} \rightarrow S_1'$.

Go to Step 3.4

Step 3.4: If $S_2' = \phi$ and $S_1' = \phi$, stop. Otherwise, go to Step 3.2.

3.4 Routing

Upon completion of the clustering process, the routing part of algorithm is called to generate a route and time schedule for every cluster in every time group. Each cluster corresponds to a single-vehicle dial-a-ride routing problem. The objective is to find the shortest route that serves all points in the cluster. This objective is consistent both with achieving high vehicle productivity and with providing relatively efficient service to the system's customers.

According to the review of dial-a-ride algorithms in the previous chapter, we have a "menu" of 5 different algorithms that can be used independently or in combination to solve the single-vehicle problems that are defined for each cluster. The menu of routing algorithms is as follows:

(1) Exact algorithm without time constraints (Psaraftis,[26]).

(2) Heuristic "Minimum Spanning Tree" algorithm (Psaraftis,[27]).

(3) Heuristic "k-interchange" algorithm (Psaraftis,[28]).

(4) Exact Dynamic Programming algorithm with time windows (Psaraftis,[29]).

(5) Heuristic "k-interchange/tardiness" algorithm.

The basic version of the multi-vehicle algorithm is implemented using the second and third algorithms sequentially, with the third algorithm implemented for k = 3. No experience exists with using any of the other algorithms within the multi-vehicle environment; however a nearest-neighbor algorithm has also been used for a specific application (see Section 3.6.2 for the circumstances that led to that choice).

3.5 An Example of Clustering

We now use an example to illustrate some of the details of the clustering algorithm. The example is based on a 6 x 6 mile service region which, intentionally, contains relatively few points. Figure 3.3 depicts each customer's origin (denoted by a "+") and destination (denoted by a "-"). Four vehicles, located respectively at V_1, V_2, V_3, and V_4, will provide service to customers in this time group. The time group includes all three categories of customers. Table 3.8 indicates the category of each customer. It should be noted that customer 20, 23 and 24 are category 1 customers, i.e., only their pick-up times fall within the current time interval. The destination points of these customers (-20, -23, -24) are shown in Figure 3.3 only because they will be used during the clustering process to assign customer 20, 23 and 24 to a cluster. However, these destination points will <u>not</u> be visited during the current time interval and do not properly belong to the current group.

With respect to category 3 customers, it is assumed for this example that customers 1 and 2 are already on board vehicle 1, while

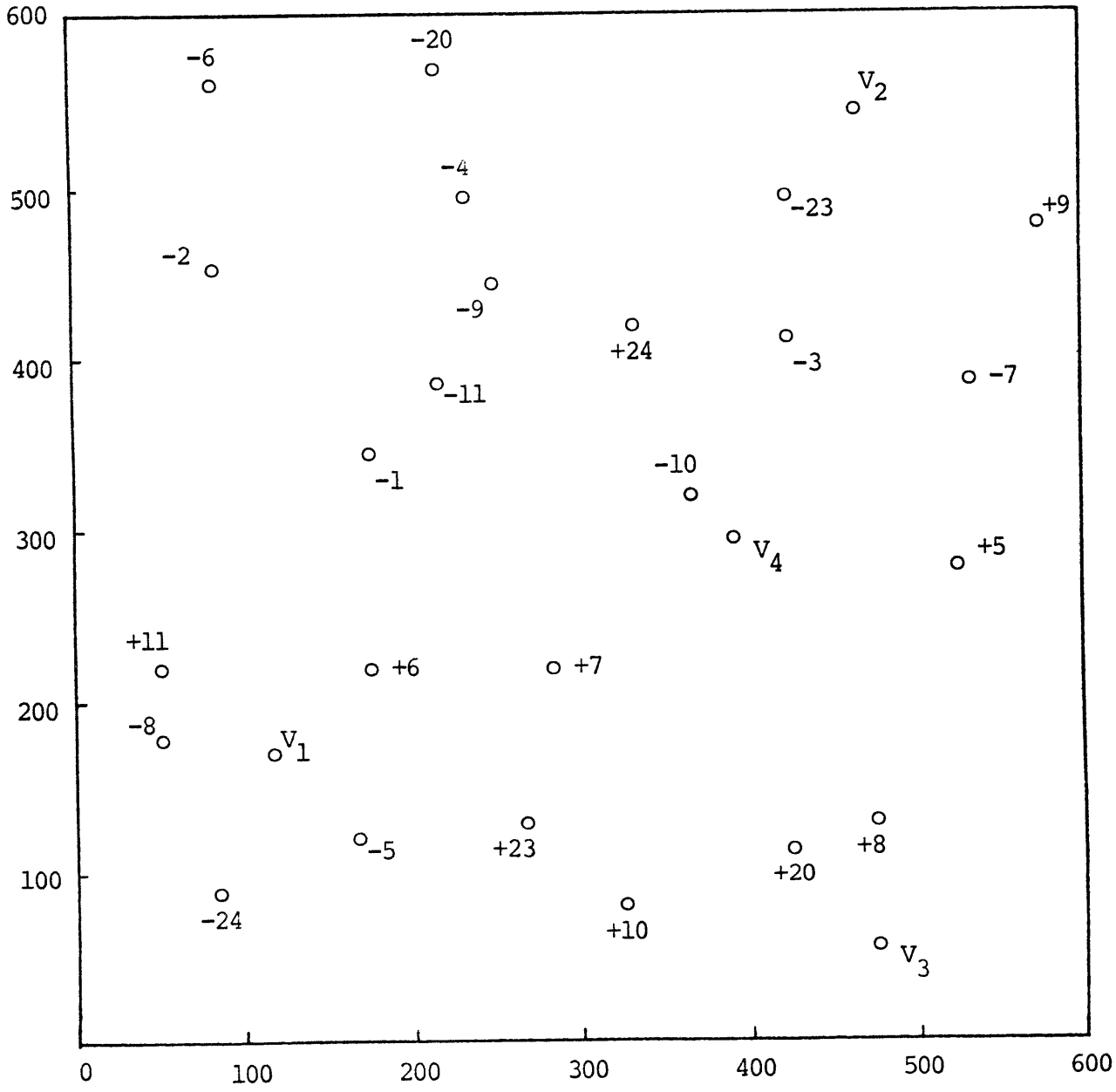


Figure 3.3
Locations of pick-ups and deliveries for the example.

Table 3.8 Customer category classification for the example

Category	Customers
1	20, 23, 24
2	5, 6, 7, 8, 9, 10, 11
3	1, 2, 3, 4

Table 3.9 Step 1: customer 8 is selected as the third seed

Customer j	$d(j,CS_1)$	$d(j,CS_2)$	$\min(d(j,CS_1),d(j,CS_2))$
5	360 + 228 = 588	177 + 382 = 559	559
6	127 + 98 = 225	282 + 169 = 451	225
7	177 + 363 = 540	241 + 117 = 358	358
8	378 + 209 = 587	297 + 374 = 671	587*
9	412 + 127 = 539	153 + 50 = 203	203
10	308 + 197 = 505	355 + 104 = 459	459
11	181 + 60 = 241	344 + 117 = 461	241

customers 3 and 4 are on board vehicle 2.

We now work through this example step by step to show how the clusters are obtained:

Step 1: Seed Selection

Since vehicles 1 and 2 already have category 3 customers on board we only need to select two more seeds one each for V_3 and V_4. In selecting the first additional seed, it is clear that situation (3) in Table 3.3 applies since $S_2' = \{ 5, 6, 7, 8, 9, 10, 11 \} \neq \phi$. From the definition of $d(j,CS_t)$ in Table 3.2, we set up Table 3.9 to identify the seed to be selected.

It can be seen that customer 8 satisfies the seed selection rule of situation (3) in Table 3.3. So we have, $CS_3 = \{8\}$. We now select the last seed needed by setting up Table 3.10.

It is clear that customer 10 is selected as the seed needed, $CS_4 = \{10\}$. The four seeds are now shown in Figure 3.4.

Step 2: Assignment of vehicles to seeds

Clusters 3 and 4 need assignments to vehicles. We first set up the assignment cost Matrix [C] as follows:

	Cluster	
Vehicle	3	4
3	81	163
4	185	227

Following the greedy approach, Vehicle 3 is first assigned to cluster 3 and then vehicle 4 to cluster 4.

Table 3.10 Step 1: Customer 10 is selected as fourth and final seed

Customer j	$d(j,CS_3)$	$\min_{\text{all } t}$ ($d(j,CS_t)$)
5	154 + 128 = 282	282
6	315 + 380 = 695	225
7	208 + 520 = 728	358
9	366 + 336 = 702	203
10	165 + 348 = 513	459*
11	433 + 265 = 698	241

Table 3.11 Step 3.1: Determination of each cluster's starting workload

Cluster	Component (A)	(B)	(C)**	W_t=(A)+(B)+(C)
1	$dir(V_1,-1)$ = 185	dir(-1,-2) = 138	-10	313
2	$dir(V_2,-3)$ = 132	dir(-3,-4) = 190	+45	367
3	$dir(V_3,+8)$ = 81	dir(+8,-8) = 425	-85	421
4	$dir(V_4,+10)$ = 227	dir(+10,-10) = 255	-40	442

** We arbitrarily assume those workloads are carried over from previous time group.

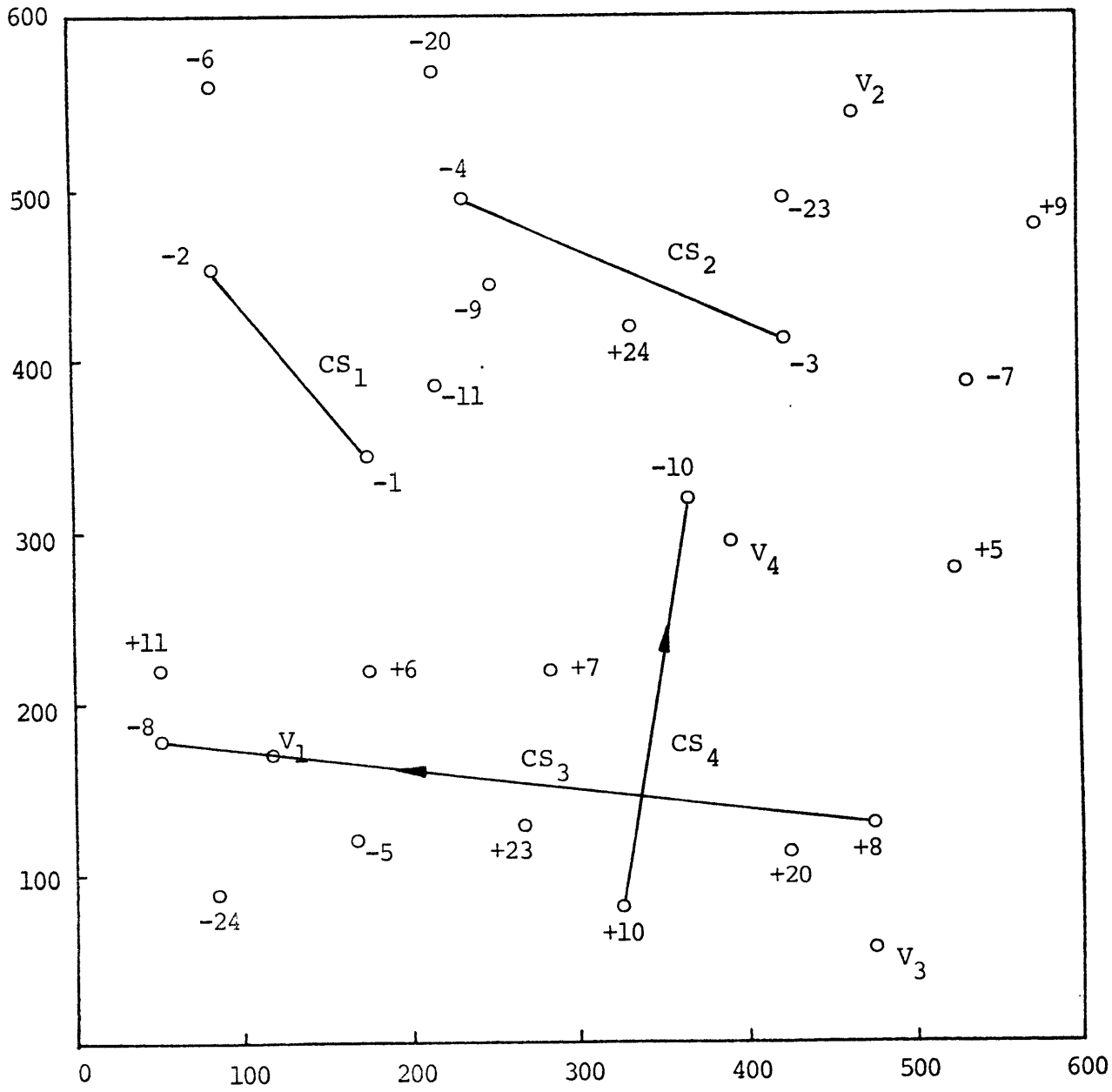


Figure 3.4
Customers selected as seeds for clustering

Step 3: Add remaining customers to clusters.

We follow the four iterative steps.

Step 3.1: Determine each cluster's starting workload.

This is performed in Table 3.11.

Step 3.2: For each cluster, find a candidate customer to join it.

We consider only category 2 customers at this stage. From the definition of $d(j,C_t)$ in Table 3.6, we have the situation displayed in Table 3.12.

It is clear that customers 5, 6, 7 and 9 are the candidates for inclusion in clusters C_3, C_1, C_4 and C_2, respectively, as indicated by the "*".

Step 3.3: Add a candidate to a cluster. As can be seen from Table 3.13, customer 6 is added to cluster C_1.

Step 3.4: $S_2' \neq \phi$, go to Step 3.2.

Step 3.2: Update the candidate list for cluster C_1. Table 3.14 shows that customer 11 in the new candidate for C_1.

Step 3.3: Add another candidate to a cluster. Table 3.15 shows that customer 9 in thus added to cluster C_2.

Step 3.4: $S_2' \neq \phi$, go to Step 3.2.

After repeating this process several times all category 2 customers are assigned to clusters in the manner shown in Figure 3.5. At that stage we have:

$C_1 = \{ 1 , 2 , 6 , 11 \}$, $W_1 = 879$

$C_2 = \{ 3 , 4 , 9 \}$, $W_2 = 570$

$C_3 = \{ 5 , 8 \}$, $W_3 = 703$

$C_4 = \{ 7 , 10 \}$, $W_4 = 755$

Table 3.12 Step 3.2: Candidate customers for inclusion in clusters are shown by "*".

j	$d(j,C_1)$	$d(j,C_2)$	$d(j,C_3)$	$d(j,C_4)$
5	588	559	282*	283 + 285 = 568
6	225*	451	695	202 + 357 = 599
7	540	358	728	140 + 173 = 313*
9	539	203*	702	471 + 162 = 633
11	241	401	698	297 + 154 = 451

Table 3.13 Step 3.3: Customer 6 is added to cluster C_1.

Cluster	Candidate i	W_t	$\Delta W_{t,i}$	Expression (1)
1	6	313	225	538*
2	9	367	203	570
3	5	421	282	703
4	7	442	313	755

Table 3.14 Step 3.2: Update candidate list for cluster C_1.

Customer 11 is the new candidate.

j	$d(j,C_1)$
5	351 + 440 = 791
7	113 + 476 = 589
9	464 + 197 = 661
11	124 + 217 = 341*

Table 3.15 Step 3.3: Customer 9 is added to cluster C_2.

Cluster	Candidate i	W_t	$\Delta W_{t,i}$	FW_t	$\Delta FW_{t,i}$	Expression (1)
1	11	538	341	0	0	879
2	9	367	203	0	0	570*
3	5	421	282	0	0	703
4	7	442	313	0	0	755

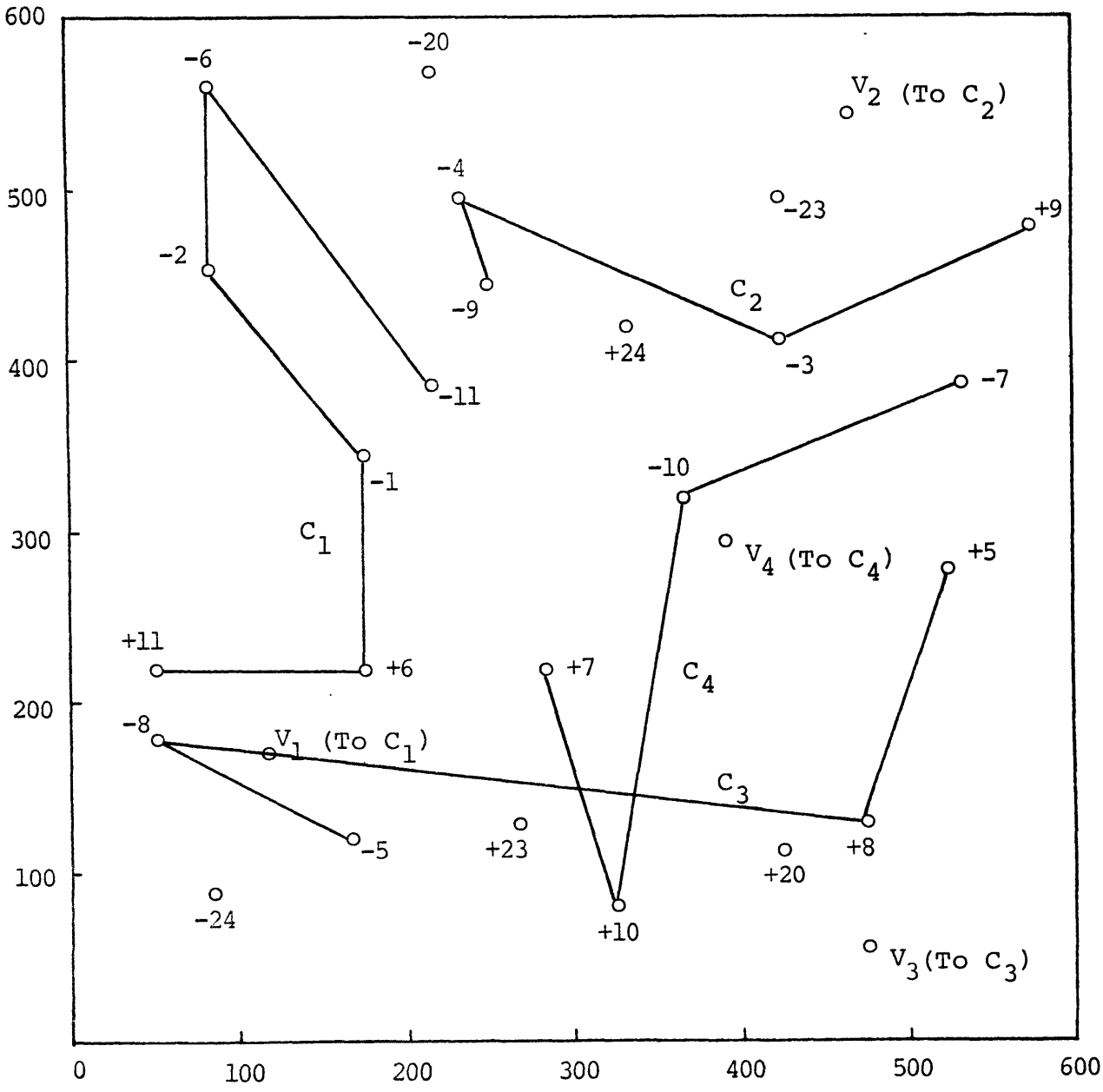


Figure 3.5

Category 2 customers are assigned to clusters

We next proceed to assign category 1 customers to clusters:

Step 3.2: Using the expressions in Table 3.7 we set up Table 3.16.

Step 3.3: (δ_j = 0.5 x (360,000/3)$^{1/2}$ = 173) Add another candidate to a cluster. This is done by setting up Table 3.17.

At this point, we have a three way tie at 879, the maximum workload. As noted in Section 3.3 the tie is broken by selecting the customer associated with the least $\Delta W_{t,i}$. Thus customer 20 is added to cluster C_3.

Step 3.4: $S_1' \neq \phi$, go to Step 3.2.

Step 3.2: Update the candidate list for cluster C_3. Table 3.18 shows that customer 23 is the new candidate for C_3.

Step 3.3: Add another candidate to a cluster. (Table 3.19).

Again, we use the tie-breaking rule of choosing the candidate associated with the minimum $\Delta W_{t,i}$ and assign customer 23 to cluster C_4.

Step 3.4: $S_1' \neq \phi$, go to Step 3.2.

Step 3.2: Update the candidate lists for clusters C_3 and C_4 (Tabel 3.20)

Step 3.3: Add another candidate to a cluster (Table 3.21). It can be seen that customer 24 is added to cluster C_2.

Step 3.4: $S_1' = \phi$, $S_2' = \phi$, stop.

We conclude the clustering procedure with the result shown in Figure 3.6:

C_1 = { 1 , 2 , 6 , 11 }

C_2 = { 3 , 4 , 9 , 24 }

C_3 = { 5 , 8 , 20 }

C_4 = { 7 , 10 , 23 }

Note that in Figure 3.6 how the disparate locations of the destinations of category 1 customers (20, 23 and 24) affected the

Table 3.16 Step 3.2: Candidate customers for inclusion in clusters are shown by "*".

j	$d(j,C_1)$	$d(j,C_2)$	$d(j,C_3)$	$d(j,C_4)$
20	265 + 173	301 + 173	57 + 173	108 + 173
23	137 + 173	321 + 173	102 + 173	85 + 173*
24	125 + 173	80 + 173	233 + 173	93 + 173

Table 3.17 Step 3.3: Customer 20 is added to cluster C_3.

Cluster	Candidate i	W_t	$\Delta W_{t,i}$	FW_t	$\Delta FW_{t,i}$	Expression (1)
1	24	879	125	0	0	1004
2	24	570	80	0	0	879
3	20	703	57*	0	0	879
4	23	755	65	0	0	879

Table 3.18 Step 3.2: Update candidate list for cluster C_3. Customer 23 is the new candidate.

j	$d(j,C_3)$
23	102 + 224*
24	233 + 485

Table 3.19 Step 3.3: Customer 23 is added to cluster C_4.

Cluster	Candidate i	W_t	ΔW_t	FW_t	ΔFW_t	Expression (1)
1	24	879	125	0	0	1004
2	24	570	80	0	0	879
3	23	760	102	0	224	879
4	23	755	65*	0	0	879

Table 3.20 Step 3.2: Update candidate lists for cluster C_3 and C_4.

j	$d(j,C_3)$	$d(j,C_4)$
24	233 + 485	93 + 531

Table 3.21 Step 3.3: Customer 24 is added to cluster C_2.

Cluster	Candidate i	W_t	ΔW_t	FW_t	ΔFW_t	Expression (1)
1	24	879	125	0	0	1004
2	24	570	80	0	0	879*
3	24	760	233	0	485	993
4	24	820	93	0	531	913

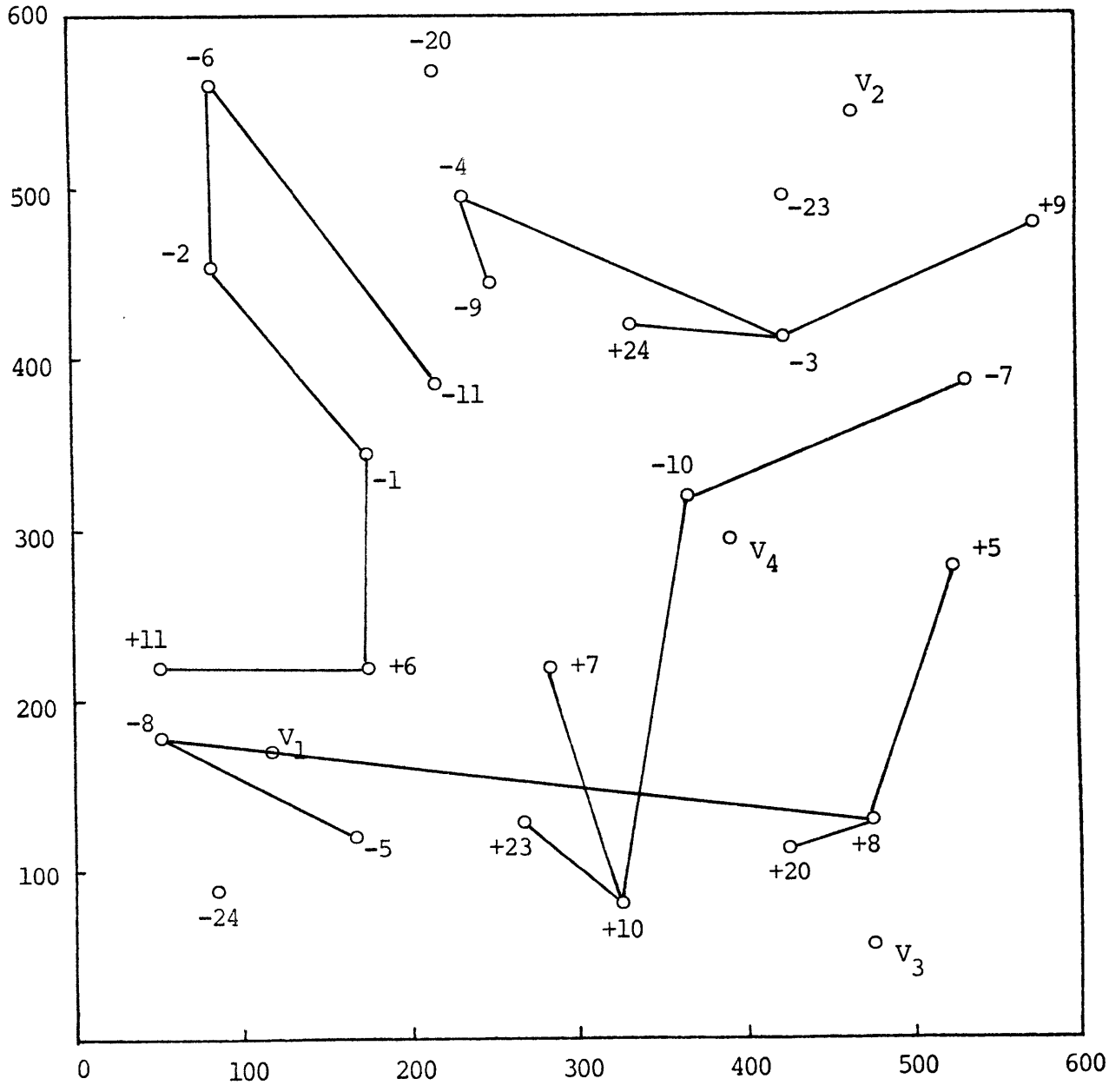


Figure 3.6
Final results of the clustering step.

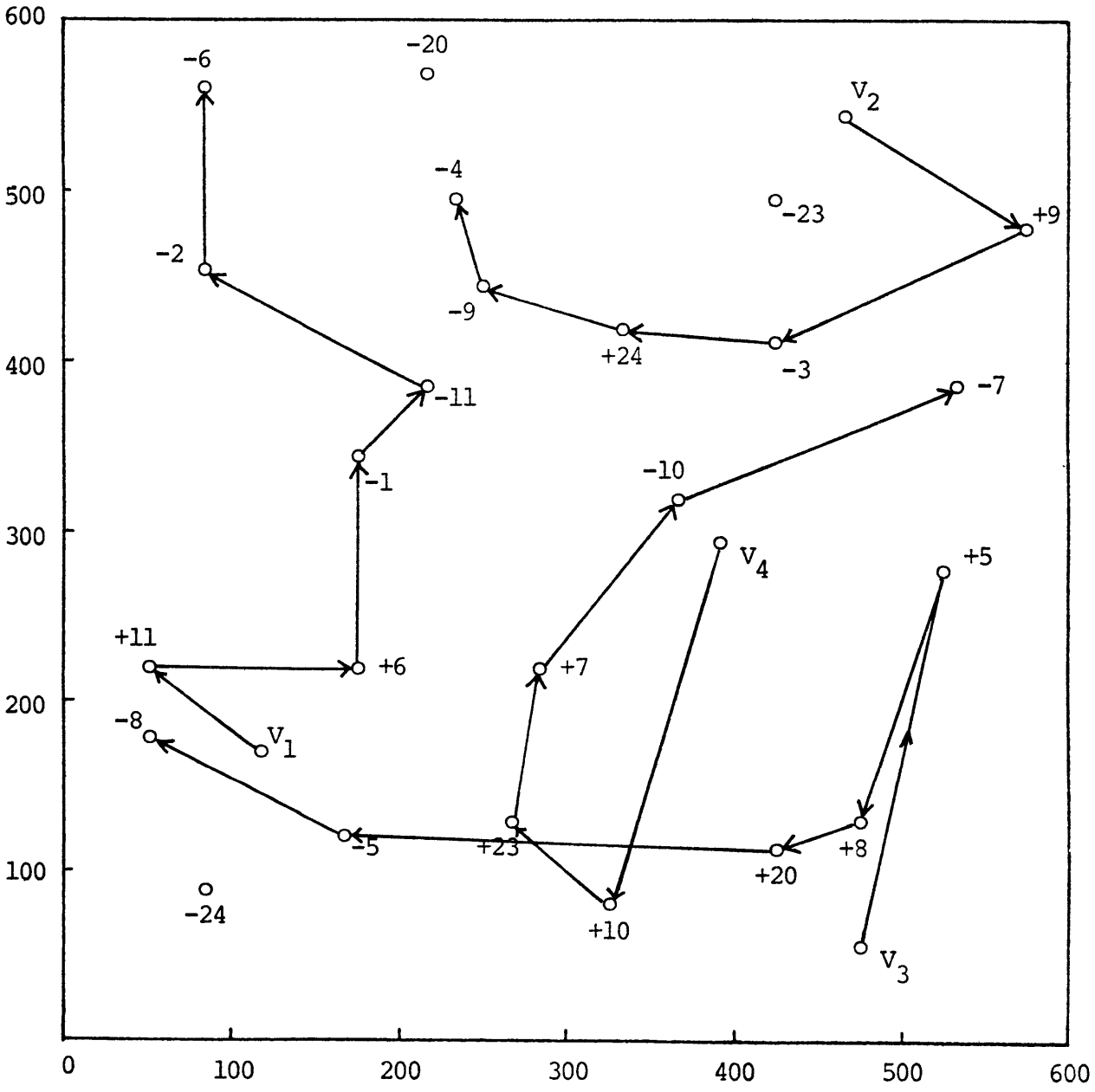


Figure 3.7
Vehicle routes after the routing step.

allocation of the origins of these customers to a different cluster for each. The routes which will eventually result from the clusters are provided in Figure 3.7. They are efficient and have good directivity characteristics.

3.6 Computational experience

This algorithm is coded in FORTRAN and has been implemented on VAX 11/750 minicomputer at M.I.T.

A series of computer runs was conducted to gain some insight on the performance and computational effort of the algorithm. They can be divided into two categories: (a) runs using simulated data and (b) runs using real data.

3.6.1 Runs using simulated data

Runs in this category involved simulations of several hours of service in a rectangular area of given dimensions. Without loss of generality, customer origins and destinations were assumed to be distributed uniformly and independently over the area of interest. Requests for service were distributed over the simulation interval at a user-specified (and not necessarily constant) rate per hour. Also specified externally were the number of available vehicles as a function of time, the desired length of time interval for grouping and the conversion factor CF.

The routing option exercised in all those runs was the MST heuristic followed by the 3-opt breadth-first search procedure without the tardiness option (which could be used for further minimizing the number of late deliveries). Table 3.22 presents sample results from this series of simulation runs. All runs involved 4 simulated hours of service in a 6 x 6 square mile geographical area. All vehicles were initially located at the center of the square and their speed was 15 mph. all runs assumed that the probability of a random customer having specified a desired pick-up time (as opposed to a desired delivery time) was 0.5. Those times are assumed uniformly and independently distributed within each hour of the 4-hour simulation interval. Delivery time deviations (positive for lateness and negative for the earliness) were evaluated only for customers who had specified a desired delivery time (similarly for pick-up-specified customers). In Table 3.22 we present various statistics of system and algorithm performance as functions of the following four parameters:

1. Customer demand rate: 100, 300 and 500 customers/hour.
2. Number of vehicles used (assumed here constant throughout the 4-hour interval): 10, 20, 30, 40, 50 and 60 vehicles.
3. Length DT of grouping interval: 30 and 60 minutes.
4. Conversion factor CF: 1.0 and 1.5.

(It should be mentioned that all runs referring to the same customer demand rate (e.g. 100) correspond to the same customer origins, destinations and desired pick-up and delivery times. Also, DT and CF were varied for calibration purposes.)

