NBSIR 75-676

Path Finding Algorithms and Data Structures For Point-to-Point Trip Management

Judith F. Gilsinn
Patsy B. Saunders
Martin H. Pearl

Operations Research Section
Applied Mathematics Division
Institute for Basic Standards
Washington, D. C. 20234

January 1975

Final

Technical Report to
Systems Analysis and Evaluation Division
Urban Mass Transit Administration
Department of Transportation
Washington, D. C. 20590

NBSIR 75--676

PATH FINDING ALGORITHMS AND DATA STRUCTURES FOR POINT-TO-POINT TRIP MANAGEMENT

Judith F. Gilsinn
Patsy B. Saunders
Martin H. Pearl

Operations Research Section
Applied Mathematics Division
Institute for Basic Standards
Washington, D. C. 20234

January 1975

Final

Technical Report to
Systems Analysis and Evaluation Division
Urban Mass Transit Administration
Department of Transportation
Washington, D. C. 20590

U. S. DEPARTMENT OF COMMERCE, Secretary

NATIONAL BUREAU OF STANDARDS, Richard W. Roberts, Director

ABSTRACT

This report identifies and characterizes the data base and computer software requirements for a Point-to-Point Trip Management (PTPTM) System which provides detailed transit trip itineraries in response to inquiries made by prospective passengers. The requirements fall into four categories, corresponding to successive stages in processing such an inquiry: 1) reception and interpretation, 2) location and connection, 3) path calculation, and 4) report. Procedures for path calculation are discussed in detail, including techniques for improving shortest path algorithm performance both through optimized computational schemes and through special methods of representing and manipulating the data base describing transit routes and schedules. Estimates, based on step-by-step schemes presented in an appendix, indicate that computation time will be sufficiently small (less than one second per request) that on-line path computation is feasible. Since path finding represents only a fraction of the total time spent in answering a request for itinerary, a queuing model is developed to establish how many formerly lost calls would be captured by a computerized system which achieved a prescribed fractional saving in service time; illustrative application of this model indicates a sharp reduction in lost calls.

Key Words: Algorithms; networks; paths; shortest paths; transit information systems; transit networks; transportation.

ACKNOWLEDGEMENTS

We wish to thank Dr. Christoph Witzgall for suggesting the time-expanded network scheme, the bipartite route/stop scheme and the method of compactly representing path data given in Appendix D, and for his helpful comments on many other aspects of the work. We also wish to thank Dr. Carl Harris of George Washington University, who suggested initial references to queuing with reneging which led to reference [1] below.

TABLE OF CONTENTS

LIST OF FIGURES

page

1. INTRODUCTION

Point-to-Point Trip Management (PTPTM) is the capability of an urban public transportation system to provide individualized trip planning and trip execution information to a prospective rider, through use of a readily accessible communication medium such as the telephone. The two major elements of PTPTM are the communication system and the transit company's "PTPTM Processor". This report discusses the various components of the PTPTM Processor.

The idea of providing transit trip itineraries in response to telephone requests is certainly not new. This service has been furnished in many localities for many years by transit system telephone operators who scan city maps, transit network maps, transit schedules, and update sheets in order to identify a trip which will serve a caller's needs. The idea which is somewhat new is that of transforming the various maps and schedule information into computerized data bases, automating the look-up procedures, and utilizing a computerized path finding process to single out a "best" trip or trips for each caller.

The current Energy Crisis has provided an impetus to developing automated transit information systems. With gasoline prices increasing and supplies decreasing, there has been a renewed interest in efficient utilization of transit systems. Some people are using mass transit for the first time or using it for more types of trips, e.g., commuting, shopping, visiting, dining out, etc. They need information in order to plan their trips, and it is plausible that even more people would utilize public transportation if trip information were more readily available. An additional impetus for transit information systems comes from the awakening interest in regional cooperative efforts to improve the quality of life for all, efforts in which coordinated approaches to transportation play a major role. Providing transit trip information on a regional basis, it is hoped, will improve all citizens' access to facilities throughout the area.

In a manual system, for example that provided by Washington D.C.'s Metrobus Company, a call for transit information can take up to 7 to 8 minutes to serve. Some people are no doubt discouraged from calling again by the length of calls, or by the resultant waits for contact with an operator. Furthermore, even when a large number of operators are answering calls, when all of those operators are busy only a certain number of additional calls can be placed in a queue on "hold", and once the queue is filled all other calls are lost to the system. Automating the service processes will significantly reduce the average time per call, thereby providing faster service on all calls and capturing at least some of the calls which are now lost.

In addition to serving more calls faster, there are several other benefits from an automated transit information system. Under the manual system operators search for a trip which meets the caller's needs, i.e. carries the caller from a transit stop near his origin to a transit stop near the desired destination, arriving and/or departing at times close to those preferred by the caller. Although the reported trip itineraries are reasonable, they are not necessarily the "best" trip for each caller. Procedures for finding "best" paths in a network, often referred to as "shortest path algorithms", can utilize any one of many different criteria for selecting an optimal path for each caller. (Possible criteria for path selection will be discussed later in the report.) Furthermore, because the algorithms are well-defined procedures, there is the added advantage of repeatability, i.e. for the same input the algorithms always select the same path, which means that a caller will receive the same response no matter when he calls or which operator handles his inquiry. In a manual system this is not necessarily true: two operators, or even one operator answering several calls, given the same information requests, may not come up with the same trip itinerary. In addition to optimality and repeatability, another advantage of an automated system is its ability to incorporate changes promptly. If, for example, a schedule is changed or a transit route altered, the data bases can be modified to reflect the changes and the PTPTM Processor will respond immediately. These advantages of an automated transit information system make it clear that such a system warrants closer investigation and analysis of its costs and benefits.

We will describe the components of a PTPTM Processor in Section 2 and discuss for each its requirements and characteristics under various levels of automation. Section 2 also contains descriptions of the data base and software requirements of each component together with an assessment of anticipated problem areas in Processor development. Section 3 focuses on the path finding component, describing computational procedures to aid in finding shortest paths in a transit network, detailing the related data structures and data processing requirements and providing timing estimates for path computation. Section 4 summarizes the findings of Sections 2 and 3 and recommends next steps.

The report also contains four appendices. The first of these includes an examination, for possible use in PTPTM, of that part of UMTA's Urban Transportation Planning System (UTPS) which finds paths in a transit network. Appendix B presents a queuing model of a PTPTM information system and an illustrative application of that model to investigate how significantly the number of lost calls would be affected by a reduction in service time resulting from automation of the path finding component. Appendix C takes up several topics, peripheral to the main thrust of this report, which nevertheless are relevant to PTPTM and were noted during our study. The final appendix gives the details of step-by-step computational schemes for path finding in a transit network.

2. COMPONENTS OF A PTPTM PROCESSOR

Some of the functions which are now carried out by the transit system information operators will be carried out by the PTPTM Processor. These functions and the associated sources of information, i.e. data bases, can be thought of as components of four major functional categories. The diagram shown in Figure 2.1 illustrates one conceptualization of the system. Although the PTPTM Processor may, in final form, differ somewhat from the diagram, the functions and data bases shown therein will necessarily be ingredients of an operational system. This section will discuss the various functions and data bases associated with each of the components.

2.1 Reception and Interpretation

The functions of this category are now performed somewhat "automatically", perhaps even subconsciously in some cases, by the transit system information operator. The caller provides an origin, a destination, and information indicating a time preference for the trip, e.g. weekday morning to arrive by 0830, or Saturday evening departing no sooner than 1700.* The operator must "receive and interpret" the caller's information. A typical conversation is subject to several basic types of communication difficulty, e.g. mispronunciation, accent variations, and multiple names for the same location. While operators handle these difficulties readily, asking for clarification when necessary, considerable effort would be required to develop a computer processor with comparable interpretation capabilities.

2.1.1 FUNCTIONAL REQUIREMENTS

The PTPTM Processor will presumably be developed and implemented in stages, and reception and interpretation may be the last stage to be automated. In fact, the information requested from callers may differ depending upon whether this part of the PTPTM Processor is or is not fully automated. In a partially automated version operators would still answer the phones, request information from the callers, and recognize and/or clarify pronunciation ambiguities. The operator would then enter this information into an automated location and connection component. Such a component must be capable of recognizing and treating some of the communication difficulties now handled by the operators as well as new problems introduced by the operators' entering the information. The following is a list of some of the complications with which this component must cope:

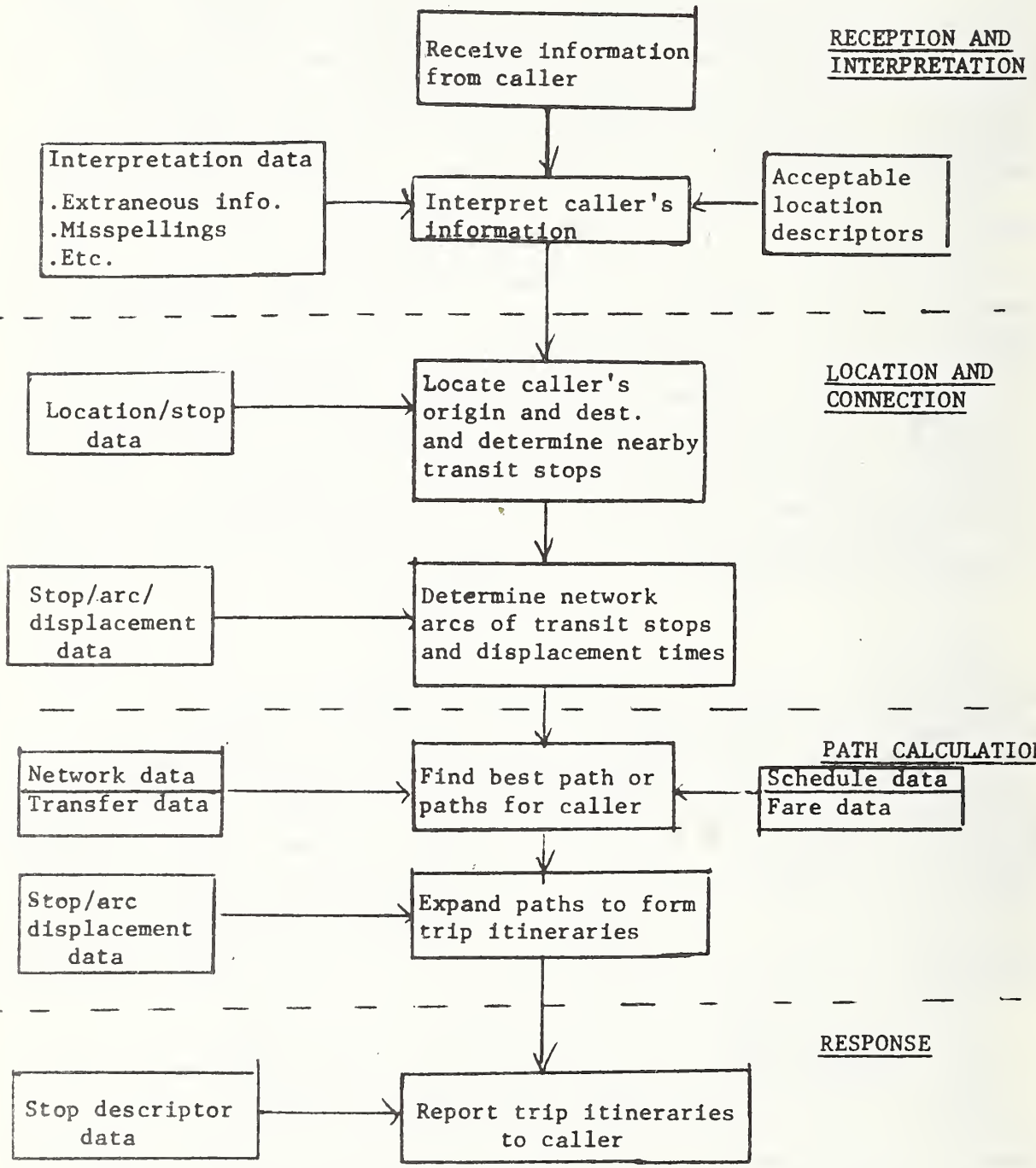
(i) multiple names for the same location, e.g. The Kennedy Center or the John F. Kennedy Center for the Performing Arts, Landover Mall or Landover Shopping Center, etc.

---

*Although military time will be used throughout this section, callers would not necessarily be required to give their time preferences in this manner.

FIGURE 2.1

PTPTM Processor Components



(ii) reordering of words, e.g. Commerce Department or Department of Commerce, Maryland University or University of Maryland, etc.
(iii) abbreviations, e.g., S., St., St, for Street, Blvd. Bvd., or B for Boulevard, etc.
(iv) misspellings, e.g., Glenn Dale Rd. or Glen Dale Rd., Chaplain St. or Chaplin St., etc.

Many difficulties of the types mentioned above can be anticipated and allowed for in advance.

After "recognizing" a communication difficulty, the location component would then indicate each correction that has been made. If ambiguities remain, it may print out a list of names from which the operator, on behalf of the caller, can select the correct choice or reenter the system with more precise information. Methods for detecting, correcting, and verifying input errors have been developed*, and these general techniques, as well as some of their details, can be utilized in the PTPTM Processor.

In a fully automated system, i.e. one in which there is operator assistance only when a problem develops, it will probably not be feasible to allow callers their present great freedom in formulating requests. Although some work is being done to develop "voice decoders", i.e. systems which translate spoken words into machine readable input, such devices are far from perfected. Variations in accent, enunciation, tone of voice, etc. create great difficulties. Present technology therefore requires efforts to minimize these variations by restricting the words to be decoded. The capability of decoding spoken numbers is far more promising than that of decoding arbitrary words. Thus, a fully automated PTPTM Processor may require that a caller supply the telephone numbers of the origin and destination locations. A more restrictive but technologically simpler approach would be to have the telephone numbers relayed not by voice but by the caller dialing (or pushing buttons on a "Touch-Tone" phone) the origin and destination telephone numbers and desired time of arrival or departure in response to appropriate recorded requests by the PTPTM Processor.

Because of the desirability of developing the PTPTM Processor in stages such that system changes are minimized as each new stage is implemented, there may be a tendency to suggest that callers initially state the origin and destination phone numbers to operators who then enter these numbers into the system. Although this process would minimize the communication difficulties described above, there are many problems which would be introduced, including those that follow:

(i) An extra burden is placed on callers who may not know the correct phone numbers and would have to obtain them from some source.
(ii) It may be difficult or impossible to obtain some phone numbers, e.g. residences with unlisted numbers, locations served by unlisted pay telephones, etc.
(iii) There are many instances where there is not a one-to-one correspondence between location and phone number. Obviously there are apartment buildings, office buildings, etc. where

*See description of the PARIS system appearing below and in [3].

there are many phone numbers for the one location. Not so immediately obvious are situations where a single phone number represents numerous locations. This occurs most frequently with businesses that have one central switchboard where an operator answers all calls and transfers calls to branch offices which have extensions of the central switchboard telephone number.

Because of the difficulties described above, it is recommended that telephone numbers not be used as origin/destination descriptors until such time as this is required for use in a fully automated system. At that time, steps can be taken to overcome some of these difficulties and the increased burden on the callers can be weighed more meaningfully against the advantages of a fully automated system.

2.1.2 DATA BASES

There are several types of data which will be required to assist in the reception and interpretation of caller supplied input. First, there must be a list of immediately acceptable location descriptors. If the callers are specifying street names, either as explicit location addresses, e.g. "223 W. Elm Street," or as nearby street corners, e.g. "Elm and Oak", the list will include all street names in the area served by the transit system. If landmarks are to be accepted, their descriptors must also be included, e.g. "Washington Monument" or "Korvette's Lot - Rockville." It will be the function of the interpretation component to determine which of the descriptors on the acceptable list is intended by the caller/operator supplied input.

Extraneous information and some abbreviations can be removed from the input by checking another data list. Such a list might include such words as "the" "in", "of", "West", "Street", etc. with each word on the list associated with a group of letters to which the words on the list should be changed. For example, "the", "in", "of", etc. might be changed to a single blank space, while "West" or "Street" might be changed to "W" and "St" respectively. This list would include fairly general changes which would not vary significantly from city to city. After the input has been checked and, if necessary, extraneous information removed, the (possibly modified) input can be checked against the list of acceptable descriptors.

If the input information is found within the list of acceptable descriptors, then the input can be passed along to the second component of the PTPTM Processor. If it is not found, additional checks will be needed. Multiple names and reorderings are probably best handled by anticipation, experience, and rejection. Some mismatches of this type can be anticipated and the multiple names or reordered phrases can be included in the acceptable list. When the system is in use, the acceptable list can be augmented to include any descriptors which frequently have resulted in mismatches. Finally, there will be cases where the caller coins his own descriptor and the system will reject the input and ask for clarification.

Another source of mismatches will be spelling errors. Some of these can be "caught" by using a list of multiple spellings of the same sound. Such a list would, for example, include the pairs, "f" and "ph", "ay" and "ey", "ay" and "eigh", "k" and "c", "c" and "s". A system for correcting spelling errors based on sounds is in use in the PARIS system which will be discussed later in this report. If, after checking for spelling errors, the input still cannot be located on the acceptable list, it must be rejected for further clarification from the user/operator.

To some extent there is a general philosophy which can be followed in developing this component of the PTPTM Processor. Obviously, it would be undesirable for the system to reject a large number of input descriptors. Yet at the same time, to provide the system with the capability of "recognizing" almost any conceivable input would require massive data bases and an extensive amount of time in modifying and checking the input. The best approach is somewhere between the two extremes, i.e. provide enough interpretation capability so that "most" input is acceptable, and periodically monitor the inputs which are rejected so that frequent mismatches can be avoided by including the non-acceptable descriptors in the acceptable list.

A second general point is that great care must be taken to avoid changing the input into the wrong descriptor. For example, "Elm Street" and "Elm Avenue" may or may not refer to the same place. If both exist in the area served by the transit system, the distinction must be maintained. If only the second exists, it could be safe to assume that a caller's reference to "Elm Avenue" really meant "Elm Street". To help avoid making the wrong changes, ambiguities can be printed out so that the user/operator can make the appropriate determination.

2.2 Location and Connection

After the caller's origin and destination have been clearly identified by the Reception and Interpretation component, they must then be located. This process corresponds to the operators, in the manual system, searching city maps to find these locations and then scanning the transit system maps to locate nearby transit stops. This component of the PTPTM processor must perform the same functions: given a caller's origin and destination descriptors, it must determine appropriate transit stops, i.e. points of access and egress for the caller, and provide all necessary information in a form acceptable to the path finding component.

2.2.1 FUNCTIONAL REQUIREMENTS

In carrying out the location and connection processes, this component of the PTPTM Processor will be primarily concerned with scanning various data bases to obtain the appropriate information. Considerable effort will be required to prepare these data bases and to organize their storage in such a way that the searches can be accomplished with a minimum of scanning. The choice of storage schemes will depend on both the path finding algorithm and the particular transit network being modeled. In the following sections which discuss the necessary data bases, the descriptions and examples are

meant to illustrate the general nature of the information requirements rather than a particular storage plan.

2.2.2 LOCATION/TRANSIT STOP DATA

One of the major efforts in developing a PTPTM Processor is that of providing the capability to locate a caller's origin and destination and to determine appropriate transit stops near both of these locations. It would, of course, greatly simplify the system if callers were required to specify these transit stops. Such a requirement would, however, place an undue burden on callers who are unfamiliar with, or unable readily to describe, the locations of appropriate transit stops. Another alternative is to have the transit system information operators continue to determine appropriate transit stops manually, i.e. by scanning maps, and then "punch in" the stop numbers associated with these locations. It is, in fact, quite likely that such a partially automated system will be used for some time during the development of a more advanced PTPTM Processor. However, a location data base would be required in order to achieve a fully automated system.

The assumption that an operator assisted location system will be needed for some time during the development of the total system is based in part upon the present lack of location data in most municipalities. It is anticipated that the DIME* files presently being developed by many local jurisdictions in cooperation with the Census Bureau will aid in locating any given address within the boundaries of the municipality. These data are expected to aid various public service agencies, e.g. fire departments, ambulances, etc., in rapidly locating addresses in need of emergency services. If useful for this purpose, they could be of similar use in locating the origins, destinations, and transit stops needed by the PTPTM Processor. Unfortunately, since the DIME files are still under development for most large cities, it is impossible to evaluate them regarding accuracy, completeness, etc. As a result, there is a decision to be made between (a) developing the same type of data base and possibly duplicating the DIME effort and (b) waiting until the DIME files are available and hoping that they are sufficiently accurate and complete for PTPTM purposes (or can be made so without excessive effort). For purposes of this report, the discussion will specify the nature of the information which must be available, regardless of its source.

