A COMPUTERIZED BUS SCHEDULE INFORMATION SYSTEM

by

Robert Wyer Simpson

A Thesis Submitted to the Faculty of the

DEPARTMENT OF CIVIL ENGINEERING
AND ENGINEERING MECHANICS

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN CIVIL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1973
STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED

Robert Wyler Simpson

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Martin C. Nizlek

MARTIN C. NIZLEK
Assistant Professor of Civil Engineering

Date

4-12-73
ACKNOWLEDGMENTS

The author wishes to thank Dr. Martin C. Nizlek for his guidance in the preparation of this thesis.

For conscientiously reviewing this thesis, sincere thanks are given to Dr. Rudolf A. Jimenez and Dr. Ralph E. Griswold.

Special appreciation is given to Leslie A. Follett of the University of Arizona Computer Center whose guidance in the use of the remote terminal made possible portions of this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>LIST OF ILLUSTRATIONS</td>
<td>vii</td>
</tr>
<tr>
<td></td>
<td>ABSTRACT</td>
<td>viii</td>
</tr>
<tr>
<td>I.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>The Problem</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Objectives of the Study</td>
<td>3</td>
</tr>
<tr>
<td>II.</td>
<td>A REVIEW OF LITERATURE RELATING TO BUS SCHEDULE PROBLEMS</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Minimum Path Algorithms and Their Applications</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Computer Application to Bus Scheduling</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Consideration of Public Preference to Transportation Systems</td>
<td>18</td>
</tr>
<tr>
<td>III.</td>
<td>THE BUS SCHEDULE INFORMATION SYSTEM PROGRAM</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Setting up the Program for a Particular Transit System</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Special Transit Network Characteristics</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Resultant Program Constraints</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Procedure for Setting up the Program</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>The General Program Logic</td>
<td>42</td>
</tr>
<tr>
<td>IV.</td>
<td>MAKING THE PROGRAM INTERACTIVE AND ANALYSIS OF SAMPLE RUNS</td>
<td>52</td>
</tr>
<tr>
<td>V.</td>
<td>RECOMMENDATIONS AND CONCLUSION</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Improving the Capability of the Program</td>
<td>64</td>
</tr>
<tr>
<td>TABLE OF CONTENTS—Continued</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conclusion</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td></td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>APPENDIX A: COMPUTER PROGRAM LISTING</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>LIST OF REFERENCES</td>
<td>92</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Operating Schedule for the Hypothetical Bus System</td>
<td>29</td>
</tr>
<tr>
<td>3.2 A List of Messages Generated by the Program</td>
<td>46</td>
</tr>
</tbody>
</table>
## LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A Typical Minimum Time Path Problem</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Solution to the Minimum Time Path Problem for Nodes X and Y</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Route Map of the Hypothetical Bus System</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>Flow Diagram for the Setup Procedure of the Program</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Flow Chart for the Main Program</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Flow Chart for SUBROUTINE TRNFER</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Sample Run Using an Origin Point of (6,17) and a Destination Point of (18,33)</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Sample Run Using an Origin Point of (9,27) and a Destination Point of (18,11)</td>
<td>56</td>
</tr>
<tr>
<td>4.3</td>
<td>Sample Run Using an Origin Point of (13,26) and a Destination Point of (12,4)</td>
<td>57</td>
</tr>
<tr>
<td>4.4</td>
<td>Sample Run Using an Origin Point of (2,32) and a Destination Point of (18,5)</td>
<td>58</td>
</tr>
<tr>
<td>4.5</td>
<td>Sample Run Using an Origin Point of (7,15) and a Destination Point of (17,9)</td>
<td>59</td>
</tr>
</tbody>
</table>
ABSTRACT

A computer program is developed that will determine the best route for a potential bus rider to take in order to get from point A to point B. The program allows for one transfer and a "desired travel time" option. The program is developed so that it can be made interactive with a remote terminal in order to allow for near instantaneous process time.

A system of this type could possibly make public transportation more desirable by providing an exact, easily accessible form of information regarding bus schedules. The reasons for the public's rejection of mass transportation are very complicated and involved. One area of rejection, however, could stem from the fact that the public has a poor knowledge of exactly where buses run and at what times. The information system program described above will help to solve this particular problem.
CHAPTER I

INTRODUCTION

The Problem

Today's urban transportation systems are very complicated, intricate systems, whose success are dependent not only upon technology but also upon the public's wants and desires. To date, the design of most of these systems has been completely dominated by the private automobile. Examples of such systems are freeways, and local and arterial streets whose intersections are, more than likely, controlled by progressive or traffic-actuated light signals. Little attention has been directed towards implementing other forms of intra-city transportation systems such as bus or rapid-rail systems. In fact, the attitude of many people is that public transportation exists only for those who have no other mode of travel available to them.

The reasons for the dominance of the private automobile and the failure of public transportation are many and varied and in many cases, highly indeterminate. The automobile is, of course, a highly convenient, personal, and private form of transportation, but these features alone have not been completely responsible for its success. The automobile has been marketed, packaged and presented to
the public as if there were no attractive alternative. This possibly then, leads to the conclusion that other forms of transportation (i.e., public transportation) should be thought of as existing within a competitive, consumer-oriented market. Lack of such thought, in the past, could quite possibly be one of the reasons for the public's rejection of public transportation.

Until recently, many transit companies have really not thought of themselves as existing within a competitive market. Many of them, accepted the fate of having a small ridership percentage and were happy if they had a break-even year in the income-expenditure column of their financial report. One area where marketing techniques are still not applied to their maximum is in the area of providing information regarding schedules of individual buses operating within the system. A person, unless regularly dependent upon a bus for transportation, usually has a poor knowledge of where buses stop and at what times. Along these lines, it is interesting to note that several public preference surveys have indicated that the public puts a very high importance on waiting a minimum amount of time for transportation. For a person then to minimize his waiting time, he must have a good knowledge of departure and arrival times of the bus he would take.
Along more economic lines, a recent article in Metropolitan magazine (1) describes improvements made to the Denver, Colorado, transit system after the city and county had taken over control of it in 1971. Because of these improvements, increased ridership was realized. This was due, in part, to the creation of a communications program center in order to improve customer access to information about the transit system. Five specialist positions were created to handle information calls inquiring into possible routes to take and other related matters. Up to 1,800 calls a day were received; showing the need for such an information system.

Such a system (using humans) has obvious limitations placed on it by the heavy reliance on the human. Speed of response and accuracy of response are two areas where such a system could be deficient. For instance, the most obvious bus route is not always the best route to take, especially in cases requiring a transfer. The above limitations suggest that the development of an improved information retrieval procedure might have considerable merit.

**Objectives of the Study**

The purpose of this study was to develop an information system that would be easily accessible to the public and attempt to alleviate the problem of the public's lack
of knowledge regarding schedules of buses within the transit system. The approach taken was to utilize the digital computer in the handling of the information so that it could be given to the public quickly and in an efficient manner. Based on airline experience and the desire to make information available almost instantaneously, input and output to and from the computer would be handled interactively with the use of a remote terminal.

The best bus route, in terms of travel time, would be found for a potential rider wanting to get from one point in a city to another. Based on the public's desire to avoid transfers, routes would be found, if available, that required no more than one transfer. The system would be developed, however, so that expanding acceptable routes to allow for two transfers could be accomplished without too much difficulty.

Of prime importance in the development of the system, would be to allow for a "desired travel time" option. This option would override the finding of the best route in terms of travel time. Instead a route would be found, if available, that would allow the rider to arrive at the destination point within a given amount of time (i.e., 30 minutes) of a desired time as specified by the rider. If more than one route were found, then the route with the minimum travel time would be selected. It was
realized that in some cases, the best route in terms of travel time, would also arrive by the desired time.

Along with providing information regarding schedules, the information system program could also benefit public transportation by always finding the most efficient route for a potential rider to take. The public could thus gain confidence in the bus system by knowing that they were taking the best route out of several possible available routes.
CHAPTER II

A REVIEW OF LITERATURE RELATING TO BUS SCHEDULE PROBLEMS

Before attempting to write the information system program, a search was made in literature pertaining to scheduling and routing of mass transit systems. The purpose of this was threefold. First, it had to be determined if work had already been done in this area and, if so, to what extent. Second, it was desired to see if any concepts used in other, nonrelated scheduling programs could be used in this program. Finally, assumptions had to be made regarding human desires (i.e., how far is a person willing to walk to get to a bus stop); by reviewing several public preference surveys, these assumptions could be crystalized to the point needed in detailed, explicit computer programming. Thus this chapter reviews pertinent literature and describes possible ways in which certain ideas could be incorporated into the program.

Minimum Path Algorithms and Their Applications

One of the major functions to be performed by the information system program was to find the best route for a bus rider to take. It was important then to review the many
minimum path algorithms developed for use on digital computer systems. As implied, many algorithms of this type have been developed, however, by far the most common is the Moore algorithm (2) developed in 1958. It is felt that a short description of this procedure is necessary in order to better understand the problems associated with finding the best route in a transit system.

The best way to describe this algorithm is with a simple example. Considering Figure 2.1, find the minimum time path from X to Y. The figures between a set of nodes represent travel time, in minutes, needed to transverse the link connecting those nodes.

Using the Moore technique the first step is to initial the starting point to zero (i.e., place a zero on node X). The next step is to place a 5 on node A, a 6 on node B, a 4 on node H, and a 5 on node G. The number in node A then represents the cumulative travel time needed to get to node A from the starting point, in this case, node X. For each succeeding step, establish the adjacent node(s) J, that connects to an adjacent node I, that already has an assigned value. Place in node J the sum of the number found in node I and the travel time needed to get to node J. If node J already has an assigned value replace it with the newly calculated time only if this new time is less. Thus, the number to place in node E would be 7, and in node D
Figure 2.1. A Typical Minimum Time Path Problem.
would be 8. Note that the cumulative travel time to get to node D from node B is 9, however, the number that remains in node D is 8. Repeat these steps until all possible travel times into node Y have been determined. The travel times to each of the nodes is as shown in Figure 2.2.

The final step, involves tracing back from node Y through consecutively adjacent nodes having the property that the two numbers appearing in the near and far end nodes of each link differ by the travel time to traverse that link. This process continues until a path to node X is established. Note that from node Y the path could either go by way of node K or node F. However, since the travel time to traverse link KY is 5 and this equals the difference between the numbers appearing in node K and node Y, the minimum path would go through node K. Applying the same logic to the other nodes, the minimum travel time path from X to Y would be the way of path XHEDKY.

It is interesting to note that although not required, minimum paths have also been established from node X to all other nodes. For instance, the minimum path from X to C would be XBC. This leads to the familiar term that the Moore algorithm is a "tree-building" procedure whereby, from any one node, minimum paths (branches) are found to all other nodes in the system.
Figure 2.2. Solution to the Minimum Time Path Problem for Nodes X and Y.
When applying this algorithm to a transit schedule program where the best route between only one set of nodes is required, several problems occur. Perhaps the biggest problem is that the algorithm assumes instantaneous transfers between nodes. In the case of a highway network this is perhaps a valid assumption, where it can be said that, for all practical purpose, an automobile desiring to exit a freeway in order to get to a city street can make an instantaneous transfer at an interchange leading to that street. However, in making a bus trip requiring a transfer from bus A to bus B at node I, the transfer wait time depends on the arrival time of bus A to node I and the frequency of bus B at node I. Very seldom will an instantaneous transfer occur where bus A arrives just before bus B departs.

Another problem in the Moore algorithm is that it is incapable of finding the best route based on a desired arrival time, if this route is not the minimum. Recall, that a desired option in the information system program is the ability to find a bus route based on a desired arrival time and, although this route would be the minimum of those routes satisfying the desired arrival time, it may not be a minimum when compared to all possible routes.

Finally, the Moore algorithm finds, for one given node, minimum paths to all other nodes. Although this is not a deficiency, it is rather inefficient considering that
a bus rider is only interested in finding the best route between two points.

Dial (3) has attempted to solve some of the above problems in his transit pathfinder algorithm, and to a large extent has successfully solved the problem of instantaneous transfers at node points. His program was developed to aid the transportation planner in comparing modes of travel by determining expected paths of travel for passengers using a transit system. The criterion for determining expected paths was travel time. No exact departure and arrival times were used as input, since projecting exact schedules 20 years into the future would be a questionable, if not fruitless effort. The program still suffers from some of the drawbacks of the Moore algorithm. It is unable to find the best path based on a desired arrival time and it still finds, for a given node, minimum paths to all other nodes. Further, it does not deal with exact time tables and thus does not satisfy one of the major goals of the information system program—to give the public exact times of departures and arrivals so that they may better plan their trip and minimize wait time. Finally, the pathfinder algorithm is written in MAP, a computer language requiring an assembler probably unavailable to most municipal computer installations.
In applying minimum path algorithms to transportation systems, Butas (4) and Beimborn (5) have arrived at some interesting concepts. Butas has basically used the Moore algorithm and extended it so that it has the ability of finding the second and third shortest paths. There is, though, no guarantee that the second or third shortest path will arrive by the desired arrival time, if required. Delays, such as time delays at intersections, have been incorporated into the program; a realization that a transfer at a node in some cases is not instantaneous. These delays, however, are constant and are not variable such as in the case of a bus transfer, where delay is dependent on the departing bus's frequency.