Although the runs shown in Table 3.22 are not necessarily representative of the full potential of the algorithm, they seem to

Table 3.22 Sample test runs of GCR algorithm on Vax 11/750

No. of customers in 4 hours	400 (seed #57)										1200 (seed #234)				2000 (seed #98)		
No. of vehicles	10				20				30		30	40	50		50		60
DT (mins)	30		60		30		60		30	60	60	60	30	60	30	30	30
CF	1.0	1.5	1.0	1.5	1.0	1.5	1.0	1.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.5	1.5
No. of late deliveries	202	197	90	160	102	118	32	25	42	17	120	81	263	66	803	459	141
Average deviation given late delivery (mins)	62	56	18	24	10	15	12	12	6	11	10	8	9	7	19	12	5
Average deviation given early delivery (mins)	-8	-8	-21	-13	-8	-8	-31	-27	-13	-38	-25	-29	-9	-30	-7	-11	-14
No. of late pick-ups	162	167	32	105	43	98	10	25	19	8	33	43	156	20	667	458	159
Average deviation given late pick-up (mins)	63	51	11	19	8	10	6	10	6	7	7	6	8	5	17	12	7
Average deviation given early pick-up (mins)	-8	-9	-27	-16	-11	-9	-37	-30	-15	-39	-33	-35	-11	-34	-9	-13	-15
Average ride time ratio	1.99	2.15	2.38	2.69	1.54	1.74	1.84	1.86	1.30	1.50	2.72	1.88	1.47	1.74	1.64	1.79	1.62
Productivity (pass/veh. hr)	6.46	6.69	9.54	8.98	5.15	4.86	5.63	6.18	3.62	3.96	10.4	8.22	6.02	6.72	8.13	8.23	7.38
CPU time (mins)	0.91	2.12	4.36	4.78	0.52	1.07	1.30	1.50	0.62	1.17	15.7	12.9	6.29	12.3	15.3	15.2	15.7

support the following general conjectures:

1) With the possible exception of the average ride time ratio which predictably increases when CF increases, there seems to be no overwhelming evidence in favor of or against using a specific value for CF. None of the remaining statistics of Table 3.22 is an unambiguously monotonic (either increasing or decreasing) function of CF.

2) By contrast, the length DT of the grouping time interval seems to be an important calibration parameter. The first observation is counter-intuitive: The number of late deliveries decreases when DT goes from 30 minutes to 60. A possible explanation of this behavior may be that the algorithm is less successful in linking adjacent time groups if their duration is very short and thus its performance deteriorates when the number of time groups increases. Other quantities that decrease when DT increases are the average deviation given a late delivery, the number of late pick-ups and the average deviation given a late pick-up. By constrast, the number of early deliveries, the average deviation given early delivery, the number of early pick-ups, the average deviation given early pick-up, the average ride time ratio as well as the overall vehicle productivity and CPU time increase when DT increases.

It is conjectured that these trends may not necessarily remain the same if DT increases substantially more than 60 minutes.

3) The behavior of the system as the number of vehicles increases is predictable. All "service" parameters ameliorate if more vehicles are present but productivity goes down. However, the CPU time is not necessarily a monotonic function of the fleet size. The reason seems to be the tradeoff in computational effort between the clustering and routing parts of the algorithm.

3.6.2 Runs using real data:

The most interesting test of this algorithm to date has been with data associated with the advanced-reservation flexible-route system operated by Rufbus GmbH Bodenseekreis in the city of Friedrichshafen, West Germany. The particular database used to test the algorithm included information covering approximately 16 hours of operation of the system. Of the 2,617 customers of the data base, 2,397 were pick-up-specified, while the remaining 220 were delivery-specified. Some of the pick-up-specified customers in the database were actually "immediate-request" customers by defining their DPT as the time of the request (Such a conversion of course has certain implications regarding our ability to compare results obtained from the algorithm with results from the actual Rufbus schedules).

The direct trip times between all possible vehicle stop pairs were part of the database. Vehicle stops were located at "checkpoints" across the entire Rufbus system. It should be noted here that due to one-way streets and other "peculiarities" of the system, the direct trip time matrix was not symmetric. The database also included information on the actual vehicle schedules for the period of interest with a 28-vehicle fleet, consisting of one 33-passenger vehicle, four 9-passenger vehicles, and twenty-three 17-passenger vehicles.

Applying the algorithm on the above database presented some initial difficulties: For instance, the direct trip time matrix was not symmetric. Since this algorithm is designed for symmetric matrices, we replaced every

entry D_{ij} of the original matrix with $0.5 \cdot (D_{ij} + D_{ji})$. A second handicap we had to face was that it was impossible to identify the locations of customer origins and destinations (say, on a city map) just by looking at the trip time matrix. Our inability either to observe, or, a fortiori, interactively to improve upon the results of the algorithm by looking at visual display of either clusters or routes, put us at a disadvantage in comparison to Rufbus schedulers, who had full access to (and fully exploited) such information. In addition, the lack of such information forced us to abandon the use of the MST heuristic in the routing part of the algorithm, since this would require information on the relative orientation of customer locations, which was not available. The routing option exercised instead was a nearest-neighbor procedure followed by a 3-opt breadth-first search.

Despite these difficulties we decided to test the algorithm with this database since it is the only database with a large number of customers that we were able to obtain.

Statistics of six different runs of our algorithm together with Rufbus' own performance are displayed in Table 3.23. In all runs DT and CF are set to be 30 minutes and 1.0 respectively. The number of vehicles specified in the runs was varied throughout the service period quasi-arbitrarily (matching the demand for service) and having a peak of 21 or 22 vehicles. From the table we can also observe that the algorithm achieves productivities ranging from about the same as, to about 20% higher than that achieved in the actual schedule. This was accomplished with about a 22% reduction in the fleet size and with no serious

Table 3.23 Rufbus database runs. For explanations see text.

	Actual Schediling	Run #1	Run #2	Run #3	Run #4	Run #5	Run #6
Fleet size (maximum)	28	22	21	21	22	21	22
No. of late deliveries	0	148	130	104	92	57	64
Average deviation given late delivery (mins)	0	15	13	12	11	10	8
Average deviation given early delivery (mins)	-6	-10	-10	-12	-14	-15	-15
No. of late pick-ups	2337	1250	1237	1222	973	763	639
Average deviation given late pick-up (mins)	13	13	17	14	11	13	10
Average deviation given early pick-up (mins)	-3	-11	-12	-12	-13	-13	-14
Average ride time ratio	N.A.	2.3	2.3	2.2	2.2	2.1	2.2
Productivity (pass/veh. hr)	8.87	10.52	10.53	9.93	9.74	9.26	8.97

N.A. : Not available

deterioration in service quality. Running time in all runs averaged between 6 and 7 CPU minutes on a VAX 11/750.

We consider this experience in testing this algorithm with realistic data particularly encouraging, especially since the Rufbus database had features that did not match the design assumptions of the algorithm. It is believed that the performance of the algorithm would certainly be much better if it were tailored to the special conditions of the database, as is usual when an algorithm is implemented on a real system.

3.7 Computational Complexity of the Algorithm

Table 3.22 has shown some empirical computation times required for the algorithm to solve a range of medium to large size problems. In this section we investigate the computational complexity of the algorithm from a theoretic point of view.

Since the algorithm consists of three independent sequential steps, grouping, clustering and routing, we can analyze its complexity by addressing each step separately. For the following discussion we assume that the problem deals with N customers whose desired service times are distributed equally among t time groups and there are m vehicles in each time group.

(i) Grouping

The grouping procedure determines which customer's desired service time falls in which time group. The effort involves the comparison between each customer's desired pick-up and delivery times with the starting and ending times of every time group. For one customer this procedure can take at most $2 \cdot t$ comparisons. Thus the total number of comparisons required to process N customers can only be as large as $2 \cdot t \cdot N$. In other words the required effort for the grouping step increases linearly with the number of customers.

(ii) Clustering

The clustering procedure divides the set of customers in each time group into smaller clusters. Let n denote the number of customers in a given time group. According to the clustering procedure, if m vehicles are in service, m clusters must be formed. At the start of clustering, we select seeds for each of the m clusters. The selection of each seed takes O(n) comparisons and additions. The additions refer to the calculation of customer-to-seed distances. The comparisons refer to the selection of the customer who maximizes the minimum-customer-to-seed distances. The total effort required to select m seeds is therefore O(mn). After seed customers have been selected, the clustering step continues with the assignment of the rest of the customers to clusters. The initialization of the customer-to-cluster distance matrix requires O(mn) work. When one customer is assigned to a cluster an update of the distance matrix requiring O(n) effort is needed. For n customers, the updating effort thus increases as $O(n^2)$. In summary the clustering step involves $O(mn+n^2)$ effort.

(iii) Routing

The routing step uses one or a combination of two single-vehicle routing algorithms to find the optimal route. Therefore, the computational complexity of the routing step depends on which routing algorithm is used. For ease of discussion we assume that the n customers in the given time group are evenly distributed among m vehicles, i.e. a vehicle would carry n/m customers. Assume now that a 2-opt tour-improvement procedure is used for routing. Such a 2-opt procedure involves computational effort that increases in proportion to the square of the number of customers in the route (see Psaraftis [28]). As a result, the computational complexity of finding m routes for m vehicles is $O(m \cdot n^2/m^2)$. (If customers are not evenly distributed among clusters, i.e., one cluster has most of the customers, the total routing effort would be at most of order n^2.)

From the above discussion, we see that the computational complexity of the GCR (denoting Grouping, Clustering, and Routing) algorithm depends mainly on the routing algorithm used. If a $O(n^2)$ routing algorithm is used, the complexity of the clustering and the routing steps together is $O(mn+n^2)$, where n is the average number of customers in a time group. If all N customers of the problem are in one time group, the overall computational complexity is at most $O(mN+N^2)$. It can now be concluded that the algorithm is a polynomial-time algorithm.

Further discussion of this Grouping-Clustering-Routing algorithm (referred as "GCR algorithm" from now on) is provided in Chapter 6.

CHAPTER 4 AN ALGORITHM FOR DARP WITH STRICT SERVICE QUALITY CONSTRAINTS

4.1 Introduction

This chapter describes a second heuristic algorithm, Advanced Dial-A-Ride With Time Windows (ADARTW), for the advance-request version of the multi-vehicle DARP with service-quality constraints that include time windows. Service quality constraints - as described in detail in Section 4.2 - refer to guarantees that (i) each customer's ride time will not exceed a pre-specified maximum and (ii) the time of pick-up or delivery of a customer will not deviate from the most desired time by more than a pre-specified amount ("the time windows").

The algorithmic approach to be described is interesting for two reasons. First, it addresses what is probably the most applicable and realistic version of the real-world problem in a way that avoids excessive abstraction and simplification. It also emphasizes flexibility and convenience to the users of the algorithm. Second, it can generate at low computation cost what are apparently good-quality solutions to problems of much larger size (e.g., 2,500 customers and over 30 simultaneously active vehicles) than have hitherto been attempted in this area.

ADARTW derives some of its fundamental concepts from the pioneering work of Wilson et al. [42,43]. It builds up vehicle tours through sequential insertion of customers and uses a nonlinear objective function to guide such insertions. A recent paper by Solomon [34] presented substantial evidence supporting the use of insertion-oriented algorithms

for vehicle routing and scheduling problems having time window constraints. Our experience with ADARTW has also been very favorable. Chapter 5 will report all our computational findings.

This chapter focuses on the structure of ADARTW and the heuristic techniques that it uses. These are to a large extent dictated by the operating scenario within which the algorithm has been conceived. This scenario is described in Section 4.2. Table 4.1 defines the mathematical notation. Section 4.3 then presents an overview of ADARTW, Section 4.4 describes the search for feasible insertions of customers into vehicle work-schedules and finally, Section 4.5 the optimization procedure used to assign customers to vehicles and to fix pick-up and delivery times.

4.2 Operating Scenario

The dial-a-ride system with which ADARTW is designed to operate is assumed to have a number of characteristics. The most important among those is that each of the system's customers is asked and willing to specify either a desired pick-up time (DPT) at his origin or a desired delivery time (DDT) at his destination, but not both. This implies that the customer is able to decide for himself whether he is constrained by a pick-up time or by a delivery time in his intended trip. For example, most individuals are constrained in the morning by a desired "delivery" time (e.g., the time one has to arrive at the work place) and adjust their "pick-up" time (e.g., the time when they leave their homes) accordingly. In a similar fashion, a dial-a-ride system's customer who specifies a desire to be delivered at a commuter train station (or an outpatient

Table 4.1 : Definitions of quantities not explicitly defined elsewhere in the text.

N : the number of customers on the subscriber list

m : the number of available vehicles

DPT_i (DDT_i) : the desired pick-up (delivery) time of customer i

EPT_i (EDT_i) : the earliest possible time at which the pick-up (delivery) of customer i can be made

LPT_i (LDT_i) : the latest possible time at which the pick-up (delivery) of customer i can be made

APT_i (ADT_i) : the time when the pick-up (delivery) of customer i will actually take place according to the schedule

DRT_i : the time it would take a vehicle to go directly from the origin to the destination of customer i, [$DRT_i = D(+i,-i)$]

ART_i : the actual ride-time of customer i , $ART_i = ADT_i - APT_i$.

D(x,y) : the time it takes a vehicle to go from point x to point y (using fastest route)

+i (-i) : the event "pick-up (deliver) customer i"; the indication "+i" ("-i") is also used to denote the point of origin (destination) of customer i

DV_i : the deviation in the time schedule of customer i from his desired pick-up or delivery time [for DPT-specified customers $DV_i = APT_i - DPT_i$; for DDT-specified customers $DV_i = DDT_i - APT_i$]

WS_i : the maximum acceptable deviation of customer i from his desired pick-up or delivery time. ($DV_i \leq WS_i$)

d : the number of stops (pick-ups and deliveries) in a schedule block.

$SLACK_j$: the duration of vehicle slack time before schedule block j. If there are n schedule blocks, $SLACK_{n+1} = \infty$.

clinic) by time X, will be a "DDT-specified" customer. In our system, this customer would rely on the operator of the system to tell him at what time he will be picked up from his origin so that he can be delivered at his destination by time X. The reverse is, of course, true for DPT-specified customers (e.g., "I can be picked up from the shopping mall at 2 P.M. for transportation to my house"). Normally, one would expect a preponderance of DDT-specified customers in the morning and DPT-specified customers in the afternoon and evening.

A second and related assumption is that DPT-specified customers will be asked to give as their DPT, the earliest time at which they can be picked up. Similarly, DDT-specified customers will give the latest time at which they can be acceptably delivered at their destination as their DDT. This actually implies no loss of generality, but is particularly convenient for the algorithm, since the time-window during which a DPT-(DDT-) specified customer can be picked up (delivered) can be defined as beginning (ending) with the specified DPT (DDT).

In the dial-a-ride system in question one would certainly expect a commitment to quality of service on the part of the system's operator. Consider, for instance, a DDT-specified customer who lives 15-minutes away (by car) from a community center and who desires to be at that center by 10 A.M. It would clearly be unreasonable to deliver that customer at the center at, say 8 A.M. or to offer him a 90-minutes circuitous ride - picking up and/or delivering many other customers on the way. For these reasons it will be assumed that the system will operate under three types

of service quality constraints :

(i) No DPT- (DDT-) specified customer will be picked up (delivered) earlier (later) than his DPT (DDT).

(ii) No customer's actual ride time will exceed a given maximum ride time for that customer - the maximum ride time being specified as a function of the direct origin-to-destination ride time for that customer.

(iii) The difference ("time deviation") between the actual pick-up (delivery) time and the desired pick-up (delivery) time of a customer will not exceed a given maximum for DPT- (DDT-) specified customers.

The values of the maximum ride-time and of the maximum time-deviation can either be determined unilaterally by the system's operator and applied universally or, can be left open to negotiation between the operator and each individual customer. In the former case, an operator might advertise, for example, that a customer's ride time would "under no circumstances" exceed twice his direct ride time and that he would be delivered (picked up) no earlier (later) than 20 minutes prior to (after) his desired delivery (pick-up) time.

The problem

The version of DARP solved by ADARTW can now be summarized as follows: Given a subscription list of N customers, each specifying either a DPT_i or DDT_i ($i = 1,2,\ldots,N$) and a fleet of m vehicles, find an

effective allocation of customers among vehicles and an associated time schedule of pick-ups and deliveries such that:

1. For all customers i:

$$ADT_i - APT_i \leq MRT_i \tag{4-1}$$

2. For DPT-specified customers:

$$DPT_i \leq APT_i \leq DPT_i + WS_i \tag{4-2}$$

3. For DDT-specified customers:

$$DDT_i - WS_i \leq ADT_i \leq DDT_i \tag{4-3}$$

Several comments are in order concerning this formulation:

(a) We have not yet specified what is our measure of effectiveness (see Section 4.3 and 4.5).

(b) It may prove <u>infeasible</u> to serve some of the N customers with the given vehicle resources and service-quality constraints.

(c) MRT_i, the maximum ride time for customer i, will normally be specified as a function of the direct ride time, DRT_i. In our work we have used:

$$MRT_i = A + B \cdot DRT_i \tag{4-4}$$

where A and B are user-specified constants (e.g. A = 5 minutes , B = 1.5). A reasonable alternative might be:

$$MRT_i = \begin{cases} DRT_i + A & \text{if } DRT_i \leq T_0 \\ B \cdot DRT_i & \text{if } DRT_i > T_0 \end{cases}$$

where, again, A, B and T_0 are operator-specified constants (e.g., A = 10 minutes, T_0 = 20 minutes, B = 1.5) satisfying the relationship $T_0+A=B \cdot T_0$. Other functional forms can, of course, be used to specify MRT_i, if desired.

In conclusion, ADARTW is designed for use under the following

scenario: The system's operator would advertise the dial-a-ride service and the service-quality guarantees that will be offered. Customers will call and specify origin, destination and a desired DPT (= EPT) or DDT (= LDT). A subscription list will be compiled up to a closing time (no sign-ups accepted after that time). With the data available on the subscription list and with the assistance of ADARTW, the system's operator will then prepare a detailed work-schedule for each of the vehicles, listing times and locations for each of the pick-ups and deliveries. These work-schedules will be distributed to the drivers of the vehicles. Each customer will also be called and given an (approximate) APT and ADT, satisfying constraints (4-1) and (4-2) or (4-1) and (4-3) as the case may be. Those customers, if any, whom it is infeasible to serve will also be so notified.

Before proceeding to the description of the algorithm a number of additional assumptions in our scenario will now be listed. While all these assumptions are natural ones, they are mentioned here because they create complications that some existing DARP algorithms cannot deal with:

a) The capacity of vehicles is assumed finite and is not necessarily the same for all vehicles.

b) Dwell times - the amount of time needed to pick up and deliver customers - can be non-zero and different customers are allowed to have different dwell times. In our following discussion, we assume that the dwell times are all zero for every customer. It should be noted that non-zero dwell times can be handled by adding them to the distance matrix in a way consistent with the definitions of APT and ADT.

c) A vehicle is not allowed to wait idly when it is carrying passengers. For example, it is not permissible in ADARTW to have a vehicle, with one or more passengers on board, arrive at the pick-up point of a DPT-specified customer i prior to DPT_i and then wait idly until time DPT_i to pick up i (remember that due to (4-2) customer i cannot be picked up earlier than DPT_i). Such idle waiting periods by non-empty vehicles are accepted by some vehicle-routing algorithms with time window constraints (see Sexton and Bodin [32]). It is felt, however, that, in practice, such idle waiting would not be tolerated by dial-a-ride customers and ADARTW has been designed accordingly. Actually, this can be viewed as a fourth service-quality constraint, imposed in addition to (i) - (iii) above.

Finally, it is useful to define availability periods, active periods and slack periods for vehicles. As shown in Figure 4.1 for a particular vehicle j, a vehicle can be unavailable during periods of a day (usually due to driver constraints, labor union agreements or vehicle maintenance requirements). An available vehicle can be either in a slack period (i.e., waiting idly) or it can be active (on the way to pick up its first customer during an active period, transporting, picking up or delivering customers or returning to a depot). Note that because of asssumption c) above, a vehicle cannot be in a slack period as long as it has even one customer on board.

4.3 Overview of the Algorithm

ADARTW is a heuristic algorithm that processes ride requests

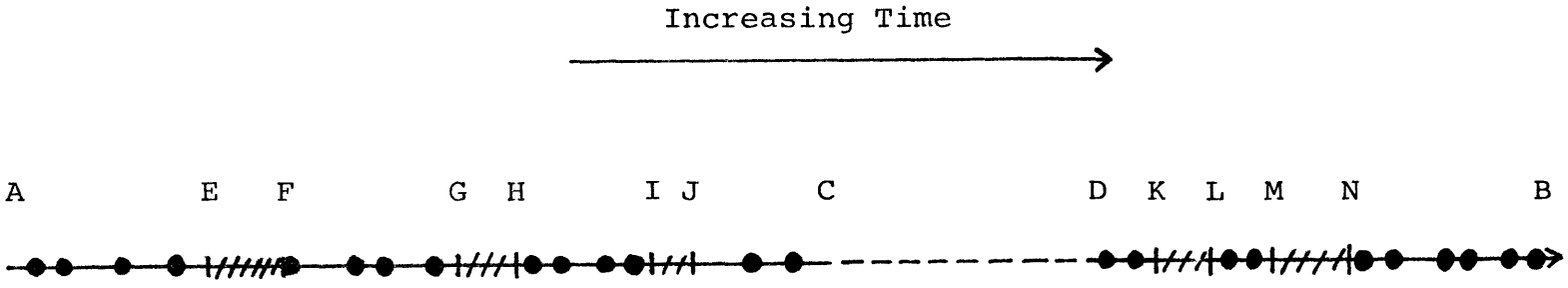


Figure 4.1

A typical schedule of a vehicle during a service period AB.

The vehicle is available during AC and DB and unavailable during CD. The vehicle leaves a depot at time A and returns there at time C, leaves again at D and returns at B. Intervals AE, FG, HI, JC, DK, LM, NB are active periods while EF, GH, IJ, KL, and MN are slack periods. Black dots within active periods indicate pick-up or delivery of a customer.

sequentially, inserting one customer at a time into the work-schedule of some vehicle until all ride requests have been processed. This section describes in qualitative terms how this procedure works.

Central to the process of assigning customers to vehicles are : a search for feasible insertions of customers into work-schedules; and a sequence of optimization steps designed to find the most desirable one among all the feasible insertions on each occasion. An insertion of a particular customer i into the work-schedule of a specific vehicle j is feasible only if it does not lead to violation of any service-quality constraints for customer i and for all other customers already assigned to vehicle j. The optimization steps deal with minimizing the additional "cost" due to inserting customer i into a vehicle's work-schedule. The cost function that we use is a weighted sum of disutility to the system's customers (due to excess ride times and to deviations from tha most desirable pick-up or delivery times) and of system costs as represented by a function that quantifies the "consumption" of available vehicle resources.

Consider now a case in which there are N (advance-request) customer demands for service and m available dial-a-ride vehicles. ADARTW begins by indexing customers in the order of their "earliest pick-up times", EPT_i ($i = 1,2,\ldots,N$), i.e. according to the earliest time at which they are expected to be available for a pick-up. The first customer in the sequence is the one with the smallest EPT_i (Section 4.4 shows how the EPT_i are computed).

The algorithm then processes the first, second, third, etc. customers in the list, one at a time (see also below), and assigns each customer to a vehicle until the list of customers is exhausted. The processing of customer i goes as follows:

Step 1 : For each vehicle j (j = 1,2,...m),

(i) Find all the possible ways in which customer i can be inserted into the work-schedule of j (details in Section 4.4). If it is infeasible to assign customer i to vehicle j, examine the next vehicle j+1 (re-start Step 1); otherwise:

(ii) Find the insertion of customer i into the work-schedule of vehicle j that results in minimum additional cost (details in Section 4.5). Call this additional cost, $COST_j$.

Step 2 : If it is infeasible to insert i into the work-schedule of any vehicle, then declare i a "rejected customer"; otherwise assign i to a vehicle j^* for which $COST_{j^*} \leq COST_j$ for all j = 1,2,...,m.

The above is only the "generic" version of the algorithm. We have, in fact, developed a number of options which are available at various points in the procedure:

(a) Customers can be indexed and processed according to criteria other than EPT. For example, one can also process customers "backward" by ordering them according to their latest delivery time, LDT - with the customer having the largest LDT being processed first (see Section 4.4 for computation of LDT). Such alternative processing orderings can be used to

generate several alternative solutions to any given problem instance.

(b) Instead of processing one customer at a time, the user can specify how many (yet unassigned) customers should be considered in Step 1 above. For example if the number "5" is specified, the top 5 (in terms of their ordering index) unassigned customers will be considered on each occasion as candidates to be assigned next to a vehicle. It should be emphasized that each of the candidates is considered separately in Step 1, so that in Step 2 the one among (the 5, in our example) candidates with the smallest $COST_j^*$ will be assigned to vehicle j^*. We can term such a group of candidates as the candidate "pool" from which the next customer for insertion is selected. There exist two possible strategies for bringing unassigned customers into the pool. One strategy would be to bring a new customer into the pool every time a customer in the pool is inserted into some vehicle's work-schedule. In this way, a pool always maintains the same number of candidate customers in it. A customer will leave the pool either when he is selected for next insertion in Step 2 or when it is infeasible for any vehicle to carry him. Such a strategy is termed "immediate-refill". An alternative is to let the candidate pool become smaller and smaller as customers in the pool are inserted, one at a time, to vehicle work-schedules. When the pool is finally exhausted, refill it with the next batch of customers in the list. This strategy is termed "periodic-refill". Computational experience with both strategies will be reported later in Chapter 5.

This multi-candidate option is provided for the purpose of improving the performance of the algorithm by making it less "myopic", i.e. by giving it an opportunity to select among more than one customer

for the next assignment. The penalty, of course, is that as the number of candidates that are considered each time increases the efficiency of the algorithm decreases.

(c) If the user so desires, ADARTW will not reject any customers. Instead, if it ever proves infeasible to assign a customer i to any of the m initially available vehicles, an option of ADARTW will introduce an additional vehicle to serve that customer. This additional vehicle will join the fleet of vehicles from that time on and will be available to serve subsequent customers. If at some later time, it turns out that customer k cannot be feasibly introduced into the work-schedule of the m + 1 available vehicles an additional vehicle, m + 2, will be introduced, and so on.

4.4 Search for Feasible Insertions

We now turn to an outline of the steps taken to identify feasible insertions of customers into vehicle work-schedules. A notion which plays an important role in this respect is that of a "schedule block". This can be illustrated through Figure 4.2, which shows part of the work-schedule of a vehicle j. A schedule block is a period of continuous active vehicle time between two successive periods of vehicle slack time. A schedule block always begins with an empty vehicle starting (either from a depot or after an inactive period) on its way to pick up a customer and ends with a period of slack time or at the end of the vehicle's entire work-schedule. It should not be confused with the "Base-Trip" definition as described in Hung [15] which is a period of active vehicle time between two consecutive

vehicle-is-empty statuses. In the case of a schedule block, a vehicle might become empty several times before the schedule block ends. Associated with a schedule block is a "schedule sequence" indicating the sequence of stops in the block and a "time schedule" indicating the time when each stop is scheduled to take place. For example in Figure 4.2, the schedule sequence associated with the middle schedule block is {+k, +m, -m, +n, -k, -n} while the time schedule is {APT_k, APT_m, ADT_m, APT_n, ADT_k, ADT_n}.

Suppose now that we wish to examine whether a yet-unassigned customer i can be inserted into the work-schedule of vehicle j. The objectives of the search for feasible insertions are:

(i) To identify new feasible schedule sequences.

(ii) For each of the feasible schedule sequences to obtain a feasible time schedule and its associated bounds within which the time schedule can be advanced or delayed and yet remain feasible.

ADARTW accomplishes these objectives by examining systematically and efficiently all possible schedule sequences associated with each and every schedule block on the work-schedule of vehicle j. For example, with respect to the middle schedule block of Figure 4.2, the possible schedule sequences involving the insertion of customer i are {+i, -i, +k, +m, -m, +n, -k, -n}, {+i, +k, -i, +m,...., -n}, ..., {+k, +m,...., -n, +i, -i}. In all, if there are $\underline{d}$ stops already on a schedule block, there are (d+2)(d+1)/2 possible schedule sequences that involve the insertion of a new customer into that schedule block, while adhering to the constraint

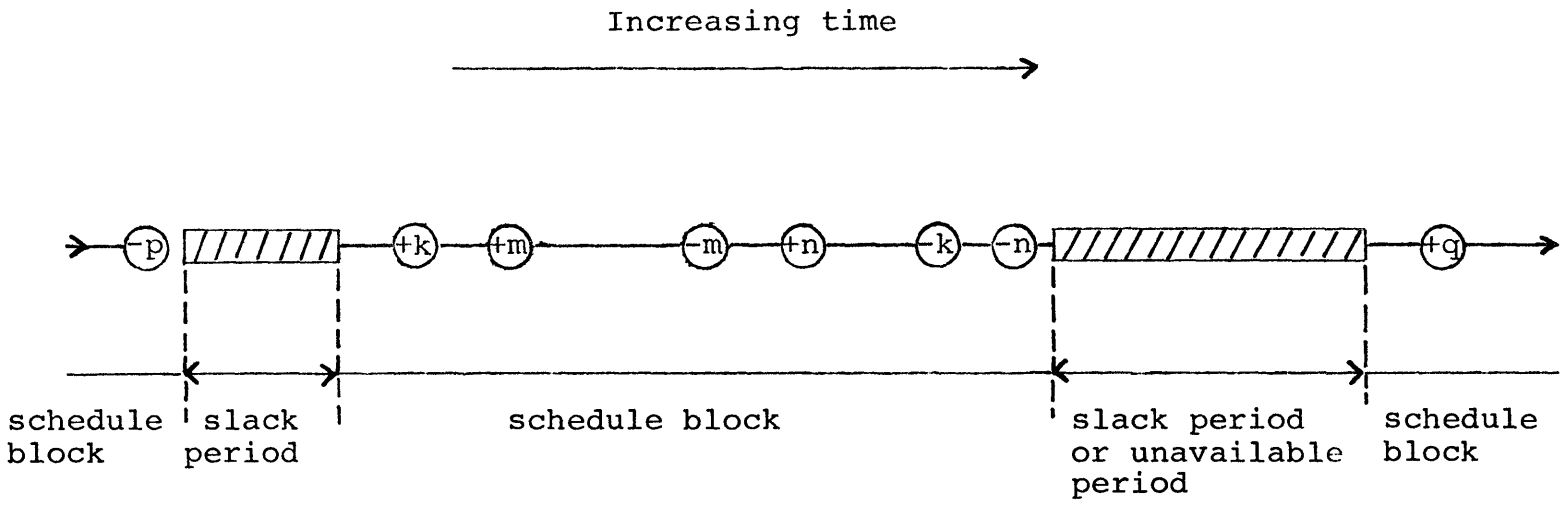


Figure 4.2

Part of the work-schedule of a vehicle j.

that the "+i" stop must precede the stop "-i". In view of the maximum ride time and maximum time-deviation constraints of our version of DARP (relations (4-1) - (4-3)) only some (and possibly none) of the above possible sequences may be feasible.

More schedule sequences are possible if we consider the insertion of stop "+i" and stop "-i" into different - not necessarily consecutive - schedule blocks. Such insertions will result in merging all the schedule blocks between (and including) the ones where the pick-up and delivery are inserted. ADARTW can consider both types of insertions, but uses different methods (see below) to test the feasibility of schedule sequences formed.

It should be noted that, in addition to inserting customer i into one of the already existing schedule blocks of any vehicle j, ADARTW will, naturally, consider creating an entirely new schedule block for vehicle j, as shown in Figure 4.3, in order to accommodate customer i. For example, the first customer ever assigned to a vehicle will obviously always create a new schedule block. Such new schedule blocks are added to the list of existing schedule blocks to which ADARTW attempts to add new customers through the insertion procedure.

A Fast Screening Test For Insertions Within The Same Schedule Block

To facilitate the feasibility search, ADARTW includes a procedure that increases greatly its efficiency and is fundamental to its viability in dealing with large-scale problems. This procedure involves defining two

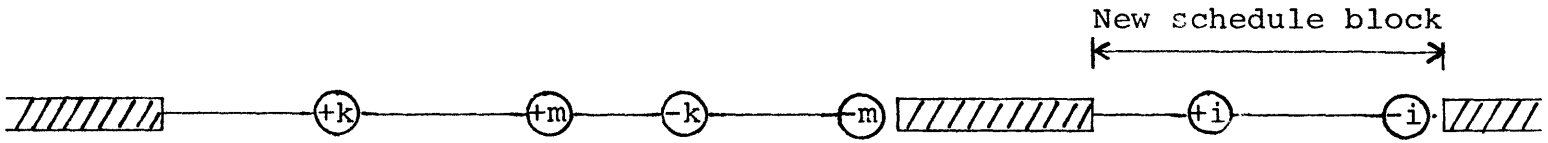


Figure 4.3

A new schedule block may be created for customer i

time windows for each customer as follows:

For DPT-specified customers, we define:

$$EPT_i = DPT_i \tag{4-5}$$

$$LPT_i = EPT_i + WS_i \tag{4-6}$$

$$EDT_i = EPT_i + DRT_i \tag{4-7}$$

$$LDT_i = LPT_i + MRT_i \tag{4-8}$$

These earliest and latest pick-up and delivery times are shown in Figure 4.4(a). Note that (4-5) and (4-6) contain the same information as (4-2).