Given a caller's origin and destination, the location component must respond with two items corresponding to transit stops "near" the caller's origin and destination. The DIME files are expected to provide x and y grid coordinates for locations within the municipality. Using this type of data, an off-line computer code can be executed to produce a data base consisting of a set of nearby transit stops for "each" location. In actuality it will

---

*Further information about the DIME (Dual Independent Map Encoding) files may be obtained from Documentation for the Geographic Base (DIME) File, U.S. Bureau of the Census, Geography Division, revised August 1, 1974.

not be necessary to have an entry for every location, but only for certain general location descriptions, e.g. "200-400 Elm Street use stops 2, 3, or 17." Specific details on the storage of these data will depend on the transit network and the city layout, since these will determine the extent to which many locations can be grouped together for purposes of determining appropriate transit stops. If total trip time is to include time spent traveling to and from these transit stops, estimates of average access/egress times to and from these transit stops will also be required. Thus the total information which must be provided by the location/transit stop data base will include the following:

(i) location descriptions
(ii) lists of transit stops
(iii) estimated access/egress times.

2.2.3 TRANSIT STOP/ARC DATA

Including all transit system stops as nodes in the network on which the shortest path algorithm operates would unnecessarily increase the computation time and storage requirements in this part of the processor. Just a little reflection makes obvious the reason that all stops are not needed for the algorithm. Consider for the moment a representation of the entire transit system network in which every transit stop is a node and a directed arc joins every pair of nodes (i,j) if a transit route stops at node i just prior to stopping at node j. Augment this (abstract) network by adding additional directed (transfer) arcs joining every pair of nodes (k,ℓ) if passengers are likely to disembark at node k and walk to node ℓ to board another vehicle. The resulting diagram is then representative of the detailed transit network.

A simple example is given in Figure 2.2. It contains a total of ten transit routes, 50 arcs, and 20 stops or nodes. The dashed arcs connecting nodes 7 and 9 represent arcs over which passengers walk to transfer between routes E or F and routes G, H, I, or J. The table at the bottom of the figure specifies the stops for each route. It should be noted from the figure that an arc from node i to node j is not the same as an arc from node j to node i, i.e. all arcs are directed arcs.

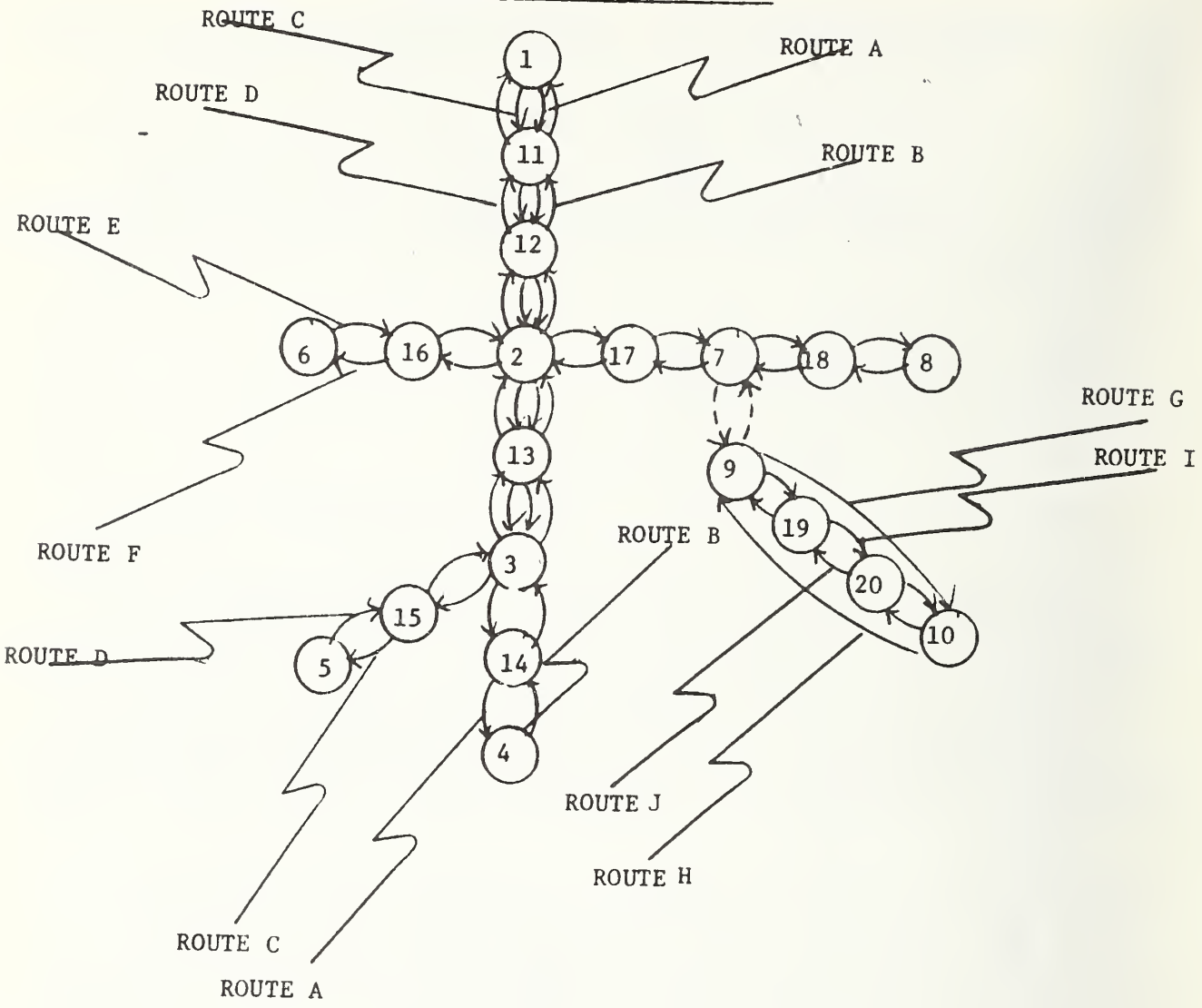
We will next illustrate how the example network can be reduced in size. We first define a note to be a "major" node if it has one or more of the following characteristics:

(i) beginning of a transit route,
(ii) end of a transit route,
(iii) beginning of a transfer arc,
(iv) end of a transfer arc,
(v) transfer point
(vi) certain other "important" transit stops.

All of the above characteristics are somewhat self-explanatory with the possible exceptions of (v) and (vi). A transfer point is defined as any stop where passengers might logically disembark and board another vehicle

FIGURE 2.2

A Detailed Transit Network



ROUTE	TRANSIT STOPS
A	1,11,12,2,13,3,14,4
B	4,14,3,13,2,12,11,1
C	1,11,12,2,13,3,15,5
D	5,15,3,13,2,12,11,1
E	6,16,2,17,7,18,8
F	8,18,7,17,2,16,6
G	9,10
H	10,9
I	9,19,20,10
J	10,20,19,9

to continue a trip. If two routes have a common segment containing several nodes, it is not necessary that all common nodes be treated as "major" nodes even though all are potential transfer points. Only those nodes meeting one of the other criteria, (i) through (vi), and usually the first and last nodes of the common segment, need be included as major nodes. In Figure 2.2, Node 2 is a transfer point because, for example, passengers on Routes E or F can transfer there to routes A, B, C, or D. Other transit stops may be termed "important" if, for example, an unusually large number of callers might originate or terminate rides at that stop. Furthermore, if fares are to be reported for a zone fare transit system, transit stops just across a zone boundary will be important and must be included in the reduced network.

In the reduced network, there will be a directed arc from major node i to major node j for every route in the detailed network such that node i is the last major node stop prior to node j on that route. All transfer arcs in the detailed network must also be included in the reduced network.

Figure 2.3 illustrates the reduced network corresponding to the detailed network shown in Figure 2.2. In this example, the reduced network includes only 10 nodes and 4 arcs. This type of reduced network is all that is needed by the shortest path algorithm for purposes of determining best paths between any pair of transit stops in the detailed network.

The transit stop/arc data will provide the information necessary to derive paths in the detailed network from paths in the reduced network. These data will consist of a set of arc numbers for each transit stop, including major node transit stops, in the complete network. Since each arc in the reduced network is associated with a portion of a route in the complete network, a transit stop will be associated with an arc if, in the complete network, that stop is on that portion of the route to which the arc corresponds. This seemingly complex explanation can be easily clarified by an example. Referring back to Figure 2.2 it can be seen that stop 16 is between stops 6 and 2 on both routes E and F. In the reduced network, arc 13 represents that portion of route E from stop 6 to stop 2 and arc 18 represents that portion of route F from stop 2 to stop 6. Thus stop 16 would be associated with arcs 13 and 18. The complete set of stops and their associated arcs for the example network are given in Figure 2.4.

2.2.4 TIME DISPLACEMENT DATA

In addition to a set of arc numbers for each transit stop, there must be available a set of times associated with the arcs for each transit stop. They can be thought of as "displacement times" since they indicate by how much time a particular node is displaced from a major node on each arc. Again, an example will clarify this explanation. If in the network of Figure 2.2, stop 16 is 2 minutes from stop 6 on route E, then in Figure 2.3 stop 16 is displaced by 2 minutes along arc 13. Thus if the schedule of departures at stop 6 along route E is known, it will be easy to determine when the vehicles will be at stop 16. If the displacement times are not constant

FIGURE 2.3

Reduced Network From Figure 2.2

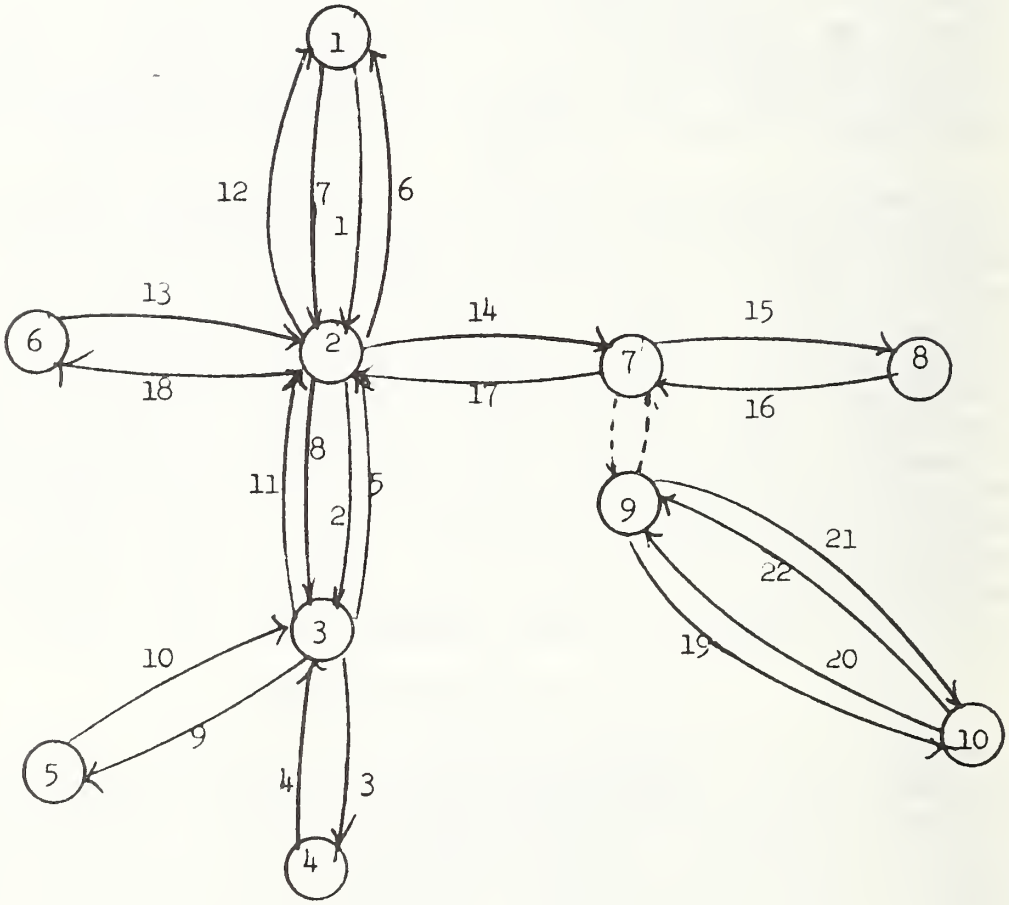


FIGURE 2.4

Stop/Arc Data From Figure 2.3

Transit Stop	Arcs
1	1,6,7,12
2	1,2,5,6,7,8,11,12,13,14,17,18
3	2,3,4,5,8,9,10,11
4	3,4
5	9,10
6	13,18
7	14,15,16,17,23,24
8	15,16
9	19,20,21,22,23,24
10	19,20,21,22
11	1,6,7,12
12	1,6,7,12
13	2,5,8,11
14	3,4
15	9,10
16	13,18
17	14,17
18	15,16
19	21,22
20	21,22

throughout the day, the displacements could be represented as fractions of total arc time, e.g. the trip from node 6 to node 16 might consume 35% of its total time in going from node 6 to node 2 on arc 13.

2.2.5 LOCATION/CONNECTION: SUMMARY

As discussed previously, the location and connection component of the PTPTM Processor will provide the information necessary to associate the caller's origin and destination with appropriate stops in the transit system network. It is the interface between the caller's request and the path finding component, and its development will require a more extensive software effort than that for any other component of the PTPTM Processor. The developmental process will require obtaining the needed information in a machine readable form, processing this information to obtain the basic data, "debugging" the data to insure accuracy and completeness, and developing clever computer storage arrangements to minimize the time needed to scan the data bases in search of the particular data needed to reply to an individual caller request. All of these tasks are major efforts.

In summary then, the Location/Connection component will, given as input a caller's origin and destination, first determine nearby transit stops and then produce, as output, sets of arc numbers and displacement times associated with these transit stops. These arc numbers are then used as input to the path finding component to define the corresponding origin and destination of the reduced network in which path finding is done. The displacement times are added or subtracted from the total trip times calculated by the path finding algorithm.

2.3 Path Finding Component

There are many "shortest path algorithms" which have been developed and are in current use. Section 3 and Appendix D of this report will describe several of these algorithms in detail. This section will concentrate on more general descriptions of the concepts and data of path finding processes.

2.3.1 PATH SELECTION CRITERIA

Although path finding techniques have traditionally been referred to as "shortest" path algorithms, it need not be the actual (distance-based) length of a path which is the basis for selection. Indeed, almost any characteristics of the various arcs and nodes which comprise a path can be combined to determine a value for the path, and the values for other paths can be compared to it to determine which path has the "best value." One frequently used criterion is time to traverse a path - total elapsed trip time. Time is of course related to path length, but in a transit system other factors, e.g. congestion, number of stops, etc., add additional time which is not necessarily a function of the length of the path. In fact, total trip time is often calculated as a function of several components, e.g. access and egress time, in-transit time, and transfer time. These components have been defined in a number of different ways and frequently are multiplied by different numerical "weights" prior to

being added together. The weights are indicative of relative preferences for the components, e.g. if in-transit time is less irritating than waiting time, the weight associated with waiting time is larger than that for in-transit time. The weights are necessary to distinguish between trips which, for example, have the same total elapsed time but differ in the distribution of this time among the various components. Such a situation might arise if a caller has the choice of walking six blocks to board an express vehicle or walking half a block to board a "stop at every corner" vehicle. Obviously the value of these two choices may differ from caller to caller and even from day to day for the same caller. Weights on the individual components of elapsed time would distinguish between the two trips whose unweighted total elapsed time is the same.

Although total trip time is probably the most frequently used value for determining best paths, there are many other criteria to be considered. In planning some trips, callers may be more concerned with departure and/or arrival time than with total elapsed time. For example, commuters may be concerned with arriving at their place of business just prior to a certain a.m. time and leaving after work shortly after a certain p.m. time. If using the routes associated with the shortest trip times required a.m. arrivals and/or p.m. departures quite different from the desired arrival and/or departure times, then routes with longer trip times might be preferable. In other words, there is an (unspecified) trade-off in the minds of these riders between total trip time and deviation from desired departure and/or arrival time. Furthermore, deviations on either side of a desired time may not be equivalent, e.g. a rider may not object to arriving at his place of business 20 minutes early, but could not even consider a trip which arrives one minute after the start of business. There are ways of defining trip value so that preferred departure and/or arrival times are included.

Still other factors which may be of importance in determining trip value include number of transfers, total fare, amount of walking required, etc. The following list, which is by no means all-inclusive, suggests some possible criteria upon which path selection can be based.

(1) Minimum time (weighted or unweighted),
(2) Minimum cost,
(3) Minimum out-of-vehicle time,
(4) Arrive closest to desired arrival time,
(5) Arrive latest before desired arrival time,
(6) Leave closest to desired departure time,
(7) Leave soonest after desired departure time,
(8) Minimum number of transfers,
(9) Minimum walking distance,
(10) Depart as late as possible to arrive prior to a specified time,
(11) Depart after a specified time and arrive as early as possible,
(12) Minimum time with:
    (a) No more than n transfers
    (b) No more than t minutes out-of-vehicle time
    (c) Cost less than d dollars or cents

Any of the criteria listed above, and perhaps even others which are not listed, could most nearly meet the needs of some particular caller. It will be necessary to decide in advance which set of criteria will be available in the PTPTM Processor. Although it is conceptually possible to allow each caller to select, from a group of criteria, the one(s) which describe his preferences, this degree of flexibility could be costly in terms of both computer storage requirements and computational time. Furthermore, although the various shortest path algorithms are capable of working with each of the criteria listed above, some of the criteria lend themselves to this more naturally, resulting in more efficient computation and storage. In particular, the criteria listed above as numbers 10 and 11 closely resemble those criteria now used in manual systems and are especially well-suited for several algorithms. In addition, it will be shown in Section 3 that these two criteria are so closely related that very little additional effort is required to allow the caller the option of choosing among them. It can be seen that both of these criteria will result in the selection of a minimum time trip subject to the stated limit on arrival or departure. The distinction is simply whether the caller wants to be sure of arriving prior to a certain time or whether he cannot depart until after a certain time. A simple input code such as punching a "D" or an "A" along with the caller's specified time can be used to indicate which of the two criteria is to be used. Use of these criteria will be discussed in greater detail in Section 3.

2.3.2 REDUCED NETWORK DATA

The type of data needed to describe the reduced network has been, to some extent, already described in connection with the location/connection data bases. In fact, the reduced network data themselves constitute the information needed to construct the reduced network from the detailed network. They can be organized such that for every arc in the reduced network there is a route number and two node numbers corresponding to the major node transit stops at the beginning and end of the arc. The reduced network data for the example network of Figures 2.2 and 2.3 are given in Figure 2.5. In this example, letters rather than numbers are used to identify the routes. The dashes as route identifiers for arcs 23 and 24 simply indicate that these are transfer arcs, which are treated differently from transit route arcs. These data completely describe the reduced network.

2.3.3 SCHEDULE DATA

Although describing the reduced network is fairly simple and straightforward, the description of schedules on the network is more complex. There are several alternative approaches, and the choice among them has important consequences. Before discussing these alternatives, some general comments can be made regarding the information to be provided by the schedule data.

As the name implies, the schedule data base will be used to determine when vehicles serving the various transit routes (arcs) will be at a particular transit stop (major node) to load and discharge passengers. Since each arc is associated with a particular route and particular origin and destination major nodes, the schedule data can be organized by arc. Thus for each arc

FIGURE 2.5

Reduced Network Data From Figure 2.3

ARC	ROUTE	MAJOR NODES
1	A	1-2
2	A	2-3
3	A	3-4
4	B	4-3
5	B	3-2
6	B	2-1
7	C	1-2
8	C	2-3
9	C	3-5
10	D	5-3
11	D	3-2
12	D	2-1
13	E	6-2
14	E	2-7
15	E	7-8
16	F	8-7
17	F	7-2
18	F	2-6
19	G	9-10
20	H	10-9
21	I	9-10
22	J	10-9
23	-	7-9
24	-	9-7

*The - in the column for route denotes a transfer arc with no route identification.

included in the reduced network there will be a set of data describing actual trips across that arc. These data can be either explicit schedules, i.e. a list of times when vehicles start to traverse the arc, or average headways, i.e. a single time interval which represents the average headway between successive vehicles serving the arc. Which of these two alternatives is selected affects both the computer storage requirements and the information which can be provided to the caller.

The difference between computer storage requirements for explicit schedules and those for average headways is rather obvious. Explicit schedule representations require one time for every daily trip across each arc in the reduced network. Average headways require only a single time increment, e.g. 15 minutes, for each arc in the reduced network. If the headways differ significantly throughout the day, it may be necessary to divide the day into time periods and specify different headways for each time period. Even in this case, there would probably not be more than 5 time periods per day so that far less storage is required to provide schedules as average headways. With either average headways or explicit times, schedule data will be required for different types of day, e.g. weekdays, Saturdays, Sundays, and holidays.

