Transit Pathfinder Algorithm

ROBERT B. DIAL, Alan M. Voorhees and Associates, Inc.

This paper represents one element of the initial phase in the development of a package of computer programs to assist in mass transportation planning. These programs provide a flexible means of predicting ridership on each line of a proposed multimode transportation system within a given land-use configuration. This paper describes an IBM 7090 computer program and its enabling algorithm which finds minimum time paths through a large multi-modal transportation system. The program's capabilities and its underlying assumptions are summarized, and terms are defined. Step-by-step descriptions and flow charts are presented of the widely used Moore "tree-building" algorithm and of the transit "pathfinder," an extension of the former algorithm accommodating the peculiarities of transit minimum paths.

•FOR purposes of long-range planning, analyses of the efficiencies and economies of a proposed transit system require knowledge of the expected path of passengers traveling between any two points on the system. The best approximation of this path is the minimum weighted time path, wherein the times to traverse various link types are weighted to reflect differences in the values the transit user places on the time spent walking, waiting or riding by various modes.

Heretofore, the transportation planner has not had the means, i.e., a computer program, to find minimum paths joining nodes of a large intra-regional transit network. No available program accepted an economical network description which allowed sufficient size and detail. Existing programs either permit only a very small network and require a real-time fixed arrival schedule at each transfer point (2), or they require summarized "trunk line" descriptions in which route discrimination and transfer information are lost. Further deficiencies result from the inability of these programs to cope realistically with wait and transfer time. All of these factors have seriously handicapped the transit system analyst.

CAPABILITIES OF THE PATHFINDER PROGRAM

The pathfinder program reduces the number and degree of the problems confronting the transit planner. Among the program's important characteristics are the following:

•The transportation network is input as two items: a link description and a line description. The former is similar to coded networks read by existing highway network analysis programs (4). The latter resembles a bus schedule. System description and updating are greatly facilitated by dual files.

•The transit network is stored in the computer as a set of "trunk-line links." Each trunk-line link is defined as (a) a pair of nodes, (b) a time, (c) a mode, and (d) a set of line numbers representing all routes traversing the link in the given time via the given mode. This permits the computer to accommodate a vastly larger network than if it had to list each line/link combination explicitly. The trunk line description also reduces computer running time. Although the program uses a trunk line description, no information is lost. Its output lists all line numbers on every minimum path.

Paper sponsored by Committee on Origin and Destination.

•Transfer penalties need not be explicitly listed. Previous methods required the analyst to code links which represented the possibility of transfer between two lines. The traverse time associated with each of these links accounted for the wait time. The number of these transfer links rose factorially, precluding a complete network description. The pathfinder program does not use transfer links. It assumes that all line number pairs at a given node constitute legitimate transfer possibilities, and that the time spent transferring depends only on the frequency of the recipient line. (This is discussed below.) Thus the majority of both intra- and intermode transfers can be handled implicitly, requiring only each line's frequency as input.

•The minimum paths can be required to satisfy user-specified constraints. The user can preclude transfer between certain modes. Any of the 64 possible ordered modal pairs can be specified as either a legitimate or illegitimate transfer. The user may also put an upper limit on the amount of time to be assessed for a transfer to a given mode, as well as the total number of transfers on a minimum path. Wait times can be similarly constrained.

•Eight modes (walk, auto, and six transit modes) can be accommodated, and the time spent on each can have a different cost factor, as can the time spent waiting or transferring. In order to implement information uncovered on the different unit values a passenger places on the time he spends on rapid transit vs that on a local bus vs that walking, etc., the program allows the user to weight these components of travel time in the selection of minimum weighted time paths.

•The program can analyze a large network. Due to the peculiarities of a transit network, any description of it in a simple node-link form becomes quite large, and the size of the problem the program can handle is of prime importance. The pathfinder's limitations are quite relaxed and permit even the largest networks to be described in fine detail. The program can process a network with the following limiting characteristics (many of the limitations are results of required interface with other programs): (a) 2800 transfer points (nodes); (b) 11,000 trunk-line links; (c) 31 lines on any one link; (d) 8 modes; and (e) 255 (optionally) round-trip lines (routes) within each of 6 modes. The implicit size of the network becomes evident when it is noted that the 11,000 links with the attendant line limitations can account for over 32,000 line/link combinations and over a quarter of a million transfer links.

When appropriate coding procedures are established and adhered to, the program's limitations do not pose any significant constraints. We are, nonetheless, anxious to recode the program for a larger capacity IBM System/360. Not only could the maximum parameters be increased, but running time could be improved and new wrinkles added.

ASSUMPTIONS OF THE ALGORITHM

There are two principal assumptions underlying the pathfinder algorithm:

1. That the time to traverse a trunk line link is constant, and
2. That the time spent waiting to transfer is satisfactorily approximated by one-half the inverse of the frequency of the recipient line(s).

Assumption 1 states that link times cannot change during the pathfinding process. The time to traverse a given link must be the same for all paths. The algorithm by itself cannot adjust for such things as volume-dependent ("capacity-restrained") speeds or times. This is a common assumption in network algorithms (1).

Assumption 2 states that, in the absence of an inviolate, real-time schedule of arrivals of each line at each node, the algorithm can assume a "random" arrival of the bus (or train, or walker) from which the transfer is to be made. Indeed, a more complicated assumption could be made, but it is not worth the effort.

The program could assume an actual bus schedule as input. But in so doing it would be imposing a hardship both on itself and on the planner. First, it would require the planner to produce these schedules for systems proposed for use 20 years hence—an exercise of questionable value. Second, the minimum time through a fixed schedule of stops varies with the time of departure from the home node. To find the minimum

time over all departure times is an exercise of considerable worth, but only to computer suppliers, insofar as long-range transit planning is concerned. For the planners the marginal cost/marginal utility ratio is nearly infinite.