Similarly, for each DDT-specified customer, we define (see also Figure 4.4(b)) :

$$LDT_i = DDT_i \tag{4-9}$$

$$EDT_i = DDT_i - WS_i \tag{4-10}$$

$$LPT_i = LDT_i - DRT_i \tag{4-11}$$

$$EPT_i = EDT_i - MRT_i \tag{4-12}$$

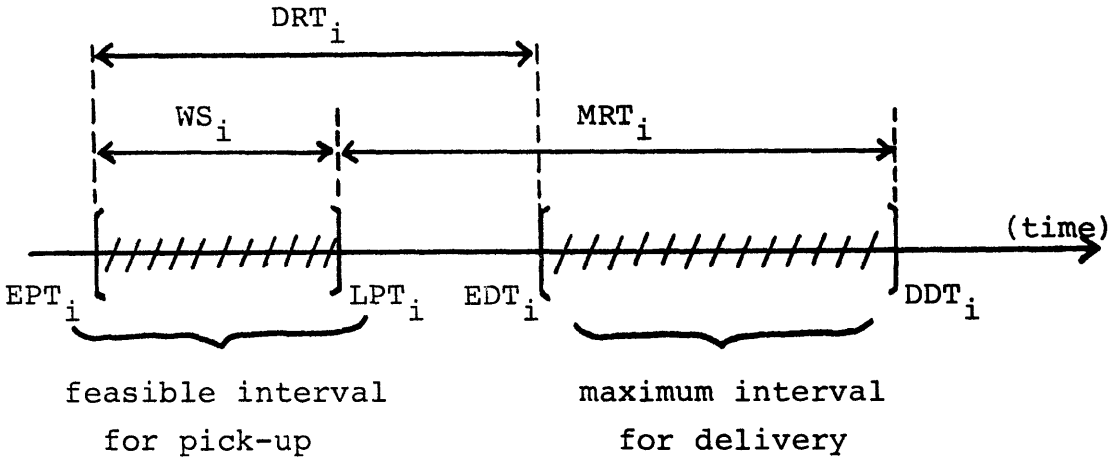
((4-9) and (4-10) contain the same information as (4-3))

For any customer i, whether DPT- or DDT-specified, a set of necessary, but not sufficient conditions for feasibility is then provided by (4-13) and (4-14):

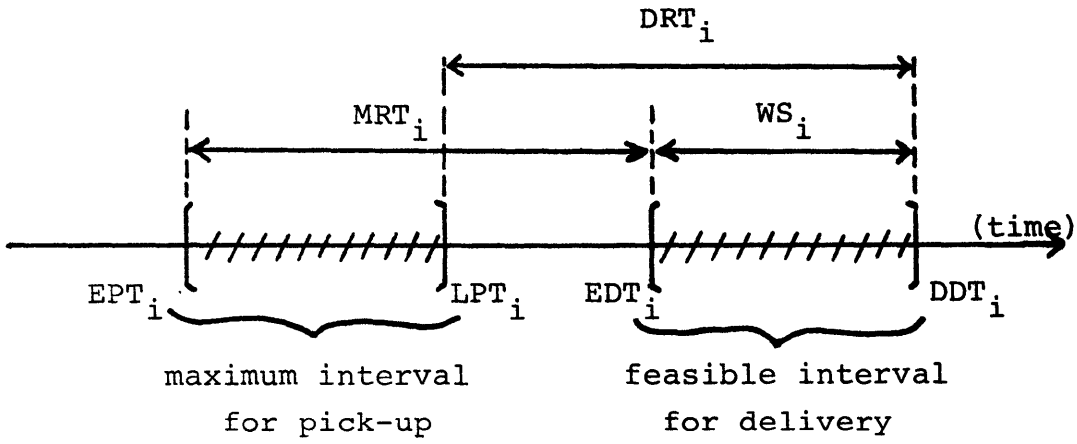
$$EPT_i \leq APT_i \leq LPT_i \tag{4-13}$$

$$EDT_i \leq ADT_i \leq LDT_i \tag{4-14}$$

(From Figure 4.4, it can be seen that (4-13) and (4-14) are not sufficient because it may be possible that APT_i and ADT_i satisfy them but (4-1) is nonetheless violated. It should also be noted that LPT_i need not be



(a)



(b)

Figure 4.4

Time windows for (a) DPT-specified and (b) DDT-specified customers.

smaller than EDT_i.)

The conditions (4-13) and (4-14) are particularly convenient to work with. The quantities EPT_i, LPT_i , EDT_i and LDT_i computed for all customers i = 1,2,...,N define "fixes" on the time axis within which the customer must be picked up and delivered. To understand the usefulness of these fixes let us return to the problem of checking the feasibility of inserting customer i into a particular schedule block "p" of vehicle j.

Let us consider such a schedule block and let us index the successive stops on the schedule sequence with the subscript r = 1,2,....,d. Note that from (4-13) and (4-14) we have an upper and lower bound for each element in the time schedule associated with our schedule block ((4-13) providing the bounds if the entry is an APT and (4-14) if the entry is an ADT).

For convenience let us now drop the indication of whether a particular stop on the schedule block is a pick-up or a delivery and use ET_r, AT_r and LT_r to denote earliest, actual (scheduled) and latest time, respectively, for stop r. For instance in Figure 4.2, we would indicate EPT_k, APT_k, LPT_k by ET_1, AT_1 and LT_1, respectively, and EDT_m, ADT_m and LDT_m by ET_3, AT_3 and LT_3, respectively, for the middle schedule block.

In ADARTW, we compute and store four statistics for each stop r (r = 1,...,d) on each schedule block, defined as follows:

$$BUP_r = \text{Min} \left[\min_{1 \leq t \leq r} (AT_t - ET_t) , SLACK_p \right] \quad (4\text{-}15)$$

$$BDOWN_r = \min_{1 \leq t \leq r} (LT_t - AT_t) \quad (4\text{-}16)$$

$$AUP_r = \min_{r \leq t \leq d} (AT_t - ET_t) \quad (4\text{-}17)$$

$$ADOWN_r = \text{Min} \left[\min_{r \leq t \leq d} (LT_t - AT_t) , SLACK_{p+1} \right] \quad (4\text{-}18)$$

where $SLACK_p$ and $SLACK_{p+1}$ denote, respectively, the duration of the slack period immediately preceding and immediately following the schedule block p in question.

There is a very real intuitive meaning associated with the four quantities defined by (4-15) - (4-18). Specifically, BUP_r ($BDOWN_r$) represents the maximum amount of time by which every stop preceding but not including stop r+1 can be advanced (delayed) without violating the time-window constraints. Similarly, AUP_r ($ADOWN_r$) represents the maximum amount of time by which every stop following but not including stop r-1 can be advanced (delayed). Essentially, the quantities BUP, BDOWN, AUP and ADOWN indicate by how much, at most, each segment of a schedule block (e.g. the segment that precedes the pick-up of the inserted customer i) can be displaced in order to accommodate an additional customer i.

The uses of the four statistics defined at each stop for checking the feasibility of an insertion differ depending on where in the schedule block the pick-up and delivery of the new customer are inserted. ADARTW divides all insertions into four basic cases (as shown in Figure 4.5(a)-(d)) which account for the total of (d+2)(d+1)/2 combinations mentioned

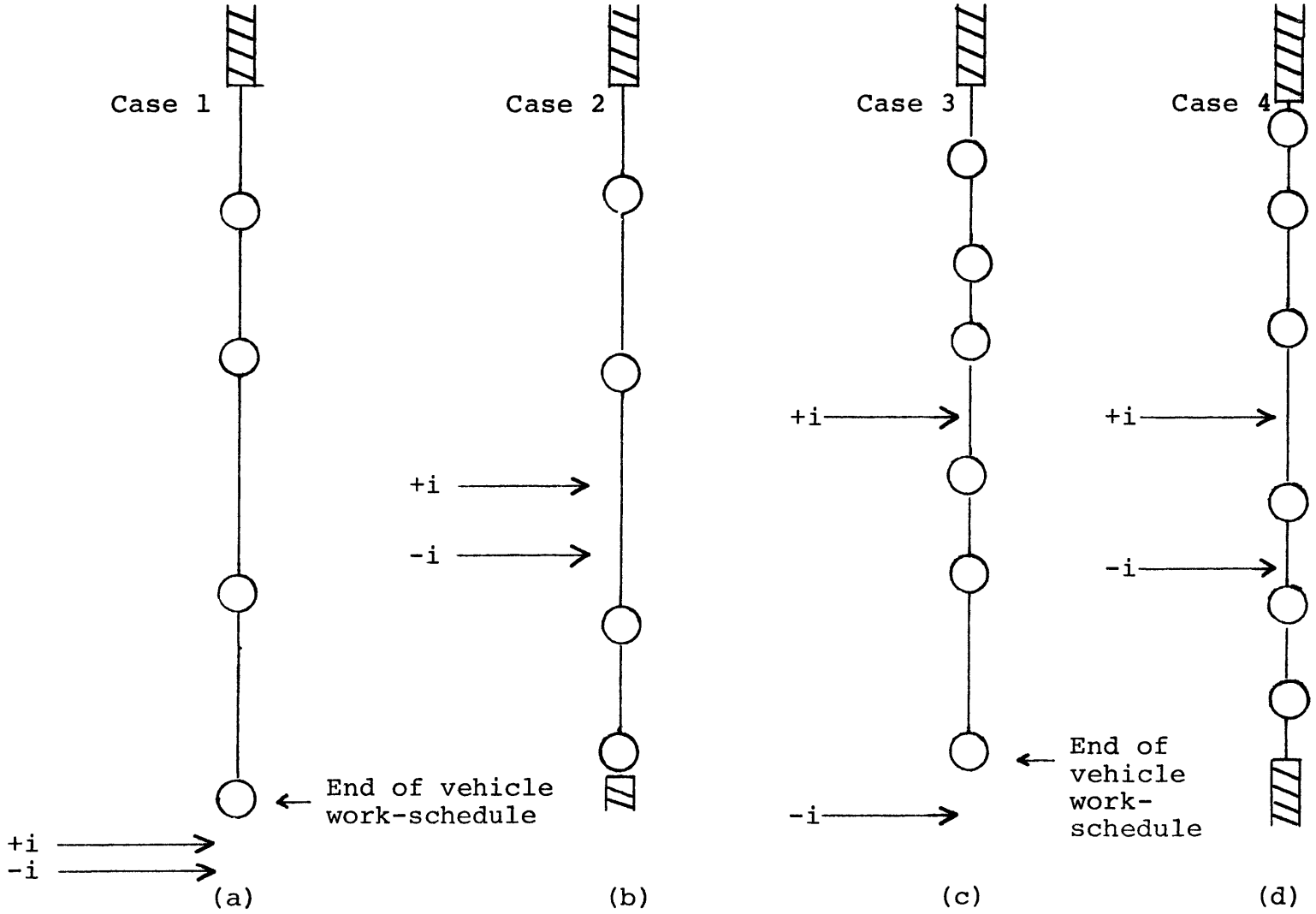


Figure 4.5 Possibilities for insertion of customer i

earlier:

(i) Both the pick-up and delivery of customer i are inserted at the end of the last schedule block, i.e. they become the last two stops in the vehicle's work-schedule. We note that this is the only case where a new schedule block can be created.

(ii) Both the pick-up and delivery of customer i are inserted between two consecutive stops on the schedule block. This includes the insertions of pick-up and delivery immediately before or after a slack period.

(iii) The pick-up of customer i takes place somewhere within the last schedule block while his delivery is inserted at the end of the vehicle's work schedule.

(iv) The pick-up and delivery of customer i are separated by at least one other stop and the delivery of i is not the last stop in the vehicle's work-schedule.

The details of the algorithm's logic for each one of these four cases are provided in Figure I.1(a) - (d) of Appendix I. Each flowchart depicts how it is determined whether it is feasible, as far as the time-window constraints are concerned, to insert customer i within a schedule block in the way defined by one of the four cases above.

Besides checking for violations of the time-window constraints, we have to check that no maximum-ride-time constraints are violated for the newly inserted customer and for the customers already in the schedule block. This can be done very easily by scanning through the list of these customers and comparing the respective actual ride-times and maximum

allowable ride-times. Finally, vehicle loads at each stop between the inserted pick-up and delivery of customer i are checked so that vehicle capacity is not exceeded.

Feasibility Test for Insertions into Different Schedule Blocks

The insertion of the pick-up and delivery of customer i into different schedule blocks requires the elimination of all the slack time contained between these schedule blocks since, otherwise, the vehicle would wait idly with customer i on board. The previous approach which employs four statistics at each stop becomes unnecessarily complicated in this case in dealing with the slack time involved. We have chosen instead to design another approach which is straightforward in terms of logic and yet efficient enough to meet our computational requirements. This new approach also achieves the two objectives in our search for feasible insertions, namely, to obtain a time schedule for each new feasible sequence and to find corresponding bounds on the displacement of each time schedule.

For demonstration purposes let us consider the case of inserting the pick-up of customer i somewhere in schedule block k and the delivery of customer i into schedule block k+n ($n \geq 1$). We limit our search for feasible insertions to those that will not affect other customers on schedule blocks q, $q < k$ or $q > k+n$. Consequently, as a priori condition for an insertion to be considered in our search is:

$$\Delta P_i + \Delta D_i \leq \sum_{j=k}^{k+n+1} SLACK_j \qquad (4\text{-}19)$$

where ΔP_i (ΔD_i) is the extra vehicle time required to serve the pick-up (delivery) point of customer i.

The algorithmic approach works as follows:

Form a new schedule sequence by merging schedule blocks k to k+n and with customer i inserted. Let d denote the number of stops in the new schedule sequence. Construct an earliest time schedule (without considering the time constraints at each stop) for the new sequence with all the slack time eliminated, i.e. by taking $SLACK_k = 0$, $SLACK_{k+1} = 0, \ldots, SLACK_{k+n} = 0$. Let us denote this tentative time schedule by "T." For every stop r (r = 1, 2, ..., d) in the new schedule sequence, compute two statistics, R_r and A_r:

$$R_r = \left| \text{Min} \left(T_r - ET_r , 0 \right) \right| \qquad (4\text{-}20)$$

$$A_r = LT_r - T_r \qquad (4\text{-}21)$$

where T_r is the visiting time of stop r in time schedule T.

R_r denotes the minimum time by which the visit time of stop r, T_r, should be delayed for it to fall within the time window of stop r. A_r denotes the maximum time by which T_r can be delayed for it not to violate the time window constraint of stop r. After R_r and A_r are computed for all r in the new schedule sequence, we can define R_{min} and A_{max} for time schedule T as:

$$R_{min} = \underset{\text{all } r}{\text{Max}} \left(R_r \right) \tag{4-22}$$

$$A_{max} = \underset{\text{all } r}{\text{Min}} \left(A_r \right) \tag{4-23}$$

R_{min} represents the minimum amount of time by which the time schedule T should be delayed so that every stop in T is visited at or after its earliest feasible time. A_{max} represents the maximum amount of time by which the time schedule T can be delayed without a stop being visited later than its latest feasible time. For a schedule sequence to be feasible we must have :

$$R_{min} \leq A_{max} \tag{4-24}$$

(4-24) indicates the fact that the amount of time by which T has to be delayed must be less than or equal to the maximum amount of time that T can be delayed. (4-24) is the necessary and sufficient condition for a new sequence to be feasible with respect to satisfying the time window constraints. To check for violations of vehicle capacity and customer's maximum-ride-time constraints, a screening test through the new time schedule would suffice. (Such screening can be performed at the same time when R_r and A_r are computed. See Figure I.2 in Appendix I.)

We have so far accomplished our first objective by identifying new feasible sequences. The second objective in the feasibility test is to obtain a feasible time schedule and its associated bounds on the displacements. Recall that T is the earliest schedule for the new sequence and it has to be delayed by at least R_{min} to remain feasible. If the condition of (4-24) is met, we can obtain a feasible schedule T' by

shifting T by R_{min}. By the definitions of T and R_{min}, T' becomes the earliest feasible time schedule for the new sequence. Consequently, the lower bound, LB, on the displacement of T' is equal to zero.

$$LB = 0 \tag{4-25}$$

In other words, no schedule earlier than T' would be feasible.

The upper bound, UB, representing the maximum amount of time T' can be delayed is obtained by :

$$UB = A_{max} - R_{min} \tag{4-26}$$

The detailed logic of the approach described above is depicted in the flowchart of Figure I.2 of Appendix I..

In summary we have designed two different methods for testing feasibility of insertions, one for insertions within the same schedule block and the other for insertions into two different schedule blocks. Both methods achieve objectives i) and ii) described above. We now proceed to discuss the optimization procedures used to choose the most desirable insertion.

4.5 The Optimization Procedure

In order to select among all feasible insertions of customer i into the work-schedules of the available vehicles, we use an objective function designed to evaluate the incremental "cost" of each insertion. This cost is taken to be a weighted sum of disutility to system's customers and of operator costs - the latter measured in terms of "consumption" of available vehicle resources.

Assume that it is feasible to insert customer i into the work-schedule of vehicle j. Then the incremental disutility of that insertion to the system's customers consists of the sum of two parts: the disutility to customer i, i.e. the customer being assigned to a vehicle; and the additional disutility suffered by all other customers already assigned to that vehicle because of the insertion of customer i. The first part (disutility to customer i) is given by

$$DU_i = DU_i^d + DU_i^r \tag{4-27}$$

where DU_i^d = disutility due to deviation from most desired time

$$= C_1 \cdot x_i + C_2 \cdot x_i^2 \qquad 0 \leq x_i \leq WS_i \tag{4-28}$$

and,

DU_i^r = disutility due to excess ride time

$$= C_3 \cdot y_i + C_4 \cdot y_i^2 \qquad y_i \geq 0 \tag{4-29}$$

In (4-28) and (4-29), C_1, C_2, C_3 and C_4 are user-specified constants and

$$x_i = \begin{cases} APT_i - DPT_i & \text{for DPT-specified customers} \\ DDT_i - ADT_i & \text{for DDT-specified customers} \end{cases} \tag{4-30}$$

and $$y_i = ART_i - DRT_i \tag{4-31}$$

The quadratic terms allow the modeling of situations in which disutilities are believed to increase nonlinearly with x_i and/or y_i. Clearly, by varying C_1, C_2, C_3 and C_4 (including assigning the value of

zero to some of them) many different types of behavior can be represented.

The second part (additional disutility to other customers) is given by:

$$DU_i^o = \sum_{k \text{ on } j} (DU_k^{new} - DU_k^{old}) \qquad (4\text{-}32)$$

where DU_k^{new} and DU_k^{old} are, respectively, the disutilities to customer k after and before the insertion of customer i into the schedule of vehicle j. The summation is over all customers k who were already assigned to vehicle j prior to the assignment of customer i.

The incremental cost, VC_i, to the system's operator due to inserting customer i into the work-schedule of some vehicle is quantified in somewhat unusual terms by our objective function. We have,

$$VC_i = C_5 \cdot z_i + C_6 \cdot w_i + U_i \cdot (C_7 \cdot z_i + C_8 \cdot w_i) \qquad (4\text{-}33)$$

where C_5, C_6, C_7 and C_8 are externally set constants.

z_i is the additional active vehicle time required to serve the customer i

w_i is the change in vehicle slack time due to the insertion

U_i is an indicator of system workload defined as:

$$U_i = \frac{(\text{No. of system customers in } [EPT_i - W_1, EPT_i + W_2])}{(\text{Effective no. of vehicles available in } [EPT_i - W_1, EPT_i + W_2])} \qquad (4\text{-}34)$$

In (4-34), W_1 and W_2 are externally specified constants. For example, if W_1, W_2 = 60 minutes, U_i is equal to the ratio of the number of customers demanding service to the available number of vehicles during the

two-hour time period that has the earliest pick-up time of customer i as its mid-point. The word "effective" is used in the denominator of (4-34) to account for the fact that some vehicles may be available for service only for a fraction of the time interval (e.g., two hours) in question - usually because of driver constraints, union rules, etc.

The following should be noted with respect to (4-33):

(i) The change in vehicle slack time, w_i, can be positive or negative. It will be negative if the insertion means that slack time in the original schedule will now be utilized to serve customer i; it will be positive if, in order to serve customer i, additional vehicle slack time must be created (this will happen if a new schedule block is created to serve customer i).

(ii) In practice, the cost per unit of time of a vehicle in the "active" state is greater than in the "slack" state (e.g. no fuel consumption in slack state). Therefore, we must have $C_5 \geq C_6$ and $C_7 \geq C_8$.

(iii) Obviously U_i will be larger during periods of heavy demand. Since the total "cost" of an insertion is given by $DU_i + VC_i$ - i.e. by the sum of (4-27), (4-32) and (4-33) - it is clear that during heavy demand periods, the objective function places more emphasis on the system operator's costs (relative to customers' disutility) than during low demand periods. This is as it should be, since during periods of high utilization vehicle resources are scarcer and it is thus important to "conserve" these resources as much as possible. For all test runs of ADARTW to be described in Chapter 5, W_1=0 and W_2=60 minutes are chosen to compute U_i in (4-34). U_i in this case can be explained as the average

number of customers a vehicle would have to carry in the next hour starting from the earliest pick-up time of customer i. As a result, the objective function in the optimization step which includes U_i as a parameter will take into consideration the customer demands for the next hour and adjust automatically the weight placed on the vehicle resource term. In the last hour of dial-a-ride operation U_i will become smaller as we approach the end of the subscription list. This is a desirable feature since at the end of a day no urgent conservation of vehicle resources is necessary.

Given this background, the optimization steps of ADARTW can be summarized as follows: Consider customer i who is to be inserted into the work-schedule of one of m available vehicles. Assume that for each vehicle j (j = 1,2,...,m) all feasible sequences for inserting customer i into the work-schedule of j have been identified in the manner outlined in Section 4.4. Then, to select the optimum insertion, we use a sequence of three steps:

(a) For each and every feasible sequence pertaining to vehicle j, find a time schedule (including the pick-up and delivery times APT_i and ADT_i for customer i) that minimizes the incremental cost $DU_i + VC_i$. Note that this requires shifting the feasible time schedule associated with each feasible sequence (obtained in section 4.4) within its bounds of displacements.

(b) Choose the one sequence among those examined above which results in the smallest incremental cost. This smallest incremental cost is $COST_j$, the additional cost associated with inserting customer i into the work-

schedule of vehicle j as defined in Section 4.3 above.

(c) Finally, assign customer i to a vehicle j^* such that $COST_{j^*} \leq COST_j$ for all $j = 1,2,\ldots,m$.

In steps (b) and (c) above, the goal is to find the minimum among a finite number of values. Such procedures are straightforward. However, step (a) presents some difficulties. Finding the optimal time schedule (i.e. the one with the least incremental cost) for a schedule sequence is very similar to the problem defined and solved in Sexton and Bodin [32]. However, our problem is different in two aspects : (i) A combination of DPT-specified and DDT-specified customers are present in our problem (in [32], only one type of customers is allowed); (ii) Our objective function is nonlinear if non-zero C_2 or C_4 is chosen in (4-28) and (4-29). Bodin and Sexton transformed their version of the problem into a maximum profit network flow problem which can be solved using noniterative procedures. For our case, we need a highly efficient way to solve the problem since it is encountered for every instance of a feasible insertion.

Before presenting our solution approach to this optimization problem, we first use an example to illustrate the nature of the problem involved.

Example 4.1:

Consider a subscription list of three customers, each having service quality constraints and distinct pick-up and delivery locations as shown in Table 4.2.

Table 4.2 Subscription list of three customers.

Customer #	EPT	LPT	EDT	LDT	DRT	MRT	Time Specified
1	7:30	8:00	8:10	9:20	40	80	DPT
2	7:35	8:30	8:25	8:55	25	50	DDT
3	7:35	8:05	8:10	9:15	35	70	DPT

Table 4.3 D(x,y) matrix (in minutes)

	1	-2	-3	+1	+2	+3
-1	0	20	10	40	38	45
-2	20	0	15	30	25	30
-3	10	15	0	50	38	35
+1	40	30	50	0	15	10
+2	38	25	38	15	0	10
+3	45	30	35	10	10	0

In Table 4.3, the distance matrix between any pair of points is given in terms of vehicle travel time. The current schedule sequence and the corresponding time schedule are provided in Table 4.4.

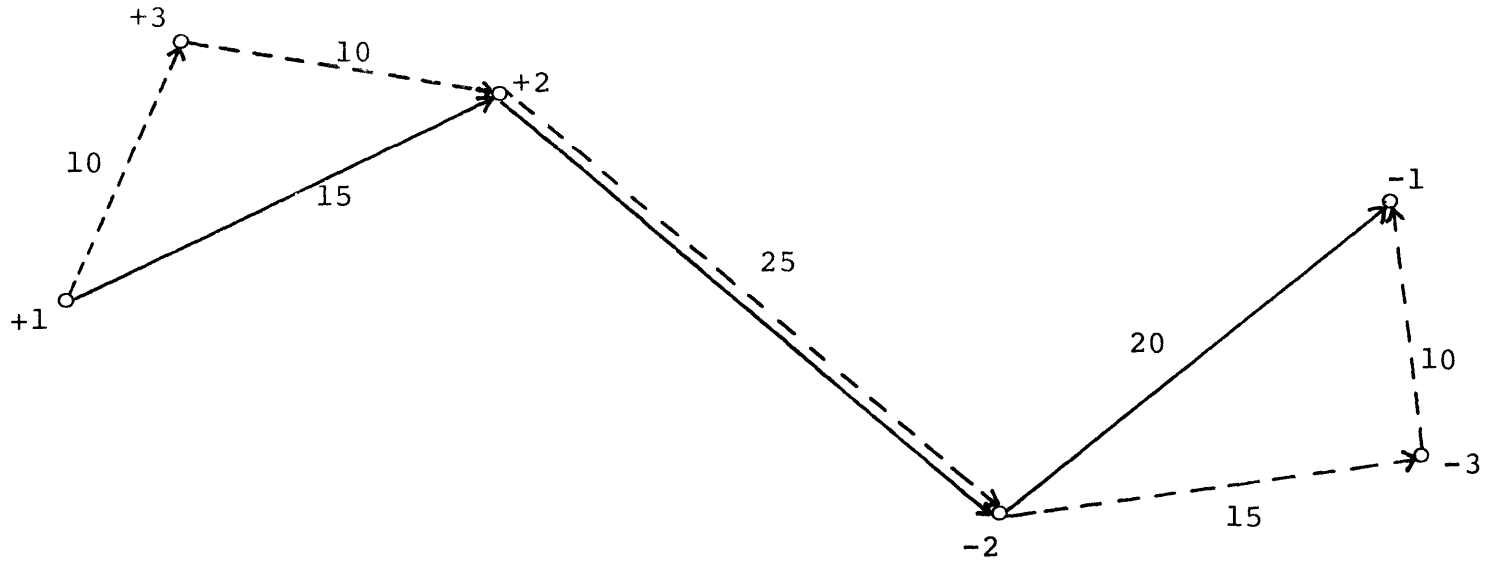
For demonstration purposes, let us focus on one particular insertion, that of customer #3 to this schedule block as shown in Figure 4.6. The new sequence {+1,+3,+2,-2,-3,-1} is feasible and can correspond to many different time schedules. Table 4.5 lists three such feasible time schedules, the first one being the earliest possible, the second one the latest possible while the third is the one obtained as a result of the feasibility test described in Section 4.4.

Table 4.4 Current Time Schedule

STOPS	TIME-SCHEDULE	DEVIATION
+1	7:52	+22
+2	8:07	
-2	8:32	-23
-1	8:52	

Table 4.5 Three Feasible Time Schedules After The Insertion

STOPS	EARLIEST-SCHEDULE	LATEST-SCHEDULE	INITIAL-SCHEDULE
+1	7:30	7:55	7:47
+3	7:40	8:05	7:57
+2	7:50	8:15	8:07
-2	8:15	8:40	8:32
-3	8:30	8:55	8:47
-1	8:40	9:05	8:57



Route after the insertion

Route before the insertion

Figure 4.6 An incidence of insertion of customer #3
(Distance in minutes)

The time difference between the earliest and latest time schedules is 25 minutes. Any time schedule which falls in this 25-minute span is considered feasible for this new sequence. The problem is to find the time schedule that minimizes the incremental cost caused by this particular insertion of customer #3.

(End of Example 4.1)

Let T^0 denote the initial time schedule which has been obtained from the feasibility test. Recall that in Section 4.4 we have also obtained the associated upper and lower bounds on displacement for this initial time schedule. Based on this initial time schedule it can be shown that our optimization problem is equivalent to minimizing a single-variable convex fuction, the variable being the amount by which the initial time schedule should be shifted. Let f(a) denote the objective fuction to be minimized. Variable "a" represents the amount of time by which the initial time schedule, T^0, is shifted. Let T^a denote the time schedule after shifting T^0 by time a. The optimization problem can be formulated as follows:

$$\text{Minimize} \quad f(a) : C(T^a) - C(T^0) \tag{4-35}$$

$$\text{s.t.} \quad LB \leq a \leq UB \quad (\text{ "a" can be negative }) \tag{4-36}$$

where C() denotes the incremental cost of a time schedule.

Note that by shifting T^0 by time a, the customers' excess on-board time disutility and the vehicle's active time remain unchanged. Only customers' service time deviation and vehicle slack time will vary with

different time schedules. Let ΔDU^a and ΔVC^a denote respectively the change in customers' disutility and vehicle resources after shifting T^0 by time a. Then we can express ΔDU^a and ΔVC^a as follows:

$$\Delta DU^a = (A) - (B) \qquad (4\text{-}37)$$

$$(A) = C_1 \cdot [\sum_{\text{all } p} (DV_p + a) + \sum_{\text{all } d} (DV_d - a)] + C_2 \cdot [\sum_{\text{all } p} (DV_p + a)^2 + \sum_{\text{all } d} (DV_d - a)^2]$$

$$(B) = C_1 \cdot (\sum_{\text{all } p} DV_p + \sum_{\text{all } d} DV_d) + C_2 \cdot (\sum_{\text{all } p} DV_p^2 + \sum_{\text{all } d} DV_d^2)$$

where DV_i represents the time deviation between the actual service time of customer i in T^0 and customer i's desired service time,

p is the index for DPT-specified customers,

d is the index for DDT-specified customers.

Part (A) and part (B) in (4-37) represent respectively customers' disutility after and before the shifting of T^0 by time a. (4-37) can be simplified to :

$$\Delta DU^a = C_1 \cdot (N(P) - N(D))a + C_2 \cdot [2a(\sum_{\text{all } p} DV_p - \sum_{\text{all } d} DV_d) + (N(P) + N(D))a^2] \qquad (4\text{-}38)$$

where N(P) (N(D)) denote the number of DPT-specified (DDT-specified) customers in the new schedule sequence.

ΔVC^a can be expressed by the change in vehicle slack time:

$$\Delta VC^a = (C_6 \cdot a + C_8 \cdot a \cdot U_i) \cdot I \qquad (4\text{-}39)$$

where I = 0 if the schedule block containing customer i is contained between two other schedule blocks or the schedule block is the only schedule block in the vehicle's work-schedule,

= -1 (= +1) if the schedule block containing customer i is the first (last) schedule block in the vehicle's work-schedule.

We can now express function f(a) in terms of ΔDU^a and ΔVC^a:

$$f(a) = \Delta DU^a + \Delta VC^a$$

$$= [C_1 \cdot (N(P)-N(D)) + 2C_2 \cdot (\sum_{\text{all } p} DV_p - \sum_{\text{all } d} DV_d) + (C_6 + C_8 \cdot U_i) \cdot I] \cdot a$$

$$+ C_2 \cdot (N(P)+(N(D)) \cdot a^2 \qquad (4\text{-}40)$$

We observe in (4-40) that f(a) is a convex quadratic fuction with respect to a. According to the property of a convex fuction, f(a) is minimized at a_* which satisfies the relationship df(a)/da = 0. The value a_* might not be the true minimum in our optimization problem due to the presence of the bounding constraint in (4-36). However, since f(a) is convex, we can easily find the true minimum a^* by comparing a_* with the constraining bounds.

We now summarize the above discussion : The optimization problem of (4-35) and (4-36) can be solved by first finding the minimum a_* for the convex fuction in (4-40). Then by comparing a_* with constraint (4-36), we can obtain the true minimum a^*.

Results can be classified into two cases:

(i) $C_2 = 0$

This is the case of linear objective function. The function f(a) in (4-40) becomes:

$$f(a) = [\ C_1 \cdot (N(P)-N(D)) + (\ C_6 + C_8 \cdot U_i\ ) \cdot I\ ] \cdot a \qquad (4\text{-}41)$$

For the minimization of (4-41) subject to (4-36), we have:

If $C_1 \cdot (N(P)-N(D)) + (\ C_6 + C_8 \cdot U_i\ ) \cdot I > 0$, then $a^* = LB$

If $C_1 \cdot (N(P)-N(D)) + (\ C_6 + C_8 \cdot U_i\ ) \cdot I < 0$, then $a^* = UB$

If $C_1 \cdot (N(P)-N(D)) + (\ C_6 + C_8 \cdot U_i\ ) \cdot I = 0$, then a^* can be set arbitrarily between UB and LB.

(ii) $C_2 > 0$, the case of nonlinear objective function:

$$a_* = \frac{-[C_1 \cdot (N(P)-N(D)) + 2C_2 \cdot (\sum_{\text{all } p} DV_p - \sum_{\text{all } d} DV_d) + (C_6 + C_8 \cdot U_i) \cdot I]}{2(N(P)+N(D)) \cdot C_2} \qquad (4\text{-}42)$$

If $a_* < LB$, then $a^* = LB$

If $a_* > UB$, then $a^* = UB$

If $LB \leq a_* \leq UB$, then $a^* = a_*$

After the optimal "shift" a^* is found, we can obtain the optimal time schedule, T^*, for the new sequence by shifting T^0 by time a^*. From T^* the incremental cost caused by this insertion can be evaluated.

To further demonstrate the process of finding a^*, we choose the same situation used in Example 4.1 and work through the entire procedure of computing the incremental cost for that particular insertion.

Example 4.2 :

Table 4.6 lists the initial time schedule, T^0, obtained as a result of the feasibility test (see flowchart in Figure I.1(d) in Appendix I. and Table 4.5). The upper and lower bounds on the displacement of T^0 are: UB = 8, LB = -17, i.e., $-17 \leq a \leq 8$. Other information pertaining to T^0 required for finding a^* are:

$N(P) = 2$, $N(D) = 1$, $I = 0$, $DV_1 = 17$, $DV_2 = 23$, $DV_3 = 22$.