The most significant difference between average headways and explicit times is the type of information which can be provided to the callers. This difference is best explained by describing the manner in which average headway data are prepared. Published schedules for transit systems usually include a list of explicit times for each route which state when vehicles will arrive at the various stops along the route. These times would be used exactly "as is" for the explicit schedule data. However, if average headways are to be used, the average headway data must be derived from the explicit times in the published schedules. In the derivation process specific times are "lost" and all that remains are approximations of these times. As an example, consider the following two departure schedules during the time period stipulated as 0700 to 0900.

Departure No.	Route 1	Route 2
1	0710	0700
2	0725	0715
3	0740	0815
4	0755	0825
5	0810	0835

Both routes have five departures during the 120 minute time period, i.e. average headways of 24 minutes. Yet the two sets of explicit schedules are obviously quite different. For Route 1, describing the average headway as 15 minutes would be accurate, except that there is no way to indicate that there are no departures on this route after 0810. For Route 2, the departures are irregularly spaced. Except for the periods 0715 to 0815 and 0835 to 0900, departures are rather frequent. With the average headway representation of this schedule, there is no way to convey the variation in headways along Route 2.

In the above example, the departures on Route 1 are regularly spaced, but they do not continue throughout the entire time period. On Route 2, however, the situation is just the opposite: departures are irregularly spaced but do, to a greater extent, "cover" the time period. A third situation in which average headways are inadequate is that of very infrequent departures, e.g. one or two trips per day. In these cases, specifying an average headway of 24 hours, 12 hours, or even one or two hours is, for obvious reasons, totally inadequate information for a caller. Thus average headways can be appropriate only for frequent, regularly spaced departures which occur throughout the duration of a time period. It may be possible to utilize a mixture of schedule data, i.e. average headways where they would provide sufficient information and explicit schedules everywhere else. This arrangement would save computer storage, but the complexities associated with recognizing and utilizing two different types of schedule data may outweigh the reduction in storage.

2.3.4 TRANSFER DATA

When passengers transfer from a vehicle serving one route to a vehicle serving another, some amount of time is consumed just in making the transfer. In constructing both the detailed transit network and the reduced network, transfer arcs were added to cover the situation where it was necessary for a traveler to walk from one node to another to make the transfer. Each of these transfer arcs must be associated with a minimum time to traverse the arc, and "trips" across the arc can be assumed to begin when a vehicle arrives at the origin of the transfer arc.

In addition to these transfer arcs, it may be necessary to include certain other transfer data to cover the situation where the passenger would not have to walk between nodes to make a transfer, i.e. when two or more routes intersect. In this case a certain amount of time must be allowed for the traveler to get off and on the vehicles. If this transfer time is fairly constant at all nodes, the time can simply be added to total trip time whenever a transfer occurs. On the other hand, if transfer time varies from node to node, then of course the transfer time must be stored for each node.

There are actually two types of transfers which can occur at a single node. In one case passengers disembark and wait at exactly the same spot to board the next vehicle. In the other, they have to cross a street to board the next vehicle. If the nodes of the reduced network correspond to street intersections at locations where transferring passengers would have to cross the street before boarding, it may be necessary to split the node, i.e. have two different nodes for the one intersection with transfer links joining these nodes. Although the difference between remaining at the same stop and crossing the street to another stop at the same intersection may seem negligible, when connections are closely timed, crossing the street might add just enough additional time that the transfer cannot be completed. The need for splitting nodes can be determined by examining schedules at transfer points to see how closely timed the connections are.

If vehicles are instructed to wait at transfer points until transferring passengers arrive, then these additional same-intersection nodes and transfer links would not be needed and the transfer time can simply be the difference between the arrival of the first vehicle at the node and the departure of the second.

2.3.5 FARE DATA

Except in transit systems where there is a single fare for all trips, if fares are to be reported to the callers and/or included in the criteria determining best trip, some type of fare data will be needed. If the transit system employs a zone fare structure, there are at least two ways of organizing these data. With such a structure, there is usually a fixed rate plus additional charges each time the passenger crosses into a new zone. Each crossing into a new zone will correspond to an arc in the reduced network. These arcs can be flagged so that whenever they are included in a path, additional charges are added. Furthermore, callers with origins or destinations along these arcs could be advised that the additional charges can be avoided by using other transit stops. Alternatively, all transit stops could be numbered in such a way as to indicate their zone, e.g. stops 1-200 are zone 1, stops 201-378 are zone 2, etc. Then if the origin stop and the destination stop are in different zones, the complete path can be checked to determine the number of zone changes involved in making the trip. If the zone changing arcs were also flagged, it would again be possible to advise callers how to avoid the additional charges. This approach would require less checking for zone changes since the path arcs would only be checked if the origin stop and destination stop are in different zones. However, it does require special numbering of the transit stops, whereas some other numbering scheme, e.g. numbering all major node stops before numbering the other transit stops, may be more advantageous for other data bases or for certain algorithms.

Some transit systems employing either a fixed charge or a zone fare impose additional fare for transfers. Sometimes the transfer charges are incurred only if more than some specified number of transfers are performed. Sometimes transfer charges may depend on the location of the transfer and/or the modes of transportation involved in the transfer. Although these fare structures appear complex, they can be treated by counting transfers, noting where or between what modes they occur, etc.

With a fare structure such as that now used by most inter-city transportation systems, the charges differ by origin-destination pair. Such a fare structure would require storing a fare for each origin-destination pair of transit stops. Additional fare schemes are discussed in Appendix C.

In summary, there are two main points to be made regarding fare data. First, although fare structures vary from one transit system to another so that no single method of representation is sufficient for all systems, all known fare structures can be handled by appropriate representation. The second point concerns where, within the computations, the fares must be

determined. If fare is to be included in the determination of best trips, fare data must be available to the path finding component. Depending upon the fare structure, the algorithm may or may not have to accumulate fares as the paths are traced. If, on the other hand, fares are only to be reported to the caller, then all necessary fare calculations can be performed by checking only the best trip(s) determined by the path finding algorithm.

2.3.6 TRIP EXPANSION

After the shortest path algorithm has determined the best trip(s) between the appropriate major node origin and destination transit stops, it remains to expand this information back to the detailed network so that the origin and destination transit stop information can be reported to the caller. The data needed to expand the reduced network are the same as those needed to reduce the complete network.

The expansion process is perhaps best explained by an example. Referring back to Figure 2.2, suppose that a caller's origin and destination transit stops are stops 11 and 18 respectively. Referring to Figure 2.3, suppose that the best trip determined by the shortest path algorithm is the following.

Nodes	Arc	Route	Depart	Arrive
1-2	1	A	0800	0825
2-7	14	E	0830	0835
7-8	15	E	0835	0845

Suppose too that the displacement times for stops 11 and 18 on arcs 1 and 15 are 10 and 5 minutes respectively. By adding 10 minutes to the departure time at node 1, the caller's boarding time is determined as 0810 at stop 11; by adding 5 minutes to the departure time at node 7, the caller arrives at the transit stop destination node 18 at 0840, for a total trip time of 30 minutes. The need to transfer at node (stop) 2 as well as the details on routes and transfer times are all available directly from the trip information for the reduced network without expansion.

This example illustrates the relative ease with which the expansion process can be executed. Besides the information which is produced by the shortest path algorithm, all that must be referenced is the stop/arc data base, to determine which displacement times are needed, and the displacement time data to retrieve these times. Section 3 and Appendix D will illustrate how some of the various shortest path algorithms compute and store the path descriptions in the reduced network.

2.4 Response to Caller

After the best trip(s) have been determined, the information describing these trip(s) must be reported to the caller. There are at least two different ways in which this transfer of information could be accomplished. They correspond to different degrees of PTPTM Processor automation.

In a partially automated system, the computer output would be in written form from which an operator would read the information to the caller. In order to produce the necessary written trip description, each transit stop is associated with a location descriptor, e.g. stop 18 might be "Eighth and Elm". Then the numerical trip description output by the path finding component could be automatically translated into a written verbal trip description for the operator to read to the caller.

An automated version of this component might operate in a manner similar to the way in which the telephone system now responds to calls which reach an inoperative number. In this case, a computer generates a voice response, which, for the telephone system, is similar to "The number you have dialed, 555-8000, has been changed. The new number is 555-6000." At least part of the message is usually repeated twice. This type of response is generated by retrieving the appropriate pre-stored words, numbers, or phrases and piecing them together to form the complete reply. Although the responses needed to describe a transit trip will be more complex, the same approach can be used. Voice recordings of transit stop location descriptions, of the numbers from zero through fifty-nine, and of certain words and phrases can be pieced together to form a transit trip description. The following sentence will illustrate a possible trip description response where the various components of the response are underlined.

> "Board route 6 0 3 at 8 43 a.m. at the northwest corner of Eighth and Elm. Get off at Twenty-first and Elm and walk to Twenty first and Main to board route 5 3 5 at 9 o'clock. Get off at Twenty-first and Oak. The total trip time is 25 minutes. The fare is 50 cents."

In summary, the purpose of this component of the PTPTM Processor is to respond to the caller with a description of the best trip(s). Whether the component requires operator assistance or not, data will be needed to translate the totally numerical trip descriptions into written or oral verbal descriptions.

2.5 Auxiliary Updating Procedures

The preceding sections have discussed the various PTPTM components, the data bases required by them, and procedures (or computer programs) needed for answering a caller's request for a point-to-point transit itinerary. Since these data bases will periodically be in need of updating, it will be necessary to provide programs for accomplishing this. Some updates are primarily additions or corrections to a file (a new street name, a frequently misspelled location, a revised schedule for a particular route) and will require only simple procedures, but others affect several files and require a more complicated treatment. A change in routing, for instance, may require revision of the location/stop file, the stop/arc displacement data, the network and transfer data and perhaps the stop descriptor data. Any change affecting the network can require much reorganization of that data base for some of the algorithms described below, and will thus require fairly

sophisticated updating procedures for automatically determining the level of reorganization required. Most changes and updates can be done in a backup mode during slack time or even completely off-line, but some updates may be desired immediately. Indeed, since one of the advantages of a computerized system is its ability to respond quickly to changes, it will be desirable to include the facility for entering updates without significant interruption to the system's operation.

Since the need for automated updating procedures is evident, the data files should be designed with that requirement in mind. One suggestion in this regard is the inclusion of additional, redundant information in each record to facilitate easy file search and consistency checking. The inclusion of latest record change date in the file structure and the design of a program to produce audit trails of updates for perusal at periodic intervals will also aid in insuring updates are correctly made. It will be necessary, in addition, to design editing programs to check the consistency of changes, internally among new updates at one time, and between updates and the current files. Careful choice of update input formats, to permit easy description of the required change in terms familiar to those with knowledge of the transit system (rather than those with backgrounds in computer science or mathematics) is desired so that changes can be made promptly, without "sending for the specialist".

Besides updating programs, one can anticipate the need for programs for monitoring system performance measures, such as calls answered per station by time of day, frequently misspelled location identifiers, average waiting time for a call, nodes which are frequently asked for, routes which frequently appear in best paths, and any other information which might be used in aiding efficiency of operation or indicating desirable updates. Although not directly part of PTPTM, such programs will enhance the utility of the system by providing information to help management analyze its performance.

3. SHORTEST PATH SCHEMES FOR PTPTM

3.1 Introduction

A major computational effort involved in providing point-to-point transit itineraries is that of finding shortest (least time, least cost, fewest transfers, arrive soonest, arrive latest before desired arrival time, or some combination of the above) paths in a transit network. Shortest path algorithms are widely used in transportation planning (see the discussion of UTPS in Appendix A as an example), and have been for many years. The most frequent application has been in computing interzone auto travel times as part of planning an urban region's road system. Computing shortest paths in transit networks is a more recent development, but here too we are not treading new ground. How does the shortest path problem for PTPTM differ from that in these previous efforts? In transportation planning, average travel times including average time spent in transferring, are desired (with perhaps a distinction between rush hour periods and off peak times), since system planning is set up for the average transit or auto rider. For PTPTM, employed in an operational setting rather than a planning context, exact boarding time and specific information concerning transfers are desired so that the person requesting it can be told precisely which vehicle to board, where to get off, and when and which one to board next. Algorithms similar to those used by transportation planners are applicable, but their implementing computational schemes will be different since the data base used is different.

In describing shortest path calculations we shall use the following definitions:

> By an algorithm we will mean a rule or procedure for solving a mathematical problem. Algorithms are usually described in general and abstract terms without reference to a particular application. The more concretely defined procedure which specifies which data structures are used and which variables refer to which data, as well as a step-by-step description of the implementation of the procedure (in sufficient detail for a computer to be programmed from the description), will be termed here a computational scheme.
>
> A network consists of a finite set V of nodes and a finite set A of arcs. To each arc $a \in A$ there corresponds an ordered pair (u, v) of nodes, the origin node u and the destination node v. In addition, each arc has associated with it a non-negative length $\ell(a)$. (For some applications each arc will have several "lengths" associated with it, one representing for example transit time on the arc, and a second representing a fare for the arc or possibly some other disutility.)

A path in a network is a sequence $P=(a_0, a_1, \ldots, a_k)$ of arcs such that the destination node of arc a_{i-1} is the origin node of arc a_i. The origin node of arc a_0 is called the path origin and the destination node of arc a_k is called the path destination. The path length $f(P)$ is the sum of the lengths of arcs in P:

$$f(P)=\ell(a_0)+\ell(a_1)+\cdots+\ell(a_k).$$

A shortest path from node v to node w is a path P (there may be several) with origin v and destination w for which $f(P)$ is minimum.

A cycle is a path which begins and ends at the same node. A network is acyclic if it contains no cycles. A network is bipartite if its nodes N can be partitioned into two sets N_1 and N_2, whose union is N, whose intersection is empty and which are such that every arc a has its origin node u in one subset and destination node v in the other. Note this means that no arcs have both origin and destination nodes in the same subset, either N_1 or N_2.

3.2 Basic Shortest Path Algorithms

Although the title of this chapter only refers to algorithms, the chapter will be concerned both with algorithms and with computational schemes. Good algorithms for solving the shortest path problem are well-known and have been available for some time. The problem we face here is not primarily one of choosing an algorithm for use in point-to-point trip management, but that of deciding upon an efficient computational scheme to employ in this particular application.

Algorithms for solving the shortest path problem may be divided into two classes, matrix algorithms which calculate paths between all pairs of nodes at the same time and labeling algorithms which calculate shortest paths from one node to some or all other nodes. Matrix algorithms must store the matrix of shortest path lengths between all node pairs simultaneously in the computer, requiring N^2 locations for a network containing N nodes. For networks of several thousand nodes, several million computer storage locations are thus required. For this reason matrix algorithms are seldom used for path calculations in large transportation networks.

Labeling algorithms involve labeling each network node with the length of a path from the origin to that node. Computer storage for labeling algorithms is usually several locations for each node and two locations for each arc. See [5] and [11] for a more detailed description of the labeling approach.

There are two basic labeling algorithms for computing shortest paths, the label correcting algorithm and the label-setting algorithm. In the basic label correcting algorithm the origin node r is initially labeled 0 ($d(r)=0$) and all other nodes are labeled ∞. At each stage, one searches for an arc a with origin u and destination v such that

$$d(u)+\ell(a) < d(v),$$

and replaces the current value of $d(v)$ by $d(u)+\ell(a)$. The algorithm terminates when no such arc remains; then for each node u, $d(u)$ is the length of the shortest path(s) from r to u, and such a path can readily be "backwards-traced". The basic label setting algorithm starts out in a similar manner with the origin labeled 0 and all other nodes labeled ∞. At each stage, the algorithm examines all arcs a whose origin node u has finite label and whose destination node v has infinite label. That node v for which $d(u)+\ell(a)$ is minimum is then labeled $d(v)=d(u)+\ell(a)$. The algorithm terminates when all nodes have finite labels (or, if the path to a particular node w is desired, when w receives a finite label). In the label correcting algorithm a finite label may be changed at a later stage of the algorithm, but in the label setting algorithm once a node receives a finite label that label is permanent and represents a correct shortest path length from the origin to that node.

3.3 Sequencing Methods

Computational schemes for these algorithms utilize a variety of techniques to increase efficiency of operation. Most of these techniques involve ways of recognizing which arcs should be examined and in what order. All require the use of additional lists to store a partial history of the latest stages of the algorithm as an indication of how to proceed next. Four of these techniques will be mentioned here. The first, used to improve the operation of the label correcting procedure, will be called the alteration flag and involves flagging a node whenever a better path is found to it. One then examines only those arcs originating at flagged nodes, since other nodes either do not have any path yet found to them or else have already been examined without improvement. The flag is removed once path extension from its node has been tried. Of course it can later be reinstated if a shorter path is found to that node.

A second technique, also used to improve the label-correcting method, is called sequencing by alteration and involves retaining a list of nodes in the order that their labels have been changed. One then examines arcs whose origin is the top node on the list before other arcs. Once all arcs from that node have been examined, it is removed from the list and the next node becomes top. Pictorially this means that one starts at the path origin and "fans out" searching for a path extension. At each stage one is either trying to extend a path to an unlabeled node or testing an alternate path to an already labeled node.

Other sequencing rules are possible, such as sequencing by cardinality distance in which nodes which are a greater number of arcs from the origin node appear further down the list from those closer (in number of arcs, although not necessarily in path length) to the origin. This is especially relevant for transit networks since with appropriate network representation, a sequence of arcs will describe a sequence of transit routes with transfers occurring at nodes at which the arcs meet. The list then insures that paths with fewer transfers are investigated before those with more. If one wishes to consider only paths with fewer than, say, two transfers, sequencing by cardinality distance allows one to examine all, and only those paths meeting this criteria.

A fourth technique is used in improving the label setting procedure. In this procedure one has a set of nodes which are labeled and a set which are not, and at each step the algorithm labels a new node not previously labeled. This requires an efficient method of determining which unlabeled node is a good candidate for labeling next. During the course of examining arcs for constructing a new label, the algorithm usually calculates several potential labels but retains only the minimum. The distance list is a method of retaining these temporary labels and using them as a method of selecting the next node to receive a permanent label and the value of that label. Nodes are placed on the distance list in a position determined directly by their label. (That is, node j is placed in position i equal to d(j) modulo one plus the maximum arc length in the network.) The fact that the length of the distance list is determined by the maximum network arc length makes this computational procedure suitable for most transit networks and expected path-selection criteria.

Acyclic networks lend themselves to especially efficient processing, since nodes in the network can be ordered (and numbered in order) at the outset in such a manner that if u is the origin node of any arc a and v its destination node, then $u<v$. Such an ordering eliminates the need for establishing an ad hoc order during the progress of the algorithm. This is especially important if the network is so large that it cannot be contained wholly within the computer at one time. The once-and-for-all ordering allows the network to be stored serially in peripheral storage, so that one portion of it (generally called a page) can be processed internally at one time without reference to the rest still retained outside the computer. Acyclic networks have other nice properties for obtaining shortest paths in transit systems.

As discussed in greater detail in Appendix D, transit schedules may be represented as an acyclic network in which nodes are geographical transit stops at precise times of day and arcs are specific scheduled transit trips between stops. Such a network is acyclic since all arcs go forward in time. Allowable transfers may be represented as arcs in the network so that minimum time required for inter-transit route transfers can be included directly in the network. This dual (location in space and in time) nature of a node also overcomes what is called the overtake problem, which can be illustrated by considering a person faced with taking a slower local vehicle immediately

or waiting to take a faster express service along the same line. If he is going only a short distance the local vehicle may arrive sooner, but if he goes further, waiting for the express may be the better choice. Applied to the network whose nodes are transit stops, most shortest path schemes would require him to start out on the local and transfer mid-trip to the express, since each beginning segment of a shortest path to any node must itself be a shortest path to the intermediate node. The acyclic network in which nodes are geographical points at particular times avoids this difficulty, since both the local service to an intermediate transit point (arriving earlier at this point) and the express service (arriving later at the intermediate point but earliest at the destination) can be included. This difficulty also arises when minimum times required to transfer at an intermediate point may make a route which seemed slower for the initial portion of the trip be the best for the whole trip. Here again the acyclic network representation described above avoids the problem.

3.4 Criteria for Path Selection

A further discussion of list processing techniques which can improve the performance of shortest path algorithms can be found in [4], [5] and [11]. Actual implementation of one of the algorithms as a computational scheme requires, as illustrated in the discussion of the acyclic network above, not only the choice of algorithm and precise list processing scheme, but also a definition of the data elements to be used in the process. For the shortest path problem this involves specification of three items:

1. What network nodes represent,
2. What network arcs represent, and
3. What values (arc lengths) are to be placed on the arcs.

The latter determines the criteria for "shortest" path, in the sense that path "length" is the sum of arc "lengths". Other operations than summation are possible but do not seem as relevant, since the criteria described in Section 2.3.1 can be represented by summation of arc lengths.