Beimborn's approach is interesting in that a grid travel time model is developed that assumes grid shaped paths and a velocity function used to calculate travel times instead of having them furnished. This velocity function was a function of the distance from the Central Business District (CBD). It was assumed that the CBD had the lowest speeds and that speed increased with increasing distance from the CBD. (This particular technique will be discussed later when possible modifications to the information system program are explored.) A cascade algorithm was then used to compute minimum travel times between points in the road network. This type of algorithm differed
slightly from the Moore algorithm in that minimum paths were calculated between all pairs of nodes simultaneously.

Finally, Dreyfus (6) has presented a paper giving an appraisal of most of the important types of minimum path algorithms. There are many valid concepts in these techniques but when trying to apply them to a transit schedule they all show most of the problems associated with the Moore technique as described previously.

The peculiarities of bus schedules plus the constraints put on any technique used to find the best bus route, would make any minimum path algorithm very difficult to incorporate into the information system program. By limiting transfers, an alternative method would be to calculate all available routes and to pick from this group the best route based on certain constraints. This method would become inefficient if more than two transfers were involved.

Although it may seem as such, it is not intended to discredit minimum path algorithms. These algorithms have been applied successfully in traffic assignments and in other areas of transportation planning. In fact, if any information system program were to allow for an unlimited number of transfers, possibly the only way to establish a path would be with the use of a minimum path algorithm.
Computer Application to Bus Scheduling

The application of computers to the bus scheduling problem basically falls into three areas according to Elias (7):

1) Determining areas of demand and fitting a route network to satisfy these demands.

2) Determining frequencies for each bus route.

3) Determining run assignments for operators of each bus.

Recently, a new area has appeared where simulation techniques are used to schedule buses in a demand responsive transportation system. Although not directly related, all these areas were explored to see if any concepts could be utilized in the information system program. Also, it was hoped that by gaining an understanding of the problems associated with transit scheduling, the information system program could be written in an efficient and general manner.

In approaching the problem of determining a fixed route network based on demand, Lampkin and Saalmans (8) have suggested that a properly designed network should including the following features:

1) Areas of appreciable demand should be connected by routes not requiring transfers.

2) Routes should not meander, but be as direct as possible.
3) The intersection of two routes should be timed so that transfer wait time is kept to a minimum.

4) The resulting network should not contain too many different routes.

An algorithm was then developed to build a route and tie it into a network system.

To a certain extent, most of the above features have been incorporated into the information system program. For instance, Feature 3 suggests that some upper limit should be put on time spent waiting for a transfer. Just because two bus routes have a common point of intersection does not mean that this constitutes a valid transfer point if the wait time is excessive. Feature 4 suggests that the information system program need not overburden itself with excessive memory requirements since the number of bus routes in a transit system will, more than likely, be small. It should be kept in mind that the program will be interactive with a remote terminal and thus enter the computer's operating system randomly throughout the day. Thus, a large memory requirement would not only slow response time but also interfere with the regularly scheduled operations of the municipality's computer installation.

In the area of determining run assignment for operators, Elias (7, 9, 10) has done considerable work. He has developed computer programs that take existing bus routes
and frequencies and assigns operators to it. In making such assignments, consideration has been given to constraints found in labor management agreements, and minimizing undertime and overtime costs. Elias has stated that there is considerable interplay between scheduling of service and scheduling of operators. Therefore, he has proposed that a procedure be developed to blend these two related areas into one overall scheduling process. As these computer programs are more involved in establishing bus and/or operator schedules rather than finding a route through an existing schedule, no constructive ideas were found for the information system program.

Finally, Howson and Heathington (11) have reviewed several computer simulation techniques used to route and schedule demand-actuated transportation systems. These programs give special attention to the selection of vehicle for pickup and delivery of passengers, methods of assigning priorities to these vehicles, dispatching policies, and establishing levels of service. These vehicles (Dial-a-bus, Demand-Jitney, etc.) do not operate under fixed-schedules but rather vary their schedules based upon demand, desired level of service and other constraints; the ultimate objective being to maximize occupancy of individual vehicles while minimizing inconvenience to passengers. As a result, the concepts used in these simulation programs could not
be adapted to the information system program using fixed-schedule principles.

Consideration of Public Preference to Transportation Systems

Of major importance in the development of the information system program was the consideration given to the public's wants and desires. The program incorporated these factors so that a realistic route was found for the rider. If no realistic route could be found, the computer would then indicate this. In other words, major emphasis was given to the acceptability of the route selected. It was believed that the public would soon lose confidence in any system that caused them to walk a mile to a bus stop, wait 15 to 20 minutes in the hot summer sun for a transfer, or transfer three to four times before getting to the destination point.

In determining what human preference factor should be included in the information system program, several behavior and preference surveys towards public transportation were studied. Two of the better surveys studied were by Golob, Canty, Gustafson and Vitt (12) and Hartgen and Tanner (13).

In the Golob et al. study a survey and analysis were made in order to aid in the design of a demand-responsive Jitney system. The objectives of the study were
to evaluate political and social acceptance of the proposed system by sampling current consumer preferences in regards to public transportation. Although the study was directed towards a demand-responsive transportation system, the preferences determined could be applied to any public transportation system. In fact, those households that were interviewed were not told of the type system that was being designed.

The method employed for quantifying consumer preference was based on experimental psychology. In selecting measurement devices for these preferences, paired comparison and semantic scaling were utilized. Interviews, incorporating 91 system characteristics, were then conducted at 1,260 random households. From this, 1,603 individual questionnaires were processed.

Data from the study were then reduced and the system characteristics were grouped into preference factors and listed according to the importance the public put on them. Of the 32 factors, four were found that could be applied to the information system program. They were:

1) No transfer trip required.
2) Arriving when planned.
3) Less wait time.
4) Less walk to pick-up.
"Arriving when planned" was considered by the public to be the most important feature a public transportation system could have. "No transfer trip required," was considered the third most important feature. "Less walk to pick-up" was ranked in the upper third of the 23 factors and "less wait time" was ranked very high.

In the Hartgen and Tanner study (13), a behavioral approach to the modeling of mode choice was developed. The idea behind this process was to let individual travelers choose a travel mode (i.e., bus, car). Information and data for their study came from a behavior survey—the Paine Study (14). This particular survey was conducted in metropolitan Philadelphia by the University of Maryland in 1967. People were requested to rate on a scale of 1 to 7 the perceived importance of each of 33 system attributes. All attributes fell within a range of 6.4 to 3.7. Along with these ratings, people were asked to rate each of several modes of travel in accordance with its ability to satisfy each of the system attributes.

As in the Golob et al. study (12), attributes that could be applied to the information system program were isolated and analyzed. Those attributes, along with their perceived importance, were as follows:

1) Arrive at intended time, (6.4)
2) Arrive in shortest time possible, (5.9)
3) Avoid waiting more than five minutes, (5.8)
4) Avoid walking more than one block, (5.5)
5) Avoid changing vehicles, (5.0)

As with the Golob et al. study, "arriving at intended time" was considered by the public to be the most important attribute in determining a mode of travel. It is interesting to note, that unlike the Golob et al. study, the Paine study found that the public puts only an average importance on avoiding trips that required a transfer. Hartgen and Tanner have suggested that this rating is due to the fact that Philadelphia has a well developed transit system and that the public would not mind making a transfer trip under these conditions. Another conclusion that could be drawn is that Philadelphia is an area of high-density population and that a larger percentage than normal of people do not own automobiles. Therefore, these people would have to accept the possibility of a transfer trip on the transit system since this is the only mode of travel available to them. Both arguments to a certain degree are probably valid.

Incorporating the features found in the two studies into the information system program was done on a quantitative basis. Before proceeding, though, it should be noted that one of the features was not actually used in the
program but rather used as a justification for it. As discussed in the introductory chapter, this feature (avoid waiting more than five minutes) implied that if the public were to use a public transit system they would require information regarding departure and arrival times of buses operating in the system. Thus, in obtaining this information they could minimize their waiting time.

Interesting at first, but obvious upon second thought, was the importance the public put on arriving at the intended time. Both studies found that this was the most important feature of any transit system; more important even than arriving in the shortest amount of time. This stands to reason since, even though a minimum path might be available that would arrive at the destination point by 8:10 A.M., it was useless if the rider has to be at that point by 8:00 A.M. This was allowed for in the information system program by developing a "desired arrival time" option. This option would override the minimum route if this path did not arrive within 30 minutes of the desired arrival time and there was another route available that did arrive within 30 minutes. The value of 30 minutes was determined arbitrarily but could easily be changed if warranted.

The Golob et al. study (12) indicated a large degree of reluctance on the part of the public to accept a transfer
trip while using a transit system, whereas the Hartgen and Tanner study suggested more willingness on the part of the public to accept a transfer. For this reason the program was equipped with logic to search for a route involving a transfer if a single route could not be found. The subject of a transfer could be argued either way. On the one hand, it could be argued that no transfer should be allowed; on the other hand, arguments could be presented allowing for more than one transfer. Based on the above studies, though, it was felt that the problem could be realistically solved by allowing for one transfer.

Finally, both studies suggested that some upper limit should be put on the distance required to walk in getting to a bus stop. The Hartgen and Tanner study went further by defining that distance to be one block. It was felt that limiting acceptable bus stops to a one-block radius put undue hardship on the information system program by considerably reducing the search area around each stop. Furthermore, the distance value of one block was not clearly defined in the study. For these reasons, a value of 2.9 blocks was used in the program. The intent was to limit the search to a two-block area; however, it was realized that a rider might be two blocks from a bus stop but be so positioned on one of the blocks that he might have to walk a fraction of a block in one direction before
walking two blocks perpendicular to that direction in getting to the bus stop. A value slightly larger than two blocks was, thus, somewhat arbitrarily chosen. The distance value to be associated with 2.9 blocks will be discussed in the next chapter.

The public preference factors arrived at in this section were determined by analyzing several behavior and preference studies in the area of public transportation. While it was not within the scope of this study to analyze public preference in transit systems, values and emphasis areas were chosen based upon these contemporary research efforts.
CHAPTER III

THE BUS SCHEDULE INFORMATION SYSTEM PROGRAM

As stated in the Introduction, the general objective of this research was to develop a system to obtain information regarding bus route schedules. A computer program made interactive with a remote terminal was the means chosen to accomplish this objective. This decision was based on the following.

1) It was important to find the optimum route.
2) Instantaneous response time was considered essential.

Reasons for finding the best route have previously been discussed. The reason for requiring instantaneous response time was based on the reasoning that no customer would want to wait idly by for five to ten minutes while some specialist manually determined a route for him to take. Thus, a bus schedule information system program written in FORTRAN IV was developed.

The last chapter described the difficulties involved with incorporating a minimum path algorithm into the information system program. The approach used for the program, therefore, was to calculate all available routes, involving no more than one transfer, and to pick from this group, the
best route based on certain constraints. This method seemed reasonable, since the preference surveys towards public transportation indicated that the public was reluctant to make even one transfer, let alone two or three.

No attempt will be made in this chapter to describe the detailed, step-by-step logic of the program. Instead, the procedure needed to set up the program will be presented along with an example of a hypothetical bus system. Following this, the general program logic will be discussed, aided by the use of flow charts and the program listing.

**Setting up the Program for a Particular Transit System**

**Special Transit Network Characteristics**

Setting up the program requires information about the bus system that can be divided into three areas. First, a coordinate grid system in which the bus system operates has to be defined. Second, the bus stop locations for each route have to be described in terms of this grid system. Finally, arrival/departure times must be established for each stop along a route. Associated with these times, are the frequencies of each route. Any transit system that can define its operations in the above manner should be able to use the program.

To better aid in the discussion, it is convenient to introduce a hypothetical bus system. Such a system is
shown in Figure 3.1. The system consists of seven bus routes with each route having from four to fourteen stop locations. The operating schedule for this system is described in Table 3.1. In order to be compatible with the computer program, the schedule deviates from normal transit schedules in two respects. First, the bus stop locations (stop numbers) are described in terms of an X-Y coordinate system (i.e., (X,Y)) instead of actual street intersections. A procedure for converting these coordinate points to actual street names will be discussed in the chapter on Recommendations. Second, the direction of travel in a route is defined as either positive or negative, depending on whether travel is in the direction of increasing (positive) or decreasing (negative) stop numbers. Many transit companies define these directions as either inbound or outbound, depending on whether travel is in towards, or away from the Central Business District. Use of these terms become confusing, though, when a particular route passes through the CBD and as a result, can enter the CBD from two different directions. Furthermore, some routes do not even operate near the CBD, and the use of inbound or outbound becomes doubly confusing.