Assume that the user of ADARTW has specified the objective function constants as:

$C_1 = 0$, $C_2 = 0.5$, $C_3 = 0$, $C_4 = 0.3$

$C_5 = 0$, $C_6 = 0$, $C_7 = 1$, $C_8 = 0.5$

Suppose that it has also been determined that the average vehicle workload of the system, U_3, defined by (4-34), is equal to 3.

Since $C_2 > 0$, we use (4-42) to compute a_*. We get $a_* = -16/3$.

By comparing the value of a_* with constraint (4-36), we note that:

$$-17 \leq -16/3 \leq 8$$

So, $a^* = a_* = -16/3$. The optimal time schedule T^* is then obtained by shifting T^0 early by 5.33 minutes[1]. Table 4.7 shows the would-be optimal

---

[1]For simplification purposes, 5.33 is rounded off to 5 in the following discussion.

Table 4.6 The Initial Time Schedule T^0

STOPS	TIME-SCHEDULE
+1	7:47
+3	7:57
+2	8:07
-2	8:32
-3	8:47
-1	8:57

Table 4.7 The Optimal Time Schedule T^*

STOPS	TIME-SCHEDULE	DEVIATION
+1	7:42	+12
+3	7:52	+17
+2	8:02	
-2	8:27	-28
-3	8:42	
-1	8:52	

time schedule T^* if customer #3 is inserted.

With the time schedule T^* found, we can evaluate the incremental cost caused by this insertion of customer #3. The incremental cost consists of two parts:

(i) Customers' disutility

a) Customer #3's disutility:

$DU_3 = DU_3^d + DU_3^r$

$= C_1 \cdot x_3 + C_2 \cdot x_3^2 + C_3 \cdot y_3 + C_4 \cdot y_3^2$

$x_3 = APT_3 - DPT_3$ (customer #3 is a DPT-specified customer)

$= 7:52 - 7:35$ (compare Table 4.6 and Table 4.2)

$= 17$

$y_3 = (ADT_3 - APT_3) - DRT_3$

$= (8:42 - 7:52) - 35$ (see Table 4.7 and Table 4.2)

$= 50 - 35$

$= 15$

$DU_3 = (0.5 \cdot 17^2) + (0.3 \cdot 15^2)$

$= 312$

b) Other customers'(#1's and #2's) additional disutility:

$DU_3^o = \sum (DU_j^{new} - DU_j^{old})$

$= (0.5 \cdot 12^2) - (0.5 \cdot 22^2)$ (compare Table 4.4 with Table 4.7)

$+ (0.5 \cdot 28^2) - (0.5 \cdot 23^2)$

$+ (0.3 \cdot 70^2) - (0.3 \cdot 60^2)$

$= 1300.5$

(ii) Operator cost

$$z_3 = [\ D(+1,+3) + D(+3,+2) - D(+1,+2)\ ] + [\ D(-2,-3) + D(-3,-1) - D(-2,-1)\ ]$$
$$= 10$$
$$w_3 = 0$$
$$VC_3 = C_5 \cdot z_3 + C_6 \cdot w_3 + U_3 \cdot (\ C_7 \cdot z_3 + C_8 \cdot w_3\ )$$
$$= 1 \cdot 3 \cdot 10$$
$$= 30$$

The total incremental cost is the sum of (i) and (ii):

$$Cost = 312 + 1300.5 + 30$$
$$= 1642.5$$

(End of Example 4.2)

We have described the logic of ADARTW and used numerical examples to highlight parts of the algorithm's procedures. We now proceed to the discussion of some computational results.

CHAPTER 5 COMPUTATIONAL RESULTS OF ADARTW

5.1 Introduction

In order to gain insights into the performance of ADARTW under various operating environments, we have designed a set of nine computer-generated customer subscription lists based upon combinations of three levels of demand scenarios and three different sizes of time windows. The three demand scenarios correspond to low (10 customers/hr), medium (50 customers/hr) and high (100 customers/hr) demand levels that have been typical of those experienced by some of the existing dial-a-ride systems. The set of three time window sizes (10, 20, and 30 minutes) represent a possible range of service quality levels that could be guaranteed by the system operator. We also tested ADARTW on a real-world data base obtained from Friedrichshafen, West Germany. In the following sections we present our computational findings on ADARTW based upon these two categories of tests. Section 5.2 examines the computational results using simulated data. Section 5.3 discusses the computation efficiency of ADARTW and finally in Section 5.4 experience with ADARTW on the real-world data base is presented.

5.2 Runs Using Simulated Data

5.2.1 Data

The nine simulated data sets are randomly generated by the computer using the following scenarios:

Service area : 6 x 6 square miles.

Distance metric : Euclidean.

Customer origins/destinations : Uniformly and independently distributed in the area.

Depot location : Center of the area.

Vehicle speed : 15 miles/hour.

Hourly demand : 10 , 50 , 100 customers (low , medium , high).

Duration of time simulated : 9 hours.

Demand pattern : Uniformly distributed within every hour simulated.

Percent of DPT-specified customers : 50%

Time window size : 10 , 20 , 30 minutes

Maximum ride time : $MRT_i = 5 \text{ minutes} + 2 \cdot DRT_i$

Parameters W_1 and W_2 used in (4-34) : $W_1 = 0$, $W_2 = 60$ minutes.

For notational purposes, we use XN (X = L,M,H and N = 10,20,30) to represent the data set generated by using one of the three demand levels, 10, 50 and 100 customers per hour and one of the three sizes of time window. For example, M30 represents the data set of 50 customers per hour and guaranteed time window of 30 minutes.

5.2.2 Computational Results and Analysis

We divide the discussion of computational results on simulated data into two parts. The first part focuses on the question of how different parameters (C_1, C_2 and others) in the objective function affect the results of ADARTW. The second part examines the performance of one

possible variation from the basic approach of ADARTW which employs a pool of candidate customers. We also evaluate the advantage gained by considering all possible insertions, instead of just those made into the same schedule block.

5.2.2.1 Investigation on the effects of individual parameters

In the first part of the investigation, we use $C_1, C_2, C_3, C_4, C_5, C_6 = 0$, $C_7 = 1$, $C_8 = 0.8$. as the base case parameter set. By varying one parameter at a time in the subsequent test runs, we attempt to separate the effects of the individual parameters. Statistics collected at the end of each run include the following:

a) Vehicle productivity = $\dfrac{N}{\text{Total vehicle time}}$

b) Average deviation = $\dfrac{1}{N} \sum_{i=1}^{N} DV_i$

c) Average ride time ratio = $\dfrac{1}{N} \sum_{i=1}^{N} \dfrac{ADT_i - APT_i}{DRT_i}$ (5-1)

d) Number of vehicles used.

e) Maximum vehicle capacity required.

f) Vehicle utilization rate = $\dfrac{\text{Active vehicle time}}{\text{Total vehicle time}}$

We have chosen the M30 data set as the primary test data in this part of the investigation. Initially, the system has 10 available vehicles. Whenever an infeasibility condition occurs during a run, another

vehicle is introduced into the system and it stays available throughout the simulation time. Test results are summarized as follows:

(i) Parameter C_1:

In this experiment it is assumed that the customer disutility function is a linear function of the service time deviation, i.e. DU_i^d in (4-28) can be expressed as: $DU_i^d = C_1 \cdot x_i$. To investigate the effect of different values of C_1 on the solution of ADARTW, a series of test runs were conducted by varying the value of C_1 from the base case scenario in each successive run. Statistics on the quality of schedules produced by ADARTW with respect to different values of C_1 are tabulated in Table 5.1. As can be seen in column 1 of the table, a wide range of possible values for C_1 was tested. The second column of the table indicates the number of vehicles used in order to reach a feasible solution (initially there were only 10 vehicles). It is noted that as C_1 increases, the number of vehicles required also increases despite some minor fluctuations when C_1 is small. The average deviations shown in column 3 seem to decrease monotonically with increasing values of C_1 within the range of 0 to 20. After C_1 exceeds 20, the average deviation becomes somewhat unpredictable but eventually settles down to 3.22 minutes for $C_1 \geq 160$. We also observe the fact that the average time deviation is more sensitive to changes of C_1 when C_1 is small. As C_1 becomes larger the rate of decrease in average deviation as a result of increasing C_1 slows down.

The fourth column of Table 5.1 records the average ride time ratio as defined in (5-1) for each run. Although no customer disutility was

Table 5.1 Base case scenario with varying C_1

(C_7 = 1, C_8 = 0.8, Others = 0)

Run	C_1	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	13	14.59	1.60	4.54	0.94	6
#2	1	12	11.94	1.60	4.47	0.94	6
#3	3	15	7.95	1.58	3.77	0.90	5
#4	5	14	6.54	1.48	3.81	0.96	5
#5	7	15	4.65	1.47	3.61	0.95	4
#6	9	14	4.28	1.46	3.64	0.96	5
#7	20	15	3.68	1.43	3.38	0.95	4
#8	50	15	4.11	1.40	3.35	0.96	4
#9	75	15	4.85	1.37	3.37	0.96	5
#10	100	15	4.42	1.40	3.36	0.95	5
#11	150	15	4.94	1.38	3.36	0.96	5
#12	$160\leq$	17	3.22	1.38	3.18	0.94	5

assumed for excess ride time in this series of test runs ($C_3 = C_4 = 0$), the average ride time ratio recorded actually decreases as C_1 increses. One explanation for such behavior is that as more weight is placed on time deviation, customers are likely to be delivered directly from their origin to their destination so that their service times are close to their desired times. Another reason might be that as the value of C_1 increases more vehicles are introduced into the system and consequently the system will provide less circuitous ride service when the workload is shared by additional vehicles.

Another important statistic, vehicle productivity, is listed in the fifth column of Table 4.8. When no customer disutility was specified in run #1 ($C_1,\ldots,C_4 = 0$), ADARTW achieved a system productivity of serving, on the average, 4.54 customers per vehicle hour. The quality of service rendered can be evaluated by an average deviation of 14.59 minutes from a customer's desired service time and by an average of 60% more ride time than necessary. As the quality of service improves when C_1 increases, the vehicle productivity decreases until it reaches the lowest level of 3.18, a 30% reduction from the highest value of 4.54. The tradeoff between the service quality and vehicle resources is real and the objective function has demonstrated its capability of providing a wide range of choices for the system operator. However, we note that in Table 5.1 there are some solutions which clearly dominate others (e.g. the solution for run #4 ($C_1 = 5$) dominates run #3 ($C_1 = 3$) in terms of the number of vehicles used, the average deviation, the average ride time ratio, and vehicle productivity). This can happen when the difference between parameters is small and the "myopic" nature of ADARTW can make the tradeoff less

predictable. But the global trend between the quality of solution and parameter C_1 is obvious and predictable in the long run.

The utilization rate[1] of vehicles in column 6 of Table 5.1 remains more or less constant at the value of 0.96. In other words, only 4% of the total vehicle time is slack time. We would expect that as the time window gets smaller, the utilization rate would decrease since the vehicles are more likely to wait for customers in order to meet their time window constraints.

Since vehicle capacities are assumed to be unlimited during all test runs, it is important that we record the maximum vehicle capacity required by the ADARTW-generated schedule. It can be seen in the last column of Table 5.1 that the maximum capacity required is quite small (at maximum, 6) when compared with other forms of public transportation. However, it is recognized that the required capacity might increase significantly if we consider situations like groups of passengers travelling together, the demand population being unevenly distributed within the service region, the possibility of peak hour demand, etc.

(ii) Parameter C_2:

---

[1]Note that the total vehicle time used in defining the utilization rate is the sum of the vehicle time for every vehicle between it leaves the depot and after delivering the last customer assigned to it.

Here we assume that the disutility function, DU_i^d, is quadratic in the service time deviation: $DU_i^d = C_2 \cdot x_i^2$. Similar experimental runs were designed as in the case of C_1. Table 5.2 compiles statistics on the scheduling results of ADARTW for different values of C_2.

For run #1, the number of vehicles required is 13 when $C_2 = 0$. It increased to 14 and stayed at 14 for $C_2 = 0.1$ to 0.9. It again increased to 15 after C_2 reaches the value of 2. The increasing trend in this regard is consistent with our observation during previous investigation of C_1.

In Table 5.2 the average deviation decreases as the value of C_2 increases until it reaches a minimum of 3.25 minutes when $C_2 = 2$. Then when C_2 becomes disproportionally large with respect to the vehicle resource terms (i.e. $C_2 \gg C_7$, C_8) the average deviation responds to changes of C_2 somewhat randomly. This phenomenon is also observed in Table 5.1 when C_1 becomes exceedingly large. It is believed that there exists a limit up to which the average deviation can be decreased by assigning more weight to the disutility term. After the weight exceeds this limit, the result is governed more by chance than by the weight itself.

Other statistics listed in Table 5.2 follow the same pattern as those listed in Table 5.2. Specifically, both the average ride time ratio and the vehicle productivity decrease as the parameter C_2 increases. The utilization rate remains more or less constant at 0.96 throughout the runs. The maximum capacity required is similar to that of Table 5.1.

After we have examined the linear and quadratic disutility

Table 5.2 Base case scenario with varying C_2

(C_7 = 1, C_8 = 0.8, Others = 0)

Run	C_2	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	13	14.59	1.60	4.54	0.94	6
#2	0.05	12	11.01	1.62	4.44	0.95	6
#3	0.1	14	8.87	1.60	4.11	0.95	5
#4	0.3	14	5.60	1.51	3.90	0.96	6
#5	0.5	14	5.33	1.49	3.60	0.96	5
#6	0.7	14	4.87	1.47	3.67	0.96	4
#7	0.9	14	4.20	1.49	3.60	0.96	4
#8	2	15	3.25	1.44	3.37	0.95	5
#9	50	15	3.98	1.41	3.36	0.95	4
#10	100	15	5.64	1.42	3.39	0.98	4
#11	200	15	4.45	1.42	3.38	0.96	4

functions individually, it is appropriate to ask the following questions:

1) What is the difference between the two functions?

2) Which functional form should be used in practice?

It is not difficult to see that if we have two functions with the same mean, the quadratic function will assign more customer disutility to longer deviations than the linear function. To demonstrate this point, we have selected an example of two particular disutility functions, one linear and the other one quadratic. Parameters C_1 and C_2 in each function are chosen to be 1 and 0.05 respectively such that the two disutility functions when considered over a uniform distribution in a time window of 30 minutes have the same mean.

a) Linear function

$$DU_i^d = C_1 \cdot x_i$$

$$= x_i \qquad (C_1 = 1)$$

$$E(DU_i^d) = \int_0^{30} \frac{1}{30} \cdot x_i \, dx_i$$

$$= 15$$

b) Quadratic function

$$DU_i^d = C_2 \cdot x_i^2$$

$$= 0.05 \cdot x_i^2 \qquad (C_2 = 0.05)$$

$$E(DU_i^d) = \int_0^{30} \frac{1}{30} \cdot x_i^2 \, dx_i$$

$$= 15$$

The results of ADARTW using each disutility function can be found by as run #2 in Table 5.1 and Table 5.2. The linear disutility function results in an average deviation of 11.94 minutes and the quadratic

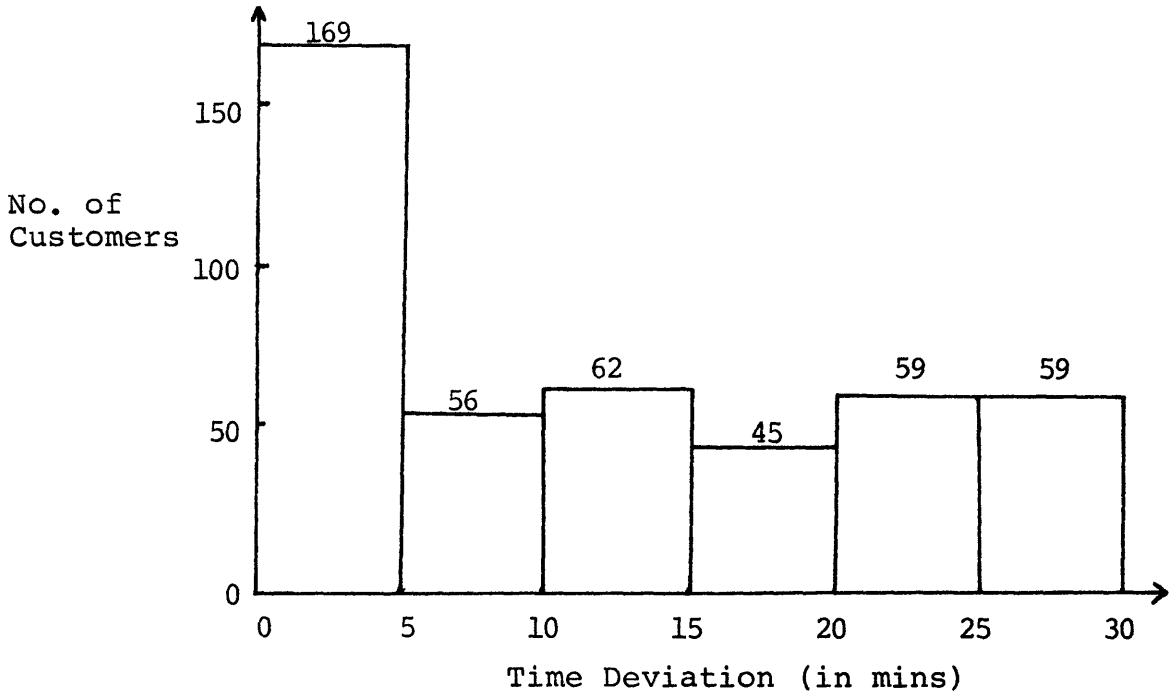
function of 11.01 minutes. Figure 5.1(a) and (b) plot the histograms of customer service time deviations under each scenario. By comparing the distribution of deviations in each figure, it can be seen that the quadratic function results in fewer deviations in the high-end range, e.g. 47 deviations in the range of 25 to 30 minutes (59 for the linear function) and 40 in the range of 20 to 25 minutes (59 for the linear function). This observation indicates that the difference between the two functions lies in the distribution of deviations in the schedules.

The second question on when to use which function refers essentially to a policy issue. There is no clear indication that the quadratic function will perform better than the linear function in areas other than generating a preferable distribution of deviations. For practical purposes, the quadratic function is suggested for large time windows since it is a more realistic representation of the disutility function in such cases.

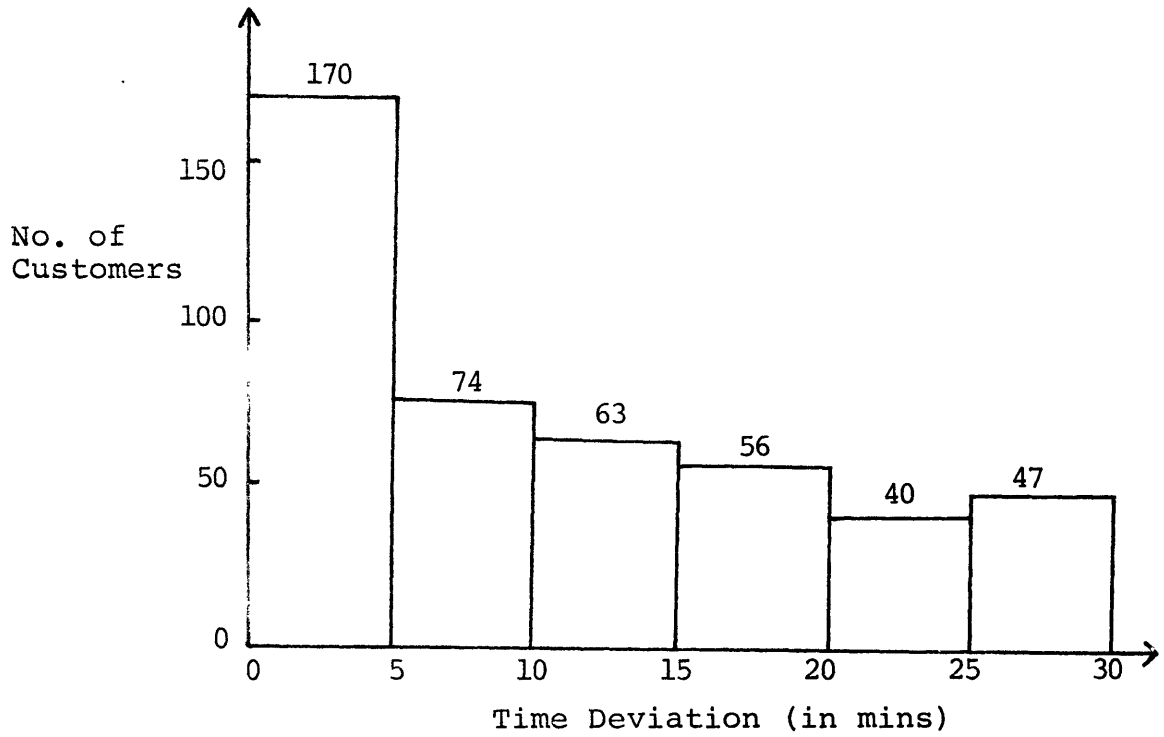
(iii) Parameter C_3:

Parameter C_3 represents the weight placed on the linear term of excess ride time in the disutility function of (4-29). In this part of the investigation it is assumed that $DU_i^r = C_3 \cdot y_i$.

Table 5.3 compiles the results from a sequence of runs of ADARTW using different values for C_3, ranging from 0 to 200. As expected, the average ride time ratio in column 4 decreased from a maximum of 1.60 when $C_3 = 0$ to a minimum of 1.04 when C_3 is sequentially increased to 100.



(a) Linear DU^d ($C_1 = 1$)



(b) Quadratic DU^d ($C_2 = 0.05$)

Figure 5.1 Histograms of service time deviations.

Table 5.3 Base case scenario with varying C_3

(C_7 = 1, C_8 = 0.8, Others = 0)

Run	C_3	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	13	14.59	1.60	4.54	0.94	6
#2	1	13	14.22	1.49	4.57	0.92	8
#3	3	13	15.15	1.28	4.00	0.89	5
#4	5	11	15.23	1.22	4.45	0.96	5
#5	7	13	15.92	1.12	4.08	0.97	5
#6	9	14	15.63	1.07	3.97	0.98	5
#7	100	15	15.62	1.04	3.39	0.98	4
#8	200	15	15.42	1.04	3.41	0.98	4

Since no disutility function was specified for service time deviation in these runs, the average deviations recorded remain at approximately 15 minutes which happens to be one half of the time window size. Also in Table 5.3, the vehicle productivity shows a small increase (4.57 from 4.54) when C_3 is increased from 0 to 1, but decreases to 4.00 in the next run (C_3 = 3). It increases again at C_3 = 5 and decreases gradually afterwards. Note that the number of vehicles used in run #4 (C_3 = 5) is 11 which is the smallest fleet size found so far to satisfy all demands in this data set. Another interesting result is that depite the small ride time ratio, e.g. 1.04 in run #7, the maximum capacity required remains at least 4. Logically, we would have expected very few shared rides at such a low level of ride time ratio. Finally in column 7, the utilization rate seems to be higher when the average ride time ratio is small.

(iv) Parameter C_4

A quadratic excess-ride-time disutility function is assumed in this test: $DU_i^r = C_4 \cdot y_i^2$. Table 5.4 gives the computational results under this scenario. The parameter C_4 was varied from 0 to 200 with smaller increments at small values. The average ride time ratio indicates a decreasing trend with repect to the increase of C_4 which is consistent with the design purpose of this function. The highest vehicle productivity and the smallest fleet size achieved are recorded in run #2 (C_1 = 0.1). This coincides with the results in Table 5.3 which also indicate that instead of assuming no disutility at all, better vehicle productivity can be achieved by assigning a relatively small weight on the excess ride time disutility.

Table 5.4 Base case scenario with varying C_4

(C_7 = 1, C_8 = 0.8, Others = 0)

Run	C_4	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	13	14.59	1.60	4.54	0.94	6
#2	0.1	12	15.13	1.42	4.73	0.96	5
#3	0.2	13	15.00	1.25	3.99	0.90	4
#4	0.3	13	15.21	1.21	4.11	0.95	6
#5	0.4	13	15.37	1.20	3.90	0.93	4
#6	0.5	13	15.71	1.15	4.03	0.99	3
#7	2	15	15.88	1.07	3.42	0.96	3
#8	100	14	15.12	1.10	3.54	0.99	5
#9	200	14	15.12	1.10	3.54	0.99	5

The difference between a linear DU^r and a quadratic DU^r also lies in the distribution of excess ride times experienced by the customers in the schedule. A quadratic DU^r places a higher penalty on longer excess ride times than the linear DU^r, and thus results in fewer excessively long ride times.

(v) Parameters C_5 and C_6:

Parameters C_5 and C_6 in the vehicle resource function provide with an alternative vehicle resource function which does not employ U_i. All runs so far have used U_i in the vehicle resource function by specifying $C_5 = 0$, $C_6 = 0$ and $C_7 \neq 0$, $C_8 \neq 0$. That is:

$$VC_i = U_i \cdot (C_7 \cdot z_i + C_8 \cdot w_i) \qquad (5\text{-}2)$$

If instead $C_7 = 0$, $C_8 = 0$ and $C_5 \neq 0$, $C_6 \neq 0$, we have the alternative function:

$$VC_i = C_5 \cdot z_i + C_6 \cdot w_i \qquad (5\text{-}3)$$

As explained in Chapter 4, U_i is used to conserve vehicle resources during high demand periods and to put relatively more emphasis on the quality of service during periods of low demand. The question to be addressed is whether the U_i term has been effective in serving the role for which it was originally conceived. During previous test runs, the data set, M30, was used. Since the data assumes a uniform distribution of demands over the simulation time, there is no benefit in using U_i. To further analyze this question, we therefore need another data set with uneven demand level. A new data set, denoted by UM30, was generated in the same way as the M30 data set except that the hourly demand distribution of the 450

customers is: 30, 40, 60, 40, 60, 90, 60, 40, and 30 for the nine hours of simulation time.

To study the effect of the U_i term in the vehicle resource function, we can take two approaches: First, we can perform test runs of ADARTW on data set UM30 using the vehicle resource functions (5-2) and (5-3) and compare the results. Since one function uses U_i and the other one does not, it should be possible to see whether the use of U_i can improve the quality of the solutions. The second approach is to use (5-2) as the vehicle resource function and compare the results of ADARTW on data set UM30 with results on M30.

Table 5.5 summarizes the scheduling results of ADARTW on the new data set UM30 using (5-3) as the vehicle resource function. Table 5.6 compiles similar results (the unbracketed numbers) using function (5-2). To compare the results of different functions directly, we note that since function (5-2) receives U_i times more weight than function (5-3), other parameters in function (5-3) must be scaled down so that the relative weight between parameters is approximately the same for each function. To do this, we assume that the average value for U_i for data set UM30 is approximately 3 (an average of 50 customers per hour divided by a predicted fleet of 16 vehicles). Note that all parameters used in Table 5.5 have been scaled down by a factor of three by comparison to those used in Table 5.6. Because this is only an approximation, it is still difficult to say which run is better than the other. For example, run #2 in Table 5.5 has almost identical service quality and vehicle productivity with run #2 in Table 5.6. By comparing two tables directly (e.g. run #1 in Table

Table 5.5 Use (5-3) as vehicle resource function (U_i not used)
($C_5 = 1$, $C_6 = 0.8$, $C_7 = 0$, $C_8 = 0$, Data set = UM30)

Run	C_1	C_2	C_3	C_4	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	0	0	0	16	15.11	1.66	5.07	0.97	6
#2	.33	0	0	0	16	11.09	1.59	4.51	0.94	4
#3	1	0	0	0	18	7.39	1.53	4.05	0.96	6
#4	1.7	0	0	0	19	5.42	1.48	3.87	0.96	5
#5	0	.03	0	0	17	7.63	1.61	4.23	0.95	6
#6	0	.1	0	0	18	6.39	1.55	3.87	0.97	5
#7	0	.17	0	0	19	5.97	1.50	3.89	0.97	6
#8	0	0	.33	0	16	13.98	1.43	4.73	0.92	6
#9	0	0	1	0	16	14.93	1.28	4.69	0.95	7
#10	0	0	1.7	0	16	15.06	1.18	4.31	0.95	5
#11	0	0	0	.03	16	14.98	1.37	4.75	0.97	5
#12	0	0	0	.1	17	15.23	1.19	4.25	0.96	5
#13	0	0	0	.17	20	14.93	1.17	3.95	0.96	4
#14	.33	0	.33	0	17	10.67	1.44	4.40	0.93	6
#15	0	.03	0	.07	18	9.04	1.25	4.01	0.97	5

Table 5.6 Use (5-2) as vehicle resource function (U_i used)
($C_5 = 0$, $C_6 = 0$, $C_7 = 1$, $C_8 = 0.8$, Data set = UM30 (M30))

Run	C_1	C_2	C_3	C_4	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	0	0	0	16 (13)	15.11 (14.59)	1.66 (1.60)	5.07 (4.54)	0.97 (0.94)	6 (6)
#2	1	0	0	0	16 (12)	11.31 (11.94)	1.60 (1.60)	4.50 (4.47)	0.92 (0.94)	6 (6)
#3	3	0	0	0	18 (15)	8.16 (7.95)	1.59 (1.58)	4.01 (3.77)	0.92 (0.90)	5 (5)
#4	5	0	0	0	19 (14)	6.37 (6.54)	1.47 (1.48)	3.86 (3.81)	0.94 (0.96)	5 (5)
#5	0	0.1	0	0	17 (14)	8.47 (8.87)	1.59 (1.60)	4.20 (4.11)	0.96 (0.95)	6 (5)
#6	0	0.3	0	0	18 (14)	6.82 (5.60)	1.52 (1.51)	3.93 (3.90)	0.96 (0.96)	5 (6)
#7	0	0.5	0	0	20 (14)	5.84 (5.33)	1.48 (1.49)	3.65 (3.60)	0.97 (0.96)	4 (5)
#8	0	0	1	0	16 (13)	15.27 (14.22)	1.46 (1.49)	4.76 (4.57)	0.93 (0.92)	6 (8)
#9	0	0	3	0	17 (13)	13.50 (15.15)	1.33 (1.28)	4.53 (4.00)	0.95 (0.89)	6 (5)
#10	0	0	5	0	18 (11)	13.90 (15.23)	1.22 (1.22)	4.42 (4.45)	0.96 (0.97)	5 (5)
#11	0	0	0	0.1	16 (12)	14.67 (15.13)	1.39 (1.42)	4.61 (4.73)	0.95 (0.96)	6 (5)
#12	0	0	0	0.3	18 (13)	14.20 (15.21)	1.23 (1.21)	4.35 (4.11)	0.97 (0.95)	5 (6)
#13	0	0	0	0.5	18 (13)	14.95 (15.71)	1.17 (1.15)	3.89 (4.03)	0.93 (0.99)	4 (3)
#14	1	0	1	0	16 (12)	11.34 (11.84)	1.41 (1.48)	4.31 (4.37)	0.93 (0.91)	5 (5)
#15	0	0.1	0	0.2	18 (15)	9.19 (8.93)	1.27 (1.25)	3.95 (3.97)	0.95 (0.97)	5 (4)

5.5 vs run #1 in Table 5.6), four runs in Table 5.5 (#7, #9, #10, #12) used fewer vehicles than the corresponding runs in Table 5.6 while two other runs (#13, #14) used more. As far as the vehicle productivity is concerned, 10 out of 14 runs in Table 5.5 score higher than those in Table 5.6. It seems at this point that no significant advantage can be gained by using U_i in the vehicle resource function.

In Table 5.6, two numbers are listed in each box, the one in parentheses represents the results when the M30 data set is used. By comparing each pair of numbers in each box, the only obvious pattern is that it requires more vehicles to satisfy customer demands with an uneven pattern, an observation which is not at all surprising. Ten out of fifteen runs using data set UM30 recorded higher vehicle productivity than those using M30. Other statistics remain more or less the same.

The lack of difference between the results of the two vehicle resource functions might be attributed to the following reasons:

a) the uneven demand pattern designed for data set UM30 is not sufficiently significant for U_i to be useful.

b) it might be misleading to consider just the average performance. A detailed study of the schedule for each hour could perhaps reveal more information on the benefits from use of U_i. For example, the average time deviation and vehicle productivity in the peak hour should be different from those of off-peak hours.

(vi) Parameters C_7 and C_8:

During test runs conducted in (i) - (iv), C_7 and C_8 were set to be constants and other parameters were changed relative to this basis. For the following discussion, we again fix the value of C_7 (= 1) as the base and vary C_8 in the successive test runs. The purpose is to find out the relationship between C_7, the weight on vehicle active time and C_8, the weight on vehicle slack time. In Table 5.7, scheduling results are tabulated for different values of C_8. It seems that when no weight is placed on vehicle slack time ($C_8 = 0$) the vehicle productivity obtained is the lowest and the utilization rate is also low. This is as expected since when no penalty is imposed on the vehicle slack time ADARTW will actually encourage those insertions which incur slack time and thus result in low utilization of vehicles. However, it is interesting to see that in run #7 a large C_8 was specified, but the utilization rate is lower than in those runs using smaller C_8. With an exceedingly large C_8, ADARTW is likely to discourage the increase of vehicle slack time in the short run. But when a customer happens to be infeasible to most of the vehicles and a case 1 insertion (see Figure 4.5(a)) with significant slack time is the only feasible insertion, ADARTW has no choice but to make that feasible insertion and thus increase vehicle slack time in the long run.