Several of the criteria suggested involve constraints, such as least time path leaving after 7:30 a.m. with fare not exceeding $.50, arriving soonest before 5 p.m. with less than 15 minutes spent in transferring, or arriving closest to 5 p.m. with less than two transfers. Under a suitable network representation the latter constraint can be handled through sequencing by cardinality distance. (See Appendix D.2 below for a more complete discussion of this procedure.) The other two types of constraints can be handled in the acyclic network described above by insuring that paths are extended only if the extension would not violate the constraint. Thus it is necessary to keep track not only of the cumulative values for the optimality criterion (time, in the examples above) but also the cumulative value of the constrained variable or variables. The decision to add arc a requires both that $d(u)+\ell(a)<d(v)$ and also that the second criterion does not violate its constraint. Whether it is necessary to include the capability for calculating constrained shortest paths remains uncertain until desired criteria are chosen, but whatever the decision, existing algorithms can handle this case.

Up to this point our discussion of algorithms has focused on finding a single shortest path, but most existing transit information systems provide several itineraries to callers. This allows patrons to alter plans at the last moment and gives them extra confidence and information on the next opportunity for service if the desired vehicle is missed. Usually in existing manual systems the operator provides only the previous and succeeding transit vehicles on the same routes. Alternative routings are not provided, unless easily available (as in the case of parallel routes, or express/local service on the same route). Once a best itinerary is found using a shortest path algorithm, it is not difficult to provide times for previous and next vehicles on the same routes for each route segment. However, the second best path may use an entirely different routing.

Algorithms for solving the kth best path problem, producing the first, second, third, etc. up to kth best paths exist [2], [8], and some experience [9] indicates that computation time on medium-sized networks is not prohibitive. Actual experience with networks of the size expected in PTPTM applications is desirable, but preliminary results indicate that for systems of this size such algorithms may well require too much time per path and/or too much computer storage. Even if second and (perhaps) third best paths are to be provided, it would probably be desirable also to include the previous and next departure along any transit route given, to help the user evaluate the criticality of making a particular vehicle or connection. This would lead to much duplication where first and second best paths use approximately the same route. The situation for which the kth-best path algorithm is of greatest value is when there are alternate transit routes available serving the same pair of nodes. The user can then be provided with information about both (or all) of them and make his choice taking into account criteria not normally considered by the algorithm (such as the neighborhood involved, the expected load factors of the lines, desired transfer points, etc.). The kth-best path algorithm is also an alternative to providing different shortest path criteria. Providing the user with several good paths all based on the same criterion allows him to choose one which is second or third best with respect to that criterion but more desirable with respect to another. Thus multiple path algorithms would be useful if true alternate routes serve the same customers, if several criteria for shortest paths are required, and if the additional computer time and storage are acceptable. Otherwise it would be better, simpler and less costly to avoid their use if possible.

All of the algorithms and computational procedures described so far have guaranteed producing the shortest path according to the chosen criterion. One should at this point ask the question, "Can one find a simpler procedure which nonetheless provides, if not the very best path, a good path?" This is certainly consistent with the itineraries provided now by existing manual systems. How "good" the itinerary is, depends in part on how experienced is the person answering the call, and several transit information callers have noted that they regularly request the same trip at least twice to compare routings.

The particular procedures used by the PARIS system [3], developed by Systems Development Corporation (SDC), are not available, and since the company is marketing the system one would expect this information to remain proprietary. One can, however, suggest possible procedures for speeding up path calculation, which may or may not resemble those used by SDC. If all network nodes, say street corners, have x-y coordinates associated with them, one could then make use of the fact that one knows the general direction in which one wants to head. All trips which deviate more than a prescribed amount from this direction might be ignored. For instance if the desired direction is northeast, one might feel justified in not going more than a fraction of the straightline distance between the two points in a south or west direction. Routes could be identified as north, south, east or west and those proceeding in the right direction scanned first. All of these procedures would be combined with the basic label correcting scheme and termination would occur when the destination node is reached and no more improvements are possible. In large networks the nodes could be divided into zones. Tables could be constructed of routes which are candidates for traveling between designated zone pairs. Perhaps some interzone routing could be defined once and for all, or at least a list of several potential routings. Such heuristic procedures would have to be tested and evaluated on large example networks, since it is impossible to anticipate all problems associated with their use.

3.5 The Role of Shortest Path Algorithms in PTPTM

It is perhaps useful at this point to discuss the role of path algorithms and computational schemes in the whole of the PTPTM process. As can be seen in Figure 2.1, shortest path calculations are a central part of PTPTM service since they actually provide the trip itinerary. It has been estimated [6] that in current manual systems about a third of the total call is spent in search for the correct path. The rest of the time is spent in ascertaining what trip the caller is inquiring about (origin, destination, time, etc.) and in relaying the information on the actual trip suggested after an itinerary has been determined. Therefore the computerized shortest path calculation can only hope to affect one third of the call, and because of additional data entry time the actual time savings may be closer to 20 or 25 percent even if path calculation were effectively negligible. This may, however, produce proportionately much greater reductions in callers' delay and turn-away rate; an analysis of this point, using queuing theory, is given in Appendix B. One example worked out in the Appendix for

a range of arrival and reneging rates and maximum queue lengths indicates that a reduction of 40 to 90 percent (typically 60 to 70 percent) in the number of lost calls would result from a reduction of 20 percent in the service rate.

Although the shortest path calculation is central to the operation of a PTPTM system, early choice of a computational scheme for it is not as critical to system development. All of the algorithms and schemes presented here use the same data base, and as discussed above in Section 2, data base development and construction is the major effort in setting up such a system. The location identifier file, the transit access file, and even the file of transit schedules are common data bases independent of choice of computational scheme. Some special processing of the basic transit schedule file may be necessary for each different scheme, but automated procedures for doing this should be relatively simple to define (and quickly programmed). The computational schemes themselves are uncomplicated, and it is expected that existing procedures and programs can be modified without great investment of time and expense. In fact, rather than making a premature choice of computational scheme, it is suggested that several be programmed and tested on a large real data base. Such a test will be necessary as a demonstration project in any case, and the additional expense of trying several computational schemes (rather than only one) will be very small compared to the effort of setting up the initial data base. The actual test would allow estimated average parameters (such as expected path length, number of transfers, etc.) to be checked so that unanticipated network and path computation anomalies can be ironed out and better estimates of actual computer times and costs can be obtained. The demonstration data base would be needed even were no further software development contemplated, as would be the case if existing software such as that available from SDC were to be adopted by DOT. Even in this case, however, further investigation into the path computational scheme seems reasonable, since this software forms a well defined module of the total system, and cost of such a study is not prohibitive.

We have in this document generally assumed that the information to be provided by a PTPTM system includes exact vehicles to be taken (the K-2 bus), exact boarding and alighting stops (8th and A to 12th and F), and boarding and alighting times (board at 9:32 and leave at 9:48). Other information, such as a description of the access trip from the caller's origin to the transit boarding stop and similarly at the trip destination, or a description of any transfers which involve walking between stops or to a different corner of the same street intersection, may also be provided in a more elaborate system. As discussed earlier in Section 2.3.3, the use of exact vehicles and exact schedules is perhaps open to question. Existing transportation planning software such as the UTPS system, developed and operating under UMTA auspices, uses average headway information rather than explicit schedules. A further discussion of the potential for use in PTPTM of UTPS is given in Appendix A. However, algorithms used in either process are essentially the same, and computational procedures differ

primarily in what constitute nodes and arcs. A more detailed description is given below of those computational schemes which may be used if average headway information is sufficient to meet the callers' needs. Testing of this approach could be part of the demonstration effort mentioned above, and although preliminary processing of the data base is perhaps more extensive in this case, the incremental cost of designing (or adapting) a prototype computer program for it and for path calculation based on the transportation planning procedures is not prohibitive.

3.6 Data Manipulation Aiding Algorithm Performance

A major part of a computational scheme is the definition of the data elements of that scheme. In the case of shortest path calculation schemes, the primary data elements are the nodes and arcs which form the network. A more detailed description of these and the way in which they enter each particular scheme will be given in Appendix D. We will here discuss general ways, of manipulating the transit network representation, which are applicable to the data structures of several schemes.

The criterion used by most existing transit information systems in providing a good trip itinerary is "least time". Clearly this does not mean the trip during the whole day requiring shortest travel time, nor usually the shortest trip during a particular time period, since actual departure or arrival time may be critical. Thus the criterion used by most systems may be phrased as "the trip which leaves after 5 p.m. and arrives soonest" or "the trip leaving latest and arriving before 9 a.m." (not both for one trip), but trips which leave before the desired departure time or arrive after the desired arrival time may be suggested to the caller if the more desirable service is not available. This is accomplished by including a penalty weight for time before desired departure or after desired arrival in the calculation. The two criteria, shortest trip departing after desired departure time and shortest trip arriving before a desired arrival time, will be called the departure oriented criterion and the arrival oriented criterion respectively. Any PTPTM system would need to include both of these criteria at the very minimum and perhaps others also.

Special list structuring for storing network arcs can speed up calculation for the two different criteria, and for some schemes it may be necessary that two copies of the network be retained (off-line, probably) so that the appropriate network structure can be recalled for use with the desired criterion. Generally, in storing a network, arcs are grouped either by arc origin node or by arc destination node. The former is termed "forward star" form and is used for the departure oriented case; the latter is termed "backward star" form and is used in the arrival oriented case. A pointer is kept for each node to the first arc in the star (forward or backward) of that node, so when in the course of the computation one wishes to "fan out" from a node, one can immediately locate the arcs which are required. In the departure oriented case, the procedure starts at the origin node and proceeds forward using the forward stars of nodes along the path (and in time) to the destination. If the same computational scheme

is used for the arrival oriented case, it proceeds backward from the destination (and in time) using the backward stars. For some of the approaches described in Appendix D, an alternative scheme may be applied to the forward star network, obviating the need to store the network in two different forms. - Storage of the network in star form requires sorting the arcs either on origin or destination node and computing the star pointer.

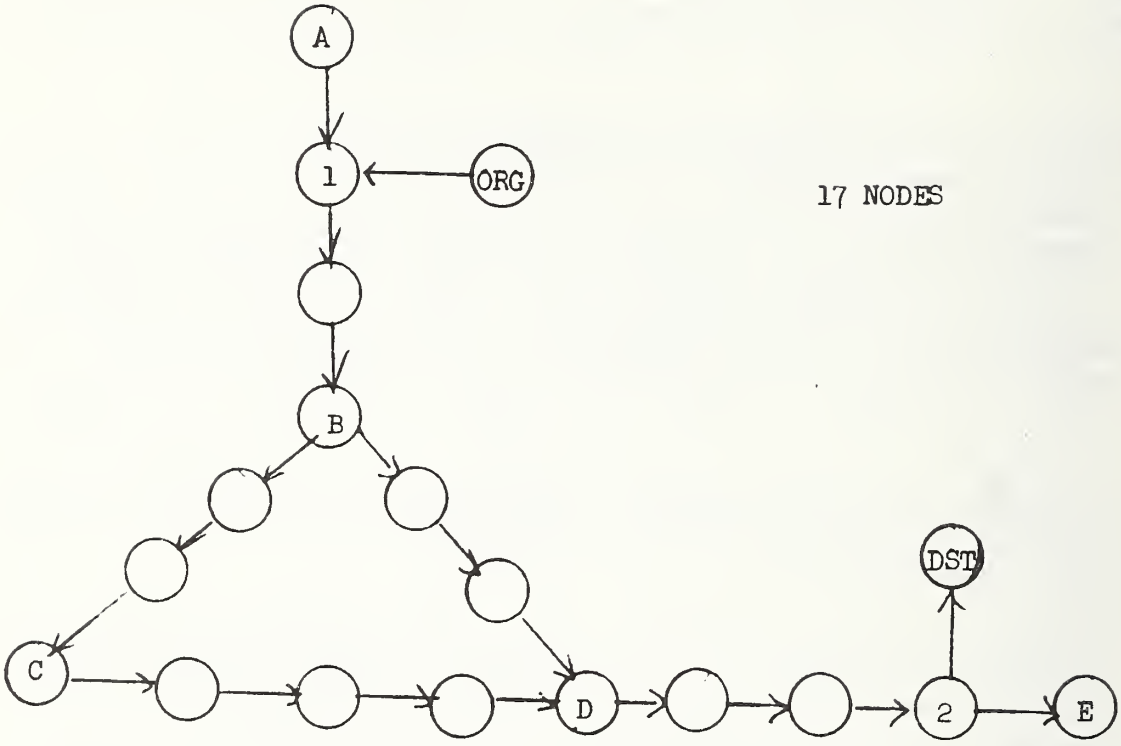
Most network arcs represent scheduled transit route segments, but it may be necessary to include some arcs representing transfers. This will clearly be necessary for cases in which two routes have no stops in common, but customers often walk from a stop on one route to a stop on the other. Time along the transfer arc must be sufficient for the walking trip and will probably also include some leeway for making the connection (unless this is handled differently within the algorithm). Even when the transfer is between different routes at the same node, it will be necessary to insure that the routing provided includes enough time for alighting one vehicle and boarding the second, as well as some extra time to allow for the first vehicle being late or the second early. This may be accomplished either by adding a network arc representing the transfer or by including a list with the minimum time required to transfer between routes at each network node, and modifying the computational scheme to add this time to vehicle arrival time before extending a path.

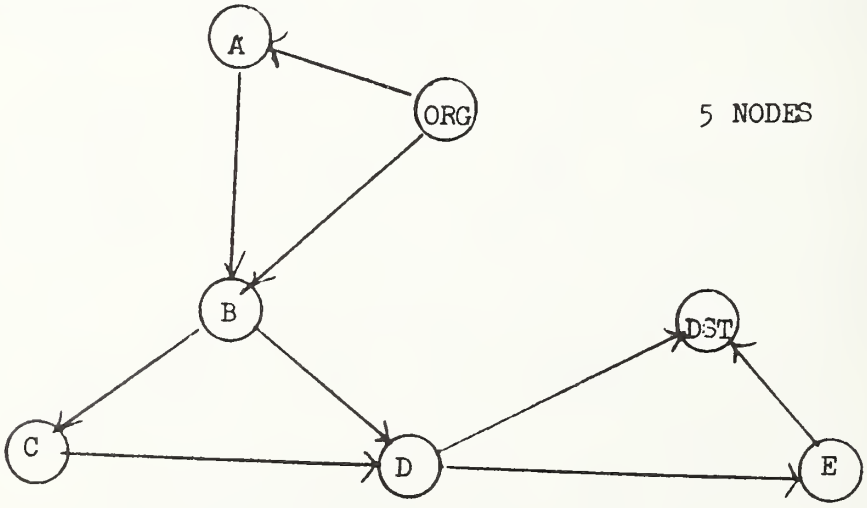
In most commuter bus and rail systems, stops are thought of as instantaneous, since boarding and alighting time is included in interstop running time and is of the order of magnitude of the variation in running time. A node at which a scheduled period of stopping occurs can be handled in shortest path calculation procedures by splitting it into two network nodes, one being the alighting node and the other being the boarding node, with a link between for the scheduled stop time.

Network arcs generally represent the schedule along a particular route. Different routes which have segments in common will still be represented as separate arcs, since transfer from one route to another will be required. Where transfer arcs are included in the network it will be necessary to have separate nodes for each line stopping at the same place, but the nodes may be the same if a single minimum transfer time is used throughout the network. When two or more routes have several arcs in common only the first and last nodes of their common segment need be considered places to transfer from one to the other, or alternatively two other nodes may be specified if they seem more natural or more convenient from a practical standpoint (at a shopping center, or busy corner say, rather than in a less traveled area). Path calculations can be made in a network consisting only of major nodes at which transfers between routes are possible. Figure 3.1 shows how such a reduction can be effected for one example, with a reduction in the number of network nodes from 14 to 5. The single access arc in the upper network is replaced by two arcs in the lower network, and the four transit arcs between nodes A and B are replaced by the single arc in the lower network. This procedure increases the number of access arcs, but reduces the number of transit network nodes. Since in

FIGURE 3.1

Reduction to Major Nodes





any path calculation access (and egress) arcs are required only for the path origin and destination nodes, only those two additional arcs are required at any one time. The reduction in network nodes is critical since the speed of calculation of shortest paths depends almost entirely on the number of nodes in the network.

Generally the headway sheets maintained by transit companies for use in manual PTPTM systems give scheduled arrival and departure times only at major stops. The time at intermediate stops is obtained by interpolation, that is, assuming direct proportionality to distance. Thus boarding or alighting at a stop one mile from the beginning of a three mile segment would occur 1/3 the total trip time after the vehicle left the segment origin node. If for example a vehicle left the origin at 9:00 a.m., covering the three miles in 15 minutes, it would reach the stop one mile from the origin at 9:05 a.m. This type of interpolation can easily be automated and will be needed in path calculations only for the transit access and egress nodes associated with the path origin and destination. (Of course several transit access nodes may be associated with one path origin since several transit lines may be available to people from that point.) The major portion of the path calculation would thus be accomplished in the "skeletal" network consisting of nodes at which transferring is possible, and information on the total network enters the process only at the beginning and at the end. The "network" referred to in the section below will thus be this skeletal network.

3.7 Computational Schemes for PTPTM

This section will include a general description of several computational schemes for use in PTPTM. (Step by step procedures for several schemes are given in Appendix D.) We will note here several special-purpose computations which can be included or will be needed in any of the schemes. The access/egress interpolation described in the previous section is one such computation which will be required as part of the schemes described below, since we are presuming that the networks to which the schemes apply will consist only of transfer nodes. Special coding will be required to add access and egress arcs for the origin and destination to the network so that access and egress can be included in the path calculation process. The networks will also have to include transfer arcs representing walking trips between transit stops whenever such a transfer is considered usual or likely. Regular network arcs represent discrete service, i.e. transit vehicles traverse the arc only at specified times. Transfer arcs represent continuous service and it would be repetitious to represent each possible transfer separately. To avoid this, arcs can be labeled as either discrete or continuous service arcs. The first may require a waiting time between when service is desired and when it is available. This waiting time must be added into total trip time along with the arc travel time. Transfer arcs require no wait time; only the arc travel time enters trip time. (Of course waiting, after walking, for the next vehicle on the second line will have to be included.) Separate treatment of the two arc types will have to be coded in the computational schemes. Some coding will also have to be included to handle minimum transfer time restrictions which ensure that enough time is allowed for transferring between routes at a node.

Such restrictions probably depend primarily on the node, but it may be necessary to have different requirements between different routes at the same node, if for instance different routes stop at different corners of the same intersection. This can always be handled by treating the different corners as different nodes and including walking transfer arcs between them. Since such a treatment increases the number of nodes, it is hoped that there will not be many cases for which it is necessary.

Some coding may also be required to obtain the most desirable path. The shortest path criterion described above produces a path leaving after a desired departure time and arriving soonest (or alternatively a path arriving before a desired arrival time and starting latest). If several paths arrive at the same time all leaving after the desired departure time, no distinction is made among them, although some may actually be more desirable than others. In fact one would expect that generally the one with least total time spent in waiting and transferring at intermediate nodes would be preferable. To break ties among trip arriving at the same time, weights may be attached to the initial wait time and to transfer time, so that initial wait time counts less in the total than does transfer time. Then the weighted sum of initial wait time and transfer time may be accumulated separately and whenever two trips have the same arrival time at a node, the one with the more desirable wait time-transfer time weighted total would be chosen. Such a procedure leads back to the same difficulty as the overtake problem: the best path to an intermediate node is the one arriving soonest at that node but may not be the most desirable first segment for the best path to a node further away from the origin. The time expanded network approach given in detail in Appendix D will overcome this problem since several paths to the same geographical node are allowed. Another possible approach would be to examine the best path after it has been calculated, from destination back to origin, to see if perhaps it is possible to leave later along the same path but arrive at the same time. Such a procedure would not necessarily arrive at the most desirable path, since only time variations of the same routing would be examined. A final alternative approach is to change the best path criterion to be a weighted average of wait time, travel time and transfer time with travel time having weight one, wait time having weight less than one, and transfer time having weight greater than one. The difficulty with this approach lies in the necessity of experimenting with actual weight values until a set is obtained that gives good paths.