Resultant Program Constraints

In setting up the program, consideration must be given to certain constraints of the program.
Figure 3.1. Route Map of the Hypothetical Bus System.
Table 3.1. Operating Schedule for the Hypothetical Bus System.

<table>
<thead>
<tr>
<th>NEGATIVE DIRECTION-READ UP</th>
<th>LOCATION</th>
<th>POSITIVE DIRECTION-READ DOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minutes after hour</td>
<td>Stop number (X,Y)</td>
<td>Minutes after hour</td>
</tr>
</tbody>
</table>

**Route 1**

| 19, 49 | 1 (13, 6) | 10, 40 |
| 17, 47 | 2 (13,11) | 12, 42 |
| 14, 44 | 3 (13,16) | 15, 45 |
| 14, 44 | 4 (13,17) | 15, 45 |
| 09, 39 | 5 (13,21) | 20, 50 |
| - -    | 6 (13,24) | 22, 52 |
| - -    | 7 (13,26) | 23, 53 |
| - -    | 8 (18,26) | 25, 55 |
| - -    | 9 (18,31) | 28, 58 |
| - -    | 10 (13,31) | 30, 00 |
| - -    | 11 (10,31) | 31, 01 |
| - -    | 12 (10,26) | 34, 04 |
| - -    | 13 (10,24) | 35, 05 |
| 07, 37 | 14 (13,24) | 37, 07 |

**Route 2**

| 10, 40 | 1 ( 8, 6) | 15, 45 |
| 07, 37 | 2 ( 8,11) | 18, 48 |
| 06, 36 | 3 ( 8,14) | 19, 49 |
| 05, 35 | 4 ( 8,16) | 20, 50 |
| 03, 33 | 5 ( 8,18) | 21, 51 |
| 01, 31 | 6 ( 8,21) | 23, 53 |
| 57, 27 | 7 ( 8,26) | 27, 57 |

**Route 3**

| 11, 31, 51 | 1 ( 8, 6) | 05, 25, 45 |
| 09, 29, 49 | 2 ( 8,11) | 07, 27, 47 |
| 07, 27, 47 | 3 ( 8,14) | 08, 28, 48 |
| 05, 25, 45 | 4 ( 8,16) | 10, 30, 50 |
| 03, 23, 43 | 5 ( 8,18) | 12, 32, 52 |
| 00, 20, 40 | 6 ( 3,18) | 15, 35, 55 |
| 58, 18, 38 | 7 ( 3,21) | 17, 37, 57 |
| 55, 15, 35 | 8 ( 3,26) | 20, 40, 00 |
| 50, 10, 30 | 9 ( 3,31) | 25, 45, 05 |

**Route 4**

<p>| 34 | 1 ( 3,21) | 15 |
| 31 | 2 ( 8,21) | 18 |
| 29 | 3 (13,21) | 21 |
| 25 | 4 (18,21) | 25 |</p>
<table>
<thead>
<tr>
<th>Negative Direction-Read Up</th>
<th>Location Stop Number (X,Y)</th>
<th>Positive Direction-Read Down</th>
<th>Minutes after hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route 5</td>
<td>17, 37, 57</td>
<td>1 (17,11)</td>
<td>05, 25, 45</td>
</tr>
<tr>
<td></td>
<td>15, 35, 55</td>
<td>2 (17,16)</td>
<td>07, 27, 47</td>
</tr>
<tr>
<td></td>
<td>12, 32, 52</td>
<td>3 (13,16)</td>
<td>10, 30, 50</td>
</tr>
<tr>
<td></td>
<td>12, 32, 52</td>
<td>4 (12,16)</td>
<td>10, 30, 50</td>
</tr>
<tr>
<td></td>
<td>07, 27, 47</td>
<td>5 (8,16)</td>
<td>15, 35, 55</td>
</tr>
<tr>
<td></td>
<td>- - -</td>
<td>6 (6,16)</td>
<td>17, 37, 57</td>
</tr>
<tr>
<td></td>
<td>- - -</td>
<td>7 (3,16)</td>
<td>19, 39, 59</td>
</tr>
<tr>
<td></td>
<td>- - -</td>
<td>8 (3,11)</td>
<td>21, 41, 01</td>
</tr>
<tr>
<td></td>
<td>- - -</td>
<td>9 (6,11)</td>
<td>23, 43, 03</td>
</tr>
<tr>
<td></td>
<td>05, 25, 45</td>
<td>10 (6,16)</td>
<td>25, 45, 05</td>
</tr>
<tr>
<td>Route 6</td>
<td>00, 30</td>
<td>1 (3,11)</td>
<td>15, 45</td>
</tr>
<tr>
<td></td>
<td>54, 24</td>
<td>2 (8,14)</td>
<td>20, 50</td>
</tr>
<tr>
<td></td>
<td>48, 18</td>
<td>3 (12,16)</td>
<td>23, 53</td>
</tr>
<tr>
<td></td>
<td>48, 18</td>
<td>4 (13,17)</td>
<td>23, 53</td>
</tr>
<tr>
<td></td>
<td>42, 12</td>
<td>5 (18,20)</td>
<td>29, 59</td>
</tr>
<tr>
<td></td>
<td>41, 11</td>
<td>6 (18,21)</td>
<td>30, 00</td>
</tr>
<tr>
<td></td>
<td>38, 08</td>
<td>7 (18,26)</td>
<td>33, 03</td>
</tr>
<tr>
<td></td>
<td>35, 05</td>
<td>8 (18,31)</td>
<td>36, 06</td>
</tr>
<tr>
<td>Route 7</td>
<td>12, 42</td>
<td>1 (3, 6)</td>
<td>00, 30</td>
</tr>
<tr>
<td></td>
<td>08, 38</td>
<td>2 (8, 6)</td>
<td>03, 33</td>
</tr>
<tr>
<td></td>
<td>03, 33</td>
<td>3 (13, 6)</td>
<td>07, 37</td>
</tr>
<tr>
<td></td>
<td>00, 30</td>
<td>4 (17, 6)</td>
<td>10, 40</td>
</tr>
</tbody>
</table>
These constraints basically fall into two categories. Belonging in the first category are those constraints if relaxed that would merely require increasing the size of the DIMENSION or COMMON statements of the computer program. Belonging in the second category are those constraints if relaxed that would require modification to the program logic.

The constraints that would fall into the first category are as follows:

1) The transit system can contain up to 20 individual routes.

2) Each route is limited to 60 stop locations.

3) Each route can have a frequency of up to 20 runs per hour.

By increasing the appropriate DIMENSION or COMMON statements, these limits can be increased as required.

The constraints that would fall into the second category are as follows:

4) The grid system is presently limited to a 998 x 998 block area.

5) Individual routes must be defined as either entirely two-way (Type 1) or two-way with a one-way loop at one end (Type 2).

6) The time to traverse a route in one direction from end to end is limited to 60 minutes.
The meaning of these constraints will become clear in the discussion to follow.

Before actually quantifying information about a transit schedule, the individual routes making up the schedule must be investigated to see that they satisfy the requirements of Constraint 5. Referring to Figure 3.1, routes 2, 3, 4, 6 and 7 are classified under Constraint 5 as Type 1 routes. That is, for each stop location along a route, service is provided for in the positive and negative directions (i.e., two-way service). A small problem arises under this classification when a bus operates on paired one-way streets where travel in one direction is on a one-way street and travel in the opposite direction is on an adjacent one-way street. This problem can be solved by assigning stop locations for both directions to one of the streets and so modifying the output to indicate the proper street depending on the direction of travel. A procedure to implement this will be discussed in the chapter on Recommendations. Example of Type 2 routes are routes 1 and 5. These routes are the same as Type 1 routes except for a one-way loop at one end of the route where service at stop locations in the loop is provided for in only one direction.

Procedure for Setting up the Program

The first item to be defined for the program is the X-Y coordinate grid that covers the area the bus system
operates in. The grid is to be rectangular with one unit in the X or Y direction equalling one block. The last chapter stated that an acceptable walking distance to a bus stop was 2.9 blocks. This was arrived at considering that each block was approximately 800 feet in length. If for some reason, a transit system feels that for their particular needs each unit in the X or Y direction should be greater than or smaller than the standard block, this change would be allowed. However, the acceptable walking distance would then have to be decreased, or increased accordingly. For instance, if a block were to be defined to equal 400 feet instead of 800 feet, the walking distance factor would have to be increased to 5.8 blocks since the acceptable walking distance would still have to equal 2,320 feet. Incorporating this into the program requires that the variable BLOCK be set equal to that block length, in feet, used in setting up a particular grid system. For the sample bus system, the standard block length of 800 feet was used (i.e., BLOCK = 800). 

No particular orientation for the grid is required. However, the format of Figure 3.1 is suggested. This shows the origin of the grid to be in the lower left-hand corner with the grid covering enough area so that there is at least a three-block distance between the X and Y axis and the extreme stop location of any one route. The only
requirement for the program is that the grid be broken into four quadrants so that the search procedure, to be discussed in the next section, can be shortened. The one rule that has to be followed is that the two lines dividing the grid into quadrants be parallel to the X and Y axis, respectively. These lines are defined in the program as NXN (parallel to the X axis) and NYN (parallel to the Y axis). For the grid system of Figure 3.1, values of NXN = 16 and NYN = 11 were chosen. Although not required, it is suggested that the setting up of the four quadrants be done so that each quadrant has approximately the same number of bus routes operating in it as compared to another quadrant. This will reduce the search procedure since no one quadrant will contain an overload of routes to be investigated.

For each quadrant, the bus routes operating in it or within an acceptable walking distance of it must be defined. This is accomplished by the use of the NBUS(I,J) integer array where I represents the quadrant number and J represents the number of routes operating in or near this quadrant. Data for the NBUS array appears in the form of a DATA declaration found in the data section of the program.¹ As an example, the bus routes operating in or near the third quadrant.

---

¹. A data section is not formally a part of a FORTRAN program. For convenience, it is used here to indicate that section where all DATA declarations appear.
quadrant of the sample bus system are routes 1, 2, 3, 5, 6 and 7. Route 1 is included since it operates within 2.9 blocks, the acceptable walking for this sample system. Note that Route 1 is also included in the NBUS array for the other quadrants since it operates in or near each of the four quadrants. Additionally, the NQ(I) integer vector must be defined where I represents the quadrant number. This vector indicates the total number of buses operating in or near each quadrant.

It is realized that hardly any city will have all parallel and perpendicular streets, let alone blocks that are all of the same length. Therefore, approximations will have to be made where street intersections, not falling exactly on a X-Y coordinate point, will have to be assigned to the nearest coordinate point. A great deal of judgment, then, is needed in arriving at a grid coordinate system for any particular city.

Once the grid system has been established, data input for stop locations and arrival/departure times of individual bus routes merely becomes a time-consuming exercise. Instead of employing the normal READ statements to input this data into the program, use is made of the DATA declaration where data actually becomes a part of the program. The main advantage to be gained by this is speed. Not only is time saved by allowing the program and data to
be stored in memory as one integral unit but valuable time is saved once program execution has begun by not requiring the computer to search for some input file located perhaps somewhat on a disc unit. As in the case of the NBUS array, all DATA declarations containing individual route information appear in the data section of the program.

The first item to be defined in this series is the number of stop locations for each bus route. This is accomplished through the use of the NSTOP(I) integer vector, where I represents the route number.

The NBUSXY(I,J) integer array describes the coordinate grid location for each stop along an individual route. In this array, I represents the route number and J represents the stop number along this route. Of course, the number of stop locations described for any one route in the NBUSXY array must equal the number of stop locations as defined in the NSTOP vector for that route.

Certain rules must be followed in setting up grid locations for a route. For a Type 1 route, a six-digit number is used to define each stop location. The first three digits define the X coordinate and the last three digits define the Y coordinate. Using this single number technique to define a grid location saves considerable memory; however, logic must be provided in the program to reduce this number to two three-digit numbers. As a result,
a six-digit number must be used every time even though the coordinate points may be very small numbers. Thus, the six-digit number representing \( X = 3 \) and \( Y = 12 \) would be 003012. For a Type 1 route it does not matter which end of the route is used as the starting point. Using Route 4 of the sample bus system as an example, it can be seen from Table 3.1 and Figure 3.1 that the left end of the route was chosen as the starting point. The complete DATA declaration for Route 4 is then:

\[
\text{DATA (NBUSXY}(4,I),I=1,4)/003021, 008021, 013021, 018021/}
\]

A few additional rules must be followed in setting up a Type 2 route. The starting point must begin at the non-loop end of the route and once in the loop, stop location numbers must increase in the same direction as the direction of travel. Furthermore, the stop location where the loop begins (pivot node) must be defined twice; the first time when the bus passes the pivot node and later when the bus returns to this node. Note that the loop does not necessarily have to be rectangular in shape. The same six-digit numbering technique is used to define the \( X \) and \( Y \) coordinate points for that portion of the route that is two-way. The only difference for the loop portion is that a one precede the number, resulting in a seven-digit number. The pivot node is considered to be a part of the loop. To clarify the setup procedure for this type route, the NBUSXY
DATA declaration, for Route 5 of the sample bus system is as follows:

\[
\text{DATA (NBUSXY(5,I),I=1,10)/017011,017016,013016,012016,008016,006016,1003016,1003011,1006011,1006016/}
\]

Note that the starting point (17,11), is at the non-loop end of the route and that the pivot node (6,16) is defined twice.