Other statistics in Table 5.7 seem to be more influenced by chance and no significant trend can be discerned. It is conjectured that varying C_8 will not result in much systematic change of the results of ADARTW.

(vii) Simultaneous presence of DU^d and DU^r:

In (i) - (iv) above, we vary one parameter at a time so that we can

Table 5.7 Base case scenario with varying C_8

(C_7 = 1, Others = 0)

Run	C_8	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0	12	14.73	1.63	4.35	0.85	7
#2	0.2	11	14.41	1.65	4.57	0.90	6
#3	0.4	13	13.73	1.61	4.70	0.96	6
#4	0.6	13	13.73	1.61	4.70	0.96	6
#5	0.8	13	14.59	1.60	4.54	0.94	6
#6	1	12	13.95	1.61	4.65	0.95	6
#7	100	13	14.01	1.61	4.42	0.93	7

differentiate the effects of one parameter from another. Based upon these results, we now selectively investigate several cases where parameters were varied simultaneously. Two basic scenarios are assumed: First, DU^d and DU^r are both linear functions. Second, DU^d and DU^r are both quadratic functions. Table 5.8 gives the results when DU^d and DU^r are linear. Table 5.9 provides the results when DU^d and DU^r are quadratic.

Table 5.8 provides the results of nine test runs using linear DU^d and DU^r. The first set of four runs (using $C_3 = 1$) seems to be inferior to the second set of four runs (using $C_3 = 2$) in terms of the number of vehicles used, the average deviation, and the average ride time ratio. When C_1 and C_3 become dominant by comparison to C_7 and C_8 in the objective function (run #9, C_1=100 , C_3=100), the number of vehicles required increases to 16, but no reduction in average deviation is realized. This is consistent with the results in Table 5.1 and Table 5.2 when C_1 and C_2 are unilaterally increased to become dominant in the objective function.

Table 5.9 shows the results of another nine test runs using quadratic DU^d and DU^r. The average deviation and ride time ratio responded reasonably well to the relative changes of C_2 and C_4. For $C_2 = 100$ and $C_4 = 100$ in run #9, the result is similar to that of $C_1 = 100$ and $C_3 = 100$ in Table 5.8.

5.2.2.2 Computational results on variations of ADARTW

For the second part of our investigation using simulated data, we focus on the effects of considering multiple candidates for each

Table 5.8 Linear disutility functions, DU^d and DU^r.

($C_7 = 1$, $C_8 = 0.8$, Others = 0)

Run	C_1	C_3	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	1	1	12	11.84	1.48	4.37	0.91	5
#2	3	1	13	8.90	1.48	4.13	0.97	5
#3	5	1	13	6.41	1.44	3.96	0.99	5
#4	7	1	15	5.39	1.40	3.74	0.98	5
#5	1	2	11	12.67	1.42	4.52	0.92	5
#6	3	2	13	8.16	1.39	4.11	0.95	5
#7	5	2	13	5.44	1.38	3.90	0.96	5
#8	7	2	14	4.49	1.32	3.65	0.95	5
#9	100	100	16	5.55	1.15	3.33	0.96	4

Table 5.9 Quadratic disutility functions, DU^d and DU^r.

(C_7 = 1, C_8 = 0.8, Others = 0)

Run	C_2	C_4	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required
#1	0.1	0.1	14	9.25	1.34	4.24	0.97	4
#2	0.3	0.1	13	6.51	1.42	3.90	0.94	5
#3	0.5	0.1	14	5.48	1.36	3.62	0.96	5
#4	0.7	0.1	14	4.98	1.38	3.64	0.97	5
#5	0.1	0.2	15	8.93	1.25	3.97	0.97	4
#6	0.3	0.2	13	6.64	1.29	3.85	0.96	4
#7	0.5	0.2	14	6.80	1.35	3.79	0.98	5
#8	0.7	0.2	14	5.38	1.29	3.62	0.98	5
#9	100	100	16	5.64	1.17	3.21	0.96	3

insertion. Both pool-refilling strategies are tested. We also evaluate the benefits gained by considering insertions made into different schedule blocks.

(i) Multiple candidates

As described in Chapter 4, one variation of ADARTW is to consider multiple candidates for the next insertion, instead of just considering one each time. Here we conduct tests runs using various numbers of simultaneous candidates to see whether the performance of ADARTW can be improved. Three data sets, L30, M30, and H30, were used so that the impact of different levels of demand in this respect can be examined. The size of the candidates pool is varied from 1 to 50 candidates by arbitrary increments. Moreover, two sets of parameters are used for the objective function in this experiment. The first set, $C_1 = 0.1$, $C_3 = 0.5$, $C_7 = 1$, $C_8 = 0.8$, others = 0, is used primarily to find out whether multiple candidates will improve the vehicle productivity since C_7 is the dominant parameter in this objective function from our previous experience. The second set, $C_1 = 3$, $C_3 = 1$, $C_7 = 1$, $C_8 = 0.8$, others = 0, is used to investigate whether the quality of service can be improved by considering multiple candidates. Table 5.10(a), (b) and (c) summarizes the statistics of ADARTW schedules for the three data sets using the first set of parameters. Similarly, Table 5.11(a), (b) and (c) summarizes the results using the second set of parameters. Both pool-refilling strategies were tested for each case. The statistics listed at the upper-left corner in each box of the two tables is obtained by using the "immediate-refill" strategy. The one on the lower-right corner uses the "periodic-refill"

Table 5.10(a) Multiple candidates using two pool-refilling strategies

(C_1 = 0.1, C_3 = 0.5, C_7 = 1, C_8 = 0.8, Others = 0)

(Data set = L30)

Run	Pool Size	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.
#1	1	5	13.17	1.35	2.89	0.84	4	4885
		5	13.17	1.35	2.89	0.84	4	4885
#2	3	4	11.95	1.29	2.58	0.75	4	5394
		5	12.02	1.36	2.81	0.83	3	4990
#3	5	4	13.85	1.30	2.85	0.79	3	5054
		4	13.06	1.37	3.02	0.88	4	4968
#4	10	4	14.82	1.35	2.74	0.80	4	5704
		4	12.83	1.36	2.87	0.80	4	5225
#5	15	5	13.22	1.32	2.72	0.80	3	5550
		4	14.43	1.38	2.94	0.85	4	4883
#6	20	5	12.72	1.26	2.90	0.85	4	4786
		4	13.17	1.34	2.93	0.84	4	4929
#7	30	4	15.13	1.44	3.01	0.81	4	5193
		5	13.17	1.35	2.89	0.84	4	4885
#8	40	6	14.16	1.28	2.98	0.87	4	5313
		6	14.76	1.28	2.94	0.86	4	4386

Table 5.10(b) Multiple candidates using two pool-refilling strategies

(C_1 = 0.1, C_3 = 0.5, C_7 = 1, C_8 = 0.8, Others = 0)

(Data set = M30)

Run	Pool Size	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.
#1	1	12	13.89	1.51	4.32	0.88	6	27232
		12	13.89	1.51	4.32	0.88	6	27232
#2	5	14	13.48	1.50	4.72	0.94	6	27267
		12	14.37	1.51	4.43	0.88	6	27200
#3	10	13	13.61	1.51	4.57	0.94	5	28495
		12	14.09	1.52	4.90	0.97	7	27292
#4	20	12	13.68	1.55	4.57	0.95	7	28741
		12	13.73	1.49	4.64	0.92	6	25969
#5	30	12	13.36	1.53	4.39	0.91	7	28823
		11	13.34	1.53	4.63	0.93	6	27744
#6	50	13	13.94	1.57	4.31	0.89	6	28284
		12	13.99	1.50	4.44	0.91	6	27000

Table 5.10(c) Multiple candidates using two pool-refilling strategies

(C_1 = 0.1, C_3 = 0.5, C_7 = 1, C_8 = 0.8, Others = 0)

(Data set = H30)

Run	Pool Size	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.
#1	1	20	14.44	1.60	5.38	0.94	7	51120
		20	14.44	1.60	5.38	0.94	7	51120
#2	10	20	13.87	1.57	5.36	0.95	7	51368
		20	13.64	1.55	5.57	0.96	7	49704
#3	20	20	14.04	1.56	5.37	0.95	7	50844
		20	13.82	1.56	5.47	0.94	6	49977
#4	30	20	13.52	1.58	5.17	0.94	7	52920
		20	13.54	1.58	5.54	0.95	7	49530
#5	40	20	13.56	1.56	5.41	0.97	8	51020
		20	14.18	1.57	5.82	0.98	8	47373
#6	50	21	13.71	1.58	5.07	0.92	8	53621
		20	13.29	1.56	5.44	0.92	7	49805

Table 5.11(a) Multiple candidates using two pool-refilling strategies

(C_1 = 3, C_3 = 1, C_7 = 1, C_8 = 0.8, Others = 0)

(Data set = L30)

Run	Pool Size	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.
#1	1	6	4.72	1.27	2.50	0.89	3	5961
		6	4.72	1.27	2.50	0.89	3	5961
#2	3	6	3.83	1.26	2.09	0.73	3	5840
		5	6.01	1.18	2.53	0.86	3	6597
#3	5	5	5.67	1.29	2.63	0.84	3	6517
		5	4.96	1.19	2.54	0.86	3	6379
#4	10	5	6.18	1.22	2.31	0.79	3	7523
		5	4.40	1.21	2.56	0.85	3	6328
#5	15	5	4.95	1.28	2.59	0.87	3	7070
		5	4.46	1.23	2.51	0.80	3	6046
#6	20	6	6.52	1.32	2.31	0.72	3	6990
		4	6.51	1.26	2.59	0.83	3	7465
#7	30	5	5.55	1.31	2.88	0.89	3	6638
		5	4.62	1.29	2.62	0.83	3	6157
#8	40	5	6.12	1.30	2.65	0.81	3	7279
		5	4.80	1.29	2.37	0.76	3	6781

Table 5.11(b) Multiple candidates using two pool-refilling strategies

(C_1 = 3, C_3 = 1, C_7 = 1, C_8 = 0.8, Others = 0)

(Data set = M30)

Run	Pool Size	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.
#1	1	13	8.90	1.48	4.13	0.97	5	41776
		13	8.90	1.48	4.13	0.97	5	41776
#2	5	14	6.81	1.44	3.87	0.91	5	37828
		13	8.08	1.48	4.11	0.94	5	40290
#3	10	14	7.61	1.46	3.75	0.92	5	39950
		13	7.44	1.45	4.13	0.97	5	39346
#4	20	14	7.57	1.45	4.08	0.96	5	40742
		14	6.86	1.44	4.18	0.96	5	38300
#5	30	13	7.45	1.42	3.94	0.92	5	41360
		14	5.58	1.40	3.88	0.93	5	36535
#6	50	13	7.59	1.45	3.95	0.93	5	43600
		13	5.64	1.43	3.92	0.92	6	36358

Table 5.11(c) Multiple candidates using two pool-refilling strategies

(C_1 = 3, C_3 = 1, C_7 = 1, C_8 = 0.8, Others = 0)

(Data set = H30)

Run	Pool Size	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.
#1	1	21	8.40	1.49	4.67	0.95	5	80348
		21	8.40	1.49	4.67	0.95	5	80348
#2	10	24	7.63	1.48	4.50	0.96	6	77131
		21	7.96	1.51	4.78	0.96	6	78365
#3	20	23	7.55	1.47	4.68	0.96	6	77935
		23	7.17	1.48	4.62	0.95	6	75091
#4	30	22	7.94	1.49	4.75	0.97	7	79433
		22	6.85	1.45	4.74	0.97	6	73996
#5	40	23	7.60	1.50	4.44	0.92	6	79362
		22	5.77	1.49	4.70	0.96	6	72039
#6	50	22	7.26	1.46	4.62	0.95	6	77260
		23	5.36	1.45	4.64	0.95	6	70070

strategy. The objective function value (O.F.V.) on the last column in the tables indicates the accumulated total insertion cost.

We divide the discussion of results into two parts. In the first part, we examine the immediate-refill strategy and in the second part we examine the periodic-refill strategy. Within each part, the results using each of the two different sets of parameters will be discussed separately. Recall that the first set of parameters is biased towards producing high vehicle productivity and the second set is biased towards producing better service quality.

1) immediate-refill strategy

(a) $C_1 = 0.1$, $C_3 = 0.5$, $C_7 = 1$, $C_8 = 0.8$. Others = 0.

In Table 5.10(a), the vehicle productivity is recorded at 2.89 customers per vehicle hour for the basic version of ADARTW (run #1). By considering 3 candidate customers each time (run #2), the number of vehicles required decreases to 4, but the vehicle productivity also decreases to 2.58. Similar results are observed for runs #3 and #4 when compared to run #1. It is not until the pool size reached 20, that the vehicle productivity begins to outperform the basic version. For the objective function value, nearly all runs using multiple candidates result in higher total cost in this regard. Other statistics show no clear trends. In Table 5.10(b) which compiles the results for data set M30, vehicle productivities are higher than the basic version for 4 (run #2, #3, #4, and #5) out of 5 runs using pool sizes other than one. But the

objective function values are also higher for these runs. Again in Table 5.10(c), after comparing runs from #2 to #5 with run #1, it seems that by increasing the number of candidates for each consideration using the immediate-refill strategy does not necessarily improve vehicle productivity. Sometimes, the solution even deteriorates.

(b) $C_1 = 3$, $C_3 = 1$, $C_7 = 1$, $C_8 = 0.8$. Others = 0.

In Table 5.11(a), no indication of any improvement over the service quality statistics (i.e. the average deviation and the average ride time ratio) can be found by increasing the pool size. However, in Table 5.11(b), a smaller deviation and a smaller ride time ratio (as compared to run #1) are recorded when pool size is increased. Similar results can also be found in Table 5.11(c), but at the cost of using additional vehicles.

2) periodic-refill strategy

(a) $C_1 = 0.1$, $C_3 = 0.5$, $C_7 = 1$, $C_8 = 0.8$. Others = 0.

Vehicle productivity statistics shown in Table 5.10(a) for the periodic-refill strategy indicate possible improvements using pool sizes greater than one. Runs #2 and #4 recorded 3% and 1% less vehicle productivity, but the other five runs resulted in higher productivity than run #1. Among all runs, only run #8 used more vehicles than run #1, while other runs used the same number or fewer vehicles. In Table 5.10(b), vehicle productivity increased by the order of 3% to 12% for various pool sizes while using the same number or fewer vehicles. This observation is

further supported by the results shown in Table 5.10(c) in which higher vehicle productivity was achieved by using pool sizes greater than one.

(b) $C_1 = 3$, $C_3 = 1$, $C_7 = 1$, $C_8 = 0.8$. Others = 0.

Statistics for this set of parameters are listed in Table 5.11(a), (b) and (c). In Table 5.11(a), the average deviation decreased for three out of seven runs using larger pool sizes. This can be considered to be an improvement if we take into account the fact that only four or five vehicles were used during those runs (run #1 uses six vehicles). Lower ride time ratios are also observed in runs # 2 to #6. Again, the increase of ride time ratio for runs #7 and #8 can be partially attributed to the smaller fleet sizes used. In Table 5.11(b), better service quality is consistently obtained by implementing pool sizes greater than one. Results for the immediate-refill strategy in Table 5.11(c) support this observation, but at the cost of additional vehicle resources.

(ii) Is the consideration of insertions into different blocks worthwhile?

In Chapter 4 we presented two fast screening tests to determine the feasibility of an insertion depending on whether the insertion is made to the same schedule block or to different schedule blocks. It is conceivable that in considering insertions into different schedule blocks the algorithm might consume a significant additional amount of computer time. It is therefore necessary to examine whether this additional effort is worthwhile in terms of improving the quality of solutions. We will evaluate the addtional computation effort involved when we examine the

computational efficiency of ADARTW in the next section. In this section we focus on the quality of solutions produced by such additional effort.

Again we use M30 as the primary test data set. Table 5.12 displays the test results obtained from a set of runs of ADARTW with a limited combination of parameter values. Each box in Table 5.12 contains two numbers. The number on the upper-left corner of each box is obtained by considering only insertions into the same schedule blocks. The other number at the lower-right corner represents the results obtained by considering all possible insertions.

The last column of Table 5.12 shows the number of insertions actually made into different schedule blocks when they are considered in ADARTW. A number 0 indicates that although such insertions are considered in the run, none is actually made. We note in Table 5.12 that those runs with zero count (run #9 through #14) used a significant excess ride time disutility function. Usually, insertions made into different schedule blocks involve a long excess ride time for the customer being considered and possibly others already assigned to the vehicle (since the slack time between schedule blocks involved must be converted to active time so that the vehicle will not wait with that customer on board). A large penalty on the excess ride time will thus certainly discourage such insertions. As far as the number of vehicles required is concerned, runs #3 , #5, #8, #15 used fewer vehicles when additional insertions are considered, but runs #6 and #7 used fewer vehicles. For the average deviation, we examine runs #2 through #6 and runs #14 and #15 in which a disutility function was specified in this regard. We note that runs #3, #5, #6, #7, and #15

Table 5.12 Comparison between considering all possible insertions and considering only insertions into the same schedule block. ($C_5 = 0$, $C_6 = 0$, $C_7 = 1$, $C_8 = 0.8$, Data set = M30)

Run	C_1	C_2	C_3	C_4	No. of Vehicles Required	Average Deviation (minutes)	Average Ride-Time Ratio	Vehicle Prod.	Utiliz. Rate	Maximum Capacity Required	O.F.V.	DB* COUNT
#1	0	0	0	0	13 13	14.61 14.59	1.60 1.60	4.54 4.54	0.93 0.94	6 6	25481 25471	0 1
#2	1	0	0	0	12 12	11.87 11.94	1.61 1.60	4.21 4.47	0.90 0.94	6 6	31084 29908	0 1
#3	3	0	0	0	13 15	8.25 7.95	1.57 1.58	3.90 3.77	0.91 0.90	5 5	37185 36832	0 4
#4	5	0	0	0	14 14	6.42 6.54	1.50 1.48	3.80 3.81	0.97 0.96	6 5	41289 41622	0 21
#5	0	0.1	0	0	13 14	9.00 8.87	1.60 1.60	4.14 4.11	0.94 0.95	5 5	33216 32546	0 1
#6	0	0.3	0	0	15 14	6.53 5.60	1.51 1.51	3.79 3.90	0.96 0.96	5 6	36560 35010	0 10
#7	0	0.5	0	0	15 14	5.57 5.33	1.44 1.49	3.61 3.60	0.95 0.96	5 5	44025 42894	0 8
#8	0	0	1	0	11 13	13.79 14.22	1.48 1.49	4.63 4.57	0.92 0.92	8 8	28069 27681	0 2
#9	0	0	3	0	13 13	15.15 15.15	1.28 1.28	4.00 4.00	0.89 0.89	5 5	30901 30901	0 0
#10	0	0	5	0	11 11	15.23 15.23	1.22 1.22	4.45 4.45	0.96 0.96	5 5	33531 33531	0 0
#11	0	0	0	0.1	12 12	15.13 15.13	1.42 1.42	4.73 4.73	0.96 0.96	5 5	30611 30611	0 0
#12	0	0	0	0.2	13 13	15.00 15.00	1.25 1.25	3.99 3.99	0.90 0.90	4 4	31664 31664	0 0
#13	0	0	0	0.3	13 13	15.21 15.21	1.21 1.21	4.11 4.11	0.95 0.95	6 6	33229 33229	0 0
#14	1	0	1	0	12 12	11.84 11.84	1.48 1.48	4.37 4.37	0.91 0.91	5 5	33400 33400	0 0
#15	0	0.1	0	0.1	13 14	9.28 9.25	1.36 1.34	4.27 4.24	0.98 0.97	4 4	37623 37776	0 2

* DB COUNT denotes the number of insertions actually made into different schedule blocks

recorded a smaller average deviation when all possible insertions are considered. Only runs #2 and #4 have larger average deviation. The vehicle productivity and the utilization rate listed in Table 5.12 show no significant difference between the two considerations. The accumulated objective function values are smaller for 7 out of 9 runs (where there is a difference) in favor of considering all possible insertion. It seems at this point that a slight advantage could be gained by considering all possible insertions, but no guarantee of this is possible.

5.3 Computational Efficiency

ADARTW has been coded in PL/I and implemented on a VAX 11/750 minicomputer. Table 5.13 indicates the average CPU times that a single run of ADARTW will take for various problem sizes and time windows. The CPU times listed in the table include the data input time and computation time of ADARTW, but do not include the time to print out the vehicle schedules. It can be seen that for the same problem size, wider time-window cases take more CPU time than shorter time-window cases. For example, for the set of medium demand data (M10, M20, and M30), an average of 24% increase in CPU time is observed whenever the time window is increased by 10 minutes each time. The extra computation time involved can be attributed to the fact that wider time windows will result in more feasible insertions and consequently it takes more time for ADARTW to evaluate the cost of such feasible insertions. The table also records computation times for considering different-block insertions in ADARTW as opposed to the case of not considering them. The extra computation effort required for considering different-block insertions are negligible for the cases with

Table 5.13 Computation time of ADARTW on VAX 11/750.
(CPU time in seconds)

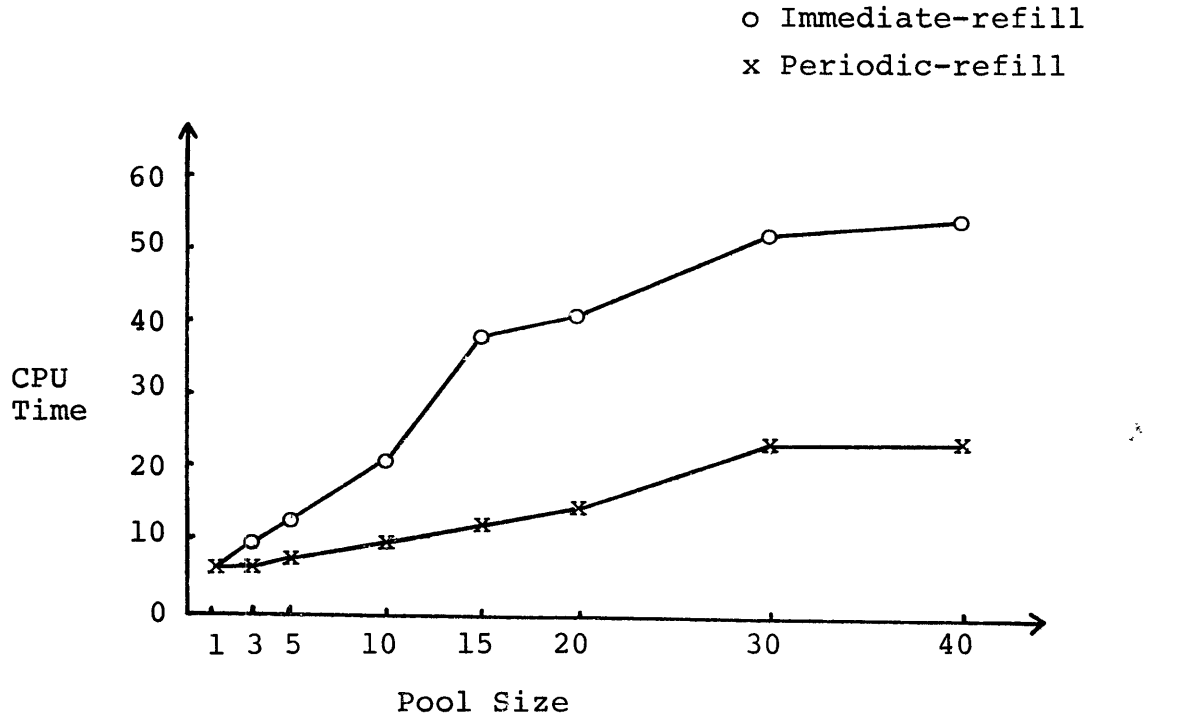
Runs \ Data Sets	L10	L20	L30	M10	M20	M30	H10	H20	H30
(A)*	4.7	5	5.4	47	58	69	162	203	250
(B)**	4.7	5	5.6	48	59	72	164	206	259

* The set of runs in (A) does not consider insertions into different schedule blocks.

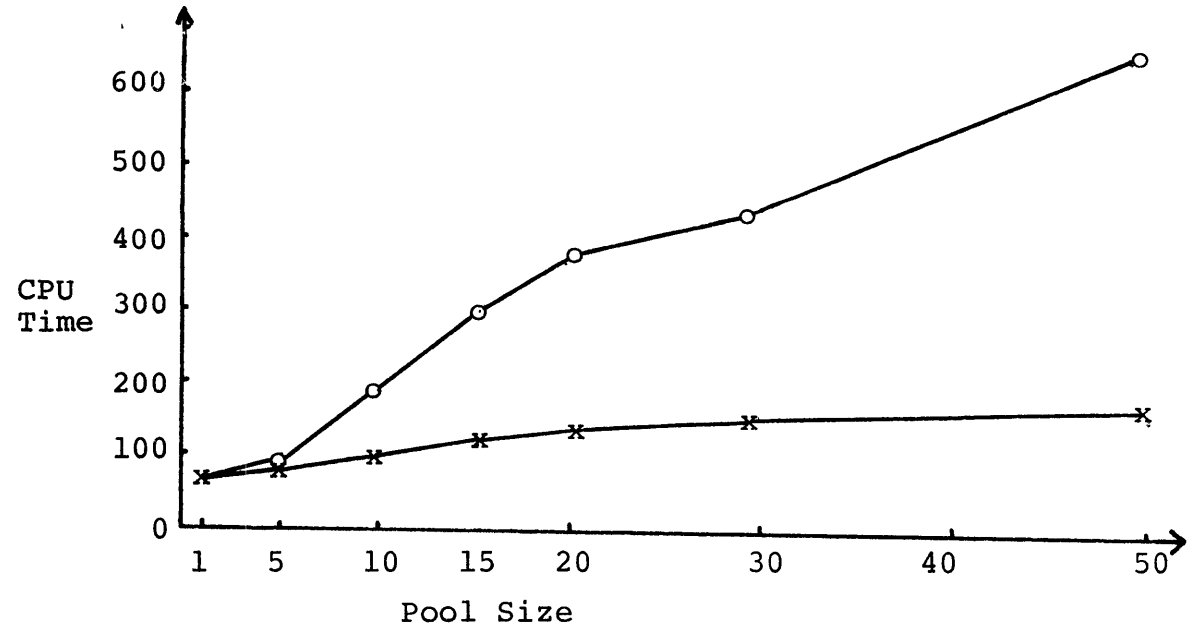
** The set of runs in (B) considers all possible insertions.

time windows not greater than 20 minutes. A small increase in computation time is observed for the cases with 30-minute time window. This observation is consistent with the earlier statement that wider time windows will result in more feasible insertions. This is especially so in the case in which we allow different-block insertions.

The computation effort required for considering simultaneously more than one customer at a time as candidates for the next insertion is illustrated in Figure 5.2(a), (b), and (c). Both pool-refilling strategies were investigated over a range of pool sizes. The positive correlation between computation time and pool size, as depicted in Figure 5.2(a), (b), and (c), is well expected since more candidates in the pool would certainly require more computations before ADARTW can choose the best candidate. However, such increasing trends are more significant for the immediate-refill strategy than for the periodic-refill strategy. The differences in computation time between the two strategies are mainly due to the updating efforts involved after a customer is chosen to leave the pool. To demonstrate this point, let us assume that the pool size is d and there are N customers on the subscription list. When a customer in the candidate pool is inserted into the work-schedule of vehicle j, it is required that for each of the remaining candidates in the pool their insertion costs to vehicle j be updated. We use the term "one update" to denote the evaluation of the insertion cost of a customer to a vehicle. Under the immediate-refill strategy, the number of updates required when a customer leaves the pool is d-1 since there are always d-1 unassigned customers in the pool. Consequently, the updating procedure would be executed $(d-1)\cdot N$ times in total since there are N customers in the list.

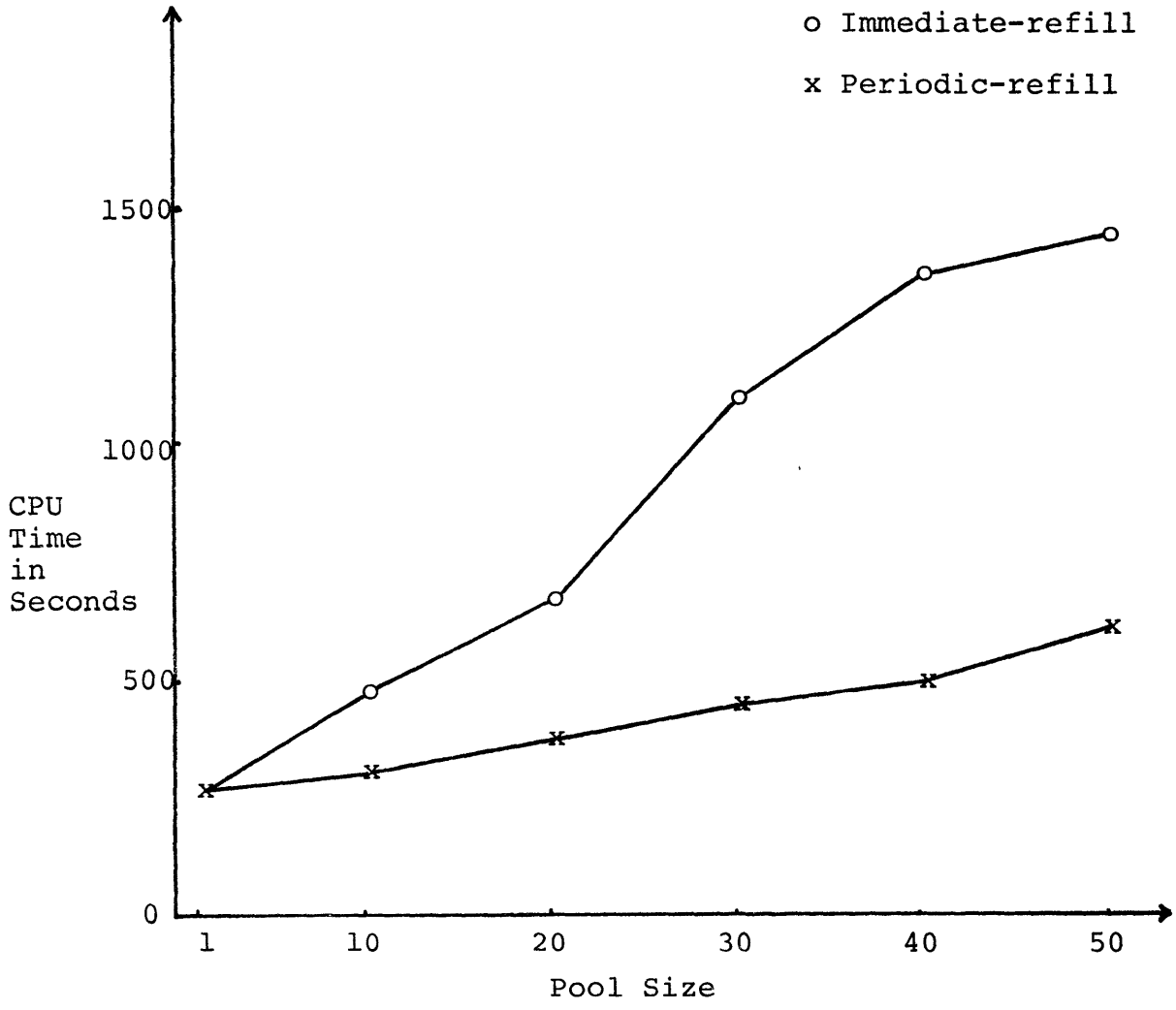


(a) Data Set L30 - 90 customers



(b) Data Set M30 - 450 customers

Figure 5.2 Computation time of ADARTW when multiple candidates are considered. (CPU time in seconds)



(c) Data Set H30 - 900 customers

Figure 5.2 Computation time of ADARTW when multiple candidates are considered.

Under the periodic-refill strategy, the number of updates required when a customer leaves the pool is not constant. It varies between 0 and d-1 depending on the number of unassigned customers in the pool at the time of updating. The total number of updates required for a batch of d customers can be computed to be (d)(d-1)/2. Since there are N/d batches of customers to be processed, the total number of updates required for the periodic-refill strategy is (d-1)N/2 which is only half of what is required for the immediate-refill strategy.

Another somewhat minor contributing factor to the significant increase of computation time under the immediate-refill strategy is the effort involved in the selection of the best customer from a pool of candidates. Since the pool is always full when a selection is made under the continuous-refill strategy, ADARTW has to perform d comparisons in order to choose the minimal cost insertion. Whereas under the periodic-refill strategy, an average of d/2 comparisons is required each time. The total difference between the two strategies is again proportional to (d/2)·N.

So far we have examined empirically the computational efficiency of ADARTW. It is also important that we study the computation complexity of ADARTW from a theoretical point of view. Normally, the computation complexity of an algorithm can be determined by the number of elementary operations required in the process. For ADARTW, the number of possible insertions for a customer can go as high as N^2 (worst case scenario when all N customers are assigned to one vehicle). For each insertion a feasibility check is required. Despite the fast screening test we have

designed to cut down the computation time, a linear search has to be performed through the list of vehicle work-schedules which can take as many as c·N comparisons (c is some constant). So for each customer, $O(N^3)$ elementary operations are necessary for the worst case scenario. Since there are N customers, the computational complexity of ADARTW is $O(N^4)$. When considering multiple candidates in ADARTW, the extra effort involved is also of order N^3 (if $d << N$) for both pool-refilling strategies. So the computational effort of ADARTW remains at $O(N^4)$ when multiple candidates are considered at the same time.