Some special coding in the computational scheme and/or some special network structuring may be required to calculate fares associated with the best time path. Many areas have gone to a flat fare system in which all trips, usually limited only by a maximum number of transfers or by a requirement that the patron may not reverse direction, cost a single flat fare. This type of fare system can easily be handled within the computational structures described below. The other most common fare structure is a zone fare system in which fixed rates are charged for trips within a zone, subject usually to the restrictions mentioned above for defining a trip in a fixed fare system, but trips from one zone to another incur an additional fee at the zone boundary. This can be handled in one of two ways in a computerized PTPTM

path calculation process. A matrix of zone to zone fares may be included, if the number of possible zones is fairly small, and the fare for the appropriate zone pair for the transit origin and desination may be obtained directly from this table. Alternatively the incremental fare for crossing a zone boundary may be accumulated during the progress of the path computation, along with the usual time accumulation. This requires that the incremental fare for boundary crossing be associated with an arc which crosses the boundary. Some special coding may be required to ensure that only trips which cross boundaries have this incremental fare associated with them. Care must also be taken that routes which criss-cross back and forth along the same boundary are not charged more fare than would actually be the case. It is expected, however, that this situation would seldom arise, since route structures would rarely favor crossing back and forth along a zone boundary. It would also be necessary to have incremental fares associated with arcs of express lines when the cost of express service is higher than that of local service. (This is a case where it might be desirable to compute a least fare route for those unwilling or hesitant to pay the cost of express service.) Combinations of these fare structures are likely and it may be necessary to include additional coding within the computational scheme to accomodate them, but this is not expected to greatly complicate the scheme or to be very difficult.

Some post-processing of the best path will be required before it is displayed to an operator or relayed to the caller. The path calculated by the algorithm may have more segments than actually required, since if no transfer was necessary at a transfer node the path will have two successive arcs along the same transit route. All successive arcs on one route must be combined, and only points at which actual inter-route transfers are required should be listed. In addition it is desirable to print the preceding and succeeding departures for each route taken. These must be fetched from appropriate storage and listed along with the best route. Other information may be accumulated and printed by the scheme, including fare, time spent in transferring, number of transfers, distance to be walked (if intermediate walking arcs are required), or any other characteristics which are included in the network description and can be associated with the path.

The items described in the present section up to this point have involved operations which affect all schemes for transit path calculation. More detailed descriptions of alternative computational schemes for PTPTM are given in Appendix D and when reading that section one should keep in mind the common operations noted above since they are not generally included as separate items in each of the schemes' descriptions. The appendix will include three major schemes, to be termed the time-expanded network scheme, the bipartite route/stop scheme, and the transportation planning scheme. In addition we will note, in conjunction with the second scheme, a method of compactifying best path information which may lend itself efficiently to storing precalculated paths. All schemes will for convenience be described in terms of the depart after/arrive soonest criterion, but they can just as easily handle the leave latest/arrive before criterion. A separate scheme

for that criterion is described for the time expanded network case because it is particularly useful to have both schemes with one network representation.

In the time-expanded network scheme, nodes represent a particular transit stop at a particular time and arcs are specific transit vehicle trips departing one stop at a given time and arriving at another stop at a specific later time. Therefore there are as many arcs as there are vehicle trip segments. Since arcs only go forward in time, the network is acyclic and a special computational scheme, examining just once a subset of the nodes in time order, can be used. Appendix D gives two schemes based on a time-expanded network stored in forward star form, one for the departure oriented case and a second for the arrival oriented case. The bipartite route/stop scheme uses a network containing two classes of nodes, one representing transit stops and a second representing routes. Arcs connect each stop with the routes stopping there and each route with the stops along it. A path in this network consists of an alternating sequence of stops and routes, beginning and ending with a stop, giving the itinerary for the trips as a sequence of stops and the routes to take between those stops. Two schemes are presented in Appendix D using this network, one a label-correcting scheme using a sequence list and a second based on the label-setting procedure using a distance list. A special method of representing and calculating paths for a whole time period is presented for use with the label-correcting procedure, for obtaining and compactly storing pre-calculated paths for on-line retrieval. Finally, a variant of the transportation planning approach using average running times and headways is presented. In this scheme time to transfer is represented as half the headway of the vehicle to which one is transferring.

3.8 Choice of Shortest Path Scheme

In Appendix D we describe several computation schemes for calculating optimal paths in a transit network. The question we address here is that of choosing among these computerized path calculation schemes for use in the PTPTM environment.

Currently transit information systems obtain paths manually as they are needed. Usually this means that the operator, who has received the call and has processed the query resulting in a precise description of the request being made, decides the best transit path using maps, schedules, and his/her intuition and experience. Such a procedure can be very efficient if the operator is experienced. Some calls may require virtually no reference to documents by an operator who knows the system well. However, it is much more difficult for operators to become expert in the large, complex, perhaps multi-mode or multi-property transportation systems. In addition, operator training costs are high as are personnel turnover rates, so that it is likely that many operators are unfamiliar with routings and will have to use reference materials in answering most requests. There is also a need for rapidly incorporating temporary or permanent data changes into responses.

As noted earlier, obtaining the "best" path takes on the average a third of the total call time. The computerized path finding schemes described in this report would be aimed at reducing this portion of the call. We are not addressing schemes for reducing the rest of the call, although the computer might be applied usefully here too (for example, to repeat slowly the suggested transit trip so that the caller can record the information at whatever speed he desires while the operator is free to go on to a new caller). One can imagine a completely automated system where the caller "dials" codes on his telephone, indicating the origin and destination of his trip and the desired departure or arrival time. The computer then obtains a best transit path and relays a description of it back to the caller. In such a system the computer would assume all functions of the operator. Without discussing the merits or pitfalls of such a system, it is clear that path-finding schemes are a central requirement for its realization.

The degree to which other activities are automated has little effect on the operation of the path-finding scheme once input to that scheme is decided upon. Its output may be a printed path description on paper or CRT scope for reference by an operator, or may be a computer generated voice response. The choice does not affect the feasibility or relative desirability of the various path finding computation schemes.

Whereas the degree of automation of other portions of the PTPTM process has no effect on the path finding scheme, that is not true of all decisions about the process. In general a path may be found in two ways - 1) it may be calculated as needed (on-line calculation) or 2) it may be retrieved from a store of previously calculated paths (precalculation). Which of these approaches is taken does have an effect on choice of scheme. Appendix D, for instance, includes a description of a method of compactly representing the path information for use in storing precalculated paths. In addition, the transportation planning scheme (Appendix D.4) is presented primarily for use in precalculating paths. An evaluation of the merits and problems associated with the two approaches, on-line calculation and precalculation, will be given below, as an aid in comparing the schemes.

To mirror the manual system, paths would be calculated as the need arises, meaning an on-line system. This system has the advantages of requiring peripheral storage only for the network and schedule data, and of being able to react immediately to changes in the network or parameter values. Temporary outages, re-routings, delays etc. can be input to the network and schedule data, and can thus be reflected in path calculations in a very short time. The crucial question in implementing such an approach is its feasibility, determined primarily by the time required for each path computation. One such system, the PARIS program, has been implemented, but so far it has been tested only in a relatively small city, Santa Monica. Since the major usefulness of computerized transit information systems is in larger cities with more complex systems, the true feasibility of PARIS must be demonstrated in such an environment. We will make some timing estimates below which turn out to support the feasibility of on-line calculation for a large city, but confirmation requires actual testing on a realistically large, complex data

base. The main point to be made in regard to the timing estimates is that path computation time is overshadowed by network paging time, for any of a set of good computational schemes. Analysis to reduce network size is thus an important part of the process.

The alternative to on-line calculation is to precalculate all possible paths and retrieve them as needed. The time required to answer a query is just that to retrieve and unpack a path, significantly less than network paging. It also eliminates the need for recalculating frequently requested paths, but at the expense of calculating some which are never asked for. Another disadvantage is that all paths must be recalculated when the network changes. In practice this means that small changes will not be permitted to trigger a recalculation, and a decision will have to be made as to what level of network alteration warrants redetermination of paths. Temporary conditions will seldom be reflected in the path calculation, but may be handled through additional message relays by the operator or by the system itself directly if the process is fully automated.

The major feasibility question for a precalculation approach is the storage required for paths. The compact representation presented in Appendix D offers a way around this difficulty, but at the cost of access requests to the file of compactly stored paths. An alternative approach to precalculation would be to use the transportation planning scheme presented in Appendix D.4 to calculate best path routings for 3 to 5 different time periods during the day. This assumes, of course, that a single routing remains "good" throughout a particular time period, an assumption which would have to be tested. It also assumes that a suitable partition of each day into time periods can be fixed at the beginning of the day, and that problems at the times of changeover from one period to the next are sufficiently few to be disregarded. In operation, when a caller requests information about a particular trip, the routing based on the transport planning scheme would be retrieved and then the precise schedules for the desired time of departure or arrival would be obtained for that routing.

The choice between on-line calculation and precalculation is difficult and cannot be wholly resolved here. Our considered opinion at this point is that the on-line approach is inherently more desirable, because of its more flexible reaction to network changes and its need to calculate and accommodate only those paths about which inquiries are made. (Some mixture of the two may be appropriate.) The critical question thus revolves around the feasibility on the on-line approach, which hinges on its requirement for paging time. For either the time-expanded network scheme or the bipartite route/stop scheme, it will be necessary to access schedule data for a time period whose length is determined by the shortest path from ORG to DST.*

---

*ORG is the path origin and DST is the path destination node.

We will now make several assumptions on system size as an aid in estimating the number of pages of data required. Assume:

1) a path length of 60 minutes
2) 2000 vehicles active at one time
3) a vehicle takes 20 minutes to traverse a route
4) 10 arcs per route
5) 3 pieces of information per arc
6) each piece of information needs 2 bytes of storage.

These assumptions lead* to a requirement for 360,000 bytes of storage for the network required for one path calculation. Under the assumption of a machine size of 250,000 bytes of internal memory, we will require 2 to 3 pages (allowing storage for the path description and program and estimating conservatively). Better, more exact estimates would be available only with more detailed descriptions of sample transit systems, but preliminary figures indicate that the values chosen are not unreasonable, and generally overestimate the average situation. Actual network storage, for instance, will require less than the 3 words per arc estimated here unless fares or other items are also required, and most trips would be less than 60 minutes. The figure of 2000 vehicles is based on the fleet size of the Washington, D.C. Metrobus system; the route length figures are pure conjecture but 10 arcs/route seems large while the 20 minutes/route may be small (both yielding a pessimistic estimate). Machine size is given for one of the larger currently available computers (e.g. IBM 360/65, UNIVAC 1108, CDC 6600), but does not take into account such advances as virtual memory and thus it too may be conservative. The estimates are made based on a longer than average trip, so that actual average paging would be less, again indicating a pessimistic estimate.

"Paging" refers to the fact that not all of the network can reside in the computer's central processing unit (CPU) at the same time. Accessing information in the CPU is limited solely by electronic processes and is measured in nanoseconds. In contrast, accessing information stored on a peripheral device (disk or drum) depends in part on the physical movement of the device and requires access times measured in milliseconds. Transfer rates from peripheral storage are also greater than from the CPU since the information must be relayed among several devices and must be checked for accurate transmission. Using currently available access and transfer rates, it is possible to access and transmit 3 pages of the size estimated above in about half a second. Shortest path calculation time using the schemes described in Appendix D is likely to be of the order of .2 to .5 seconds (see [5] for reported timings), which to some extent may occur simultaneously

---

*During the 60 minute trip duration, one vehicle will be able to traverse 3 routes, at 20 minutes per route, so 2000 vehicles will be able to make 6000 route trips. At 10 arcs per route this yields 60,000 arcs. If each arc requires 3 pieces of data and each piece of data requires 2 bytes of storage, the 60,000 arcs will require 360,000 bytes of storage in total.

with the paging process, leading to a conservative estimate of one half to one second per response. Even when combined with the additional time for peripheral references required in obtaining the access and egress information, total time is unlikely to be increased by more than an additional .1 to .5 seconds. It is therefore expected that with current technology one second average responses are certainly possible even for large transit systems. This would allow 60 responses per minute; e.g. if average call length were as low as about 30 seconds, it could handle a maximum of 30 operators answering calls simultaneously. The technology of computer peripheral devices is rapidly advancing, so that much of this estimate may be reduced by a factor of 2 to 3 in the next few years, and responses of one third to half a second may be easily possible. We note that the estimates above refer solely to the time needed by the computer to process the itinerary request. It does not, of course, include data entry time, but it is anticipated that much of that activity can be done as the information is elicited from the caller in parallel with the reception and interpretation phase of the call.

It is clear from this analysis that the on-line calculation of shortest paths should be feasible even for large transit systems. In addition (ignoring differences in cost between the two approaches), on-line calculation is preferable to a precalculation approach because it allows changes in the schedule data to be input as they become known, making the information being provided much more current than otherwise available.

Choice among the two schemes, the time expanded network and the bipartite route/stop scheme, is more difficult. But since, as noted earlier, programming the shortest path scheme is neither difficult nor is it likely to be a bottleneck in the development of a PTPTM system, it is possible and desirable that both schemes be programmed and run on a large scale transit system data base, thus allowing a more direct operating comparison between them.

It is possible that some combination of the on-line and precalculation approaches may be most desirable, but time constraints limited the current effort to comparison of their "pure" forms. Specifically, pre-calculation could be used for speeding up answers for frequently asked origin/destination pairs, if such exist, or even for selected destinations which are often requested. Precalculation of a set of reasonable alternative routes can also speed up calculation of long paths, but at the expense of more storage. All of these possibilities should be investigated as ways of speeding up and streamlining a PTPTM system.

4. CONCLUSION

4.1 Summary

This report describes the components of a PTPTM system, characterizing them in a way which facilitates discussion of automating the various funtions. Data requirements and structures are defined and necessary software subsystems are identified. Step-by-step computational schemes are provided for the path finding component of the process.

From the point of view of automating responses to requests for point-to-point itineraries, the PTPTM process may be divided into four components (which are also present in handling a typical call to an existing manual information system):

1. reception and interpretation, in which the request is processed to ascertain what itinerary is desired, generally by eliciting and identifying trip origin, trip destination, and desired arrival or departure time,

2. location and connection, in which the appropriate origin and destination transit stops are identified for access to and egress from the transit system,

3. path calculation, in which a "best" transit itinerary is calculated for the desired trip request, and

4. response, in which the itinerary found in component 3 is conveyed to the person who requested the information.

Any or all of these functions may be automated, although 3 and 4 lend themselves to automation perhaps more easily than 1 or 2. How the degree of automation affects software requirements is discussed to some extent in Section 2 above. That section also describes the eight data files associated with the completely automated version of the PTPTM process:

1. Interpretation data are used in aiding translation of the words and phrases received into an acceptable list of locations.

2. Acceptable location descriptors are used in locating the caller's trip origin and trip destination.

3. Location/stop data provide for each acceptable location descriptor the transit stops which one might use from or to that location.

4. Stop/arc displacement data provide the information for treating arrivals and departures at stops not appearing in the reduced network on which the main calculations are performed.

5. Network and transfer data list the arcs of each route by giving the stops along that route, as well as any required transfer arcs between routes.

6. Schedule data list departure and arrival times at each stop along a route.

7. Fare data give the fare structure and any information required for fare computation.

8. Stop descriptor information describes each network transit stop in words, for transmission to the caller either by an operator or by a voice encoding.

Not all data files would be required for all possible levels of automation. For instance the interpretation data are not required by the current manual systems, in which this function is executed by the information operator who listens to the caller and presumably understands the request. (Of course a language barrier may in some instances make interpretation difficult.) Instead of a list of acceptable location descriptors, current manual systems usually use street maps with street names and major buildings marked on them. The list of descriptors thus becomes the list of names appearing on the map together with the interpretive ability of the operator. In a totally automated system the onus would be placed on the caller to provide a location descriptor which the system could interpret, perhaps aided by some system/caller interaction. A partially automated system might leave all the access and egress portions of the trip to an information operator, obviating need to have the first three files in computerized form. The last five files will be required by any system using automated path-finding; they are represented in a manual system by city maps, transit maps, fare charts, and headway schedule sheets.

It is estimated that of a typical call one third is spent in the reception and interpretation of the caller's request (component 1 and part of component 2 above), one third in retrieving the itinerary (part of component 2 and all of component 3), and one third in responding to the caller (component 4). The first step in automating a PTPTM system would be the computerization of the path finding process, component 3. The probable next step would be automating the response, component 4, followed by automating the location of origin and destination and connection to the transit system, component 2. The automation of reception and interpretation, component 1, would be the final step toward a totally automated process. Automating path finding could reduce call-length by less than one third, since additional data entry time will be required and exact location of trip origin and destination are unlikely to be automated at the start. A surrogate such as nearest street corner would undoubtedly be required, and its determination will use some of this one third of the call. How great a reduction is available from automated path finding depends also on the choice of computational scheme for use in the process. Previous crude estimates for large transportation systems, based on straightforward implementations of shortest path algorithms, are of the order of half a minute to several minutes per path calculation, too long to be competitive with manual methods for on-line calculation and even too long for reasonable precalculation. The estimates reached in this report, based on network reduction and carefully designed computational schemes, are of the order of a fraction of a second to one second per path calculation, making on-line path finding quite feasible.

Although automation of the path finding procedures of a PTPTM system would affect at most about one third of the total call length, the queuing model described in greater detail in Appendix B shows that a decrease of 20 percent in the time to answer a call can lead to a reduction of 60 to 70 percent in lost calls.

Section 3 and Appendix D provide two step-by-step procedures for calculating shortest paths in a transit network:

1. the time expanded network procedure in which each network node is a transit stop at a particular point in time and each arc represents one leg of one vehicle's route at a particular time during the day, and

2. the route/stop procedure in which network nodes correspond to the transit stops and the transit routes, and arcs connect a route to the stops along that route and a stop to the routes stopping there. Schedules are provided for each departure along each route with the appropriate one chosen as needed.

Two schemes are presented for each of these procedures; for the first procedure the two schemes address the departure and arrival oriented criteria respectively; for the second procedure one scheme is a variant of the basic label-correcting shortest path algorithm and the second is a variant of the basic label-setting algorithm.

A description is also provided for the scheme now used by transportation planners, which relies on average running times and headways rather than on explicit schedules. This approach is not recommended for further investigation since it does not allow accurate information on specific transfers and vehicles to be provided to the caller, unless the system is very regular. Appendix D also contains a description of a special step-function representation of path length which might be employed in compactly storing and computing precalculated paths, for use should on-line path calculation prove infeasible. This was not the case, since the estimates of on-line path calculation time obtained in Section 3 are about half to one second per path. The on-line calculation approach is recommended primarily because it allows real-time modification of the data base, thus utilizing the full capability of a computerized system for rapid response.

Thus the report contains a description of the data bases and data handling software required for a PTPTM system, together with two step-by-step procedures identified as the most promising alternaties for use in the path finding component of such a system. We find that whereas computerized shortest path schemes form a central part of the operation of an automated PTPTM system, early choice of such a scheme and its software implementation is less critical to PTPTM system development since the programs involved are relatively short and easy to code. Data base design and prototype development is a much larger effort; in particular the data base and software for use in location identification is probably the most difficult portion of the whole development effort.

4.2 Recommended Next Steps

Taking into account the findings summarized above, the following five steps are suggested by this report.

1. The work reported here addresses mainly the structuring of the data base, software, and algorithmic requirements of a PTPTM system, together with a discussion of its technical feasibility. A companion report [6] prepared concurrently with the present one raises questions about the cost-effectiveness of an automated approach to PTPTM. Further investigation of this question seems paramount. To this end it is suggested that a parametric cost-effectiveness model of a computerized transit information system be developed and applied as an aid in evaluating the utility of such a system. The model could use as a starting point the queuing model with finite queue length and reneging presented in Appendix B of this report. Much of the effort involved in such an evaluation would lie in the estimation and allocation of system costs and benefits accruing from an increase in the number of calls answered, a reduction in caller-experienced frustration resulting from shorter queues, and possibly reduced operator staff.

2. In order fully to evaluate any automated PTPTM system it will be necessary to exercise that system on a data base derived from some large existing system. The second suggested next step is therefore to set up such a data base. It might build on Run Cutting and Scheduling (RUCUS) files if they are available for any large system, or on any other existing computerized schedule data so as to minimize the manual labor required. This task could also include design and implementation of the editing and other data processing software needed to check, maintain, and update the data files, since some such software will be required for setting up even the first version of a data base.

3. In addition to an actual large-scale data base, it is desirable to design and develop a parametric procedure for generating synthetic networks to use in testing computerized PTPTM systems. Such a procedure would allow debugging of systems on smaller data bases, testing of systems during the interim before the larger data base is available, and, by varying parameter values, exercising systems on data bases with a variety of network characteristics to aid in evaluations of wider scope than possible using just one data base.

4. It is suggested that the schemes presented in Section 3.7 and Appendix D be coded and exercised on sample, small-scale networks to validate the claims made for their efficiency and to assess tradeoffs among schemes.

5. Two significant technical tasks were identified but left unaddressed by tne report:
    i. the design and evaluation of procedures for combining precalculation of some paths with the on-line approach in the hope of speeding-up query answering, and
    ii. the automation of the network reduction process to recognize efficiently a "minimum" set of important points and thus produce automatically the stop/arc displacement data base and the reduced network from the detailed network.