For stop locations along each route, arrival/departure times have to be assigned. This is accomplished through the use of the NTIME(I,J) integer array, where I represents the route number and J represents the stop location number. Arrival and departure times for the program are considered to be the same since the time lapse between arrival and departure for a bus at a stop location is negligible. For each route, only the first run of the hour has to be defined. For all type routes, times are shown as minutes after the hour and use is made of a four-digit number to describe the time for the positive and negative direction of each stop location. The first two digits indicate the arrival time in the positive direction while the last two digits indicate the arrival time in the negative direction. As in the case of the NBUSXY array, a four-digit number is needed each time even though certain arrival times may be under ten minutes.
For a Type 1 route, arrival times for the first run of the hour in both the positive and negative direction are used to populate the NTIME array. Using Route 7 of the sample bus system as an example, the NTIME DATA declaration is:

\[
\text{DATA (NTIME(7,I),I=1)/3012,2208,3703,4000/}
\]

For the positive direction, the bus arrives at stop location 1 and 4 at 30 and 40 minutes after the hour, respectively. For the negative direction, the bus arrives at stop locations 4 and 1 at 00 and 12 minutes after the hour, respectively. Note that 00 minutes after the hour is always used instead of 60 minutes after the hour.

Arrival times for a Type 2 route are described somewhat differently. The first run of the hour is chosen and arrival times are assigned in both the positive and negative directions based on tracing this run completely through the route until it returns to the starting point. Note that once in the loop, the negative direction time must be the same as the positive direction time. Consider the NTIME DATA declaration for Route 5 of the sample bus system.

\[
\text{DATA (NTIME(5,I),I=1,10)/0537,0735,1032,1032,1527,1717,1919,2121,2323,2525/}
\]

The first bus of the hour leaves stop location 1 at 5 minutes after the hour and returns to this location 32
minutes later at 37 minutes after the hour. Note that the bus enters the pivot node, (6,16), at 17 minutes after the hours and returns to this node eight minutes later at 25 minutes after the hour.

The final item to be defined in the data section is the frequency of each route. The NFREQ(I) integer vector defines this relationship where I indicates the route number. In establishing frequencies, certain assumptions were made. First, it was assumed that the number of runs per hour (frequency) for a route were divided evenly throughout the hour. Second, it was assumed that the frequency in the positive direction was equal to the frequency in the negative direction. For the sample bus system the following NFREQ DATA declaration was established for the frequencies of the seven routes:

```
DATA (NFREQ(I),I=1,7)/2,2,3,1,3,2,2/
```

Note that even though Route 4 has only one run per hour, its frequency still has to be defined.

In summary, six arrays (vectors) must be populated in the form of DATA declarations appearing in the data section of the program. They are: NBUS, NQ, NSTOP, NBUSXY, NTIME and NFREQ. Immediately following this the variables NYN, NXN and BLOCK must be defined. For clarity, the flow diagram of Figure 3.2 is provided to act as an outline of the setup procedure described in this section.
Insure that routes can be classified as either Type 1 or Type 2 routes.

Establish grid coordinate system.
   a) define NXN, NYN, and BLOCK
   b) populate NBUS(I,J) array
   c) populate NQ(I) vector

Define stop locations for each route.
   a) populate NSTOP(I) vector
   b) populate NBUSXY(I,J) array

Assign arrival times at stop locations of each route.
   a) populate NTIME(I,J) array
   b) define frequencies with NFREQ(I) vector

Figure 3.2. Flow Diagram for the Setup Procedure of the Program.
The General Program Logic

The information system program, in the execution of a problem, basically performs four steps as described below:

1) All bus routes are found that stop within an acceptable walking distance of the origin and destination points.

2) From these routes, paths are built that connect the origin and destination points.

3) The path is found that satisfies the constraints of desired arrival time and/or minimum travel time.

4) Information regarding route numbers, departure/arrival times, and travel times are printed for all paths.

The flow charts for this procedure are shown in Figures 3.3 and 3.4. Table 3.2 lists the various messages that could be printed based on certain circumstances that could arise in the execution of the program.

For input, the specialist operating the remote terminal simply enters grid point locations for the origin and destination points and the desired arrival time in 24-hour clock notation. If there is no desired arrival time, a number greater than 2400 is entered. A more detailed description of this input procedure will be presented in the next section.
Figure 3.3. Flow Chart for the Main Program.
Determine those routes that arrive within the desired time. 

Are such routes available? 

Yes  

Of these routes, find the one with minimum travel time. 

Write message; see Table 3.2. 

No  

Is there a desired time of arrival? 

Yes  

Find minimum travel time route. 

Write information on this route. 

Write information on remaining routes. 

Do transfer routes exist? 

Yes  

Call SUBROUTINE TRNFER 

No  

Is a transfer required? 

Yes  

Figure 3.3. Flow Chart for the Main Program, continued.
Identify an origin point route and a destination point route as a route combination.

Identify this combination

For this route combination, find all points in common.

Do points in common exist?  Yes

Determine travel times for all valid transfer points.

Save only the path going through the transfer point giving the minimum travel time.

Identify each point in common as a valid point if it requires no more than a 10-minute wait.

Is another route combination available?

RETURN

Figure 3.4. Flow Chart for SUBROUTINE TRNFER.
<table>
<thead>
<tr>
<th>Message</th>
<th>Circumstance</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO BUS GOES NEAR DEPARTURE POINT.</td>
<td>No buses could be found that operate within an acceptable walking distance of the origin point.</td>
</tr>
<tr>
<td>NO BUS GOES NEAR DESTINATION POINT.</td>
<td>No buses could be found that operate within an acceptable walking distance of the destination point.</td>
</tr>
<tr>
<td>BUS HAS SAME DEPARTURE AND DESTINATION POINTS.</td>
<td>Distance between the origin and destination points was so small that the same stop location number was assigned for both the departure and arrival stations.</td>
</tr>
<tr>
<td>NO BUS ROUTE IS AVAILABLE, EVEN CONSIDERING A TRANSFER.</td>
<td>No direct or transfer routes could be found.</td>
</tr>
<tr>
<td>THERE ARE NO BUSES THAT WILL ARRIVE WITHIN 30 MINUTES OF DESIRED ARRIVAL TIME. HOWEVER, THE FOLLOWING BUSES ARE LISTED FOR CONVENIENCE.</td>
<td>No desired arrival time routes could be found.</td>
</tr>
</tbody>
</table>
Step 1 above relieves the specialist of having to assign actual bus routes and stop locations (stations) for the program. This feature was incorporated into the program for two reasons. First, it is possible that response time can be improved considerably by eliminating the need for the specialist to look at a route map in order to find acceptable bus stop locations. Second, by not restricting the program to one particular combination of routes, all routes that stop within an acceptable distance of the origin and destination points can be investigated in order to find the best combination.

The accomplishment of Step 1 involves dividing the grid system that the bus system operates in, into four quadrants as described previously. The main program, ROUTE, then determines which quadrants contain the origin and destination points. This is done in order to investigate only those routes that operate in the appropriate quadrants, thus reducing execution time. SUBROUTINE SEARCH is then utilized to investigate each stop location, starting with the first station along each route operating in the indicated quadrant. This SUBROUTINE is first executed in conjunction with the origin point; then in conjunction with the destination point. The function HYPOT is used to calculate the distance between a station and the origin or destination point. For each route, the first station that
is within the acceptable walking distance is initially chosen. The succeeding stations, equalling in number the integer value of the acceptable walking distance plus one, are investigated to see if the required walking distance to one of them is less than that of the initially chosen station. Of these stations, the station is chosen that requires the least walking distance.

When control is returned to the main program, the vectors MBUS and MSTOP are populated, respectively, with those routes and appropriate stations that stop within an acceptable walking distance of the origin or destination point. If routes cannot be found for either the origin or destination points, then a message will be printed according to Table 3.2 and execution of the program will stop.

Once origin and destination routes have been established, an investigation is made to determine if routes are available that arrive at both the origin and destination points. These routes are then stored in the IBUS vector. The destination stop locations of these routes are stored in the ISARRV vector and the origin stop locations are stored in the ISDEPT vector. For a Type 2 route, however, where the origin station is in the loop and the destination station is outside of the loop, the origin station must be optimized. This is necessary when an origin station is chosen that is in the loop near the pivot node that would
require the rider to board the bus when the bus first enters the loop. Since the bus returns to the pivot node after traversing the loop, it would be better for the rider to board at this later time in order to reduce travel time. This optimization procedure is handled by SUBROUTINE LOOP. Finally, travel times for the routes are calculated and stored in the ITT vector.

If routes are not available that arrive at both the origin and destination points, then SUBROUTINE TRNFER is called. This routine finds route combinations, if available, that would require no more than one transfer. Departure routes (origin point routes) are compared to arrival routes (destination point routes) to see if they have a point in common. Realizing that one route may have more than one point in common with another route, all points in common are determined. These points constitute valid transfer points only if the transfer wait time is at least one minute but less than or equal to ten minutes. It was felt that allowing an instantaneous transfer (no wait time) was improper since the operating schedule of a transit system would probably not be that exact. Travel times are then calculated for these two-route combinations considering each valid transfer point. These travel times are compared and only that route combination containing the transfer point that results in the minimum travel time
is saved. All frequencies of this combination are also saved. This procedure is repeated for all other possible route combinations. If no route combinations can be found then a message is printed according to Table 3.2 and execution of the program stops.

Once a path (direct route or combination of routes) has been established, the program checks to see if there is a desired arrival time. If so, those paths that allow the rider to arrive at the destination point no earlier than 30 minutes before the desired time are separated from the other paths. Of these selected paths, the one having the minimum travel time is considered to be the best path. If no path can be found satisfying the constraints of the desired arrival time option, a message indicating this is printed.

If there is no desired arrival time or if no desired arrival time path can be found, the program compares all paths to find the one having the minimum travel time. This one is then considered to be the best path.

The final phase of the program involves printing information regarding route numbers, departure/arrival times and travel times for all paths. Information on the best path is printed first. If this path happens to be a desired arrival time path, departure/arrival times are printed in 24-hour clock notation. Information regarding
the remaining paths is then printed under the heading "Other Routes Available." The printing of information for the remaining paths was considered necessary since a rider might not always consider the best path to be the most convenient.

The above paragraphs have described the general logic of the program. The actual program listing can be found in Appendix A. Comments are contained throughout this listing to further supplement the understanding of the program logic.
CHAPTER IV

MAKING THE PROGRAM INTERACTIVE
AND ANALYSIS OF SAMPLE RUNS

Adapting the information system program for remote terminal usage in order to obtain instantaneous responsive was a major objective of this research. The program was initially written in such a way as to minimize the difficulty in this procedure. However, minor modifications will probably have to be made for each computer installation it is to be run on. The discussion to follow will describe the procedure needed in adapting the program to the Control Data 6400 computer and its related INTERCOM system. It is hoped that this procedure will generally apply to any computer system.

The information system program must first be compiled\(^1\) and an object deck obtained. Although the program could be compiled each time it is to be run, this would be extremely time-consuming and expensive; thus, the need for an object deck. This object deck must then be given a file name and made a permanent file in the operating system of the computer. In order for the program to be interactive

---

1. Procedure by which the program is translated into machine language.
with the terminal, the object deck file must be attached to the terminal's list of files. In INTERCOM, the input and output files of the program are automatically attached to the terminal when the terminal is connected to the computer. To begin execution of the program requires that the object deck file be named when INTERCOM is in the COMMAND mode.

The program needs information regarding origin point, destination point, and desired arrival time before routes can be found. When the program is ready to accept this information it will print three messages on the terminal as follows:

ENTER THE ORIGIN POINT
ENTER THE DESTINATION POINT
ENTER THE DESIRED ARRIVAL TIME

Each message requires a response and succeeding messages will not be printed until this response is received. For each of the first two messages a six-digit number representing the X-Y coordinate point of the appropriate location is needed. The first three digits represent the X coordinate and the last three digits represent the Y coordinate. The response for the third message is a four-digit number in 24-hour clock notation for the desired arrival time. If there is no desired arrival time, a four-digit
number greater than 2400 must be entered. All of the above numbers must be in integer form.