Empirically, from Figure 5.2(a), (b), and (c), the computation time of ADARTW does not seem to grow as fast as $O(N^4)$ as is suggested theoretically. This is due to the fact that $O(N^4)$ is just the worst case and such a worst case rarely happens in practice. For example, in practice, the N customers are more likely to be evenly distributed among m vehicles. Thus, the computation effort of processing one customer reduces to the order of $m\cdot(N/m)^3$. The efficient screening tests introduced in Chapter 4 will also normally reduce the computation time by detecting infeasibilities early in the process.

5.4 Runs Using Real-World Data

5.4.1 Data

The real-world customer data base is obtained from the flexible-route system operated by Rufbus AmbH Bodenseekreis in the city of Friedrichshafen, West Germany. The same database was used in Chapter 3 to

test the Grouping-Clustering-Routing (GCR) algorithm. In this part of the investigation we have made similar conversions of immediate-request customers to advance-request customers as in Chapter 3 by defining their DPT as the time of request.

The schedule in the database had been obtained by Rufbus schedulers by use of a man-machine procedure, details of which were not available. We have been told that Rufbus operators tried to keep a 15-minute time window, although at times this constraint was relaxed to 60 minutes to avoid rejecting customers.

5.4.2 Computational Results

Table 5.14 highlights statistics from one particular run of ADARTW together with statistics of the actual scheduling (the latter displayed for illustration purposes rather than for comparison). Options exercised for this run of ADARTW were the following:

Time Window : 15 minutes

Maximum Ride Time : 5 minutes + 1.5 · (Direct Ride Time)

Vehicle Capacity : 17 customers (all vehicles)

Initial Fleet Size : 10 vehicles

Customer service Option : Add more vehicles if necessary

$C_1 = 3$, $C_2 = C_3 = C_4 = C_5 = C_6 = 0$, $C_7 = 1$, $C_8 = 0.8$

Table 5.14 Performance statistics of Rufbus schedule and ADARTW schedule.

	Rufbus Scheduling ("mixed" case)	ADARTW Scheduling ("advance-request" case)
Vehicles used	28	17
Vehicle productivity	8.87	12.06
Average deviation (minutes)	11.9	6.6
Ride time ratio	N.A.	1.54

Since the Rufbus procedure was essentially applied to a "mixed" demand scenario (where some of the requests were "immediate" and some were "advance") while ADARTW was applied to a case where all demands were converted to advance-request, it must be stressed that no direct comparison of the two procedures can be made from Table 5.14. However, the statistics in the table demonstrates that considerable margins (with 40% less vehicles achieving 40% smaller deviations) exist between ADARTW's performance and the Rufbus Schedule in this run. The CPU time for this run of ADARTW was about 12 minutes on the VAX 11/750.

CHAPTER 6 DISCUSSION AND CONCLUSIONS

6.1 On Grouping-Clustering-Routing Algorithm (GCR)

The development of the GCR algorithm described in Chapter 3 has made a number of contributions in the area of routing and scheduling of dial-a-ride vehicles: First, it provides a systematic methodology for tackling the problem of forming clusters, a problem that has received little or no attention in the literature to date. Second, the overall approach has been applied viably to problems of far greater size than have been attempted thus far. Finally, extensive computational experience with the GCR algorithm over a broad range of problem sizes was reported.

In this section we further explore ways through which the performance of this algorithm can be enhanced. To begin with, we address the following question: "How successful is the algorithm in producing schedules that give some guarantee to the system user that the service provided would be of acceptable quality?"

Although the GCR algorithm is not intended for problems with strict service quality constraints under which ADARTW is conceived, such a question is certainly legitimate given the way the algorithm treats the time dimension of the problem (within the "grouping" part). In this respect, it would appear (at least from a worst-case point of view) that there are definitely risks that a customer may suffer either an intolerably long pick-up (or delivery) time deviation, or a similarly long excess ride time, or both. Consider for instance the case where 30-minute

time groups are set up starting at 7:00 a.m., and a particular customer has specified a DPT of 7:25 a.m. while his direct ride time is 10 minutes. If CF is chosen to be 1.0, his calculated DDT is at 7:35 a.m. The risk for this customer is that he may be picked up as early as 7:00 a.m. and delivered as late as 8:00; in which case he would suffer a 25-minute early pick-up (which he might be unable to meet), plus, a 50-minute excess ride time (or equivalently, a ride time ratio of 6.0). Other pathological cases can be similarly devised, for instance cases where one customer has such a long direct trip time relative to others, that the value of DT_{min} calculated as per Section 3.2 is intolerably high.

Within the above context we can suggest a straightforward set of adjustment procedures in the grouping part of the algorithm that serve to reduce the risk of such worst-case scenarios. Such adjustments constitute the enhanced grouping part of the algorithm and can either be programmed within the algorithm, or, be left to the discretion of the algorithm's operator who could invoke some or all of them in an interactive way.

With the assumption that the vehicle fleet size is such that the clustering part of the algorithm prevents schedules of a time group from "spilling" consistently into the next time group, we give below an idea of some of these adjustments (the discussion could also be generalized in case the above assumption is dropped):

Enhanced grouping adjustment #1: If the difference (DDT - DPT) (as estimated from (1) or (2) of Section 3.2) for a particular customer is substantially less than DT (say, $\leq$ 0.25 DT), and, if that customer's DPT and DDT fall in adjacent time groups (k and k+1 respectively), then shift

both that customer's DPT and DDT so that (a) their difference stays the same and (b) they both fall within the same time group. Specifically, shift both of them so that DDT falls at the very end of time group k if the customer has specified a DDT, or so that DPT falls at the very beginning of time group k+1 if he has specified a DPT.

The implication of such an adjustment is that the customer in question would now be treated as a category 2 customer, hence picked up and delivered within the same time group. A customer who has specified a DPT would the be picked up no earlier than his DPT, and would experience a ride time no longer than DT. These figures are in constrast to a worst-case pick-up of DT minutes earlier than DPT and a worst-case ride time of 2DT if no adjustment is made. (Actually, since that customer's origin and destination are likely to be very close, it is also likely that, with enhanced grouping, the clustering and routing steps would yield a ride time much shorter than DT). Of course, such a customer would possibly be picked up late (in fact, as much as DT minutes late), but, for DPT-specified customers, a late pick-up may be preferable to an early pick-up - and, if the opposite is the case for the customer in question, then the algorithm's operator may choose not to excercise the enhanced grouping adjustment for that customer.

An analogous argument can be made for customers who have specified a DDT.

It can be seen that this adjustment alone "fixes" worst-case scenarios associated with customers who have relatively short direct trip

times, such as the one of the earlier example.

Enhanced grouping adjustment #2: By contrast to the previous adjustment which involves strictly manipulations of data, adjustment #2 involves a structural change in the algorithm. Such an adjustment is recommended in cases when the length of the minimum time interval DT_{min}, calculated according to Section 3.2 as $0.5 \cdot CF \cdot \max_j DRT_j$, is too long to warrent an acceptable service quality. This may happen if some of the customers have relatively long direct trip times, and is a direct consequence of the assumption that all category 1 customers in any time group k are category 3 customers in time group k+1.

This adjustment resets DT to a suitable user-specified value lower than DT_{min}. If this happens, some category 1 customers in some time group k will have their DDTs beyond time group k+1. However, under this adjustment the algorithm continues to consider such customers as category 3 customers for time group k+1. As a result, clustering for those customers is executed as usual, and since such customers have their destinations at rather remote locations, very few (or no) other customers will be clustered with them in group k+1. This is so because of the feature of the clustering step that attempts to equalize vehicle workload. Such a behavior of the algorithm is considered desirable, for it would be inefficient to cluster customers whose DDT falls within time group k+1 with customers who can only be delivered beyond that time group.

The above discussion indicates that implementing the above

adjustments should improve this algorithm's worst-case performance. Whether or not significant improvements in the algorithm's average performance are to be expected too, depends on the relative frequency of occurrence of customers prone to such worst-case service within the entire population of customers.

To further improve the quality of solution, a tour-improvement procedure can be introduced as the final step to seek possible improvements on the vehicle routes and schedules developed by GCR. Recall that in the routing step of GCR a k-interchange procedure was used to minimize the length of the vehicle route within a cluster of customers. This single-vehicle procedure can be extended to solve the multi-vehicle problem by considering additionally the interchange of customers between different vehicles. Christofides and Eilon [3] and Cook and Russell [6] have applied a similar concept to vehicle routing problems by extending the Lin-Kernighan algorithm for TSP to the multiple-vehicle routing problem. However, dial-a-ride problems may present some complications in implementing the procedure. At least four links in a vehicle route (two for a customer's pick-up point and two for his delivery point) must be replaced by another set of four links during one interchange as compared with two links in 2-opt or three links in 3-opt for the travelling salesman problem. The computation cost involved can be high. Besides, scheduling pick-up and delivery times for the new vehicle routes adds another dimension of complications to the procedure.

Many possibilities exist for the choice of an objective function for the tour-improvement procedure. Three obvious choices might be : (i)

To minimize vehicle route length while maintaining "acceptable" service quality. (ii) To improve service quality, e.g. minimizing the maximum deviation, minimizing the total deviation, etc. (iii) A combination of (i) and (ii). If vehicle productivity is the primary concern, objective (i) should be used. If a solution of better service quality is desired, objective (ii) is recommended. Objective (iii) takes into account both concerns and tries to seek a balance between objective (i) and (ii).

Besides implementing the tour-improvement procedure as a follow-up step after GCR, we can also introduce the procedure repetitively after the routing step in each time group. The extended multi-vehicle procedure would only investigate the last portion of the existing vehicle routes which was formed in the current time group. It is conjectured that through the interchange procedure, better-quality vehicle routes can be found early in the process of GCR and might have a riple effect on the quality of clusters and vehicle routes subsquently formed. Moreover, less computational effort would be involved when the number of customers is relatively small.

It is believed that the GCR algorithm can be best used in two ways: First, in <u>assisting</u> a dispatcher create a <u>preliminary</u> set of vehicle routes and schedules; and, second, as a <u>planning tool</u> for making some "strategic" decisions. The ideal environment for such use would be large-scale, multi-vehicle systems in which customers do not have "hard" time constraints but would instead find it acceptable if they were picked up (or delivered) at times reasonably close to their most preferred ones, - especially, if these pick-up and/or delivery times were quoted to them

well in advance. From discussions with dial-a-ride system operators, it would appear that some dial-a-ride system users in the United States do fall in this category.

Some forms of man-machine interaction can be beneficial to the first type of use ("assisting with preliminary design of routes and schedules") that we visualize for this algorithm. Graphical displays of customer clusters formed after the clustering step and vehicle routes that are designed for customers within the same clusters can help the operator to refine the results whenever he thinks is necessary. Such man-machine interaction appears quite feasible considering that: (i) the algorithm is very efficient computationally and therefore its repeated use would not be particularly expensive and some of the procedures in the algorithm can be carried out in "real-time"; (ii) we are dealing with an advance-request system in which the operator usually has the benefit of a considerable margin of time during which to prepare and finalize routes and schedules; and (iii) the computer program's outputs have been designed to include detailed information on each customer's processing and level of service and on each vehicle's routes and time schedules. Such interactive routing approaches have been suggested by Krolak et al. [16] and Cullen et al.[7].

In using the algorithm as a planning tool, one could address questions like (i) finding a desirable number of vehicles in service for some given expected level of demand (possibly time-varying) over a day or (ii) quantifying the tradeoffs between level of service to customers and costs to the operator (as reflected through number of vehicles and drivers used, length of routes, etc.) again for a given level of demand.

6.2 On ADARTW

ADARTW is an algorithm designed for dial-a-ride problems with "hard" service quality constraints. It addresses many of the complicated details of a realistic problem while offering some flexibility and several options to its user. The pricipal challenge in this instance was obviously the service-quality constraints, particularly the time windows for pick-ups and deliveries. ADARTW seems sufficiently efficient from the computational point of view to deal with multi-vehicle time-constrained problems of size comparable to the largest ones encountered in practice.

The development of ADARTW has made other contributions to solving the advance-request dial-a-ride problem. First of all, ADARTW can consider the simultaneous presence of DPT-specified and DDT-specified customers in the problem. Second, customers can have quadratic disutility function which is more realistic than linear one, especially when time window size is large. No other algorithm has made similar attempts so far. For example, in the algorithm developed by Sexton and Bodin [32,33], only one type of customers (either DPT-specified or DDT-specified) is allowed in the system which is a rather restrictive assumption. Moreover, Sexton and Bodin's algorithm does not guarantee service quality, it only uses a linear disutility function and tries to minimize the total disutility of the schedule. The work of Roy et al. [30] allows both types of customers but also uses a linear disutility function[1]. The NEIGUT/NBS algorithm

[1]The factor of customer disutility is not considered in the process of building up vehicle routes. During the so called "post-optimization step", the disutility becomes a cost term to be minimized.

[15] assumes no disutility function in this regard. Actually. the choice of the form of disutility function in ADARTW is not even restricted to linear or quadratic form. It can take any form as long as the resulting function in (4-35) is convex. The convexity of (4-35) is critical in terms of finding the optimal schedule efficiently.

Due to the heuristic nature of ADARTW, it can handle many complications that might be present in the real-world operations. For example, ADARTW can handle unsymmetric travel times, take into account nonzero dwell times, different vehicle types, etc. Moreover, a customer can specify his earliest pick-up time (a latest delivery time) in addition to his desired delivery (pick-up) time. In other words, customers can define their desired service quality as long as it is feasible to the system. The only requirement, as far as the system operator is concerned, is that a customer must specify <u>either</u> it is the desired pick-up <u>or</u> the desired delivery time on which his disutility function should be based. A customer can of course choose not to specify any disutility function in which case any service time within the time window would be equally acceptable to him.

Advanced data structures, e.g. list processing, were introduced in ADARTW. A vehicle work-schedule is represented by a linear list of stops linked by pointers. The linear list can be expanded by linking additional stops with pointers without considering the possibility of exceeding the

dimension of the list. Such characteristics are crucial in tour-building heuristics because of the unpredictability of the size of the tour. Furthermore, an insertion of a customer into a vehicle work-schedule can be done by changing neighboring pointers where the insertion is taken place. In this way, ADARTW avoids reshuffling of the sequence as required if an array is used to represent the vehicle schedule.

As in the case of most heuristic algorithms that solve large-scale and complex routing and scheduling problems, it is difficult to state in quantitative terms how good the solutions provided by ADARTW are. There are no "optimal" solution to compare with since first, no exact algorithms to solve problems of similar size exist, and second, we lack a precise closed-form objective function (the objective of dial-a-ride systems can be viewed as that of "satisfying" both operator and customers). About all that can be said is that, as dial-a-ride systems go, the solutions found by ADARTW for simulated or real database are at least as good or superior to those encountered in practice in all respects (vehicle productivity, vehicle utilization, ride circuity, deviation from desired pick-up/delivery times). In addition, of course, ADARTW provides strict guarantees on the minimum level of service quality to be provided. Through appropriate adjustmant of the constants C_1 through C_8, users of the algorithm can also give more or less emphasis to customer- or operator-oriented objectives, as desired.

One variation of the algorithm - by considering multiple candidates for the next insertion and using the periodic-refill strategy - shows promising improvements to the solutions obtained through the basic version

of ADARTW. The increase in computation time for this particular variation of ADARTW is reasonable and it can be weighted against the advantages gained by implementing such a variation.

Similar to the GCR algorithm, an interchange procedure can be introduced to improve the quality of solution obtained by ADARTW. In [6], Cook and Russel reported computational experience with an interchange procedure, MTOUR, which was applied to the vehicle routing problem with time window constraints on some of the demand points. However, it seems that such an interchange procedure can be computationally expensive for medium to large problems.

So far we have discussed how ADARTW can be used to help the day-to-day operations of a dial-a-ride system which quarantees a given level of service quality. It is conceivable that ADARTW can also be used at the planning stage of dial-a-ride systems to evaluate alternative levels of service quality to be guaranteed and their cost implications. Figure 6.1 depicts three-way relationships between service, cost, and demand involved in the design of a system. Given a particular level of service quality and a forecasted demand scenario, ADARTW can help to determine the amount of resources (fleet size, labor, etc.) required to operate the system. To demonstrate this point, we provide an example which employs the nine data sets, L10, L20,....., H30, as the possible demand/service scenarios for the system under consideration. Partial scheduling results of ADARTW using one set of parameter values for demonstration purposes are shown in Table 6.1. The second number shown in parentheses indicates the number of vehicles required to satisfy all demands. The other number in each box

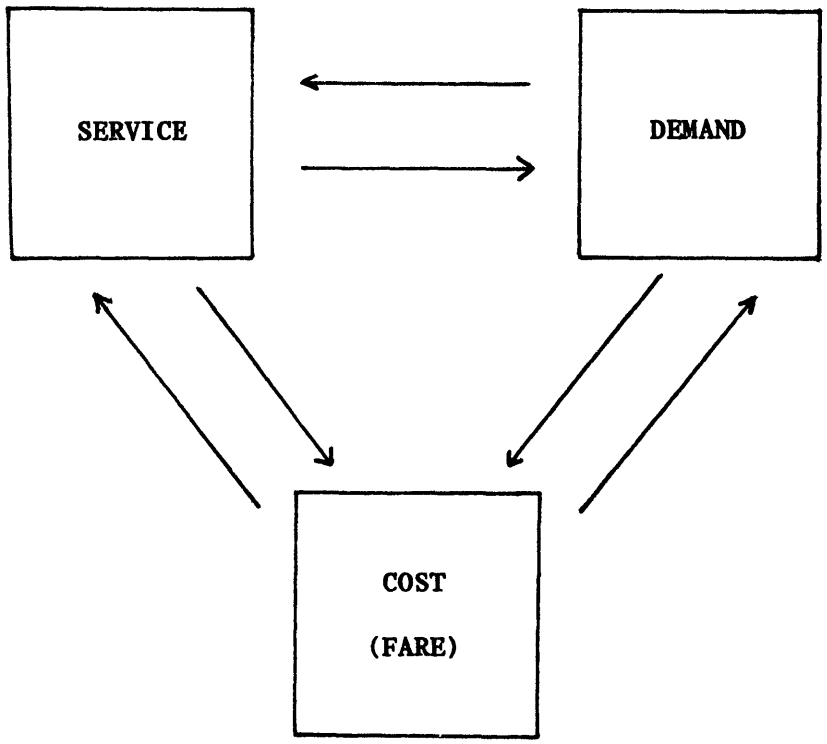


Figure 6.1

Design considerations of a dial-a-ride system

Table 6.1 Vehicle productivity and fleet size required for different system design scenarios

(C_1 = 0.1, C_3 = 0.5, C_7 = 1, C_8 = 0.8, Others = 0)

Time Window Size \ Demand Rate	10 customers/hr	50 customers/hr	100 customers/hr
10 minutes	2.28* (4,7)**	3.39 (12,17)	4.18 (25,26)
20 minutes	2.50 (4,5)	4.23 (10,12)	4.97 (20,23)
30 minutes	2.89 (3,5)	4.32 (10,12)	5.38 (20,20)

* Vehicle productivity in number of customers per vehicle hour

** (a,b): a - no. of vehicles initially specified in ADARTW.
b - no. of vehicles required by ADARTW to reach a feasible solution.

represents the vehicle productivity. As can be seen in the Table, by relaxing time window of 10 minutes to 20 minutes, the fleet size can be reduced by a fair amount. No substantial gain can be realized by further relaxing the time windows to 30 minutes. Table 6.1 also illustrates the relationship between vehicle productivity and demand levels. For example, vehicle productivity increases only about 50% when demand increases five times (10/hr to 50/hr) and about 90% when the increase in demand is about 10 times (10/hr to 100/hr).

We can apply the type of analysis described above to the design of any dial-a-ride system. However, we note that the results shown in Table 6.1 will in general be data-dependent. For any particular system in question, we suggest that the following approach be taken: First, define the service area and a range of service quality standards that may be acceptable. For each level of service quality planned, obtain forecasts of demands based on several low-to-high scenarios. The forecasts should include the likely geographical distribution of demands within the service area and the distribution of demands over time. Then, use ADARTW to construct a table of anticipated results similar to that shown in Table 6.1. Based on these results, the system planners can evaluate the cost implications of each planned service quality under different demand scenarios and can design the system accordingly.

A question that might arise in using ADARTW concerns the sensitivity of results to the setting of parameter values in the objective function and to the choice of service quality standards. By lacing relatively larger weights on the customer disutility components in the

objective function one can obtain schedules of better service quality for most of the customers. But with tighter service quality constraints, one can provide guaranteed services to all customers. It seems that in a manner similar to that described in the previous paragraph one can use a simulation approach to investigate the cost implications of each set of alternative parameter settings. One systematic approach would be to define a possible range of service quality standards that are acceptable and for each service quality standard perform a set of simulation runs of ADARTW by specifying different parameter values for the customer disutility components. With a table of results similar to that of Table 6.1, an appropriate strategy can be selected and its cost can be estimated.

6.3 Comparison Between GCR and ADARTW

The major difference between GCR and ADARTW lies in the "hard" adherence by the latter to time constraints. It might be conjectured that by relaxing the quality of service provided, the GCR algorithm would yield solutions with higher vehicle productivity than ADARTW since the time constraints in ADARTW are more restrictive. However, in comparing the results of both algorithms when applied to the German data base, it became clear that ADARTW gave superior solutions on every account, e.g. vehicle productivity, average deviation, average ride time ratio, etc. However, it is not obvious that ADARTW will outperform GCR in every application. To further clarify this issue, we provide an example to highlight the characteristics of the two approaches.

Figure 6.2 depicts a case of 25 customers (extracted from the top

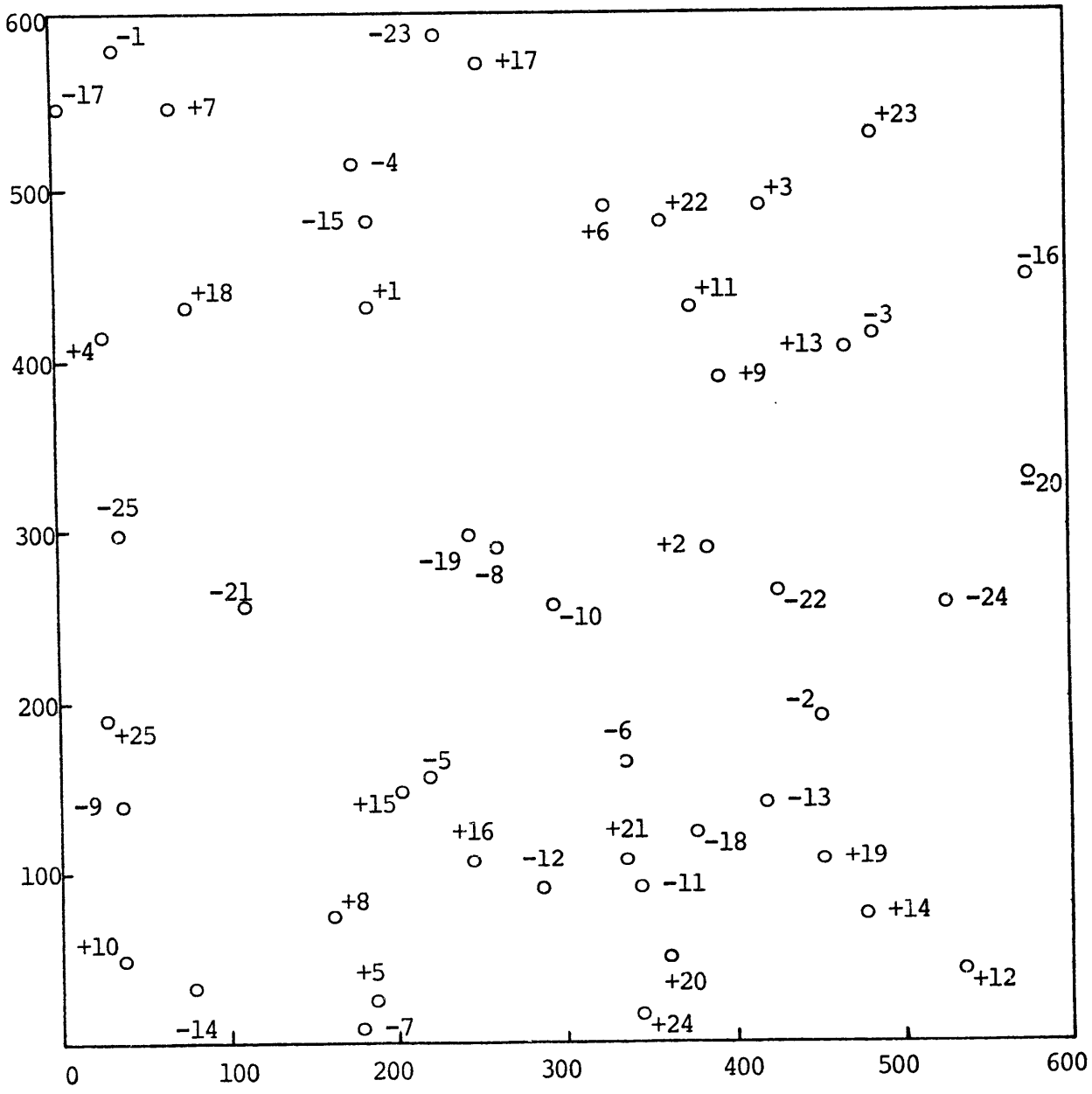


Figure 6.2
Pick-up and delivery locations of the 25 customers in the example

of the subscription list of data set L30) distributed in an area of 6 by 6 square miles. Their corresponding time constraints, pick-up and delivery locations are listed in Table 6.2. The vehicle schedules produced by ADARTW to serve these 25 customers are shown in Table 6.3. The complete schedule involves four vehicles and a total vehicle time of 389 minutes (no vehicle slack time was incurred). Figure 6.3(a) - (d) depicts pictorially the four vehicle routes as scheduled by ADARTW. Another run using the GCR algorithm with the same set of customers was performed and the results are recorded in Table 6.4. This run used a 60-minute time interval to group customers and used 1 as the conversion factor (CF). The resulting GCR schedule used the same number of vehicles as ADARTW, but required only 360 minutes of vehicle time (including 47 minutes of slack time). The four vehicle routes are plotted in Figure 6.4(a)-(d). By examining the GCR schedule in Table 6.4, we found that there are three customer (#2 on vehicle 1, #17 on vehicle 3, #11 on vehicle 4) who will be delivered later than their desired delivery time and there are another three customers (#13 on vehicle 1, #9 on vehicle 2, #14 on vehicle 3) who will be picked up earlier than their desired pick-up time. Only two DDT-specified customers (#21 on vehicle 2, #22 on vehicle 3) will be delivered more than 30 minutes earlier than their desired times and no DPT-specified customer will be picked up more than 30 minutes later than their desired times. Statistics on the performance of the two schedules are compiled in Table 6.5. Note that the average ride time ratio produced by the GCR schedule is 1.33 which is less than the 1.49 of ADARTW schedule and by further examination of the GCR schedule in Table 6.4 we find no individual actual ride time exceeds the maximum ride time constraints of ADARTW. In Table 6.5, the vehicle productivity of the GCR schedule is seen to be 7%

Table 6.2 Subcription list of 25 customers

CUSTOMER	TYPE	EPT	DPT	LPT	PX*	PY*	EDT	DDT	LDT	DX*	DY*	DRT	MRT
1	DDT	649	733	733	180	437	712	742	742	30	582	9	23
2	DDT	652	732	732	380	288	707	737	737	442	198	5	15
3	DDT	700	740	740	412	490	715	745	745	488	404	5	15
4	DPT	702	702	732	27	413	710	710	753	169	516	8	21
5	DDT	711	751	751	197	29	726	756	756	205	153	5	15
6	DPT	714	714	744	326	490	727	727	815	324	166	13	31
7	DDT	714	811	811	62	543	803	833	833	182	14	22	49
8	DDT	715	800	800	163	77	740	810	810	254	293	10	25
9	DPT	721	721	751	394	389	739	739	832	29	144	18	41
10	DPT	723	723	753	31	66	737	737	826	283	278	14	33
11	DDT	730	819	819	377	420	803	833	833	332	97	14	33
12	DDT	744	829	829	528	50	809	839	839	285	98	10	25
13	DPT	748	748	818	465	406	759	759	845	408	144	11	27
14	DPT	751	751	821	472	73	807	807	858	93	34	16	37
15	DDT	752	840	840	212	161	823	853	853	183	474	13	31
16	DDT	756	851	851	243	106	841	911	911	589	436	20	45
17	DDT	800	845	845	253	569	825	855	855	4	547	10	25
18	DDT	810	902	902	89	427	849	919	919	375	128	17	39
19	DDT	811	858	858	448	111	840	910	910	243	298	12	29
20	DDT	812	902	902	362	57	847	917	917	577	337	15	35
21	DDT	816	902	902	333	104	843	913	913	102	251	11	27
22	DDT	821	906	906	352	483	846	916	916	428	259	10	25
23	DPT	828	828	858	491	537	839	839	925	238	584	11	27
24	DDT	828	915	915	348	33	857	927	927	520	259	12	29
25	DPT	840	840	910	27	194	845	845	925	33	299	5	15

*(PX,PY) and (DX,DY) denote respectively the pick-up and delivery location of a customer using coordinates shown in Figure 6.2.