These topics merit further investigation. Clearly, if the analysis of Step 1 produces negative results then much of the remainder of the work would have only marginal value. However, a careful estimation of the costs of a PTPTM system may require at least prototype development, so that initial work on the remaining steps would in any case be required.

5. REFERENCES

1. C.J. Ancker, Jr. and A.V. Gafarian, Queuing With Reneging and Multiple Heterogeneous Servers, Nav. Res. Log. Quart. 10, No. 2, pp. 125-150, June 1963.

2. R. Bellman and R. Kalaba, On kth Best Policies, J. Soc. Indust. Appl. Math. 8, pp. 582-588, 1960.

3. George Cady, A "First" for Bus-line and Mixed Media Transportation Computerized Information Service, Bus Ride, December 1974.

4. Robert B. Dial, Algorithm 360 Shortest Path Forest With Topological Ordering [H], Comm. ACM 12, pp. 632-633, 1969.

5. Judith Gilsinn and Christoph Witzgall, A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees, National Bureau of Standards Technical Note 772, Washington, D.C., May 1973.

6. William G. Kienstra and Daniel J. Minnick, Point-to-Point Trip Management Program Preliminary Analysis, National Bureau of Standards Report NBSIR 75-665, Washington, D. C., February 1975.

7. Catherine S. Owen, John L. Vialet and Peter Wood, Application of Computers to Transit Information Services, Report MTR-6303, Vol. II, prepared by MITRE Corporation for Urban Mass Transportation Administration, January 1973.

8. Douglas R. Shier, Iterative Methods for Determining the k Shortest Paths in a Network, to appear in Networks.

9. Douglas R. Shier, Computational Experience with an Algorithm for Finding the k Shortest Paths in a Network, J. Res. NBS-B, 78B, No. 3, pp. 139-165, July-Sept. 1974.

10. UTPS Reference Manual, U.S. Department of Transportation, August 1974.

11. Christoph Witzgall, On Labelling Algorithms for Determining Shortest Paths in Networks, National Bureau of Standards Report 9840, Washington, D.C., May 1968.

APPENDIX A

UTPS: UMTA TRANSPORTATION PLANNING SYSTEM

"Developed by the Urban Mass Transportation Administration, the UMTA Transportation Planning System (UTPS) is a collection of IBM System/360-370 computer programs for use in planning multimodal urban transportation systems."[10] Each of the computer programs relates to one or more of three analytical categories: (i) network analysis, (ii) demand forecasting, and (iii) passenger loading. The network analysis programs aid in describing transportation networks, evaluating the levels of service provided by the various modes, and estimating operating costs. The demand forecasting programs utilize the results of the network analysis to estimate the number of passengers who will choose each mode of transportation. Then the passenger loading programs, using output from both network analysis and demand forecasting, assign passengers to links of the network to evaluate system effectiveness in terms of ridership versus capacity and cost. We will in this document be interested only in programs for network analysis, since these are the ones with possible relevance to PTPTM.

There are five UTPS programs concerned with network analysis: UNET, UPATH, UPSUM, UROAD, and UFMTR. Two of these programs, UROAD and UFMTR, appear to be of no relevance to PTPTM. Program UROAD is devoted to pathfinding, path skimming, and traffic assignment on highways, i.e., for automobile trips. UFMTR is primarily devoted to producing graphic and tabular reports for comparison purposes. Typical UFMTR output might include, for example, tables showing the number of trips leaving, entering, and remaining in each zone and/or plots showing the relationship between number of trips and trip fare. Neither of these two programs are relevant for use in connection with PTPTM.

There are three network analysis programs in UTPS which may be relevant for PTPTM. The three programs are related in that the output from UNET is input for UPATH and the output from UPATH is input for UPSUM. After briefly describing these three programs, the relevance of these programs to PTPTM will be discussed.

UNET. The function of UNET is to create or modify a computerized description of a transit network. The input to UNET consists of node, link, and line data as follows:

for each node,
(1) node number
(2) x and y coordinates

for each link,
(1) two node numbers
(2) mode number
(3) distance
(4) speed or time for a.m., p.m., and off-peak

for each line,
(1) mode number
(2) line number
(3) average headway for a.m., p.m., off-peak, night, and maximum*
(4) a sequence of link numbers describing the route

In addition to creating a network, UNET estimates fleet size and operating costs and produces a graphical display of the network. The following list gives the maximum (unless otherwise noted) network parameter values which UNET can accommodate.

nodes	8191
zones	2500
transit routes	1275
transit lines	255
stops per route	50
weighted** link time	25.5 minutes
number of times or speeds per link	3
zone-to-zone travel time	204 minutes
headways per route	5
minimum headway	.1 minute
maximum headway	99.9 minutes
non-transit modes (walk, auto, etc.)	3
transit modes (bus, train, etc.)	5

UPATH. The function of UPATH is to read the network description created by UNET and to calculate shortest paths between all or selected zones in the system. The criterion for selecting shortest paths is the sum of the path's weighted link times, wait times, and transfer times. The length of a path is calculated as

$$L = M + I + P$$

where

L = length of path
M = sum of weighted link times, RUN(K)*T(J)
I = sum of constrained weighted wait times, WAIT(K)*W(N)
P = sum of additional transfer penalties, ADD(K)

---

*Maximum headways are used to constrain the passenger loading programs which recompute headways based on loads.

**Weights are specified by the user for each mode.

and

T(J) = time on link J
W(N) = estimated wait time for line(s) N
RUN(K) = weighting factor for mode K in link time
WAIT(K) = weighting factor for mode K in waiting time
ADD(K) = penalty for each transfer to mode K

The following discussion attemps to clarify each of the terms involved in the path length calculation. The link times, T(J), are obtained directly from the link data input to UNET for each link J. The weighting factor, RUN(K), is a user supplied input to UPATH for each mode K. The products of these terms are summed over all links of the path to obtain M, the sum of the weighted link times.

The estimated wait times, W(N), are functions of the headways of the lines boarded when traversing the path. When more than one line can be used, a combined headway is calculated as half of the inverse of the sum of the line frequencies. For example, if two lines can be used, with average headways of 20 and 30 minutes, the sum of the frequencies is 1/12 = 1/20 + 1/30 so the combined unconstrained, unweighted headway is 6 minutes for this "line" which is actually a composite of two lines. This term is then weighted by the user supplied value WAIT(K) for mode K. The weighted value is constrained to fall within user supplied upper and lower limits for mode K, i.e., if the product W(N)*WAIT(K) falls outside the limits, it is reset to equal the limit nearest the computed value. The constrained weighted wait times are summed over all lines in the path.

The transfer penalties ADD(K) are user supplied additional times which are associated with each transfer to mode K. These values are summed over all transfers in the path. The user can also supply an upper limit on the total number of transfers.

Each run of UPATH calculates shortest paths for only one period of the day, e.g., a.m., p.m., midday, or night. For each zone pair, the output is a single shortest path which may involve travel by more than one mode. Exact departure/arrival times are not associated with these paths, and so UPATH cannot supply the shortest path at a particular time during the time period. Single mode shortest path trips can be generated by placing large weights on all but the desired mode.

Documentation of the actual computational scheme used in UTPS for path-finding is now in progress, but discussions with those familiar with the programs indicate that the scheme used is similar to the transportation planning scheme described in Appendix D.4.

In addition to calculating shortest paths, if provided with fare link data cards, UPATH computes zone-to-zone fares over the shortest paths. Fares need not be associated with every link, a desirable feature for modelling zone fare structures.

UPSUM. Program UPSUM reads the shortest path data output by UPATH and prints up to twelve zone-to-zone matrices detailing the component parts of the shortest path trips. The data available for output are: unweighted travel time on each of the (at most) 8 modes, number of transfers, initial wait time, transfer time, and total weighted time. Frequency distributions are produced for each output matrix.

UTPS was designed for use in the transportation planning process, for which the average values are both adequate for decision making and desirable because they require less work to concoct and modify. PTPTM is an operational, rather than planning, function and as such requires information at a level of detail specific enough to advise customers which vehicles to take and at which stops and at what times to board and alight. UTPS does not contain data at this level of detail and therefore is not alone adequate for use in PTPTM. In addition, as indicated in Section 3, programming the path finding element of PTPTM from scratch is not a major effort and is probably less difficult than adapting the already existing UTPS programs.

The major advantages of using the UTPS programs presumably would be (1) the utilization of multipurpose data which may already exist in some cities and (2) the association with currently existing UMTA procedures and programs. However, rather than finding the UTPS data base a satisfactory source of inputs, a PTPTM system requires data more closely associated with actual day to day operations (such as say the RUCUS system for transit scheduling). Compatibility with UMTA programming standards, if deemed important, can be accomplished by careful program design, using the UTPS approach without actually borrowing the UTPS programs.

APPENDIX B

A QUEUING MODEL FOR ANALYZING BENEFITS FROM REDUCING CALL LENGTH

We have noted earlier that automation of the path finding component of a transit information call is likely to reduce time for servicing the call by at most (and probably less than) one third. It is not the reduction in call length itself which is the main benefit from such automation, but rather the resultant decrease in time customers spend on "hold" waiting for a free operator, and the additional calls not receiving a busy signal. Lost calls, those which receive a busy signal, are a significant fraction of calls in some systems [6], and a reduction of one third in call length could (and as the analysis below demonstrates, indeed does) have a much larger effect on the number of lost calls.

To investigate the magnitude of that effect, we will develop in this appendix a queuing model of the point-to-point transit information service and then apply that model to a numerical example to illustrate its use in evaluating the lost calls recovered by reducing call length. Alternatively it could be applied to determine how a reduction in operator-time per call could be used to maintain a given quality of service with fewer operators and/or fewer "hold" lines (hence lower labor and equipment costs). The effects of faster service on other measures of system performance, such as waiting times and the number of callers who "give up" when the period spent on "hold" becomes intolerable, can also be estimated using the model.

Most large transit information systems currently use Automatic Call Distribution (ACD), a telephone system in which incoming calls are distributed on a first-come first-served basis until all operators are busy, when additional callers are requested to wait. The system has a limited capacity for handling customers on hold, and once this limit is reached subsequent customers receive a busy signal. Customers who are initially placed on "hold" may give up after a while rather than wait longer.

Such a system may be modeled as an s-server queue with maximum queue length r and with reneging (giving up) allowed. That is the approach taken in this appendix, under some simple straightforward assumptions, customary in queuing theory, which are chosen in part because they make the analysis tractable. Other assumptions might be used for alternative future model development, if it is desired to extend the model or its application in analyzing a wide range of levels of automation of a transit information system. A small illustrative example of such an analysis is also presented as an indication of the queuing model's use to evaluate the benefits, in terms of reducing lost calls, to be obtained from a decrease in part of the service time by automating the path finding procedure. The formula used to calculate the expected number of lost calls is given following the analysis of the example, and may be skipped by those readers not concerned with the mathematical details.

We consider a queue with s servers (the operators answering calls) and with space for r people to wait for service, i.e. r people can be on "hold". If the system contains s + r people, then necessarily s of them are occupying the s servers while the other r are filling up the available waiting capacity, so that any additional arrival must be turned away (a lost call). It follows that no more than s + r customers can ever be present in the system.

We make the following assumptions:

1. The queue is handled in a first come first served (FIFO) manner. (This is required only for the analysis of waiting times.)

2. The probability of one call arriving in the small time interval t to t + h is a h + o(h)*, the probability of more than one arrival is o(h), and the probability of no calls in the same interval is 1 - ah + o(h)*. Thus we assume a Poisson arrival process, sometimes called a completely random arrival pattern, with an arrival rate a. The Poisson distribution is customarily used to describe telephone call arrivals and other similar situations in which customers act independently without consulting one another. The arrival rate is constant and does not depend in any way on the condition of the system, such as when the last call arrived or how long the hold queue is.

3. The conditional probability that one person will finish service in the small time interval t to t + h, given that m people are being served at time t, where $0 \leq m \leq s$, is bmh + o(h), the probability of more than one completion is o(h), and the conditional probability that no one finishes service during that interval under the same assumption is 1 - bmh + o(h). This results from the assumptions that the service time for each server follows the negative exponential distribution with mean 1/b, and the service times of the different servers are independent of one another. These again are commonly made assumptions for situations like that here.

4. Any person waiting for service will renege (leave without service) if the time he has been waiting exceeds a number which is exponentially distributed with mean 1/c, and the waiting times before leaving of different people in the queue are independent. Thus the conditional probability that one person waiting for service will leave without it (hang up while still on hold) in the small time interval t to t + h, given that s people are being served (i.e. all servers are busy) and n people are in the queue, where $1 \leq n \leq r$, is cnh + o(h). This

---

*o(h) is used to denote any function f such that $f(h)/h \to 0$ as $h \to 0$.

assumption perhaps adheres less to the actual situation than the previous two. Whereas the negative exponential distribution decreases steadily from its peak at 0, one might posit the actual distribution of waiting time to have two peaks, one at a fairly low wait time, representing those who are very impatient and not really willing to wait, and a second at a higher value representing the average caller. In spite of this, the approximation may not be too bad for much of the analysis, and the negative exponential reneging time assumption greatly facilitates an analytic closed-form queuing description, not as available with other distribution assumptions.

In summary, the model involves five parameters: the two positive integers s and r, representing the number of operators and the "on hold" capacity of the system, and the three positive real numbers a, b, and c, representing the demand level, the rapidity of service, and the impatience of customers. Since each of these features seems an essential ingredient of the situation under study, and each is represented by just one parameter, no simpler reasonable approximation to the real system seems possible.

A formula will be given below for the expected number of calls lost to the system, i.e. those receiving a busy signal because the hold queue is full. Table B.1 records the calculated values, using that formula, for a manual system with 8 operators (s = 8) each capable of answering on the average 37.5 calls per hour (b = 37.5)*. Since we are primarily interested in busy periods during which the answering system is overloaded, three arrival rates were chosen, all greater than the service rate (8b) of 300 calls an hour: 330, 360 and 450 calls per hour. The reneging rate was arbitrarily chosen to depend on the arrival rate: .005 x a, .010 x a and .025 x a, leading to average wait before reneging times of 5 minutes to over half an hour. (These seem rather long; the actual values are unknown and would require further study.) Three cases were examined, one with maximum queue length r being 6, one at 8 and one at 16. The manual system was assumed to have the stated service rate (b = 37.5), while the reduction in average service time 1/b (slightly less than one third) from using an automated path finding procedure results in an increased service rate of 52.5.

Examination of Table B.1 shows that in all cases a reduction in service time of about 20 percent leads to a reduction in the number of lost calls of at least 40 percent, and typically 60 or 70 percent. The smallest reduction occurs when the system is very overloaded, so that it remains overloaded even with the increased service rate. Smaller reductions also occur with shorter maximum queue lengths, since the queue fills up quicker and stays filled up longer. Thus the size of the system obtaining greatest benefit from an automated path finding procedure is that for which a twenty percent

*These values pertain to characteristics of transit system 10 in [6].

TABLE B.1

Expected Lost Calls/Hr. With and Without Automated Path Finding

For an 8 Server Queue With Service Rate 37.5 Calls/Hr.

Arrival Rate (Calls/Hr.)	Reneging Rate (Calls/Hr.)	Queue Capacity 6		Queue Capacity 8		Queue Capacity 16	
		Without Computer	With Computer	Without Computer	With Computer	Without Computer	With Computer
330	1.65	49.1	15.8	41.4	11.3	24.0	2.3
330	3.3	45.8	15.6	36.8	9.9	15.6	1.4
330	8.25	37.8	12.5	25.7	6.8	4.0	.3
360	1.8	71.5	29.7	63.3	22.2	44.3	7.5
360	3.6	66.8	27.5	56.6	19.5	30.5	4.6
360	9.0	54.5	22.1	40.4	13.4	8.7	1.1
450	2.25	148.6	89.4	140.3	79.1	120.0	55.3
450	4.5	140.2	83.5	128.1	70.8	93.2	38.1
450	11.25	117.2	68.1	96.4	50.5	36.4	10.9

reduction in service time brings the total service rate up to a level for which the system is no longer overloaded. Comparison of the columns in Table B.1 headed "Queue Capacity 8 With Computer" and "Queue Capacity 16 Without Computer" shows that a greater benefit is generally to be derived from a reduction of 20 percent in the service rate than would occur from a doubling in the queue capacity. This also holds true for the situation with lower reneging rates for which there is an increase of 266 percent in queue capacity (from 6 to 16).

The 20 percent reduction in service time assumed to be available through automated path finding is equivalent to a 25 percent increase in the number of servers, or in this example an increase of 2 peak time operators. It is not apparent that use of a computer system would cost less than the salary and overhead costs for 2 or 3 more operators, though of course such saving would not be the sole benefit from the automation. Further investigation of actual computer costs and examination of benefits are needed to evaluate more fully the cost-effectiveness of an automated path finding procedure for PTPTM. If functions other than path finding were also automated, such as response to the caller, so that additional service time savings were realized, this too would affect the analysis. The example developed here is only illustrative of how the queuing model can be used for analyzing the benefits to accrue from automation of parts or all of the PTPTM process. Further development of this queuing concept, additional data collection and expansion of the cost effectiveness analysis would be required for a more definitive approach.

Reference [1] develops formulas for a queuing model similar to the one presented above, with the further option that b, the average service rate, can differ from server to server (i.e. different operators have different capabilities and some can handle calls better than others). Whereas in the current manual systems it is undoubtedly true that more experienced operators know the system better and can respond on the average more quickly, one would expect automation of the path finding portion of a call to even responses in the area in which the greatest difference occurs. Reference [1] also gives formulas for the special case of homogeneous service rate (i.e. all servers have the same average service rate). The formula below for lost calls was also developed independently by one of the authors and appears here in the format he used. Let p_i be the steady state probability that there are i people in the system including those being served as well as any who are waiting. Then, for the particular value $i=r+s$,

$$p_{r+s} = \frac{\dfrac{a^{r+s}}{s!\,b^{s}\prod_{i=1}^{r}(bs+ci)}}{\sum_{h=0}^{s-1}\dfrac{a^{h}}{h!\,b^{h}} + \sum_{h=0}^{r}\dfrac{a^{h+s}}{s!\,b^{s}\prod_{i=0}^{h}(bs+ci)}} .$$

A lost call results from the conjunction of r+s people in the system and the arrival of a call. Thus the probability of a lost call in the interval t to t+h is $ahp_{r+s} + o(h)$, from which the expected number of calls in a period of length ℓ can be shown to be $a\ell p_{r+s}$. This formula is the one used for calculating the expected calls lost to the system in the example reported in Table B.1.

APPENDIX C

ADDITIONAL AREAS OF CONCERN

This appendix contains discussions of four additional topics which are peripheral to the main thrust of this report (the path finding process for PTPTM), but which arose during the effort and we believe are worth noting.

(1) Not all inquiries to transit information systems concern trips which a customer desires to make from one point to another at a specific time. Complaints, lost-and-found requests, requests for general fare information, and requests for schedule pamphlets can be channeled to other numbers by listing in the directory separate phone numbers for the different operations and by repeating a message to those on hold giving the other numbers and preparing the customer to organize his point-to-point request. However, many requests, while not about a specific trip, may require access to the schedule data base concerning a particular line, route or departure, and it would be most logical to channel these requests to the PTPTM operator. Examples might include questions concerning the timeliness of the schedule pamphlet which the customer has (is it current? for how long?); questions about specific departures along a route (the first bus after 5 on the K-2 route? how late do express busses run down Connecticut Avenue? how often are departures along the L-6 route?); questions about routings (what routes stop at a particular corner? where does the L-2 route stop near the library? can you transfer from the K-3 to the L-2 at 16th and K?). In each of these cases it will be necessary to interrogate the schedules or routings in order to answer the request. Thus an automated PTPTM system will need to include a means of directly accessing various schedule and routing data using as keys the route number, line identification, time of day, or stop identification. Additional descriptive data will be required for the stops, identifying their actual locations and characteristics.

In spite of this, it will probably be impossible to answer all questions by referring only to the computerized data files. For instance, the question about where a route stops near the library would be so answerable only if the route actually stopped at the library, but would require additional knowledge of city geography if the route stopped two blocks away. Since operators in a manual system normally use route maps, such questions fall naturally within the scope of their current activities. The danger in a partially automated system is that operators will tend to rely increasingly on the computer to prepare responses, and will find it difficult to deal with questions which cannot easily be phrased in terms the computer has been programmed to understand. Training will tend to emphasize, as it should, communication between operator and customer and between operator and computer, while deemphasizing detailed knowledge of the transit system. Thus it may also be necessary that the transit information system provide a separate position staffed by specially trained personnel to answer questions which require greater knowledge.