When the program has completed execution of a particular problem, it will loop back to the beginning and start printing the first of the above messages in order to begin execution of a new problem. If it is desired to exit from this loop and stop any further execution of the program, the number 999999 must be entered instead of an origin point coordinate number. The terminal will go back to the COMMAND mode and the object deck file name must be entered again in order to begin execution of the program for a new problem.

To illustrate the above procedure and show the value of the program, Figures 4.1 through 4.5 show output of several sample runs using the schedule of the hypothetical bus system described previously. The example of Figure 4.1 uses an origin point of (6,17) and a destination point of (18,33). The desired arrival time is 1:30 P.M. This example shows the many route combinations available even for a small transit system such as the sample bus system. The best bus route arrives at 1:06 P.M. with a total travel time of 21 minutes. Notice that the third route, although having a total travel time of 23 minutes, arrives at the destination at 1:28 P.M., only two minutes before the desired time. Thus, this route when compared to
ENTER THE ORIGIN POINT 006017
ENTER THE DESTINATION POINT 018033
ENTER THE DESIRED ARRIVAL TIME 1330

THE FOLLOWING IS THE BEST BUS ROUTE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(6, 16)</td>
<td>1245</td>
<td>7 MIN</td>
</tr>
<tr>
<td></td>
<td>(12, 16)</td>
<td>1252</td>
<td></td>
</tr>
</tbody>
</table>

WITH A TRANSFER (1 MIN WAIT) TO

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>(12, 16)</td>
<td>1253</td>
<td>13 MIN</td>
</tr>
<tr>
<td></td>
<td>(18, 31)</td>
<td>1306</td>
<td></td>
</tr>
</tbody>
</table>

TOTAL TRAVEL TIME, INCLUDING WAIT TIME = 21 MIN

OTHER ROUTES AVAILABLE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(8, 18)</td>
<td>43</td>
<td>4 MIN</td>
</tr>
<tr>
<td></td>
<td>(8, 14)</td>
<td>47</td>
<td></td>
</tr>
</tbody>
</table>

WITH A TRANSFER (3 MIN WAIT) TO

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>(8, 14)</td>
<td>50</td>
<td>6 ** 16 MIN</td>
</tr>
<tr>
<td></td>
<td>(18, 31)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTAL TRAVEL TIME, INCLUDING WAIT TIME = 23 MIN

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(6, 16)</td>
<td>5</td>
<td>7 MIN</td>
</tr>
<tr>
<td></td>
<td>(13, 16)</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

WITH A TRANSFER (3 MIN WAIT) TO

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(13, 16)</td>
<td>15</td>
<td>13 MIN</td>
</tr>
<tr>
<td></td>
<td>(18, 31)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTAL TRAVEL TIME, INCLUDING WAIT TIME = 23 MIN

** INDICATES MINUTES AFTER NEXT HOUR.

Figure 4.1 Sample Run Using an Origin Point of (6,17) and a Destination Point of (18,33).
ENTER THE ORIGIN POINT 009027  
ENTER THE DESTINATION POINT 018011  
ENTER THE DESIRED ARRIVAL TIME 1600  

THE FOLLOWING IS THE BEST BUS ROUTE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(10, 26)</td>
<td>(13, 16)</td>
<td>1534</td>
<td>1544</td>
<td>10 MIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WITH A TRANSFER (8 MIN WAIT) TO</td>
</tr>
<tr>
<td>5</td>
<td>(13, 16)</td>
<td>(17, 11)</td>
<td>1552</td>
<td>1557</td>
<td>5 MIN</td>
</tr>
</tbody>
</table>

TOTAL TRAVEL TIME, INCLUDING WAIT TIME = 23 MIN

OTHER ROUTES AVAILABLE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(8, 26)</td>
<td>(8, 16)</td>
<td>57</td>
<td>5 **</td>
<td>8 MIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WITH A TRANSFER (2 MIN WAIT) TO</td>
</tr>
<tr>
<td>5</td>
<td>(8, 16)</td>
<td>(17, 11)</td>
<td>7</td>
<td>17</td>
<td>10 MIN</td>
</tr>
</tbody>
</table>

TOTAL TRAVEL TIME, INCLUDING WAIT TIME = 20 MIN

** INDICATES MINUTES AFTER NEXT HOUR.

Figure 4.2. Sample Run Using an Origin Point of (9,27) and a Destination Point of (18,11).
ENTER THE ORIGIN POINT 013026
ENTER THE DESTINATION POINT 012004
ENTER THE DESIRED ARRIVAL TIME 0800

THE FOLLOWING IS THE BEST BUS ROUTE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME (AFTER HR)</th>
<th>ARRIVAL TIME (AFTER HR)</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(13, 24)</td>
<td>(13, 6)</td>
<td>737</td>
<td>749</td>
<td>12 MIN</td>
</tr>
</tbody>
</table>

OTHER ROUTES AVAILABLE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(13, 24)</td>
<td>(13, 6)</td>
<td>7</td>
<td>19</td>
<td>12 MIN</td>
</tr>
</tbody>
</table>

Figure 4.3. Sample Run Using an Origin Point of (13,26) and a Destination Point of (12, 4).
ENTER THE ORIGIN POINT 002032
ENTER THE DESTINATION POINT 018005
ENTER THE DESIRED ARRIVAL TIME 1715

THERE ARE NO BUSES THAT WILL ARRIVE WITHIN 30 MINUTES OF DESIRED ARRIVAL TIME. HOWEVER, THE FOLLOWING BUSES ARE LISTED FOR CONVENIENCE.

THE FOLLOWING IS THE BEST BUS ROUTE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(3, 31)</td>
<td>(8, 6)</td>
<td>10</td>
<td>31</td>
<td>21 MIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WITH A TRANSFER (2 MIN WAIT) TO</td>
</tr>
<tr>
<td>7</td>
<td>(8, 6)</td>
<td>(17, 6)</td>
<td>33</td>
<td>40</td>
<td>7 MIN</td>
</tr>
</tbody>
</table>

TOTAL TRAVEL TIME, INCLUDING WAIT TIME = 30 MIN

OTHER ROUTES AVAILABLE

NONE

Figure 4.4. Sample Run Using an Origin Point of (2, 32) and a Destination Point of (18, 5).
ENTER THE ORIGIN POINT 007015
ENTER THE DESTINATION POINT 017009
ENTER THE DESIRED ARRIVAL TIME 9999

THE FOLLOWING IS THE BEST BUS ROUTE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(6, 16)</td>
<td>(17, 11)</td>
<td>25</td>
<td>37</td>
<td>12 MIN</td>
</tr>
</tbody>
</table>

OTHER ROUTES AVAILABLE

<table>
<thead>
<tr>
<th>ROUTE NUMBER</th>
<th>DEPARTURE POINT</th>
<th>ARRIVAL POINT</th>
<th>DEPARTURE TIME</th>
<th>ARRIVAL TIME</th>
<th>TRAVEL TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>(6, 16)</td>
<td>(17, 11)</td>
<td>45</td>
<td>57</td>
<td>12 MIN</td>
</tr>
<tr>
<td>5</td>
<td>(6, 16)</td>
<td>(17, 11)</td>
<td>5</td>
<td>17</td>
<td>12 MIN</td>
</tr>
</tbody>
</table>

Figure 4.5. Sample Run Using an Origin Point of (7,15) and a Destination Point of (17, 9).
the best route allows the rider to depart from the origin point at a later time even though the best route has a shorter travel time. This presents somewhat of a dilemma for the decision logic within the program. Of the desired arrival time routes, should the program consider the best route to be the one of minimum travel time or the one that arrives closest to the desired time? The decision to choose the route having the minimum travel time was based on the belief that this route would, in general, result in a ride requiring less stops and, in the case of transfer routes, a minimum transfer wait time. The other route combinations are listed, however, to give the rider an alternative in case the best route is not convenient.

The example of Figure 4.2 is interesting in that the best route is not the minimum travel time route. The desired arrival time is 4:00 P.M.; however, the second route having the minimum travel time of 20 minutes arrives 17 minutes after 4:00 P.M. The best route is the one arriving at 3:57 P.M. even though this route has a greater travel time.

The remaining examples are included to show direct route listings and a few of the messages generated by the program. Note that there are no routes that satisfy the desired arrival time requirements of the example in Figure 4.4. The only route available arrives at 4:40 P.M., 35 minutes before 5:15 P.M. This results in the arrival
time being too early, since a lead time of 30 minutes is the maximum allowable.

To illustrate the program logic, consider an example having an origin point of (8,17) and a destination point of (19,21). The program would find acceptable origin point routes to be Routes 2, 3 and 5. Acceptable destination routes would be Routes 4 and 6. The program would next determine if there were direct routes available. Finding none, it would investigate two-route combinations to see if any transfer route combinations were available. All possible route combinations with points in common would be: 2-4 (i.e., board Route 2 and transfer to Route 4), 2-6, 3-4, 3-6 and 5-6. Route combinations 2-4, 2-6 and 3-4 would be ruled out since the transfer wait time for these combinations would be greater than 10 minutes. For route combination 5-6, a transfer would be possible at location (12,16) if the third run of the hour for Route 5 and the second run of the hour for Route 6 were used. This would result in a travel time of 13 minutes. For route combination 3-6, a transfer would be possible at location (8,14) if the third run of the hour for Route 3 and the second run of the hour for Route 6 were used. This would result in a travel time of 17 minutes. Finally, the best route would be selected on the basis of desired arrival time and/or minimum travel time.
An informal test was conducted to compare the performance of the information system program with the performance of a system using humans. People were chosen at random and asked to find routes in the hypothetical bus system given origin and destination information. Similar information was processed through the information system program and the results compared. Problems to be solved involved both direct route and two-route combination solutions. Some of the problems also contained desired arrival time constraints.

The time required by people to obtain solutions ranged between 2.5 to 19 minutes. Problems involving direct routes normally required much less time to solve than problems involving transfers and desired arrival times. For the same problems, program execution time excluding terminal printing time ran between .036 and .112 seconds. Terminal printing time, of course, was dependent upon the printing speed of the particular unit used. Another area of importance was the ability to find the best route. Using the best routes found by the program as a basis, people found the best route 42 percent of the time. Part of the reason for this low percentage might be due to the peculiarities of the hypothetical bus system. People probably had difficulty relating it to a real system and the
use of coordinate points rather than street names undoubtedly confused some.

Since this was an informal test, no firm conclusions could be formulated. The speed of the program, though, has been shown. Furthermore, the inability of the persons involved in this test to find the best routes suggests that if a transit system is to be efficiently used, a sophisticated information system is needed.
CHAPTER V

RECOMMENDATIONS AND CONCLUSION

Improving the Capability of the Program

Very few computer programs have probably been written that could not be improved somewhat in capability and efficiency. With this assumption in mind, several suggestions will be proposed, as additional areas of research, that would hopefully improve the quality and performance of the information system program.

Probably the most obvious improvement that could be made to the program would be to modify the output so that actual street intersection names could be printed for departure and arrival points rather than the presently used grid coordinate points. This feature is not incorporated into the present program as this is more of a data management problem than a conceptual problem. A possible way to implement this feature would be to create a street listing master file. This file would actually consist of two files. Each record of the first file would contain consecutively numbered X coordinate points with corresponding street names while the second file would contain similar information for Y coordinate points. The program would still find departure and arrival points in terms of coordinate points.
Just before the printing routine, however, the program would start searching the street listing master file until matches could be found with the X-Y coordinate points. The appropriate street names would be read and subsequently printed in place of the coordinate points. This procedure, of course, would require additional execution time as locating the file and searching for the appropriate record would be a time-consuming affair. The program would also have to rewind this file before execution of a new problem.

A similar procedure could be implemented for input into the program where street names would be entered rather than coordinate points. This procedure is not recommended, though, as the probability of error is too great. For instance, the computer could not detect misspellings and different streets sometimes have similar names such as 12th Street and 12th Avenue; a situation confusing to many people. Not only would this procedure require additional execution time but an elaborate error routine would probably have to be incorporated. Besides, a qualified operator who could distinguish between coordinate points and street names would be inputing data into the program. On the other hand, the printing of street names should be required for output as this output would be read by the customer who would probably consider the printing of coordinate points as useless information.
Along these lines, a scheme was mentioned early in regards to one-way paired streets whereby output could be modified to indicate the proper one-way street. This could be accomplished by defining the stop locations of a route with the NBUSXY array except that the stop locations in the one-way section would be flagged and defined for the positive direction only. Before the output routine, the selected stop locations would be examined to see if they were in a one-way section and in the negative direction. If so, information about corresponding stop locations in the negative direction would be used. This information would be contained in an array similar to the NBUSXY array.