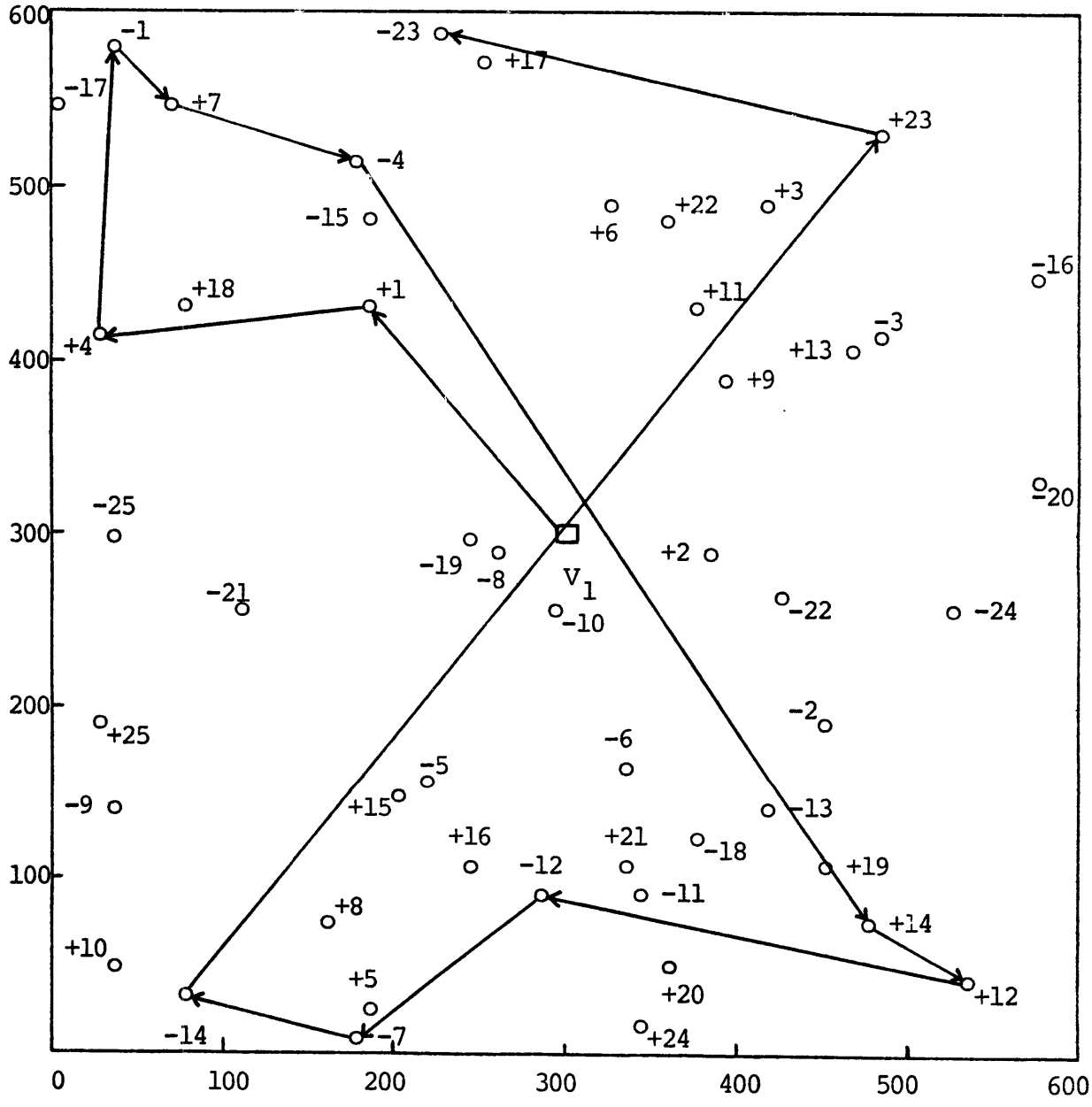


Figure 6.3(a)
Vehicle #1's route designed by ADARTW

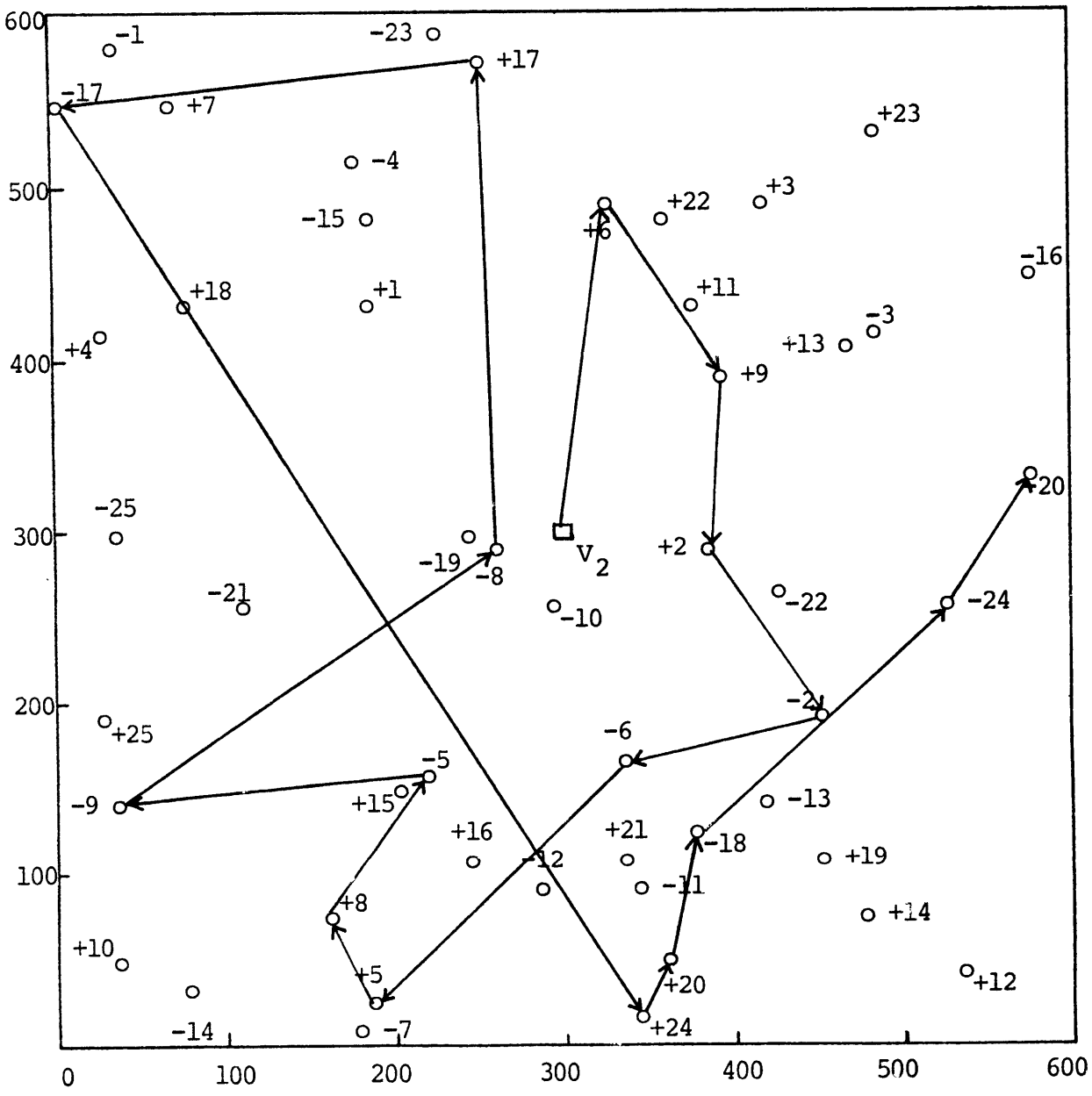


Figure 6.3(b)
Vehicle #2's route designed by ADARTW

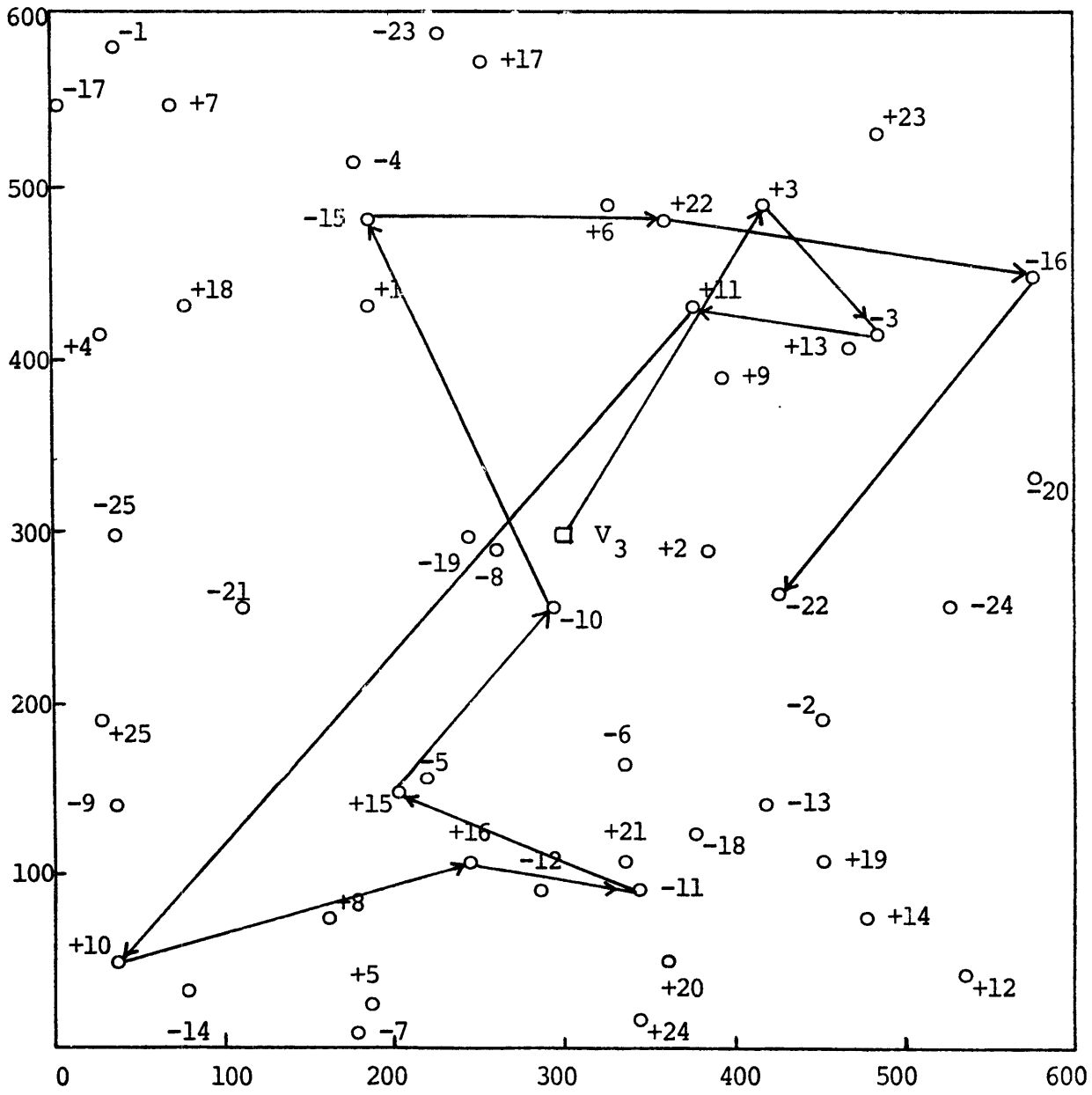


Figure 6.3(c)
Vehicle #3's route designed by ADARTW

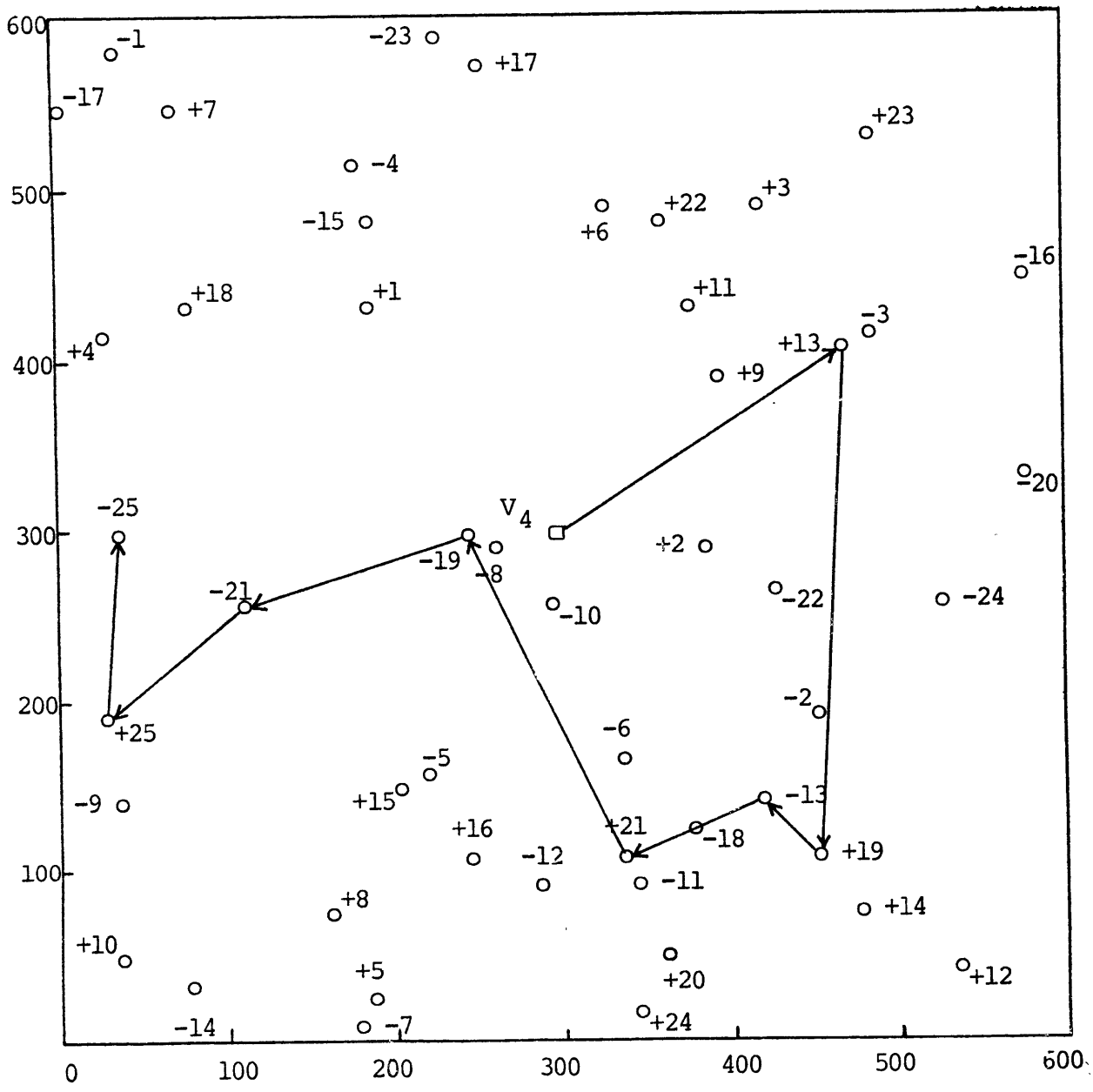


Figure 6.3(d)

Vehicle #4's route designed by ADARTW

Table 6.3 Vehicle schedules generated by ADARTW
(C_7 = 1, C_8 = 0.8, Others = 0)

VEHICLE #1

STOPS	EARLIEST	SCHEDULE	LATEST	TIME DEVIATION	RIDE TIME RATIO
0	0:00	7:17	24:00		
1	6:49	7:25	7:33		
4	7:02	7:32	7:32	30	1.87
-1	7:12	7:39	7:42	-3	1.55
7	7:14	7:42	8:11		
-4	7:10	7:47	7:53		
14	7:51	8:09	8:21	18	1.43
12	7:44	8:12	8:29		
-12	8:09	8:22	8:39	-17	1.00
-7	8:03	8:28	8:33	-5	2.09
-14	8:07	8:32	8:58		
23	8:28	8:58	8:58	30	1.00
-23	8:39	9:09	9:25		

VEHICLE #2

STOPS	EARLIEST	SCHEDULE	LATEST	TIME DEVIATION	RIDE TIME RATIO
0	0:00	7:08	24:00		
6	7:14	7:16	7:44	2	1.53
9	7:21	7:21	7:51	0	2.11
2	6:52	7:26	7:32		
-2	7:07	7:31	7:37	-6	1.00
-6	7:27	7:36	8:15		
5	7:11	7:44	7:51		
8	7:15	7:47	8:00		
-5	7:26	7:51	7:56	-5	1.39
-9	7:39	7:59	8:32		
-8	7:40	8:10	8:10	0	2.29
17	8:00	8:22	8:45		
-17	8:25	8:32	8:55	-23	1.00
18	8:10	8:38	9:02		
24	8:28	8:57	9:15		
20	8:12	8:59	9:02		
-18	8:49	9:02	9:19	-17	1.41
-24	8:57	9:10	9:27	-17	1.08
-20	8:47	9:14	9:17	-3	1.00

(Continued on the next page)

Table 6.3 (Continued) Vehicle schedules generated by ADARTW

VEHICLE #3

STOPS	EARLIEST	SCHEDULE	LATEST	TIME DEVIATION	RIDE TIME RATIO
0	0:00	7:13	24:00		
3	7:00	7:22	7:40		
-3	7:15	7:27	7:45	-18	1.00
11	7:30	7:32	8:19		
10	7:23	7:52	7:53	29	1.78
16	7:56	8:01	8:51		
-11	8:03	8:05	8:33	-28	2.35
15	7:52	8:11	8:40		
-10	7:37	8:17	8:26		
-15	8:23	8:26	8:53	-27	1.15
22	8:21	8:33	9:06		
-16	8:41	8:43	9:11	-28	2.09
-22	8:46	8:53	9:16	-23	2.00

VEHICLE #4

STOPS	EARLIEST	SCHEDULE	LATEST	TIME DEVIATION	RIDE TIME RATIO
0	0:00	8:10	24:00		
13	7:48	8:18	8:18	30	1.36
19	8:11	8:30	8:58		
-13	7:59	8:33	8:45		
21	8:16	8:37	9:02		
-19	8:40	8:46	9:10	-24	1.33
-21	8:43	8:52	9:13	-21	1.36
25	8:40	8:56	9:10	16	1.00
-25	8:45	9:01	9:25		

Table 6.4 Vehicle schedules generated by GCR algorithm

VEHICLE #1

STOPS	SCHEDULE	DESIRED	PICKUP-DEV.	DELIVERY-DEV.	RIDE TIME RATIO
0	7:07				
1	7:19				
-1	7:27	7:42		-15	1.00
6	7:39	7:14	25		1.62
3	7:42				
13	7:45	7:48	-3		1.09
-3	7:45	7:45		0	1.00
2	7:51				
-2	7:55	7:37		18	1.00
-13	7:57				
-6	8:00				
SLACK	8:18				
19	8:24				
12	8:28				
20	8:34				
24	8:35				
-12	8:39	8:39		0	1.10
16	8:40				
-19	8:47	9:10		-23	1.83
-24	8:58	9:27		-29	1.83
-20	9:01	9:17		-16	1.73
-16	9:04	9:11		-7	1.20

VEHICLE #2

STOPS	SCHEDULE	DESIRED	PICKUP-DEV.	DELIVERY-DEV.	RIDE TIME RATIO
0	7:11				
9	7:17	7:21	-4		2.11
4	7:31	7:02	29		1.13
7	7:36				
-4	7:40				
-9	7:55				
-7	8:03	7:33		-30	1.23
SLACK	8:18				
21	8:26				
15	8:31				
-21	8:37	9:13		-36	1.10
25	8:40	8:40	0		1.10
-25	8:44				
-15	8:53	8:53		0	1.62

(Continued on the next page)

Table 6.4 (Continued) Vehicle schedules generated by GCR algorithm

VEHICLE #3

STOPS	SCHEDULE	DESIRED	PICKUP-DEV.	DELIVERY-DEV.	RIDE TIME RATIO
0	7:18				
14	7:31	7:51	-20		1.06
5	7:42				
-14	7:46				
10	7:48	7:23	25		1.07
8	7:53				
-5	7:56	7:56		0	2.80
-10	8:01				
-8	8:02	8:10		-8	1.11
SLACK	8:16				
22	8:25				
-22	8:34	9:16		-42	1.00
23	8:45	8:28	17		1.10
17	8:54				
-23	8:55				
-17	9:04	8:55		9	1.10

VEHICLE #4

STOPS	SCHEDULE	DESIRED	PICKUP-DEV.	DELIVERY-DEV.	RIDE TIME RATIO
0	8:20				
18	8:31				
11	8:42				
-18	8:53	9:19		-26	1.29
-11	8:55	8:33		22	1.07

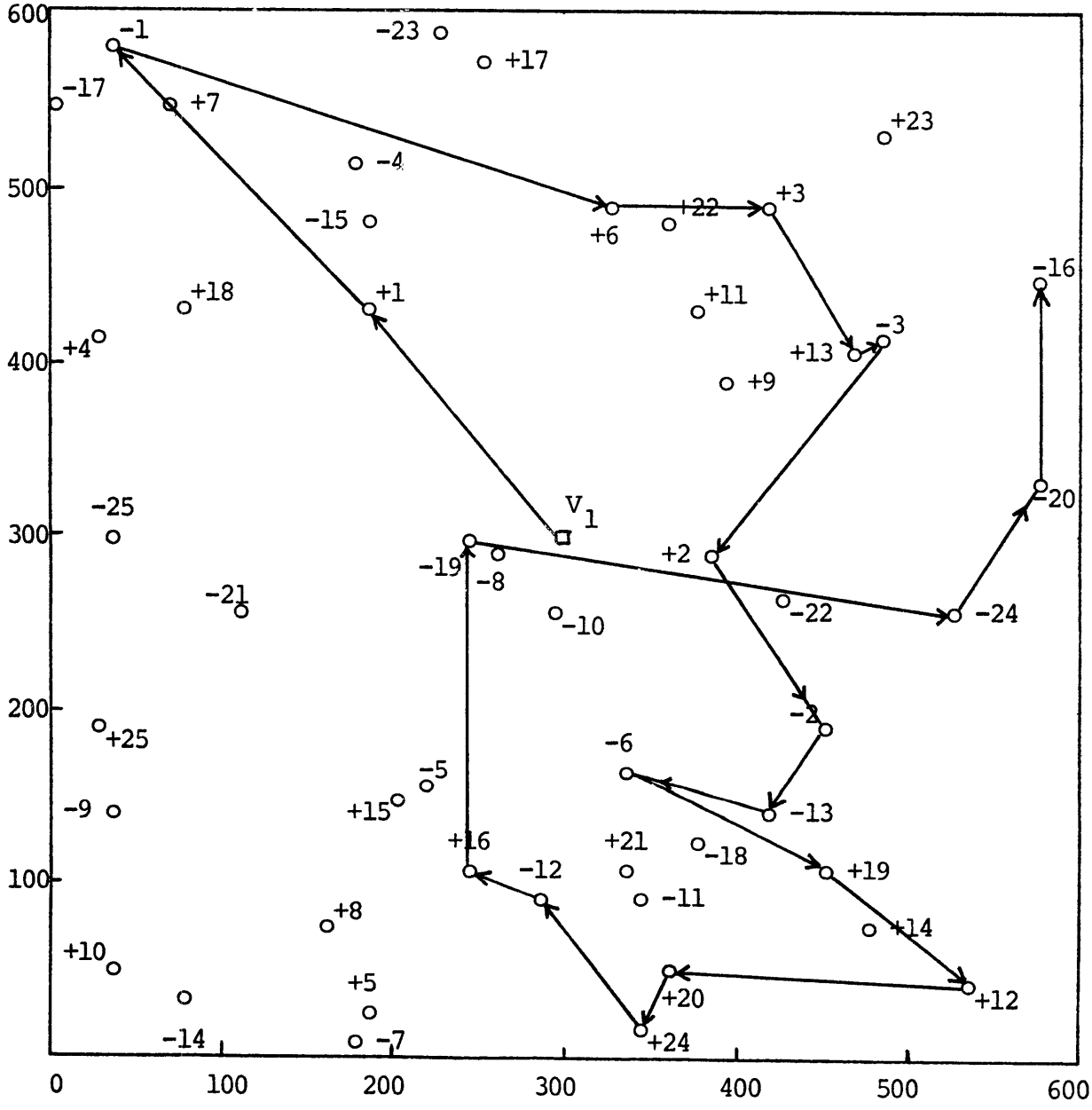


Figure 6.4(a)
Vehicle 1's route designed by GCR

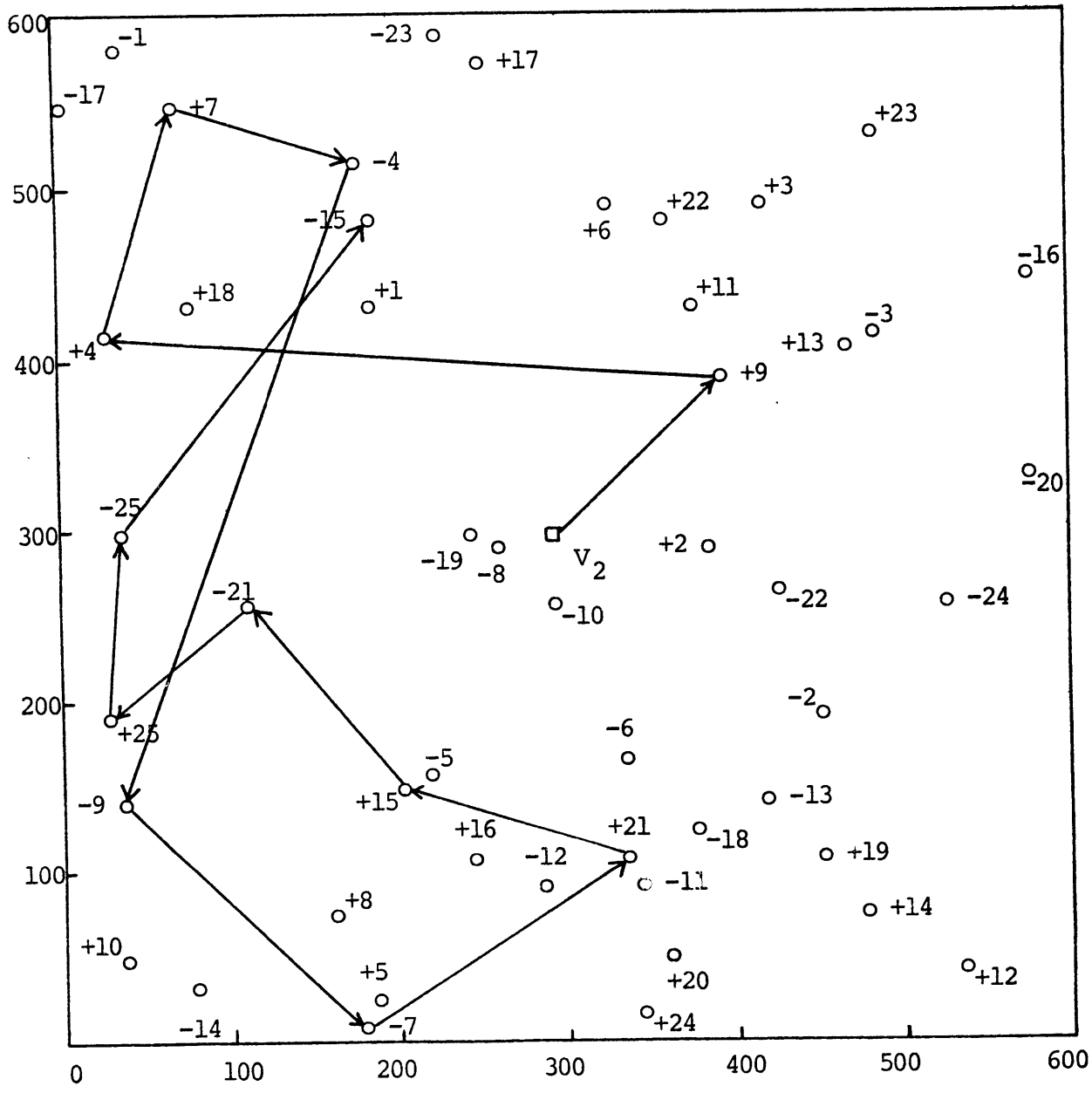


Figure 6.4(b)

Vehicle 2's route designed by GCR

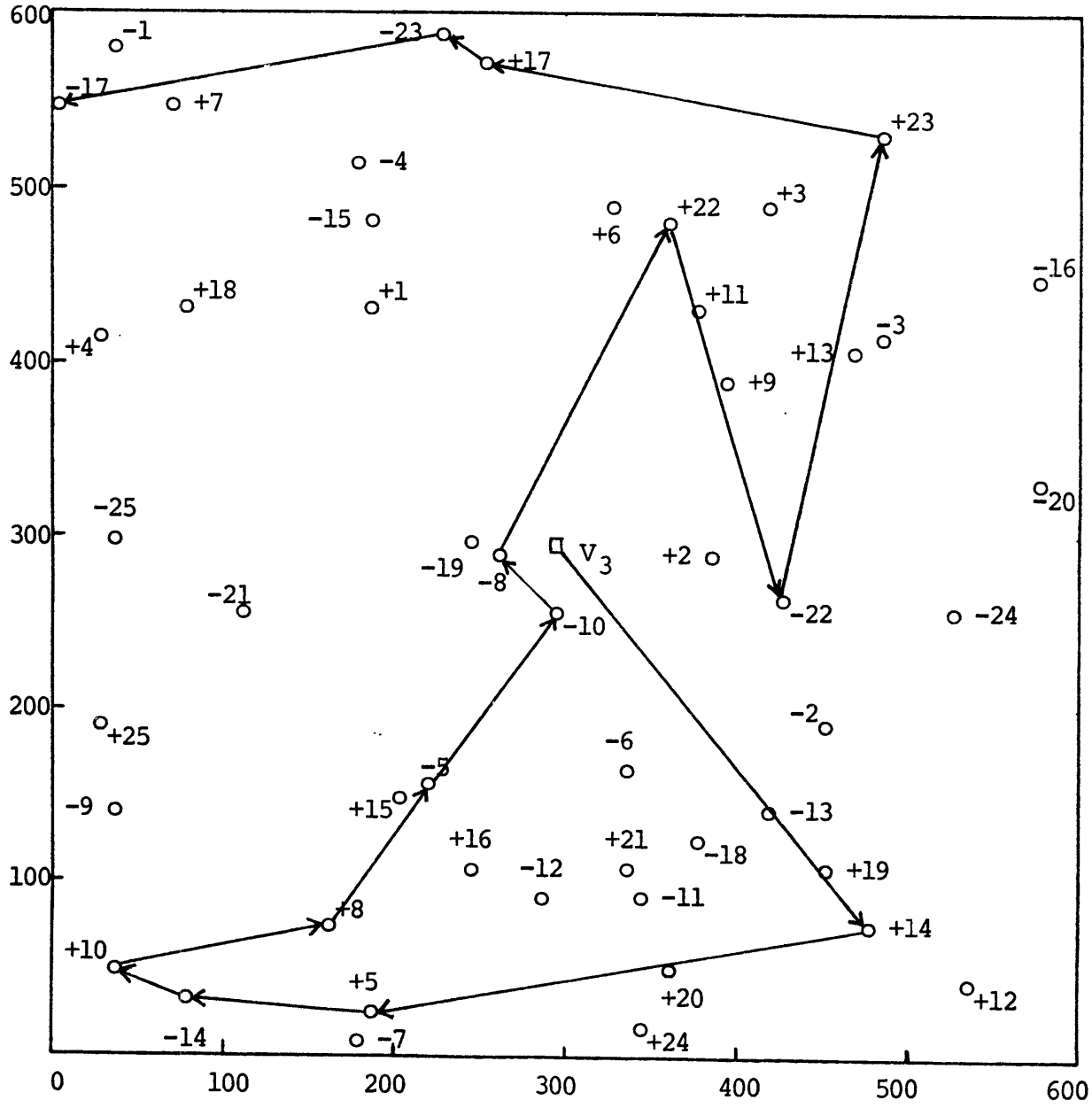


Figure 6.4(c)
Vehicle 3's route designed by GCR

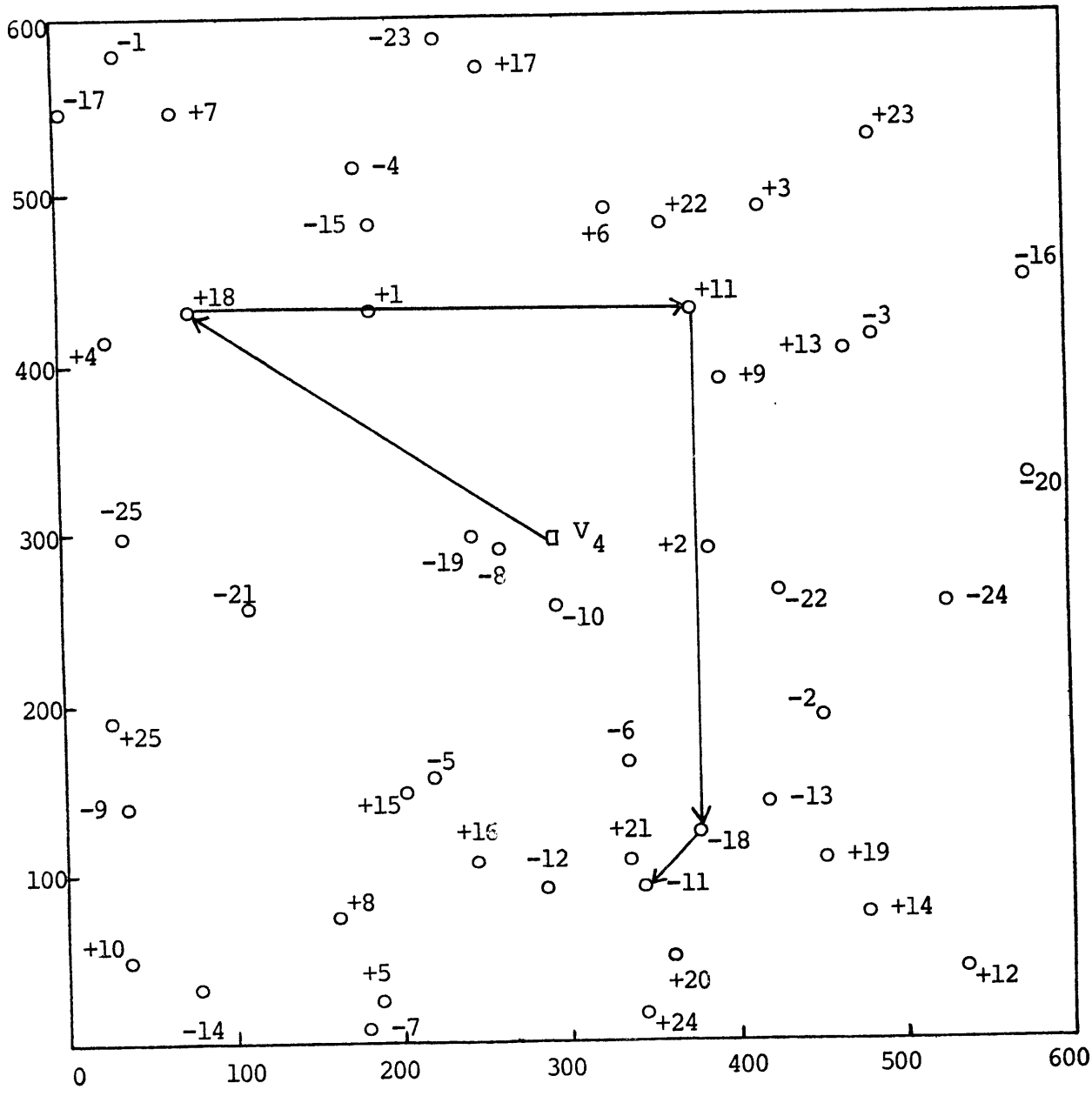


Figure 6.4(d)
Vehicle 4's route designed by GCR

Table 6.5 Performance statistics of GCR and ADARTW schedules for the example.

	ADARTW	GCR
No. of Vehicles Used	4	4
No. of Late Deliveries	0	3
Average Deviation Given Late Delivery (in minutes)	0	16.3
Average Deviation Given Early Delivery (in minutes)	15.6	16.6
No. of Early Pick-ups	0	3
Average Deviation Given Early Pick-up (in minutes)	0	9
Average Deviation Given Late Pick-up (in minutes)	19.4	19.2
Average Ride Time Ratio	1.49	1.33
Vehicle Productivity (in pass/vehicle hour)	3.86	4.17
CPU time on Vax 11/750 (in seconds)	1.4	1.12

higher than that of the ADARTW schedule. For this example, GCR has generated a more productive schedule than ADARTW, but at the expense of some customers receiving somewhat inferior service (which may, however, be quite acceptable in the environment for which GCR is designed). However, since the problem used in this example is rather small, these results might not be representative of general cases. We only use this example to illustrate concerns that might arise in comparing the two algorithms.