(2) It is unlikely that point-to-point trip information dissemination is the only procedure which a transit system would automate. One might expect that run scheduling (perhaps using UMTA's RUCUS program) and system planning (perhaps using some of the capabilities of UTPS) would also be computerized before or concurrently with automated PTPTM. Other activities, such as payroll, would undoubtedly already be automated by most larger properties. Coordination among these various automated procedures is of paramount importance for smooth operation of all systems. This can be greatly facilitated by advance planning of common data structures and interfaces. Thus any design of data bases for PTPTM should rely on already existing or planned data bases for other functions. Redesign of data base structure should be contemplated if such will provide benefit to the additional new function without greatly complicating the old. Time and effort spent in careful planning and coordination of related activities is almost always repaid many times over. Tradeoffs between the efficiencies of specialization and general applicability must be weighed, and timely updating of similar data items in different data bases should be considered. Coordination among related subprogram areas should be done at the lowest management level possible, to provide for coordination of detailed computer and data structure specifications. Changes made in one area may affect related areas, and those effects should be assessed before the changes are implemented.

(3) Fares were treated briefly in Section 2.3.5, but most fare structures are more complicated than those described there, with many different fare systems coexisting. Among such systems are:

a) the fixed charge - whereby each passenger pays the same fixed fare for a trip no matter where his origin and destination are,

b) the zone fare - in which passengers pay a fare depending on their origin and destination zones. The zones are usually set up so that an incremental fare is added whenever a zone boundary is crossed.

c) a distance fare - in which the fare depends on distance traveled. This may also be regarded as an O/D structure, in the sense that separate fares are charged for each origin-destination pair. Such charges are often found in commuter rail systems in which the passenger usually purchases a ticket from a suburban station for a round trip downtown and back.

d) transfer fares - in which a charges is levied for each transfer or for more than so many transfers. In the latter case one may have to pay a new basic fare after a certain number of free transfers. Some care is exercised, usually, to insure that the trip proceeds expeditiously from origin to destination without side trips and looping back, and that transferring is necessitated by the transit system network, not performed to provide special advantage to the passenger. Time limits, to prevent passengers from using transfer stops for accomplishing personal business rather than solely for transferring, may be imposed.

e) special fares for certain groups - The elderly and school children often receive reduced fares subsidized by other funds. The handicapped may also fall into this category. Members of the favored groups may have to purchase special tickets at designated ticket outlets or may only be required to show special identification.

f) off-peak fares - To encourage off-peak ridership, some systems have introduced reduced fares for off-peak travel. Such fares may be limited by other criteria e.g., applying only to special groups or only for certain trips.

g) multiple-ride fares - in which special tickets good for several rides are issued. Such tickets can be issued for a particular route for a fixed time period (e.g. 30-day commutation ticket from Suburb to City), for a particular number of trips on a particular route (10 trips from Suburb to City), or for any trip anywhere during a particular time period (similar to the 30-day Eurorail Pass). Even the good old round trip ticket can in a sense be thought of as a two-ride-for-reduced-price ticket.

The preceding list is not intended to be an exhaustive treatment of the fare question, but is designed to show the complexities of existing fare structures and thereby to point out the difficulties of providing fare information to the caller. We do not mean to imply that such difficulties are insurmountable or even to discourage the inclusion of fare calculation (or quotation) as part of a PTPTM system. We only wish to point out that fare systems are complex, and will thus need complex and careful treatment in the computer programs, as well as requiring additional information from the caller to establish his fare category.

(4) The access, egress and transfer times used in the automated PTPTM system would be those required by a "typical" person. Perhaps several values could be provided, representing slow and fast walkers. However, it would be difficult to represent adequately the problems encountered by the infirm or handicapped, without using special path tracing criteria and additional characterization of transit stops. Transit systems are sometimes required by law to provide facilities for the handicapped (Washington's METRO was required to design facilities for wheelchair passengers, for example), but planning trips for this special group of passengers would require additional effort. A trip which might be optimal for most transit passengers might be impossible for a person unable to walk 2 blocks to transfer. The stops available to the handicapped passenger may be fewer in number, both because of special characteristics such as curb heights and steps and because of distance. Transferring may be difficult if not impossible in some places. The handicapped or infirm may feel especially vulnerable to street attack and therefore desire to avoid certain sections of a city. It is possible that transit vehicles themselves may differ in their accessibility to those unable to ascend steps.

The extent to which these special characteristics and criteria are actually contained within the computer coding depends on the values placed by the transit system on providing services to this special group of passengers. Some accommodation can be made for the handicapped or infirm using the existing data base and the limited transfer criterion. The longer times for access and egress can be represented by requesting a trip starting later than the desired departure time to allow for additional access, and similarly adding additional egress time to the end or requesting an earlier desired arrival time. If the set of stops is limited and the passenger knows this, he can request his trip giving particular feasible stops as origin and destination. These "fixes" are not foolproof, but together with additional notations (such as "high curb", "flight of steps", etc.) about stops they may provide adequate additional information for those unable to negotiate such obstacles. More elaborate treatment would be required to satisfy fully the requests of this special class of passengers.

APPENDIX D

STEP-BY-STEP COMPUTATIONAL SCHEMES FOR PTPTM

This appendix contains step-by-step instructions for several computational schemes for PTPTM. Two schemes are presented using the time expanded network stored in forward star form, one scheme for the departure oriented and a second for the arrival oriented criterion. Next we present two alternative bipartite route/stop schemes as well as a method of compactly representing the paths output from the scheme for use in precalculating some trips. Finally, we describe a scheme based on the transportation planning approach.

D.1 Time-Expanded Network Scheme

Nodes in the time-expanded network are defined by a pair of entities, the geographical transit stop (a separate node for each route) and a time of day. Thus the geographical node Sixteenth and K Street will become several nodes, one for each time a transit vehicle stops there. The network arcs become transit trips departing one stop at a particular time and arriving at another stop at a different (later) time. Transfer arcs, representing allowable transfers (i.e. those obeying minimum transfer times) may be coded directly. An example of such a network is depicted in Figure D.1. In this example there are 4 transit stops and 3 bus lines: a local stopping at each stop (its two daily runs are represented by the paths 1→3→6→8 and 12→16→17→20), a faster vehicle starting at the second stop of the first route and proceeding directly to the last stop of that route (with four runs: 5→7, 9→10, 14→15, and 18→19), and one coming from the last stop of the first route back to the next to last stop of that route (its runs are represented by 2→4, 11→13, and 21→22). Two transfer arcs, 3→9 and 14→18, have been included. Note that in the left transfer, one is prevented by a minimum transfer time restriction from making the earliest vehicle on the second line.

The 4 stop network has been transformed into one with 22 nodes and 15 arcs. In general, time-expanding the network greatly increases the number of nodes, in fact by a factor equal to the average number of transit vehicle departures per grographical node. Generally it is desirable to decrease rather than increase network size, but the fact that the resulting time-expanded network is acyclic means that the increase is likely to be beneficial. On an acyclic network the network nodes can be numbered at the outset in such a way that for each arc (i, j), $i<j$. That is, in the numbering, arcs always lead from lower numbered nodes to those with higher numbers. Of course a similar property also holds true of paths. This limits the search for path extensions to nodes whose numbers are greater than nodes already in the path, so that nodes can be interrogated in the order of their numbering. The node order assumed below is the one determined entirely by the time component of node identity, except for "ties" which cause no trouble unless some arc requires no time to traverse. If only major stops are included this situation is unlikely to occur, but if it were, procedures exist to number nodes having identical time components.

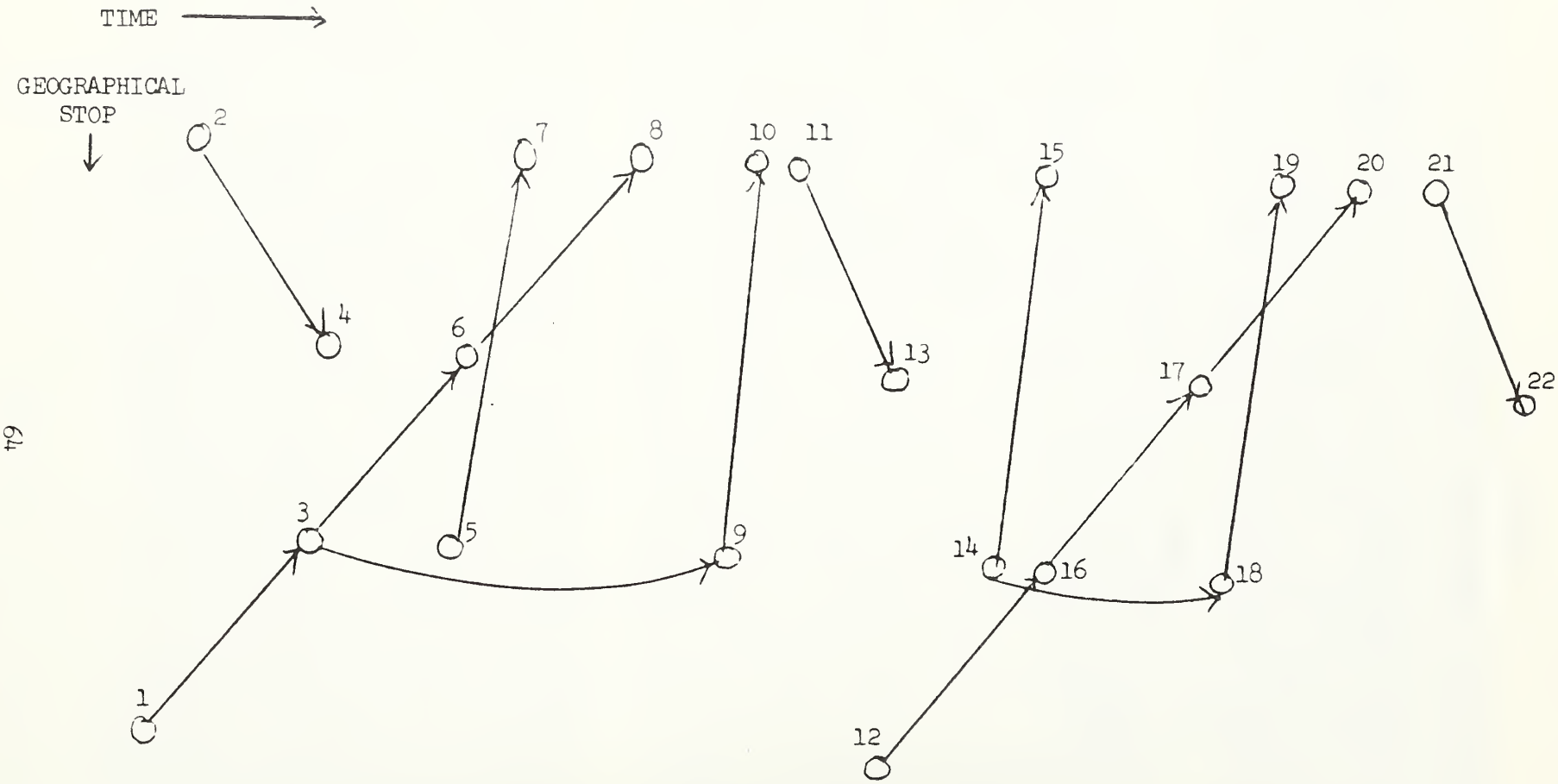


FIGURE D.1

Illustrative Time-Expanded Network

The node numbering procedure allows the network to be broken up into pieces (which will be called pages) so that the computational scheme only needs one of them at a time and can finish with the current one before needing the next.

The computational scheme proposed for computing transit paths in such a network appears below and is essentially the basic label-correcting scheme. We will use the following notation:

n(i)	the geographical transit node associated with the network node numbered i
t(i)	the time associated with network node numbered i
ORG	geographical node of transit origin (Note - this may actually be a list of several nodes, in which case it will be necessary to check the whole list when asking if n(i) is actually ORG.)
DST	geographical node of transit destination
P(i)	the network node preceding i in a best path from ORG to the network node numbered i
DONE	the first node associated with DST encountered (i.e. the one for which t is minimum) in a path starting at a node associated with ORG

The computational scheme proceeds through the following steps:

Initialization: DONE= ∞ ; P(i)=0 for each node i.

Step 1: Scan the arc list starting with the first node i for which t(i) is not less than the desired departure time. Let i be the first node encountered with n(i)=ORG.

Step 2: Let a=(i,j) be the first arc originating at node i. If there are none, go to Step 6.

Step 3: If P(j)$\neq$0, go to Step 5. Otherwise set P(j)=i.

Step 4: If n(j)$\neq$DST, go to Step 5. Otherwise set DONE=min (DONE, j).

Step 5: Let a=(i,j) be the next arc originating at node i, if there is one, and go to Step 3. Otherwise continue.

Step 6: Let i=i+1. Stop if i=DONE.

Step 7: If P(i)=0 and n(i)$\neq$ORG, go back to Step 6. Otherwise go to Step 2.

It is clear from this description that only the nodes numbered between the first departure from ORG after the desired departure time (which we shall call i') and the node DONE are examined as arc origin nodes. In most instances this should be considerably fewer than the total number of nodes in the network. To take advantage of this fact, one may store information about node i in position i-i'+1 in the P array. In fact the actual number

of active nodes at any time is considerably less than this, and information may be stored in P in a rotating fashion using storage only for the range of active node numbers. The values to be represented in P can be shortened to their increments relative to i'-1, which should generally be of the order of 10,000 or less, requiring only about 14 bits to store each. The n and t values are required only for the same set of nodes as P, and n requires storage for numbers of about the same order as P (say 12 or 13 bits), while t requires about 11 bits (since there are 1440 minutes in 24 hours).

The algorithm described above requires only one pass through the nodes, and only a subset of the nodes at that. No sorting or sequencing of nodes is necessary, since nodes are examined in numerical order. Since arcs are stored sorted by origin, only that portion of the network originating at nodes i' through DONE need be referenced for this path calculation, leading to an efficient paging scheme for the network. The computer program can be set up to store a fixed number of arcs, depending on the size of the computer or of memory-share for the PTPTM function. An arc in this application only requires identification of its origin and destination nodes since the t and n pointers describe the relevant arc characteristics. Actual estimates of page size and of the computer access and transfer times involved are given in Section 3.8. The main point to be made here is that although the time-expanded network seems to enlarge the data base to be handled, the fact that this representation is an acyclic network means that a particularly efficient computational scheme can be used and the scheme lends itself to a direct paging procedure. Two additional advantages of the time-expanded network scheme are that it handles both the overtake problem and that of constraints in the best path criterion, producing optimal paths in both situations, a characteristic not shared by most schemes which have only one node for each geographical stop. (See Section 3.7 for discussion of this point.) It furthermore provides a ready vehicle for the selective inclusion of transfer penalties.

The same scheme, used with the time-expanded network stored in backward star form, may be applied for the arrival oriented shortest path criterion. Alternatively, a modified version of the above scheme may be applied, using the forward star form and examining the nodes in reverse order starting from the last node associated with DST whose time is before the desired arrival time. This scheme will be described below. Use of two schemes has the advantage that only one copy of the network, the forward star form, need be stored to handle both the departure oriented and arrival oriented criteria. This is particularly necessary for the time-expanded network because of its large size. Different schemes are then applied to the network for the two criteria, the one above for the departure oriented criterion and the one below for the arrival oriented criterion. The following array and variable will be used in describing the scheme, together with the arrays n and t and variable ORG and DST listed above:

S(i) the network node succeeding i in a best path from the network node numbered i to DST

FIN the first node associated with ORG encountered (i.e., the one for which t is maximum) in a path ending at a node associated with DST.

The computational scheme proceeds through the following steps:

Initialization: FIN=0; S(i)=0 for all i.

Step 1: Scan the arc list backwards, starting with the last node i for which t(i) is at most the desired arrival time. Let i be the first node encountered with n(i)=DST. Set S(i)=i. Go to Step 6.

Step 2: Let a=(i,j) be the first arc originating at i. If there are none, go to Step 6.

Step 3: If S(j)=0, go to Step 5. Otherwise set S(i)=j.

Step 4: If n(i)≠ORG, go to Step 5. Otherwise set FIN=max(FIN,i).

Step 5: Let a=(i,j) be the next arc originating at node i, if there is one, and go to Step 3. Otherwise continue.

Step 6: Let i=i-1. Stop if i=FIN.

Step 7: If n(i)≠DST, go to Step 2. Otherwise, set S(i)=i and go back to Step 6.

D.2 Bipartite Route/Stop Scheme

The nodes of the bipartite route/stop network are of two types, one representing the geographical transit stops and the second representing individual transit routes. Network arcs are also of two types: for each transit stop an arc connects it to those lines stopping there, and for each route an arc connects it to the stops along that route. (The arcs associated with a route should appear in the order of the stops along the route.) Thus the network described here is bipartite as defined in Section 3.1. Figure D.2 displays an example of such a network. Note that a dummy route was introduced for a walk transfer link connecting two other routes. A path in this network is an alternating list of stops and routes, beginning with the origin stop and ending with the destination stop. The route node appearing between each pair of stops specifies the route which should be taken between them. The number of transfers is thus one less than the number of lines appearing in the list, or alternatively, since each stop other than the origin and destination represents a transfer, two less than the number of stops in the path.

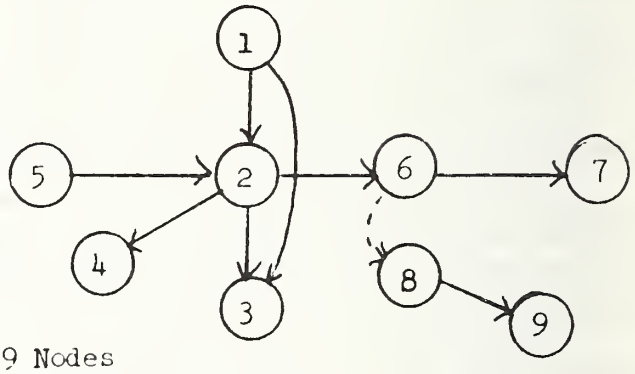
The network as described above does not have associated with it the time data specifying each discrete departure. The arcs connecting the routes to the stops actually represent a whole list of route scheduled trips, to be fetched during the course of the algorithm as needed. An example of such a list for one route is given in Figure D.3. Each column gives times at a stop and each row represents one transit vehicle's trip along the route. Thus if the arcs emanating from a route node are listed in the order of the stops

FIGURE D.2

Illustrative Bipartite Route/Stop Network

ORIGINAL NETWORK

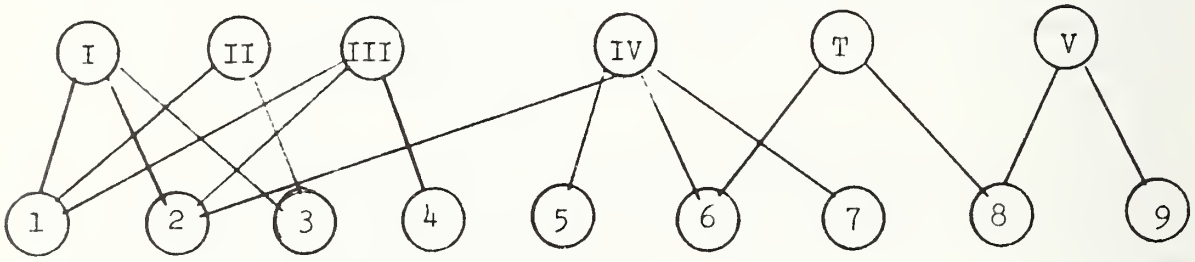
Route	Stops
I	1, 2, 3
II	1, 3
III	1, 2, 4
IV	5, 2, 6, 7
V	8, 9
T	6, 8



9 Nodes

9 Arcs

BIPARTITE ROUTE/STOP NETWORK



15 Nodes

32 Arcs (Each Connection is Two-way)

FIGURE D.3

Illustrative Route Schedule

Stop 1	Stop 2	Stop 3	Stop 4
7:00	7:10	7:15	7:30
7:30	7:45	7:55	8:10
8:00	8:15	8:25	8:40
8:30	8:40	8:45	8:55
10:00	10:05	10:09	10:15
12:00	12:07	12:15	12:25
2:00	2:05	2:09	2:15
4:00	4:07	4:15	4:25
4:30	4:40	4:50	5:05
5:00	5:15	5:25	5:40
5:30	5:45	5:52	6:05
6:00	6:10	6:15	6:25
6:30	6:40	6:45	6:55
8:00	8:05	8:09	8:15
10:00	10:05	10:08	10:13

along the route, a row of Figure D.3 represents arrival times at the arc endpoints in order. We will describe two computational schemes based on this network, one using the basic label correcting procedure and a sequence list ordered by cardinality distance (cf. Section 3.3, paragraph 3), and the second using the label setting procedure and a list ordered by temporary label (in this case trip arrival time). The following notation will be used in describing the schemes.