It was shown in the last chapter that information about individual routes is inputed into the program by the use of DATA declarations. This is a somewhat awkward procedure but execution time is saved. Although not necessarily an improvement, a more convenient and conventional procedure would be to input this information through the use of a standard input file. Using this method, a check routine could be established which would be highly desirable since data entry of this type is subject to considerable error. As an example, an initial input file could be prepared. From this, distances between adjacent stop locations for each route could be calculated. If there was an excessive distance, the appropriate stop locations could be
flagged for re-investigation. The same technique could apply for the absolute difference between arrival times at adjacent stop locations with the added advantage of being able to determine if times increase in the positive direction and decrease in the negative direction. Once the file was corrected, it could be used as input to the program.

In a more general sense, an interesting area of study would be to compare the information system program with a program utilizing a minimum path algorithm such as an adapted version of the Dial technique (3) discussed in Chapter II. Although a minimum path algorithm would be difficult to adapt to a transit network, such an algorithm would become necessary if the number of allowable transfers became unlimited. A possible way to accomplish this might be to use the Dial technique to find the general path outline for the three best paths between the origin and destination points. This technique could provide information pertaining to route numbers and transfer points. Once this information were known, actual arrival times could be assigned to the appropriate stop locations along each path. From this, exact travel times and transfer wait times could be calculated for all frequencies of each path. These paths could then be sorted in terms of minimum total travel time.
For transit systems operating in large cities where frequencies of individual routes approach 10 to 15 runs per hour, a basic minimum path algorithm could possibly be used. In this case, transfer wait times would become negligible and the Moore algorithm (2) could be used rather than the more sophisticated Dial algorithm (3). In the development of his algorithm, Dial postulated that the average transfer wait time was one-half the inverse of the recipient route's frequency. If this be the case, then the time spent waiting to transfer to a route with a frequency of 15 runs per hour would only be two minutes. For all practical purposes, this value could be considered negligible resulting in an instantaneous transfer.

Incorporating a minimum path algorithm into the information system program would require additional data input to the program. This data would consist of the usual information needed in minimum path calculations for networks such as node locations and link travel times. A way to reduce this burden might be to implement a procedure similar to that used by Beimborn (5) where a velocity function was used to calculate travel times between links rather than having these times inputed. This would introduce another approximation in in the calculation of a minimum path but since only the general outline of a path would be needed, this approximation could be tolerated.
Finally, the information system could be adapted for use as a transportation planning tool. The program could keep a record of the origin and destination points each time it was run. This data could then be reduced to find areas of demand. In cases where areas of demand were serviced by bus routes, investigations could be made to determine if the extent of demand warranted increases or decreases in the frequencies of the routes. In cases where areas of demand were not serviced by bus routes, decisions could be made as to whether or not new routes should be created or existing routes changed in order to service these demand areas. Information on desired arrival times at key destination points could also be kept in order to determine if the time schedules of the involved routes should be restructured so that desired arrival time requirements could better be met.

For transfer routes, records could be kept on key transfer points so that the time schedules of the involved routes could be restructured in order to minimize transfer wait time. A similar technique could be used on two-route combinations that were available but could not be used since the transfer wait time would have been excessive.

**Conclusion**

A bus schedule information system has been developed that provides information on the best bus routes for a
person wanting to get from one point in a city to another. This information is processed by utilizing a computer program that is made interactive with a remote terminal in order to obtain near instantaneous response.

In most cases, the best route is considered to be the one having the minimum travel time. The public's wants and desires, though, play an important role in the success or failure of any public transit system. Therefore, an attempt was made to quantify these needs into the information system program in order to find the best route in terms of these needs rather than in terms of travel time alone. The program takes into account such needs as establishing upper limits for walking distances and transfer wait times, restricting the number of allowable transfers, and determining desired arrival time requirements. Values for a certain number of these needs were arrived at somewhat arbitrarily and the program was written in such a way that these values could easily be changed if so warranted.

Throughout this study, emphasis has been placed on the information system program as an economic tool to be used by transit companies and as a needed system in order for the public to gain confidence in the bus as a viable form of transportation. The program could also be used on a promotional basis. Such a system could be set up at shopping malls or convention centers so that information
on bus routes could be easily obtained, if for no other reason than curiosity.

As the title of this thesis indicates, the information system program has been developed for a bus system. However, the program is also applicable to any transit system that operates under a fixed-schedule such as subways or surface rapid-railways.

In conclusion, it is realized that the bus schedule information system will not suddenly project public transportation into competition with the automobile. Making mass transit a viable, attractive alternative will require many steps over a long period of time. Public transportation will exist within a competitive environment and improving its marketing techniques are essential. To this end, it is hoped that the bus schedule information system presented will help.
APPENDIX A

COMPUTER PROGRAM LISTING
PROGRAM ROUTE (INPUT, OUTPUT)
REMARKS. A COMPUTERIZED BUS SCHEDULE INFORMATION
SYSTEM.
AUTHOR. RW SIMPSON.
COMMON /ALL1/ NBUSXY(20,60), NSTOP(20), NBUS(4,20),
1 MSTOP(60), NBUS(20), NTIME(20,60), NREQ(20),
2 NQ(4), NDEPT(20), NDSTOP(20), NDEUP(20), NARRIVE(20),
3 NSTOP(30), TTCV(30), ITT(30), IBUS(30), NFN(60),
4 ISARV(30), INARV(30), TRACV(30), NTAR(30), FLG(2),
5 NOAV, NOAV1, NOAV2, NOAV3, NCT, N-T, NCT1
COMMON /ALL3/ NDEPART, NARRIVE, NOX, NOY

*** DATA SECTION ***
DATA (NBUS(1,I),I=1,4)/1,4,5,6/
DATA (NBUS(2,I),I=1,6)/1,2,3,4,5,6/
DATA (NBUS(3,I),I=1,7)/1,4,7,9,4,10,9,4/
DATA (NBUS(4,I),I=1,7)/1,4,7,9,4,10,9,4/
DATA (NBUSXY(1,J),J=1,14)/013006,013011,013016,013017,013021,
1 1013024,1013026,1013026,1013031,1013031,1013031,10140026,
2 10140026,10140026/
DATA (NBUSXY(2,J),J=1,7)/038306,038306,038306,038306,038306,038306,038306/
DATA (NBUSXY(3,J),J=1,9)/038306,038306,038306,038306,038306,038306,038306,038306,038306/
DATA (NBUSXY(4,J),J=1,7)/038306,038306,038306,038306,038306,038306,038306,038306,038306/
DATA (NBUSXY(5,J),J=1,10)/017011,017011,017016,017016,017016,017016,017016,017016,017016,017016/
DATA (NBUSXY(6,J),J=1,8)/038306,038306,038306,038306,038306,038306,038306,038306,038306/
DATA (NBUSXY(7,J),J=1,4)/063006,063006,063006,063006,063006,063006,063006,063006,063006/
DATA (NTIME(1,I),I=1,14)/1049,1247,1544,1544,2039,2222,
1 2323,2525,2828,3030,3131,3434,3535,3737/
DATA (NTIME(2,I),I=1,7) / 1540,1837,1936,2035,
12133,2331,2727 /
DATA (NTIME(3,I),I=1,9) / 0531,0729,0827,1025,
11023,1526,1718,2015,2510 /
DATA (NTIME(4,I),I=1,4) / 1534,1831,2129,2525 /
DATA (NTIME(5,I),I=1,10) / 0537,0735,1032,1032,
11527,1717,1919,2123,2923,2525 /
DATA (NTIME(6,I),I=1,8) / 1530,2024,2318,2318,
12912,3611,3709,3605 /
DATA (NTIME(7,I),T=1,4) / 3012,3309,3703,4000 /
DATA (NFRQ(I),I=1,7) / 2,2,3,1,3,2,2 /
DATA FLAG/ 3H , 3H **/

NYN=11
NYN=16
BLOCK=8J0.
WFACT=2320,/BLOCK

DISPLAY MESSAGES AND ACCEPT ORIGIN AND
DESTINATION INFORMATION.

800 FORMAT

30 FORMAT("/* ENTER THE ORIGIN POINT */")
READ 1, NOX,NOY
IF (NOX.EQ.999.AND.NOY.EQ.999) GO TO 801
1 FORMAT(2I3)
POINT 31
71 FORMAT("/* ENTER THE DESTINATION POINT */")
READ 1, NOX,NOY
PRINT 32
32 FORMAT("/* ENTER THE DESIRED ARRIVAL TIME */")
READ 33, NDAV
3 FORMAT(I4)
IF (NDAV.GT.2400) GO TO 3
NDAV1=NDAV/100 * NDAV=NDAV-(NDAV1*100)
TF(NDAV.GT.300) NDAV2=60
NDAV3=NCAV1

FIND BUS ROUTES OPERATING NEAR ORIGIN AND
DESTINATION POINTS.

COUNT=1.
LOCX=NOX $ LOCY=NOY
GO TO 5

LOCX=NOX & LOCY=NOY
COUNT=2.

DETERMINE THE QUADRANT TO BE INVESTIGATED.

IF (LOCX.GE.NYN.AND.LOCY.GE.NXN) GO TO 10
IF (LOCX.LE.NYN.AND.LOCY.GE.NXN) GO TO 11
IF (LOCX.GE.NYN.AND.LOCY.LE.NXN) GO TO 12
NQUAD=3
GO TO 15

10 NQUAD=1
GO TO 15

11 NQUAD=2
GO TO 15

12 NQUAD=4

15 NCOUNT=0
CALL SEARCH (LOCX,LOCY,NQUAD,NCOUNT)
IF (COUNT.EQ.2.) GO TO 300
IF (NCOUNT.EQ.0.) GO TO 620
NDEPART=NCOUNT
DO 20 T=1,NCOUNT
NDBUS(I)=MBUS(I)