The particur example above shows that GCR is a viable tool for some dial-a-ride systems. Its strong point lies in the high vehicle productivity achieved. One can argue that the gain in vehicle productivity achieved by using the GCR algorithm may not be substantial (and possibly inferior to that achieved by ADARTW as in the case of the German data base) and it is probably better to use ADARTW in all cases to provide guaranteed service quality to all customers. It is our belief that this is a point to be genuinely concerned about in practice and one that should certainly be taken into consideration when designing a system.

6.4 Conclusions

This thesis has examined the advance-request version of the dial-a-ride problem. Two different scenarios on the treatment of time constraints have been postulated for the problem : The first scenario assumes that constraints on a customer's service time can be relaxed and the system operator will only try to develop vehicle schedules that are sufficiently close to the desired ones. The second scenario requires the strict enforcement of the service quality constraints which include upper limits

on the service time deviation and the customer on-board time. Heuristic algorithms are developed for solving the problem under each of the two scenarios. The first algorithm, GCR, developed for the first scenario consists of three steps: grouping, clustering, and routing. In the grouping step customers are separated into groups according to their desired service times. Within each group, customers are then divided into small clusters so that a vehicle can serve a cluster of customers with service times sufficiently close to the desired ones. Finally, a routing algorithm is used to develop vehicle routes that minimize travel time. The GCR algorithm was tested against simulated data and real-world data.

The second algorithm, ADARTW, assumes strict service quality constraints. It can be characterized as a tour-building heuristic which builds up vehicle tours through sequential insertions of customers into existing routes. An objective function which is a weighted sum of customer disutility and consumption of vehicle resources is used to guide the selection of the best insertion to be made. Fast screening tests were developed for finding all feasible insertions of a customer into a vehicle work-schedule and an optimal vehicle time schedule can be found for each new feasible schedule sequence through the minimization of a convex function. Variations of this basic approach were discussed and extensive computational experience with ADARTW was reported.

Although the two algorithms have been shown to be consistent in producing good solutions, it is not yet known what the worst-case performance might be for each of the two algorithms. Solomon [35] has constructed a pathological case for an insertion procedure applied to the

vehicle routing problem with time window constraints. He has shown that the error of the solution can grow as the number of customers increases. It is yet to be proved that this is also true for ADARTW in dealing with dial-a-ride problems. The existence of a customer disutility component as part of the objective function of ADARTW is likely to complicate the search for a pathological case for this algorithm.

A natural extension of the two algorithms would be to perform a tour-improvement procedure after the solution has been obtained by GCR or ADARTW. A candidate approach would be to modify the k-interchange procedure by Psaraftis [28] to solve the multi-vehicle case. However, according to the computational experience with a similar procedure for vehicle routing problem reported in Russell [31], such a procedure can be inefficient for medium to large size problems.

One of the possible areas for future research is to examine how these algorithms can be used to solve problems with a mix of immediate-request and advance-request customers. A straightforward strategy would be to use GCR or ADARTW to generate schedules for all advance-request customers on the subscription list and then have the system operator try in "real-time" to incorporate the immediate-request customers into schedules not yet executed at the time of each request. The addition of immediate-request customers to a pre-planned schedule may result in perceived extra waiting times for some customers on that schedule, i.e. those advance-request customers who have already been notified of their scheduled service time by the agency or those immediate-request customers who had already been assigned to the schedule before the new request. For

those systems which guarantee service quality through the specification of constraints such as those associated with ADARTW, the system operator will have to perform a feasibility test to ensure that no constraints are violated by the addition of a new immediate-request customer to a schedule. ADARTW can be easily modified to perform this kind of test. The major distinction between an immediate-request and an advance-request environment is that vehicle schedules, when being executed in real time cannot be advanced to an earlier time since part of the schedule has already been committed to or been executed already. The schedule can only be delayed to accomodate a new request. Actually, such a restriction (in the immediate-request environment) would simplify the feasibility test of ADARTW. Only one (i.e., ADOWN) of the four statistics defined in (4-15) to (4-18) would be needed to determine the feasibility of a new sequence. Another possible change might be to include customer waiting time in the objective function to guide the insertion of immediate-request customers. It would be interesting to compare the performance of such a strategy with other strategies, e.g. by treating all types of requests as immediate-requests [41].

It would seem that GCR is not as amenable to such modifications. Since immediate requests are received one by one, it would be inefficient to use the GCR algorithm to generate schedules in the immediate-request environment. By constrast, ADARTW has the advantage of processing one customer at a time, a characteristic which is ideally suited for the immediate-request system. Another consideration, in this respect might be the possibility of re-allocating already assigned customers among vehicles, once an immediate-request demand becomes known. At the present

time, ADARTW does not consider such a "reshuffling" of schedule sequences once they are set. Such a possibility for real-time scheduling also requires future research attention.

Another interesting future research topic is a more detailed comparison of the performances of GCR and ADARTW for actual databases other than the one used in this thesis. The possible enhancement of the performance of GCR and ADARTW through man-machine interactions should also be evaluated.

As a final note, recent developments in the area of combinatorial optimization [9,10] have shown that mathematical-programming based heuristics can solve vehicle routing and scheduling problems of reasonable size if some special structures in the formulation of the problem can be exploited. No mathematical programming formulation of the multi-vehicle advance-request dial-a-ride problem has been attempted so far. It is suggested that the initial effort in this regard should concentrate on the objective of minimizing total route length and disregard the existence of customer disutility since the nonlinearility of the disutility function might render the problem unsolvable.

APPENDIX I

APPENDIX I contains five flow charts which illustrate the procedures used by ADARTW for the feasibility test of all possible insertions. Figure I.1(a) - (d) depict procedures used for those insertions with the new pick-up and delivery points inserted into the same schedule block of a vehicle work-schdule. Figure I.2 shows the procedure used for insertions with the new pick-up and delivery points inserted into two different schedule blocks. Notations used in Figure I.1(a) - (d) are provided in Table I.1. Notations used in Figure I.2 can be found in the text.

Table I.1 Notation list (not defined in the text) for Figure I.1(a) - (d).

p,q,r,s	:	Indices denoting visit sequence in a vehicle schedule block having d stops. They are used to define where the insertion of a new pick-up (+i) and delivery (-i) is made. Four cases of insertions (also see Figure 4.5) can be defined as follows: Case 1) 1 ..... p(= d) +i -i Case 2) 1 ..... p +i -i q.....d Case 3) 1 ..... p +i q ..... d -i Case 4) 1 ..... p +i q ..... r -i s ....d
T_i (T_{-i})	:	The pick-up time (delivery time) of the new customer i.
ST_j	:	Scheduled visit time of stop j in the schedule block before customer i is inserted.
w_i	:	Amount of change in vehicle waiting time due to the insertion of customer i.
SHIFT	:	The amount of time by which a given time or a time schedule is being shifted.
ΔP_i (ΔD_i)	:	The extra vehicle time required to pick up (deliver) customer i.
PS (DS)	:	Shift in time schedule for all stops preceding stop +i (succeeding stop -i) after customer i is inserted.
MS	:	Shift in time schedule for all stops between stop +i and -i after customer i is inserted.
GT,ET,LT	:	Variables.

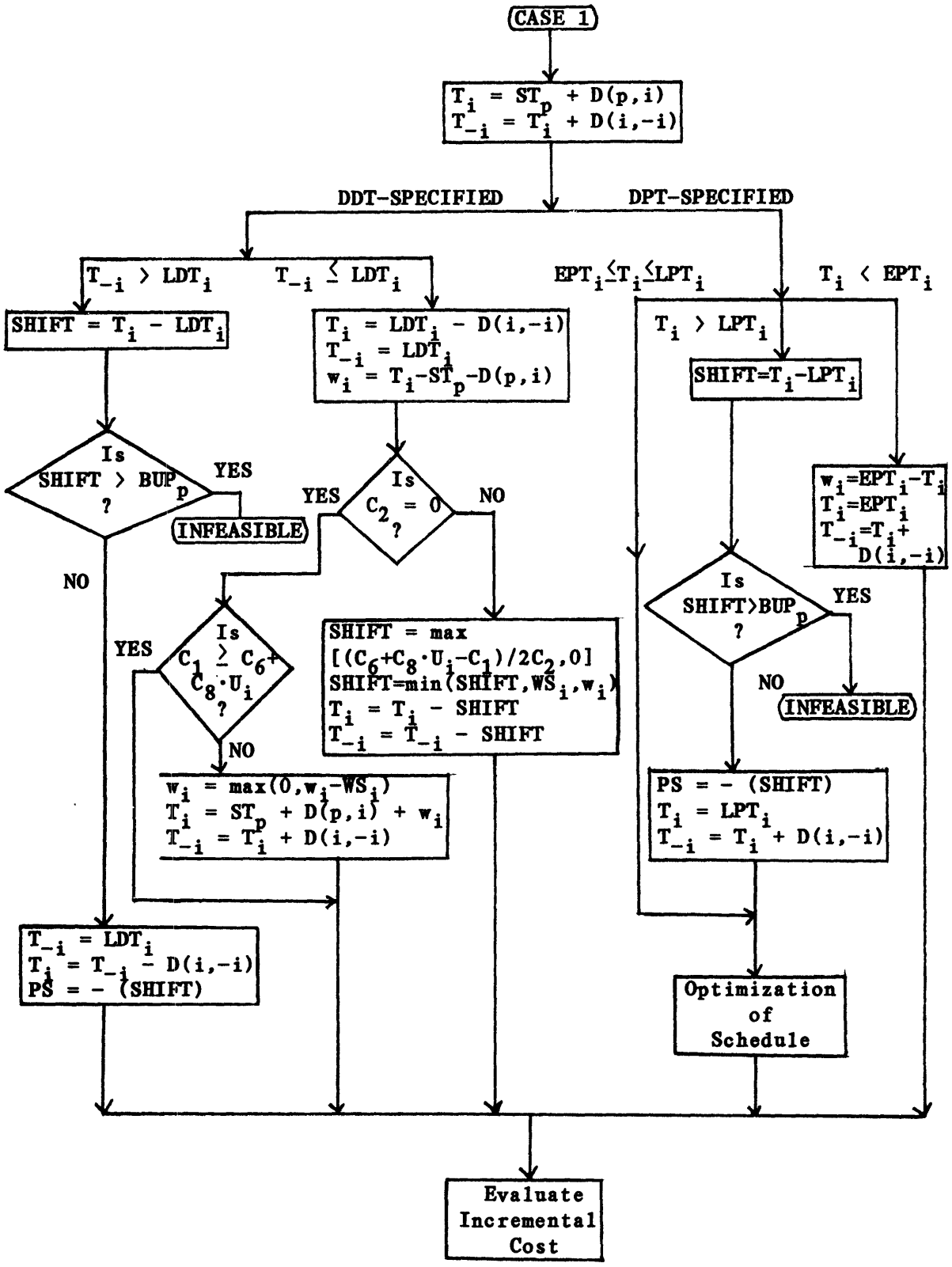


Figure I.1(a) Feasibility test for Case 1 insertions.

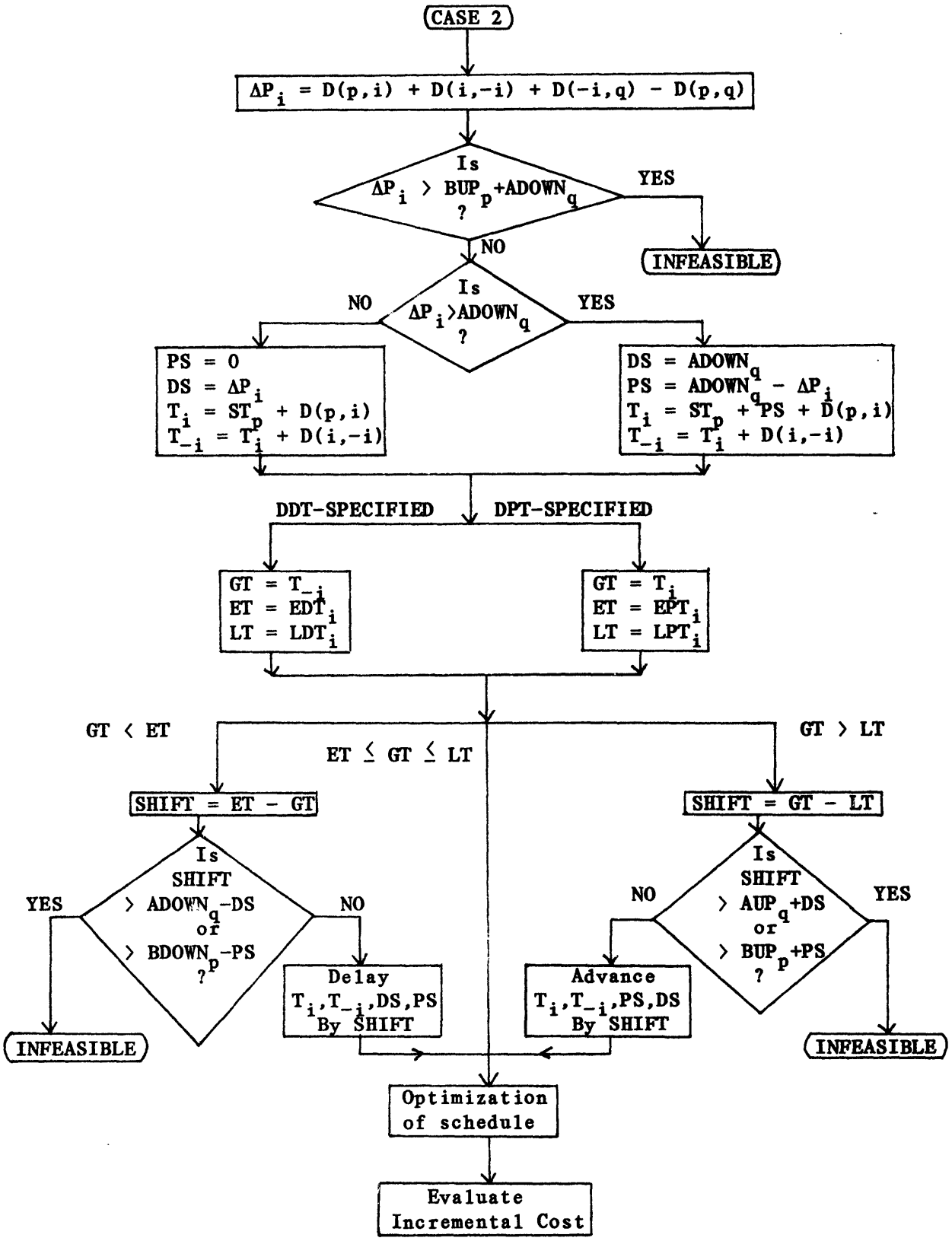


Figure I.1(b) Feasibility test for CASE 2 insertions.

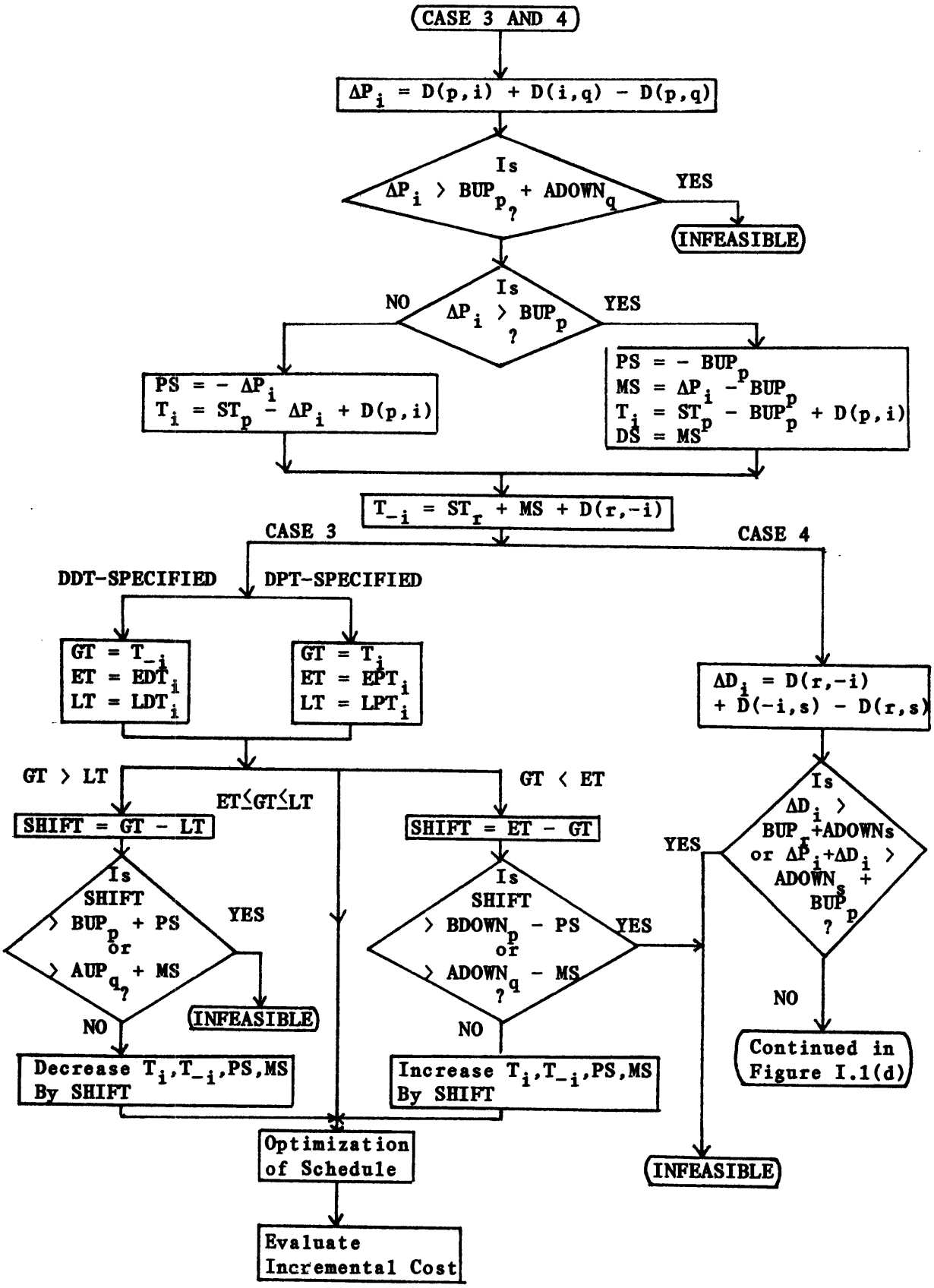


Figure I.1(c) Feasibility test for CASE 3 and 4 insertions

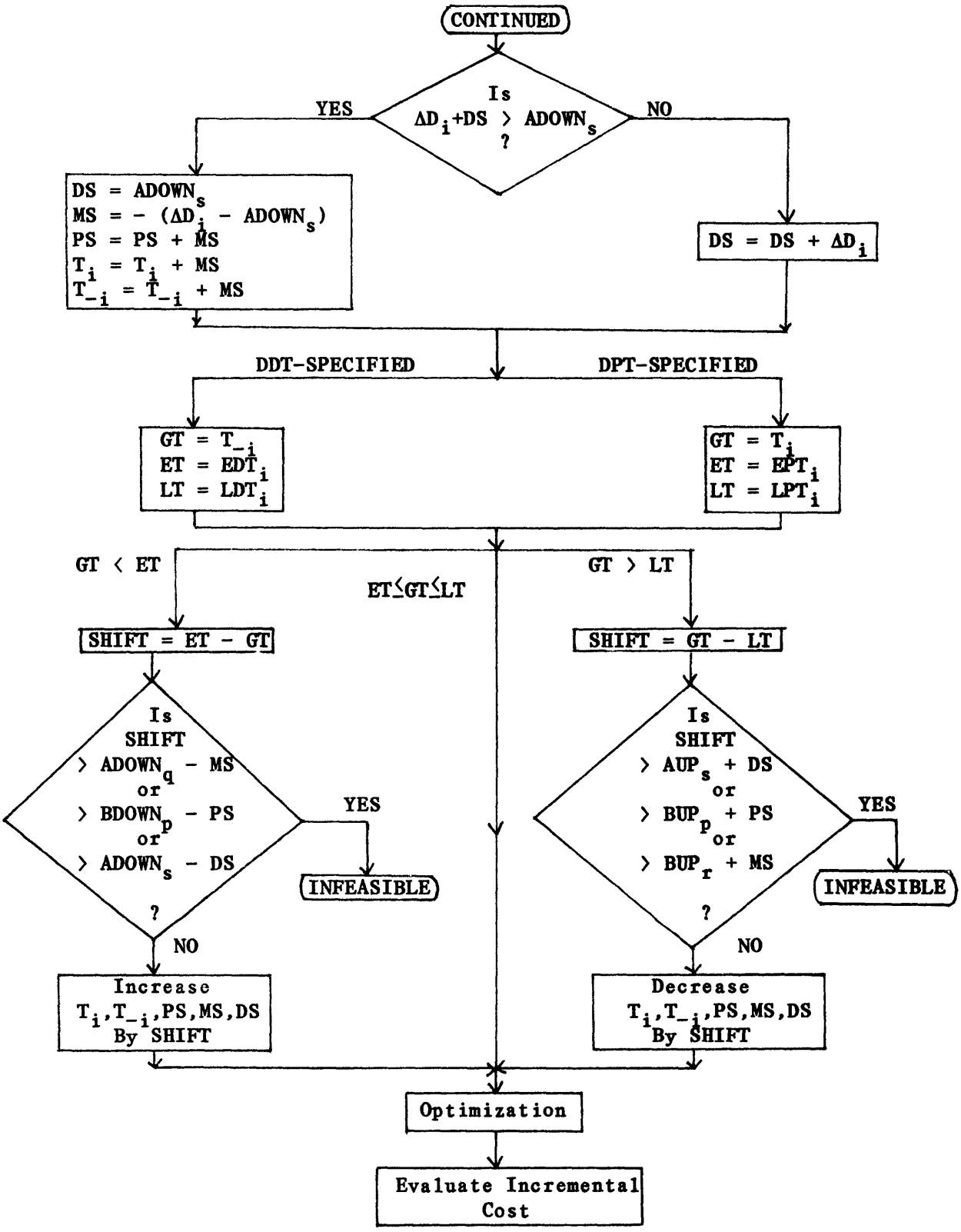


Figure I.1(d) Feasiblity test for CASE 4 insertions
(continued from Figure I.1(c))

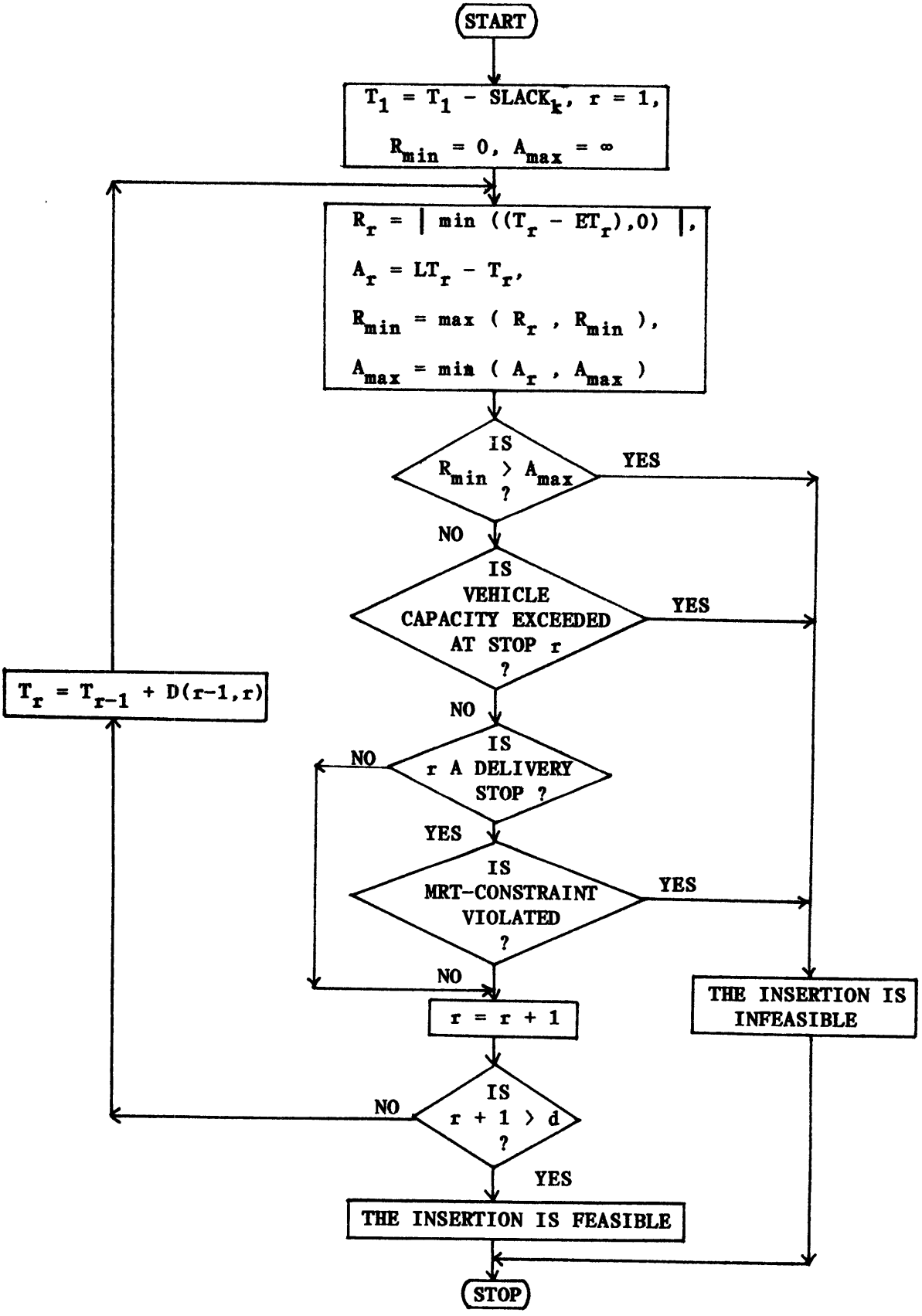


Figure I.2 Feasibility test for insertions into different schedule blocks.

REFERENCES

[1] Balinski, M.L., R.E. Quandt, "On an Integer Program for a Delivery Problem", Operations Research 12, 300-304 (1964).

[2] Bodin, L.D., B.L. Golden, A. Assad, M.O. Ball, "Routing and Scheduling of Vehicles and Crews: The State of The Art", Computers and Operations Research 10, 63-211, (1983).

[3] Christofides, N., S. Eilon, "An Algorithm for the Vehicle Dispatching Problem", Operational Research Quarterly 20, 309-318, (1969).

[4] Christofides, N., A. Mingozzi, P. Toth, "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Mathematical Programming 20, 255-282 (1981).

[5] Clarke, G., J. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", Operations Research 12(4), 568-581 (1964).

[6] Cook, T.M., R.A. Russell, "A Simulation and Statistical Analysis of Stochastic Vehicle Rouitng with Timimg Constraints", Decision Science 9, 673-687, (1978).

[7] Cullen, F.H., J.J. Jarvis, H.D. Ratliff, "Set Partitioning Based Heuristics for Interactive Routing", Networks 11 No.2, 125-144, (1981).

[8] Dantzig, G.B. and J.H. Ramser, "The Truck Dispatching Problem", Management Science 6, 80-91 (1959)

[9] Fisher, M.L., A.J. Greenfield, R. Jaikumar, P. Kedia, "Real-Time Scheduling of a Bulk Delivery Fleet: Practical application of Lagrangian Relaxation.", Decision Sciences Working Paper 82-10-11, University of Pennsylvania, (1982).

[10] Fisher, M.L., R. Jaikumar, "A Generalized Assignment Heuristic for Vehicle Routing", Networks 11 No.2, 109-124, (1981).

[11] Foster, B.A., D.M. Ryan, "An Integer Programming Approach to the Vehicle Scheduling Problem", Operations Research Quarterly 27, 367-384 (1976).

[12] Gaskell, T.J., "Basis for Vehicle Fleet Scheduling", Operations Research Quarterly 18, 281 (1967).

[13] Gillett, B., L. Miller, "A Heuristic Algorithm for the Vehicle Dispatch Problem", Operations Research, 22, 340-349, (1974).

[14] Golden, B., T. Magnanti, H. Nguyen, "Implementing Vehicle Routing Algorithms", Networks 7, 113-148, (1977).

[15] Hung, H.K., R.E. Chapman, W.G. Hall, E. Neigut, "A Heuristic Algorithm for Routing and Scheduling Dial-A-Ride Vehicles", Presentation at the ORSA/TIMS conference at San Diego, (1982)

[16] Krolak, P., W. Felts, J. Nelson, "A Man-Machine Approach toward Solving the Travelling Salesman Problem", Comm. ACM, 14, 327-334, (1971).

[17] Larson, R.C., A.R. Odoni, "Urban Operations Research", Prentice Hall (1981).

[18] Lenstra, J.K., A.H.G. Rinnooy Kan, "Complexity of Vehicle Routing and Scheduling Problems", Networks 11 No.2, 221-228, (1981).

[19] Lin, S., "Computer Solutions to the Travelling Salesman Problem", Bell Systems Technical Journal 44, 2245-2269 (1965).

[20] Lin, S., B.W. Kernighan, "An Effective Heuristic Algorithm for the Travelling Salesman Problem", Operations Research, 21, 498-516, (1973).

[21] Miliotis, P., "Combining Cutting-Plane and Branch and Bound Methods to Solve Integer Programming problems", London School of Economics working paper (1977).

[22] Miliotis, P., "Integer Programming Approaches to the Travelling Salesman Problem", Mathematical Programming 10, 367-378 (1976).

[23] Pierce, J.F., "Direct Search Algorithms for Truck Dispatching Problems" Transportation Research 3, 1-42 (1969).

[24] Psaraftis, H.N., "A Dynamic Programming Approach to the Dial-A-Ride Problem", Department of Civil Engineering, M.I.T. Report R78-34, (1978).

[25] Psaraftis, H.N., G.G. Tharakan, "A Dynamic Programming Approach to the Dial-A-Ride Problem: An Extension to the Multi-Vehicle Case", Department of Civil Engineering, M.I.T. Report R79-39, (1979).

[26] Psaraftis, H.N., "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-A-Ride Problem", Transportation Science, 14, 130-154 (1980).

[27] Psaraftis, H.N., "Analysis of an $O(N^2)$ Heuristic for the Single Vehicle Many-to-Many Euclidean Dial-A-Ride Problem", Transportation Research, 17B, 133-145, (1983).

[28] Psaraftis, H.N., "k-Interchange Procedures for Local Search in a Precedence-Constrained Routing Problem", European Journal of Operations Research 13, 391-402, (1983).

[29] Psaraftis, H.N., "An Exact Algorithm for the Single Vehicle Many-

to-Many Dial-A-Ride Problem with Time Windows", Transportation Science 17, 351-357, (1983).

[30] Roy, S., L. Chapleau, J. Ferland, G. Lapalme, J-M Rousseau, "The Construction of Routes and Schedules for the Transportation of the Handicapped", Working Paper, University of Montreal, (1983).

[31] Russell, R.A., "An Effective Heuristic for the M-Tour Travelling Salesman Problem with Some Side Conditions", Operations research 25(3), 517-524 (1977).

[32] Sexton, T.R., L.D. Bodin, "Optimizing Single Vehicle Many-To Many Operations with Desired Delivery Times : I. Scheduling", Working Paper W-83-10, (1983).

[33] Sexton, T.R., L.D. Bodin, "Optimizing Single Vehicle Many-To Many Operations with Desired Delivery Times : II. Routing", Working Paper W-83-11, (1983).

[34] Solomon, M.M., "Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms", 83-02-01, The Wharton School, University of Pennsylvania, (1983).

[35] Solomon, M.M., "On the Worst-Case Performance of Some Heuristics For the Vehicle Routing and Scheduling Problem with Time Window Constraints", working paper 84-02, College of Business Administration Northwestern University, (1984).

[36] Stein, D.M., "An Asymptotic, Probabilistic Analysis of a Routing Problem", Mathematics of Operations Research 3, 89-101, (1978).

[37] Stein, D.M., "Scheduling Dial-A-Ride Transportation Systems", Transportation Science 12, 232-249, (1978).

[38] Tharaken, G.G., H.N. Psaraftis, "An Exact Algorithm for the Exponential Disutility Dial-A-Ride Problem", working paper, MIT, (1981).

[39] Tyagi, M., "A Practical Method for the Truck Dispatching Problem", Journal of Operations Research Society of Japan 10, 76-92 (1968).

[40] Wilson, Nigel H.M., Joseph M. Sussman, Ho-Kwan Wong, Trevor Higonnet, "Scheduling Algorithms for Dial-A-Ride system", Urban Systems Laboratory, USL TR-70-13, M.I.T. (1971).

[41] Wilson, Nigel H.M., R.W. Weissberg, B.T. Higonnet, J. Hauser, "Advanced Dial-A-Ride Algorithms: Interim report", R75-27 Department of Civil Engineering, M.I.T. (1975).

[42] Wilson, Nigel H.M., Weissberg, Hauser, "Advanced Dial-A-Ride Algorithms Research Project: Final Report", Department of Civil Engineering, M.I.T. Report R76-20, (1976).

[43] Wilson, Nigel H.M., E. Miller, "Advanced Dial-A-Ride Algorithms Research Project Phase II: Interim Report", Department of Civil Engineering, M.I.T. Report R77-31, (1977).

[44] Yellow, P., "A Computational Modification to the Savings Method of Vehicle Scheduling", Operations Research Quarterly 21, 281 (1970).