R the set of all transit routes
r a particular route

S the set of all transit stops
s a particular transit stop

ORG the origin transit stop

DST the destination transit stop

N $R \cup S$ the nodes of the whole bipartite network

T(i) arrival time at stop i via best path from ORG, for $i \in S$;

P(i) node preceding i in best path from ORG (Note that if $i \in R$ then $P(i) \in S$ while if $i \in S$ then $P(i) \in R$.)

L(k) sequence list of nodes in S, developed by the scheme, indicating the order in which they are to be fanned out from. In the label-correcting method L is maintained in cardinality distance order and in the label-setting method is ordered by arrival time.

F(i) position of node i in sequence list L.

u current position in the sequence list

v last position filled in the sequence list

END last entry in sequence list L

A computational scheme for a label-correcting procedure for use with the bipartite route/stop network is given below.

Initialization: Set $T(i) = \infty$ for all $i \neq ORG$ and set T(ORG)= desired departure time. Set P(i)=0 and F(i)=0 for all $i \in N$. Set u=0 and v=0.

Step 1: Let i=ORG. Let r be the first listed route stopping at i.

Step 2: Search the schedule for route r for the first departure from i at or after T(i). Let s be the first stop occurring after i in route r.

Step 3: Compare the arrival time at s of the scheduled vehicle found in Step 2 with the current value of T(s). If it is no less, go to Step 6; if it is less, set P(s)=r and P(r)=i.

Step 4: If $T(s)=\infty$, go to Step 5. Otherwise let k=F(s). If k>0, remove s from its previous position by setting L(k)=0.

Step 5: Set T(s)= the new value at node S. Let v=v+1. If v>END, set v=1. Set L(v)=s and F(s)=v.

Step 6: Let s be the next stop on route r, and go to Step 3. If there are no more stops on r, let r be the next route stopping at i and go to Step 2. If there are no more routes stopping at i, set F(i)=0.

Step 7: If u=v, stop. Otherwise let u=u+1. If u=END, let u=1. Let i=L(u). If i=0, repeat Step 7. Otherwise let r be the first route stopping at i and go to Step 2.

These computations do actually maintain the sequence list L in cardinality distance order. Note that successive path segments are always by different routes, so that cardinality distance is associated with the number of different routes used in a path. ("Actual" cardinality length of a path in this network is twice the number of routes used since paths consist of an alternating stop-route sequence. However since L contains only stops, it can be used in obtaining directly the number of routes used.) If it is desired to consider only paths using no more than some maximum number of routes, say r_{max}, then the computational scheme given above can be modified easily to accommodate this additional constraint, utilizing two additional pointers:

m the cardinality distance (actually the number of routes) used in the current path from ORG to node i.

j the position in L of the last node of cardinality distance m.

Both m and j are initialized at 0. The computational scheme is modified by the addition of a Step 6 1/2 between Steps 6 and 7 above.

Step 6 1/2: If u=j, let m=m+1. If $m=r_{max}$, stop. Otherwise, set j=v.

A label-setting scheme for use with the bipartite route/stop network is similar to that given above, but the sequence list L is kept ordered by arrival time at each node. The length of L is determined by M, one plus the maximum arc length. See [5] for a more complete discussion of the label-setting procedure. Since for transit networks the transferring time must be included in M, it is perhaps easiest to set it at some reasonable trip length level (say 3 hours, or if desired 24 hours).

Initialization: Set $T(i)=\infty$ for all nodes $i \neq$ ORG and T(ORG)= desired departure time. Set P(i)=0 for all nodes i. Let u be one plus the desired departure time (mod M).*

*By X(mod M) is meant the remainder when X is divided by M. Its calculation is uaually available in FORTRAN through the function MOD(X,M).

Step 1: Let i=ORG. Let r be the first route stopping at i.

Step 2: Search the schedule of route r for the first departure from i at or after T(i). Let s be the first stop occurring after i in route r.

Step 3: Compare the arrival time at s of the scheduled vehicle found in Step 2 with the current value of T(s). If it is greater, go to Step 5; if it is less, replace the old T with the new value and set P(s)=r and P(r)=i.

Step 4: Let k=T(s) (mod M). Store s in position k+1 of L.*

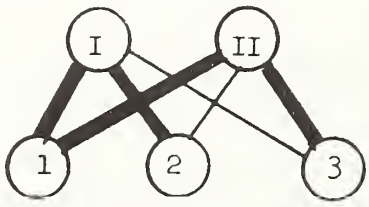
Step 5: Let s be the next stop on route r and go to Step 3. If there are no more stops on r, let r be the next route stopping at i, and go to Step 2. If there are no more routes stopping at i, continue to Step 6.

Step 6: Let u=u+1. If u>M, let u=1. If L(u)=DST, stop. Otherwise, let i=L(u). If i=0, repeat Step 6. Otherwise let r be the first route stopping at i and go to Step 2.

Note that termination occurs when DST is the node to be fanned out from. Thus if DST is fairly close to ORG the label-setting procedure requires much less calculation than the label correcting procedure, which terminates only after best paths have been found to all nodes.

Both of these computational schemes for use with the bipartite route/stop network can be combined with a paging scheme for storing the network so that Step 2 in both algorithms searches first that portion of the scheduled departures for route r which occurs in a time period containing the desired departure time. That is, instead of putting all schedules for route r together, those for all routes for a particular time period are grouped (but within the period grouping is by route).

As long as departures along the same route do not pass one another, the schemes presented above handle the overtake problem, since although a route goes through an intermediate node in reaching another, it is not necessary that it be used as a first segment of that trip if a better route is available to the intermediate node. Such a situation is pictured below. The best path



*Since paths from ORG to different nodes may arrive at the same time, it is necessary to accommodate several nodes in the same position in L. The mechanism for doing this is complicated and will not be described here, but is described in full detail in [4] and [5].

(shown in heavy lines) from stop 1 to stop 2 uses route I, but the best path from 1 to 3 uses route II. Thus the bipartite route/stop scheme handles both the overtake problem and constraints on the number of transfers.

D.3 Precalculating Paths

In the two schemes presented above, we assumed that the problem to be solved is "find a path from ORG to DST starting after a desired departure time and arriving soonest." We were in effect also assuming that the schemes are being used in an on-line mode, in which paths are calculated as they are requested. In this section we present a procedure which modifies the bipartite route/stop scheme by introducing a different mathematical representation for the best path criterion. This procedure will output shortest paths from one node ORG to all other nodes for all day long in a compact form.

Consider the following three step functions:

$$R(s,t) = \begin{cases} r_1 & \text{for } t_o \le t \le t_1 \\ r_2 & \text{for } t_1 \le t \le t_2 \\ \vdots & \\ r_{k+1} & \text{for } t_k \le t \le t_o + 24^* \end{cases}$$

$$S(s,t) = \begin{cases} s_1 & \text{for } t_o \le t \le t_1 \\ s_2 & \text{for } t_1 \le t \le t_2 \\ \vdots & \\ s_{k+1} & \text{for } t_k \le t \le t_o + 24^* \end{cases}$$

$$T(s,\tau) = \begin{cases} \tau_o & \text{for } \tau_o \le \tau \le \tau_1 \\ \tau_1 & \text{for } \tau_1 \le \tau \le \tau_2 \\ \vdots & \\ \tau_{j+1} & \text{for } \tau_j \le \tau \le \tau_o + 24^* \end{cases}$$

---

*We use 24 here as an example; the actual number depends on the units: 24 if hours, 1440 if minutes.

We interpret these three functions in the following way. R(s,t) is the last route used in the best path from ORG to s starting at ORG no earlier than time t. The values $t_0, t_1 \ldots t_k$ are times at which the best last line changes. S(s,t) is the stop immediately preceding s in the best path from ORG to s. Presumably S does not have to change whenever R does and vice versa, but it will probably most often be the case that they both change at the same time, so that setting up the time breakpoints to be the same for the two will only require extra work when exactly one of the pair change. $T(s,\tau)$ is the soonest arrival time at s when starting from ORG at time τ. It is useful to note that $j \geq k$, and in fact in most instances j would be much larger than k since one would expect that only a few different routings (i.e. sequences of routes used) would be found optimal, while T will change whenever a different vehicle of the same route arrives.

These three step functions together with the schedules contain all the information needed to retrieve a best path from ORG to DST starting at or after any time t. We will illustrate the procedure below. T(DST,t) is the arrival time at the destination transit stop, r=R(DST,t) is the last route used and s=S(DST,t) is the stop preceding DST in the path from ORG to DST. If s=ORG then we have found the best path and to get the departure time from ORG it is only necessary to look up in the schedules that departure by route r which arrives at DST at T(DST,t). If $s \neq$ ORG then we must go back one further step, retaining the r and s found so far as the last segment of the best path and setting a new r=R(r,t) and s=S(s,t). This is continued until s=ORG, retrieving departure times from the schedules as described above for each segment.

The three functions may be used in a variant of the label-correcting scheme, described for the bipartite route/stop network in the previous section, with R and S taking the place of P. Rather than replacing an old T by a new one when a better arrival time is found or storing r and s in the appropriate positions of P, a special operation combining successive T's, R's, and S's in a path is required. An example of the operation applied to T is pictured in Figure D.4 In symbols,

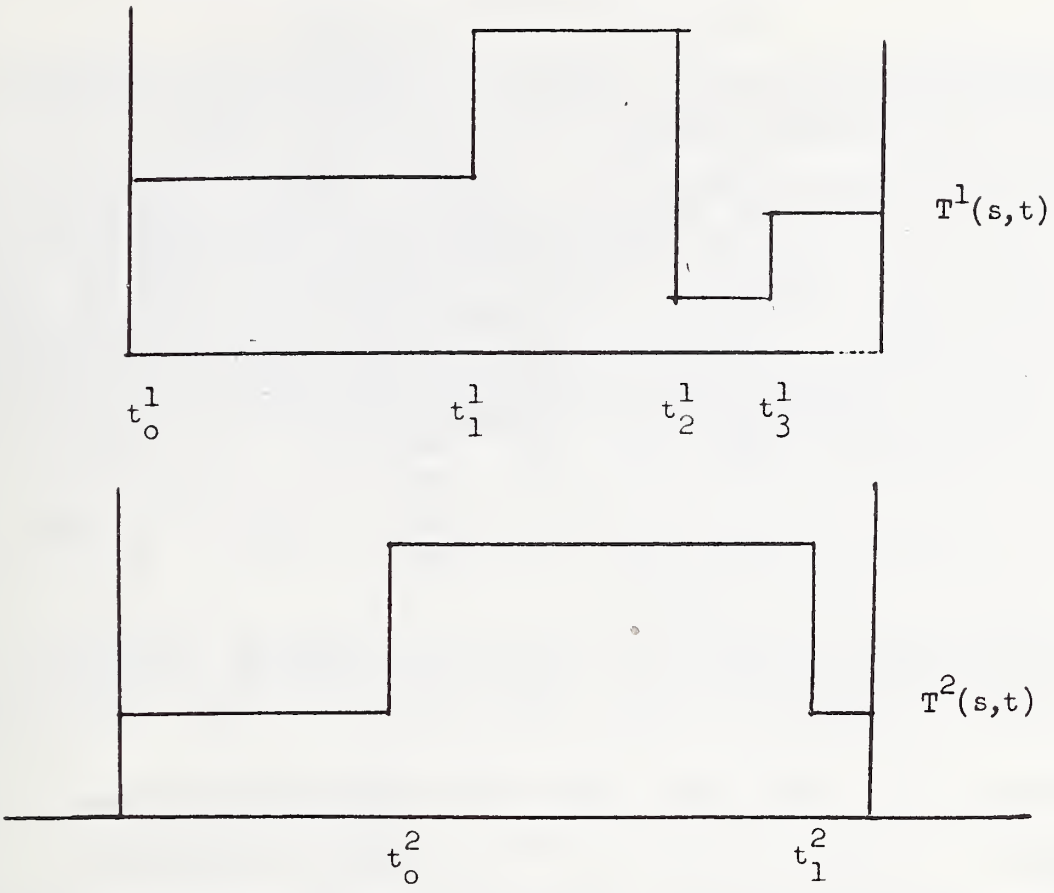
$$T^1(s,t) \oplus T^2(s,t) = \min\,(T^1(s,t),\ T^2(s,t)).$$

Whereas T^1 has four time breakpoints and T^2 has two, the sum $T^1 \oplus T^2$ has five. The point t_0^1 is not a breakpoint since T^2 is less than T^1 on both sides of t_0^1 and T^2 has the same value on both sides. In general each application of $\oplus$ will increase the number of breakpoints, at most (but not typically) to the sum of the two numbers of breakpoints. The R and S functions are combined in association with the T function as follows:

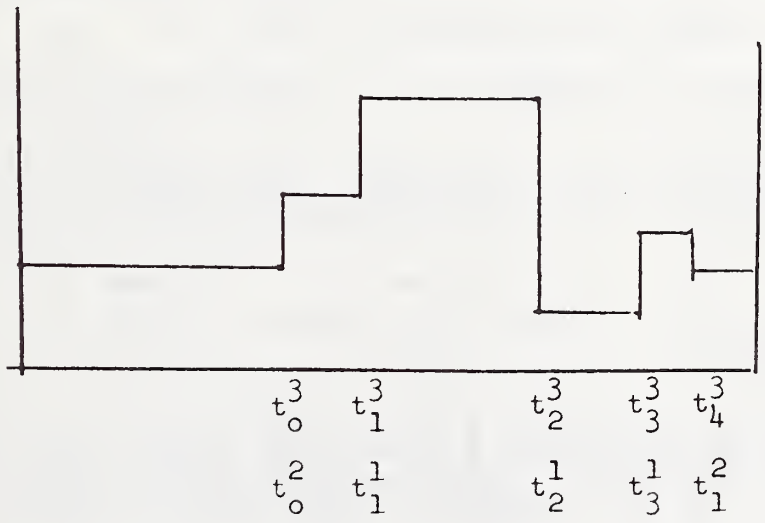
$$R^1(s,t) \oplus R^2(s,t) = \begin{cases} R^1(s,t) \text{ for } t \text{ such that} \\ \quad T^1(s,t) \oplus T^2(s,t) = T^1(s,t) \\ R^2(s,t) \text{ for } t \text{ such that} \\ \quad T^1(s,t) \oplus T^2(s,t) = T^2(s,t) \end{cases}$$

FIGURE D.4

The Operation $\oplus$ Combining Step Functions



$\oplus$ combines these two functions into:



$$T^3(s,t) = T^1(s,t) \oplus T^2(s,t)$$

and similarly for S. In more conventional mathematical notation, $R^1 \oplus R^2$ is equivalent to $\min(R^1, R^2)$. The number of breakpoints for R and S will also probably increase with each application of $\oplus$.

The computational scheme will also require simple function addition $T(s,t) + \ell(t')$, and composition of functions, in the form $\ell(T(s,t))$. The network will be stored in arc form where each arc is identified with a specific route r and segment (i, s). The length $\ell(t)$ associated with each arc will be the step function representing the arrival time at s when starting from i at time t. In referencing ℓ during the course of the computation, it will be necessary in adding ℓ to the current path length to add that value of ℓ determined by the path arrival time at the intermediate node, which is denoted by $\ell(T(i,t))$.

The computational scheme proceeds in a manner similar to the label-correcting scheme already presented above in the previous section. (We will also use the notation of that section.)

Initialization: Set $T(s,t) = \infty$ for all $i \neq ORG$, and $T(ORG,t) = t$ for all t. Set $R(s,t) = 0$ and $S(s,t) = 0$ for all s and t. Let u=v=0. Let $F(i) = 0$ for all i.

Step 1: Let i = ORG. Let r be the first route departing i.

Step 2: Let s be the first stop occurring after i on route r.

Step 3: Let $T'(t) = T(i,t) + \ell(T(i,t))$. Let $T(s,t) = T(s,t) \oplus T'(t)$, updating R and S at the same time. If there was no change in T, go to Step 6.

Step 4: Let $k = F(s)$. If $k>0$, set $L(k) = 0$.

Step 5: Let v=v+1. If v>END, set v=1. Set L(v)=s and F(s)=v.

Step 6: Let s be the next stop on route r, and go to Step 3. If there are no more stops on r, let r be the next route departing i, and go to Step 2. If there are no more routes departing i, set F(i)=0.

Step 7: If u=v, stop. Otherwise, let u=u+1. If u=END, let u=1. Let i=L(u). If i=0, repeat Step 7. Otherwise let r be the first route departing i, and go to Step 2.

This scheme can be applied with ORG being each node in the network successively. The three arrays T, R and S can then be stored for use in retrieving a path as it is requested during a call. On-line computation would consist of access calculations and unpacking the path stored in T, R and S.

D.4 Transportation Planning Scheme

This scheme may be used if headways are short (frequent service is provided) or if it is acceptable to provide only routing and a general idea of the frequency of service along pieces of the routing. Actual schedules are not used. Instead, arcs are described by a running time and an average headway. Different values of each may be provided for different times of day (for instance morning peak, evening peak, mid-day, and evening periods). For each origin one set of paths to all destinations is computed and stored using the arc data for each of the time periods separately. Since average headways and running times are available, the best path is computed as that during the time period selected whose total trip length is least. Because actual transfer information is unavailable, transfer time is approximated as half the headway of the line to which one is transferring. This is added into the trip length computation, which then consists of a series of half the headway, running time, half the headway, running time, etc.

The schemes presented in Section D.2 for the bipartite route/stop network can be modified for use here. One using the label-setting method will be presented below for consideration. Notation will agree with that in Section D.2 above, with the two additional functions describing the network:

t(a) running time on arc a

h(a) average headway on arc a.

Initialization: Set $T(i)=\infty$ for all nodes $i \neq ORG$ and set $T(ORG)=0$. Set $P(i)=0$ for all nodes i. Let $u=0$.

Step 1: Let $i=ORG$. Let r be the first route departing i.

Step 2: Let s be the first stop after i on route r and let a be the arc connecting i and s by r.

Step 3: Compare $T(i)+t(a)+.5h(a)$ with $T(s)$. If the first is larger, go to Step 5. Otherwise replace $T(s)$ with the new value and set $P(s)=r$ and $P(r)=i$.

Step 4: Let $k=T(s)(\text{mod } M)$. Store s in position $k+1$ of L. (See footnotes referring to this process in the earlier description in Section D.2.)

Step 5: Let s be the next stop on route r, and let a be the arc connecting i to s by r. Go to Step 3, unless there are no more stops on r. In that case let r be the next route departing i and go to Step 2. If there are no more routes stopping at i, continue to Step 6.

Step 6: Let $u=u+1$. If $u>M$, let $u=1$. Otherwise let $i=L(u)$. If $i=0$, repeat Step 6. Note: if the whole list has been searched and no $i>0$ was found, we can stop. Once $i \neq 0$, let r be the first route departing i and go to Step 2.

Weights may be introduced so that initial waiting time, running time, and transfer time contribute different amounts towards the final optimality criterion. A major advantage of using this scheme is that few enough paths are generated that the complete set could be stored in full, without packing, for later retrieval. In addition, in any system in which adherence to schedule is erratic, it may be preferable to provide routing with the information on average headways for each segment, since paths calculated based on the ideal detailed schedules may be unattainable much of the time. When headways are small (i.e. service is frequent), half a headway is likely to approximate transfer time better than when they are larger. In addition, when headways are small they are a smaller proportion of the total trip, and so a large percentage error in transfer time still has a relatively minor effect on the total trip time.

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 75-676	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Path Finding Algorithms and Data Structures For Point-to-Point Trip Management

5. Publication Date: January 1975

6. Performing Organization Code

7. AUTHOR(S)

Gilsinn, J., Saunders, P., and Pearl, M.

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No. 2050402

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)*

Department of Transportation
Systems Analysis and Evaluation Division
Urban Mass Transit Administration
Washington, D. C. 20590

13. Type of Report & Period Covered: Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

This report identifies and characterizes the data base and computer software requirements for a Point-to-Point Trip Management (PTPTM) System which provides detailed transit trip itineraries in response to inquiries made by prospective passengers. The requirements fall into four categories, corresponding to successive stages in processing such an inquiry: (1) reception and interpretation, (2) location and connection, (3) path calculation, and (4) report. Procedures for path calculation are discussed in detail, including techniques for improving shortest path algorithm performance both through optimized computational schemes and through special methods of representing and manipulating the data base describing transit routes and schedules. Estimates, based on step-by-step schemes presented in an appendix, indicate that computation time will be sufficiently small (less than one second per request) that on-line path computation is feasible. Since path finding represents only a fraction of the total time spent in answering a request for itinerary, a queuing model is developed to establish how many formerly lost calls would be captured by a computerized system which achieved a prescribed fractional saving in service time; illustrative application of this model indicates a sharp reduction in lost calls.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Algorithms; networks; paths; shortest paths; transit information systems; transit networks; transportation.

18. AVAILABILITY [X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[ ] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13

[ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	78
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	

USCOMM-DC 29042-P74