20 MDEPART(I)=MSTOP(T)
GO TO 2

360 IF (NCOUNT.EQ.0.) GO TO 510
NPRIVE=NCOUNT
DO 21 I=1,NCOUNT
NABUS(I)=MBUS(I)

21 MAPRV(T)=MSTOP(I)
NCX=0
C DETERMINE IF DIRECT ROUTES ARE AVAILABLE.
DO 250 J=1,NEPAPT
DO 250 J=1,MARRIVE
IF(MARY(J),GE.NAUS(J)) GO TO 250
II=NAUS(J)
JJ=MARRY(J) & JL=DEPART(TK)
IF(JJ,EQ.JL) GO TO 795
NFLAG1=NAUSYY(II,JJ)/1000000
NFLAG2=NEUSYY(JI,JL)/1000000
C OPTIMIZE ORIGIN POINT FOR TYPE 2 ROUTES.
IF(NFLAG1.EQ.1.AND.,NFLAG2.EQ.1.AND.,JL.GT.JJ) GO TO 250
TF(NFLAG2.EQ.1.OR.,NFLAG1.GT.1) GO TO 125
CALL LOOP (NOY,NOY,IT,JL)
125 MTD1=NTIME(TT,JL)/100
NTD2=NTIME(JI,JL)-(NTD1*100)
NTA1=NTIME(JJ,JJ)/100
NTA2=NTIME(JJ,JJ)-(NTA1*100)
NVEC=JJ-JL
C DETERMINE DIRECTION OF TRAVEL.
IF(NVEC)185,190,190
185 NTA=NTA?
NTD=NTD2
GO TO 191
190 NTA=NTA1
NTD=NTD1
C ASSIGN FREQUENCIES AND DEPARTURE/ARRIVAL TIMES.
191 NG=NFRFQ(II)
NSPACE=60/NG
DO 192 I=1,NG
192 NPN(I)=NSPACE*(T-1)
DO 195 JJJ=1,NG
NCT=NCT+1
ITAPPV(NCT)=NTA+NPN(JJJ)
IF (ITARRV(NCT) .GE. 60) ITARRV(NCT) = ITARRV(NCT) - 60
ITDEPT(NCT) = NTD + NPN(JJJ)
IF (ITDEPT(NCT) .GE. 60) ITDEPT(NCT) = ITDEPT(NCT) - 60
ITAV = ITARRV(NCT)
IF (ITDEPT(NCT) .GE. ITAV) ITAV = ITAV + 60
ITI(NCT) = ITAV - ITDEPT(NCT)
IPUS(NCT) = IT
ISDEPT(NCT) = JL
195 ISARRV(NCT) = JJ
250 CONTINUE
LFLAG = 0
C FIND TRANSFER ROUTES IF DIRECT ROUTES ARE NOT AVAILABLE.
IF (NCT.NE.0) GO TO 400
CALL TPNFEP
LFLAG = 1
IF (NCT.EQ.0) GO TO 800
400 NCT1 = 0
PHOUF = 1.
IF (NDAV.GT.2400) GO TO 455
C DESIRED ARRIVAL TIME OPTION.
DO 450 I = 1, NCT
IF (ITARRV(I) .GT. NDAV?) GO TO 450
NDIFF = NDAV? - ITARRV(I)
IF (NDIFF .EQ. 60) NDIFF = 0
IF (NDIFF .GT. 30) GO TO 450
NCT1 = NCT1 + 1
NPNM(NCT1) = I
450 CONTINUE
IF (NCT1 .NE. 0) GO TO 461
PRINT 700 * PRINT 701
700 FORMAT(/,** THERE ARE NO BUSSES THAT WILL ARRIVE*,
1* WITHIN*/,*. 30 MINUTES OF DESIRED ARRIVAL TIME. HOWEVER,*
701 FORMAT(* THE FOLLOWING BUSSES ARE LISTED FOR*,
1* CONVENIENCE.*)
C      FIND MINIMUM TRAVEL TIME ROUTE.
455  NTT=1
    DO 460 J=1,NCT
    IF(ITT(J).GE.ITT(NTT)) GO TO 460
    NTT=J
460  CONTINUE
461  PRINT 710 & PPINT 750
    PPINT 751 & PRINT 752
    IF(NCT1.NE.0) GO TO 470
710  FORMAT(//,"* THE FOLLOWING IS THE BEST BUS ROUTE*")
750  FORMAT(//,2AX,"* DEPARTURE ARRIVAL*")
751  FORMAT(31H ROUTE DEPARTURE ARRIVAL ,
     1 27HTME TIME TRAVEL)
752  FORMAT(34H NUMBER POINT POINT (AFTER,
     1 23H HR) (AFTER HR) TIME,/)  
    IF(LFLAG.EQ.0) GO TO 500
    CALL TIMER
    GO TO 800
C      FIND BEST DESIRED ARRIVAL TIME ROUTE.
470  NTT=NPRT(1)
    DO 480 J=1,NCT1
    L=NPRT(J)
    IF(ITT(L).GT.ITT(NTT)) GO TO 480
    NDIFF1=NDAV2-ITARRV(NTT)
    IF(NDIFF1.EQ.0) NDIFF1=0
    NDIFF2=NDAV2-ITARRV(L)
    IF(NDIFF2.EQ.0) NDIFF2=0
    IF(NDIFF2.LT.NDIFF1.OR.ITT(L).LT.ITT(NTT)) NTT=L
480  CONTINUE
    IF(LFLAG.EQ.0) GO TO 485
    CALL TIMER
    GO TO 800
485  CONTINUE
C      24-HOUR CLOCK NOTATION FOR DESIRED ARRIVAL TIME ROUTE.
IF (ITDEPT(NTT) .GE. ITARRV(NTT)) OR (NDAV2 .EQ. 60) NDAV3 = NDAV1 - 1
NPTMD = (NDAV3 * 100) + ITDEPT(NTT)
IF (NDAV2 .EQ. 60 .AND. ITARRV(NTT) .NE. 0) NDAV1 = NDAV1 - 1
NPTMWA = (NDAV1 * 100) + ITARRV(NTT)

500 NHOUR = 1
I1 = IFUS(NTT)
I2 = ISCEPT(NTT)
I3 = ISARPV(NTT)
C REDUCE STOP LOCATION NUMBER TO TWO NUMBERS.
NLP = NBUSXY(T1, I2) / 1000000
NXD = (NBUSXY(T1, I2) - (NLP * 1000000)) / 1000
NYD = NBUSYY(T1, I2) - ((NXD * 1000) + (NLP * 1000000))
NLP = NBUSXY(T1, I3) / 1000000
NYA = (NBUSXY(T1, I3) - (NLP * 1000000)) / 1000
NYA = NBUSYY(T1, I3) - ((NYA * 1000) + (NLP * 1000000))
C PRINT INFORMATION ON BEST ROUTE.
IF (NDAV1 .GT. 2 .OR. NCT1 .EQ. 0) GO TO 510
PRINT 720, T1, NXD, NYD, NXA, NYA, NPTMD, NPTMWA,
1 FLAG(NHOUR), TTT(NTT)
720 FORMAT (3X, I2, 4X, 1H(T1, I3, 1H), I3, 3H)(, T3, 1H, I3, 4H),
1 I4, 7X, T4, A3, 2X, I3, 4H MIN,/) GO TO 520
510 IF (ITARRV(NTT) .LE. ITDEPT(NTT)) NHOUR = 2
IF (NHOUR .LE. 2) PHOUR = 2
PRINT 720, T1, NXD, NYD, NXA, NYA, ITDEPT(NTT),
1 ITARRV(NTT), FLAG(NHOUR), TTT(NTT)
C PRINT INFORMATION ON REMAINING ROUTES.
520 PRINT 730
730 FORMAT (/ A18, 23H OTHER ROUTES AVAILABLE,/) GO TO 600
IF (NCT .EQ. 1) GO TO 530
GO TO 530, I = 1, NCT
NHOUR = 1
IF (T .EQ. NTT) GO TO 530
IF (ITARRV(I) .LE. ITDEPT(I)) NHOUR = 2
IF(NHOUR.EQ.0.2) PHOUR=2.
I1=IPUS(I) & I2=ISCEPT(I)
I7=ISAPRV(I)
NLP=NPUSXY(I1,I2)/1000000
NXD=(NPUSXY(I1,I2) - (NLP*100000))/1000
NYD=NPUSXY(I1,I2) - ((NXD*1000)+(NLP*100000))
NLP=NPUSXY(I1,I2)/1000000
NXA=(NPUSXY(I1,I3) - (NLP*100000))/1000
NYA=NPUSXY(I1,I3) - ((NXA*1000)+(NLP*100000))
PRINT 720, I1,NXD,NYD,NXA,NYA,ITCEPT(I),
1 ITAPRV(I),FLAG(NHOUR),ITT(I)
530 CONTINUE
GO TO 776
600 PRINT 776
776 FORMAT(2FX,5H NONE)
777 FORMAT(/,10X,38H ** INDICATES MINUTES AFTER NEXT HOUR,)
GO TO 800
610 PRINT 780
780 FORMAT(/,* NO BUS GOES NEAR DESTINATION POINT*)
GO TO 800
620 PRINT 790
790 FORMAT(/,* NO BUS GOES NEAR DEPARTURE POINT*)
GO TO 800
695 PRINT 795
795 FORMAT(/,* BUS HAS SAME DEPARTURE AND DESTINATION POINTS*)
801 STOP
END
SUBROUTINE SEARCH (LOCX,LOCY,NOUAD,NCOUNT)

COMMON / ALL1/ NRUSXY(20,60), NSTOP(20), NBUS(4,20),
MSTOP(60), MBUS(20), NTIME(20,60), NFPE0(20),
NQ(4), NDFUS(20), MDEPART(20), NFUS(20), MARPV(20),
MTHRD(30), ITDEPT(70), TTT(30), IBUS(30), NPN(60),
ISDEPT(30), ISARPV(30), NPRT(30), FLAG(2), WFAC,
NDAV, NDAV1, NDAV2, NDAV3, NCT, NTT, NCT1

DIMENSION NY(60), NX(60)
XL=LOCY  YL=LOCY
N=NO(NOQUAD)

C INVESTIGATE ALL ROUTES IN GIVEN QUADRANT.
DO 1 I=1,N
MP=NRUS(NOQUAD,I)
NN=NSTOP(MB)

C REDUCE STOP LOCATION NUMBER TO TWO NUMBERS.
DO 3 IT=1,NN
NLP=NRUSXY(MB,II)/1000000
NX(II)=(NRUSXY(MB,II)-(NLP*1000000))/1000
NY(II)=NRUSXY(MB,II)-(NLP*1000000)

C DETERMINE DISTANCE TO STOP LOCATION.
DC 9 J=1,NN
X=NX(J)  Y=NY(J)
DIST=HYPOT(X,Y,XL,YL)

C FIND STOP LOCATION WITHIN ACCEPTABLE WALKING DISTANCE.
IF(DIST.GT.WFAC) GO TO 9
NPST=J
L=J+1
IF(L.GT.NN) GO TO 20
KFAC=WFAC+.5
LL=L+KFAC

C INVESTIGATE KFAC SUCCESSING STOP LOCATIONS.
DO 8 K=LL,LL
IF(Y.GT.NN) GO TO 20
X=NX(K)
Y=NY(K)  
DIST1=HYFCT(X,Y,XL,YL)  
IF(DIST1.GT.DIST) GO TO A  
DTST=DIST1  
NPEST=K  
8 CONTINUE  
20 NCOUNT=NCOUNT+1  
MBUS(NCOUNT)=MP  
MSTOP(NCOUNT)=NPEST  
GO TO 1  
9 CONTINUE  
1 CONTINUE  
RETURN  
END

SUBROUTINE LOOP (NOX,NOY,II,JJ)  
COMMON /ALL1/ NPUSXY(20,60), NSTOP(20), NBUS(4,20),  
1 MSTOP(60), MBUS(20), NTIME(20,60), NFREO(20),  
2 NC(4), NDBUS(20), MDEPXT(20), NDBUS(20), MAVR(20),  
3 ITABV(30), ITDEPT(30), TTT(30), IBUS(30), NPN(60),  
4 ISDEPT(30), ISARRV(30), NPRT(30), FLG(2), WFAC,  
5 NDIV, NOAV1, NOAV2, NOAV3, NCT, NTT, NCT1  
C DETERMINE IF THE FINAL STOP LOCATION OF THE ROUTE  
C IS WITHIN AN ACCEPTABLE WALKING DISTANCE.  
NN=NSTOP(II)  
NNK=NN-JJ  
KNN=WFAC+1.  
IF(NNK.LE.KNN) GO TO 100
REDUCE STOP LOCATION NUMBER TO TWO NUMBERS.

\[ NLP = \text{NBRUSXY}(III,NN) / 1000000 \]
\[ NVX = (\text{NBRUSXY}(IT,NN) \cdot NLP \cdot 1000000) / 1000 \]
\[ NVY = \text{NBRUSXY}(II,NN) - (NVX \cdot 1000) \cdot (\text{NLP} \cdot 1000000) \]
\[ XL = \text{NOX} \quad \& \quad YL = \text{NOY} \]
\[ X = \text{NXX} \quad \& \quad Y = \text{NYY} \]
\[ \text{DIST} = \text{HYPOT}(X,Y,XL,YL) \]
\[ \text{IF} (\text{DIST} \leq \text{WEAC}) \quad JJ = \text{NN} \]

100 RETURN

END

SUBROUTINE TPNFER

COMMON /ALL1/ NBUSXY(20,50), NSTOP(20), NBRUS(4,20),
1 KSTOP(60), NBUS(20), NTIME(20,60), NFREQ(20),
2 NO(4), NBRUS(20), MDEPART(20), NBRUS(20), MARRV(20),
3 ITARRV(30), ITDEPT(30), IFAIL(30), IFAIL(30), IFAIL(30),
4 TSDEPT(30), ISARPA(30), NPRT(30), FLAG(2), WFAC,
5 NOAA, NOAV1, NOAV2, NOAV3, NCT, NIT, NXT,

COMMON /ALL2/ NDEPART, NRPX, NOX, NOY

DIMENSION IUSB(30), ITARRVT(20), ITDEPTT(20), NPN1(20),
1 NPN2(20), IDIFF(20), JTDEPTA(80), JTARRVA(80),
2 JDIF(80), NPNO(20), NPNQ(20), ITDEPTA(30), JTIT(80),
3 ITARRVA(80), JTARRV(80), JTDEPT(80), JSTAA(80),
4 JSTAP(80), ISARPA(30), ISDEPA(30), ITTA(30), JSTAP(80),
5 ITTP(30), JTTA(80), JTTR(80), IDIFF(30), JTARR(80),

C INVESTIGATE ALL POSSIBLE TWO-ROUTE COMBINATIONS.

DO 100 I=1,NDEPART
    TI=NBRUS(I) \& JD=MDEPART(I)
    JJ=JD
    ND=NSTOP(IT)
    TLOOP=1
    DO 100 J=1,NRPX
JJ=MARPUS(J)   &   JA=MARPV(J)
NA=NSTOP(JJ)
KFLAG=NBUSXY(JJ,JA)/1000000
CNTI=0
C
FIND ALL POINTS IN COMMON.
DO 91 TI=1,ND
IF(TI.EQ.JJ.AND.I1.NE.ND) GO TO 91
MD=NBUSXY(I1,I1)/1000000
MDXY=NBUSXY(I1,TI)-(MD*1000000)
DO 90 J1=1,NA
MA=NBUSXY(JJ,J1)/1000000
MAXY=NBUSXY(JJ,J1)-(MA*1000000)
IF(MDXY.NE.MAXY) GO TO 90
IF(TI.NE.J1) GO TO 90
JF=J1
ILOOP=1
2 JFLAG=NBUSXY(I1,JD)/1000000
C
OPTIMIZE ORIGIN POINT FOR TYPE 2 ROUTES AND
Determine legal transfers within loop.
IF(MD.EQ.1.AND.JFLAG.EQ.1.AND.T1.LT.JD) GO TO 91
IF(MA.EQ.1.AND.KFLAG.EQ.1.AND.J1.GT.JA) GO TO 91
IF(MD.GT.0.OR.JFLAG.NE.1.OR.ILOOP.EQ.2) GO TO 3
CALL LOOP (NOX,NOY,I1,JD)
ILOOP=2
C
Determine acceptable transfer points.
7 NT1=NTIME(I1,T1)/100
NTA2=NTIME(I1,I1)-(NT1*100)
NTD1=NTIME(JJ,J1)/100
NTD2=NTIME(JJ,J1)-(NTD1*100)
C
Determine direction of travel for departure route.
IF(TI-JD) F,10,10
NTA=NTA2
GO TO 11
10 NTA=NTA1
11  NGA=NFFEO(IT)
    NSPACE=60/NGA
12  DO 12  IJ=1,NGA
    NPN(IJ)=NSPACE*(IJ-1)
15  DO 16  JJJ=1,NGA
    ITAPPVT(JJJ)=NTA+NPM(JJJ)
16  IF(ITAPPVT(JJJ).GE.60) ITAPPVT(JJJ)=ITAPPVT(JJJ)-60
    DETERMINE DIRECTION OF TRAVEL FOR ARRIVAL ROUTE.
    TF(JA-J1) 17,20,20
17  NT0=NTO2
    GO TO 21
19  NTO=NTO1
21  NGO=NFFEO(JJ)
    NSPACE=60/NGO
22  DO 22  IJ=1,NGO
23  NPNI(IJ)=NSPACE*(IJ-1)
25  DO 26  JJJ=1,NGO
    TTUEPTT(JJJ)=NTO+NPNI(JJJ)
26  IF(TTUEPTT(JJJ).GE.60) ITUEPTT(JJJ)=ITUEPTT(JJJ)-60
    NCI=0
    DO 30  JK=1,NGA
            ITD=ITUEPTT(JK)
            IF(ITD.LE.ITARRVT(JK)) ITD=ITUEPTT(JK)+60
            NDIFF=ITD-ITARRVT(JK)
            SET ALLOWABLE LIMITS FOR TRANSFER/ARRIVAL TIMES.
            IF(NDIFF.GT.100.OR.NDIFF.EQ.0) GO TO 30
            NCI=NC1+1
30  CONTINUE
    IF(NCI.EQ.0) GO TO 91
    ASSIGN FREQUENCIES AND DEPARTURE/ARRIVAL TIMES.
NTD1 = NTIME (II, JD) / 100
NTD2 = NTIME (II, JD) - (NTC1 * 100)
NTA1 = NTIME (JJ, JA) / 100
NTA2 = NTIME (JJ, JA) - (NTA1 * 100)
TF (T1 - JD) = 76, 40, 40

36 NTD = NTC?
GO TO 3a

40 NTU = NTD1

78 DO 41 IJ = 1, NC1
  K = NPNI (IJ)
41 NPNI (IJ) = (60 / NFREQ (II)) * (K - 1)
43 TF (JA - JJ) = 57, 60, 60
57 NTA = NTA2
GO TO 58

60 NTA = NTA1

58 DO 59 IJ = 1, NC1
  K = NPNI (IJ)
59 NPNI (IJ) = (60 / NFREQ (JJ)) * (K - 1)

C CALCULATE TRAVEL TIMES.

63 DO 76 IJ = 1, NC1
  NCTI = NCTT + 1
  JDEPT (NCTI) = NTD + NPNO (IJ)
  IF (JDEPT (NCTI) .GE. 60) JDEPT (NCTI) = JDEPT (NCTI) - 60
  K = NPNI (IJ)
  JTARRVA (NCTI) = ITARRVT (K)
  JSTARRV (NCTI) = NTA + NFNA (IJ)
  IF (JSTARRV (NCTI) .GE. 60) JSTARRV (NCTI) = JSTARRV (NCTI) - 60
  K = NPNI (IJ)
  JDEFTA (NCTI) = JDEPTT (KK)
  JIFF (NCTI) = KDIFF (IJ)
  JSTA (NCTI) = I1
  JSTAB (NCTI) = .11
  JDEPT (NCTI) = 10

70 CONTINUE
CONTINUE

C SELECT TRANSFER POINT RESULTING IN MINIMUM TRAVEL TIME.
DO 95 JK=1,NCTI
ITAV1=JTARRVA(JK)
IF(JTDEPT(JK).GE.ITAV1) ITAV1=ITAV1+60
JTTR(JK)=ITAV1-JTDEPT(JK)
ITAV2=JTARRV(JK)
IF(JTDEPTA(JK).GE.ITAV2) ITAV2=ITAV2+60
JTTR(JK)=ITAV2-JTDEPTA(JK)
JTTR(JK)=JTTR(JK)+JTTP(JK)
JTTR(JK)=JTTR(JK)+JTTB(JK)+JDIFF(JK)
TF(JTTR(JK),GT,JTT(NBEST)) GO TO 95
TF(JTTR(JK),LT,JTTAB(NBEST).OR.JTT(JK).LT.JTT(NBEST)) BEST=JK
95 CONTINUE

C STORE INFORMATION ON ACCEPTABLE TWO-ROUTE COMBINATIONS.
DO 96 JK=1,NCTI
IF(JSTAA(JK).NE.JSTAA(NBEST)) GO TO 96
NCT=NCT+1
IDIFF(NCT)=JDIFF(JK)
ITPUS(NCT)=II
ITBPS(NCT)=JJ
ITDEPT(NCT)=JSDEPT(JK)
ISAPPA(NCT)=JSTAA(JK)
ITARRV(NCT)=JA "$ ISDEPA(NCT)=JSTAR(JK)
ITDEPT(NCT)=JTDEPT(JK)
JTARRVA(NCT)=JTARRVA(JK)
ITDEPTA(NCT)=JTDEPTA(JK)
ITARRV(NCT)=JTARRV(JK)
JTTR(NCT)=JTT(JK)
JTTR(NCT)=JTTA(JK) "$ JTTR(NCT)=JTTA(JK)
CONTINUE
100 CONTINUE
PHOUR=1.
IF(NCT.NE.0) GO TO 800
PRINT 102
102 FORMAT(/,16*NO BUS ROUTE IS AVAILABLE, EVEN *,
11*CONSIDERING A TRANSFER.*)
GO TO 800
ENTRY TIME

C
?4-HOUP CLOCK NOTATION FOR DESIRED ARRIVAL TIME ROUTE.
IF(TDPTA(NTT).GE.ITARRV(NTT).GP.NDAV2.EQ.60) NDAV3=NDAV3-1
NPTMPDA=(NDAV3*10)+TDPTA(NTT)
IF(NDAV2.EQ.60.AND.ITARRV(NTT).NE.0) NDAV1=NDAV1-1
NPTM1N=(NDAV1*100)+ITARRV(NTT)
IF(ITARRVA(NTT).GT.ITARRVA(NTT)) NDAV3=NDAV3-1
NPTMAD=(NDAV3*100)+ITDEPT(NTT)
IF(ITDEPT(NTT).GE.ITARRVA(NTT)) NDAV3=NDAV3-1
GO TO 500

C
ENTRY TIME

500 NHOUR=1
PASS=1.
I1=IXUS(NTT)
I2=TSDEPT(NTT)
I3=TSARP(NTT)

C
REDUCE STOP LOCATION NUMBER TO TWO NUMBERS.
505 NLP=NASUXY(I1,I2)/1000000
NXD=(NASUXY(I1,I2)-(NLP*1000000))/1000
NYD=NASUXY(I1,I2)-(NXD*1000000)+NLP*1000000)
NLP=NASUXY(I1,I3)/1000000
MXD=(NASUXY(I1,I3)-(NLP*1000000))/1000
NYA=NASUXY(I1,I3)-(MXD*1000000)+NLP*1000000)
IF(NDAV.GT.2400.OR.NCT1.EQ.0) GO TO 510
IF(PASS.EQ.2.) GO TO 506

C PRINT INFORMATION ON DESIRED ARRIVAL TIME ROUTE.
PRINT 720, I1, NXD, NYD, NXA, NYA, NPTIMD, NPTIMA,  
1 FLAG(NHOUR), ITTA(NTT)
720 FORMAT(3X, I2, 4X, 1H, (I3, 1H), T3, 3H)  (I3, 1H), I3, 4H)
   1I4, 7X, I4, A3, 2X, I3, 4H MIN)
PRINT 721, IDIFF(NTT)
721 FORMAT(13X, 1H WITH A TRANSFER (, I2,  
1 13H MIN WAIT) TO)
504 T1=TDUSP(NTT)
   T2=TSDEPA(NTT)
   I7=ISAFRV(NTT)
   PASS=2.
GO TO 505

506 PRINT 720, I1, NXD, NYD, NXA, NYA, NPTIMD, NPTIMA,  
1 FLAG(NHOUR), ITTA(NTT)
PRINT 722, ITT(NTT)
722 FORMAT(4X, 29H TOTAL TRAVEL TIME, INCLUDING,  
1 17H WAIT TIME = , I3, 4H MIN, //)
GO TO 520

510 IF(PASS.EQ.2.) GO TO 515

C PRINT INFORMATION ON MINIMUM TRAVEL TIME ROUTE.
IF(ITARRVA(NTT), LE, ITDEPT(NTT)) NHOUR=2
IF(NHOUR.EQ.2) MHOUP=2.
PRINT 720, I1, NXD, NYD, NXA, NYA, ITDEPT(NTT),  
1 ITARRVA(NTT), FLAG(NHOUR), ITTA(NTT)
PRINT 721, IDIFF(NTT)
GO TO 520

515 NHOUR=1 & MHOUP=1
IF(ITARRV(NTT), GT, ITDEPTA(NTT)) MHOUP=2
IF(ITARRV(NTT), LE, ITDEPTA(NTT)) NHOUR=2
IF(NHOUR.EQ.2, OR, MHOUP.EQ.2) MHOUP=2.
PRINT 723, I1, NXD, NYD, NXA, NYA, ITDEPTA(NTT),  
1 FLAG(MHOUR), ITARRV(NTT), FLAG(NHOUR), ITTA(NTT)
C
PRINT INFORMATION ON REMAINING ROUTES.

520 PRINT 730
730 FORMAT(18X,23H OTHER ROUTES AVAILABLE,/) 
   IF(NCT.EQ.1) GO TO 600
   DO 530 I=1,NCT
       NHOUR=1 & PASS=1.
   IF(I.EQ.NTT) GO TO 570
   I1=IBUS(I) & I2=ISDEPT(I)
   I3=ISAPRV(I)
   NLP=NAUSXY(I1,I2)/1000000
   NXD=(NPUSXY(I1,I2)-NLP*1000000)/1000
   NYD=NAUSXY(I1,I2)-((NXD*1000)+(NLP*1000000))
   NLP=NAUSXY(I1,I3)/1000000
   NYA=NAUSXY(I1,I3)-((NYD*1000)+(NLP*1000000))
   TF(PASS.EQ.2.) GO TO 532
   IF(ITAPRVA(I).LE.ITDEPT(I)) NHOUR=2
   IF(NHOUR.EQ.2) PHOUR=2.
   PRINT 721, I1,NX0,NYD,NXA,NYA,TTDEPT(I),
   1 ITAPRVA(I),FLAG(NHOUR),ITTA(I)
   PRINT 721, IDIFF(T)
   I1=IBUS(I)
   I2=ISDEPT(I)
   I3=ISAPRV(I)
   PASS=2.
   GO TO 531

532 NHOUR=1 & MHOUR=1
   TF(ITARRVA(I).GT.ITDEPTA(I)) MHOUR=2
   IF(ITAPRVA(I).LE.ITDEPTA(I)) NHOUR=2
   TF(NHOUR.EQ.2) OR(MHOUR.EQ.2) PHOUR=2.
PRINT 723, I1,NXD,NYO,NXa,NYA,ITDEPTA(I),
   1 FLAG(MHCPUP),ITACPRT(I),FLAG(NHOUR),ITTE(I)
PRINT 722, ITT(I)
570 CONTINUE
   GO TO 776
600 PRINT 779
770 FORMAT(26X,5H NONE,/)  
776 IF (PHCUP.EQ.1.) GO TO 830
PRINT 777
777 FORMAT(10X,3AH ** INDICATES MINUTES AFTER NEXT HOUR.)
860 RETURN
END

FUNCTION HYPOT(A,B,C,D)
   X=C-A
   Y=D-A
   HYPOT=SQRT(X**2+Y**2)
RETURN
END
LIST OF REFERENCES