Obviously, the planner must be able to provide an estimate of each line's frequency. This is the minimum requirement. It puts a handle on wait times and is needed to estimate capital and operating expenditures.

Thus the algorithm assumes that if a transfer is to be made from route A to route B, and route B's frequency is 5 buses per hour, then the expected wait time is one-tenth of an hour or 6 minutes. Further, whenever a transfer can be made to more than one route at the same point, the assumed transfer wait time is one-half the sum of the departing lines' frequencies. For example, if at a given point a minimum path requires a transfer to either route B or route C, and route B frequency is 5 buses per hour and route C goes by once an hour, then the transfer penalty assessed is $0.5/(5+1) = 1/12$ hour or 5 minutes.

Assumption 2 makes the transfer wait time independent of the arriving line(s). An important corollary results: If a minimum time path from point A to point C requires a transfer at point B, then this path begins by using the minimum time path to B. The algorithm is free to "forget" the initial origin point of a path involving transfers, provided it "remembers" the point at which the last transfer occurred. Because a minimum time path to the transfer point is known, the entire path can be traced back. For example, if the minimum time path from A to C had a transfer at B, this path would be described simply as "Take line(s) X at B." The observer, i.e., volume loading program, would note that B was not the home node A, and it would seek out the minimum path to A, namely, "Take line(s) Y at A," and the path would be complete.

This is a descriptive economy, and it provides the algorithm with a means of keeping down the number of transfer possibilities to be examined. As clarified below, the only transfers to be examined at a given point are those from a minimum path to that point.

DEFINITIONS

At this point the discussion of the pathfinder algorithm requires a few definitions of terms already used, so as to eliminate any ambiguities which henceforth might prove confusing.

The algorithm works with a network description composed of line frequencies and trunk-line links. Line frequencies are simply the average number of buses per unit time for each of the lines (routes) in the transit system. Each line can have only one frequency. If a round trip route has a different frequency outbound than inbound, then the two directions must be coded as separate lines.

The only purpose for frequencies in the pathfinder program is to assess transfer penalties. A transfer penalty X is a function of a set of line numbers S:

$$X(S) = \frac{0.5}{\sum_{k \text{ in } S} F(k)}$$

where F(k) is the frequency of line k, and the sum is taken over all lines k which are contained in the set of line numbers S.

A trunk-line link (TLL) has three immediate constituents:

1. An ordered pair of nodes,
2. A traversing time, and
3. A set of line numbers.

1. Nodes are junctions in a network at which transfers occur. There are always exactly two nodes associated with a TLL, and each is identified with a unique number. The same number must always be used when referring to the node. The first node of a pair is called the A-node and the second is called the B-node. A TLL is always "one-way." The flow is always from the A-node to the B-node. The highest node number permitted by the program is 2800.

2. Traversing time is simply the time needed to traverse the TLL. All transit lines listed with a TLL must provide identical service between the TLL's A-node and B-node. When there is more than one level of service between two nodes which would otherwise constitute a single TLL, the analyst must code a second "dummy" link to which he can assign the lines providing the second level of service. The highest time permitted is 64 minutes.

3. A set of line numbers constitutes the final element of a TLL's definition. Each link has a simple listing of all the lines which traverse it in the same time. Lines are coded as integers between 1 and 255.

MECHANICS OF THE MOORE ALGORITHM

As mentioned earlier, the pathfinder algorithm is an extension of the widely used Moore algorithm (3). Because of their similarity, a description of the latter greatly facilitates a description of the former. Therefore, the Moore algorithm is discussed, flow-charted and used in an example.

The task of the Moore algorithm is to "build a tree." A tree is the set of links comprising the minimum paths from a given "home node" to all other nodes in the network. Assuming there exists a unique minimum path to each destination node, the term "tree" is quite appropriate. The uniqueness assumption implies there can be one and only one link entering each node. Thus when all minimum paths are plotted the result is a series of "branches," spreading as they move away from the home node.

The Moore algorithm finds the number-of-nodes-minus-one links in a tree in the order of their "distance" (measured in time) from the home node. A link's distance from the home node is defined here as the minimum time to go from the home node to its B-node. The algorithm starts by finding the closest link, which must have the home node as its A-node. This link is placed in the tree along with its corresponding "distance." The remaining process consists simply of selecting the link closest to the home node among all links connected to links already in the tree but not in the tree themselves. Each link's distance from the home node is calculated by adding the distance to its A-node's tree link to its own link traverse time. The link closest to the home node is placed in the tree, and the process repeated. The process stops when nodes-minus-one links have been placed in the tree.

The flow chart of Figure 1 is an example of a computer program of the Moore algorithm. Although not intended to reflect an optimal code, it is realistic enough to use with the network of Figure 2 to build a tree from home node A. The resulting tree

```
DICTIONARY
VARIABLE    EXPLANATION
--------    -----------
N(I,J)      B-NODE DEFINING J'TH EXIT LINK FROM NODE I
T(I,J)      TIME ON LINK FROM NODE I TO NODE N(I,J)
LINKS(I)    TOTAL NUMBER OF EXITS FROM NODE I
TREE(I)     TERMINAL LINK IN MINIMUM PATH FROM H TO I
H           HOME NODE
C(I)        NUMBER OF LINKS IN SLOT I OF TABLE
TABLE(I,J)  B-NODE OF LINK WHOSE CUMULATIVE TIME IS I
            (THIS IS THE LINK SEQUENCING TABLE)
MAXN        TOTAL NUMBER OF NODES IN SYSTEM TO
NODE        COUNTS THE NUMBER OF NODES WHICH MINIMUM
            FROM H HAVE BEEN FOUND
```

Figure 1. Flow chart of Moore algorithm implementation.

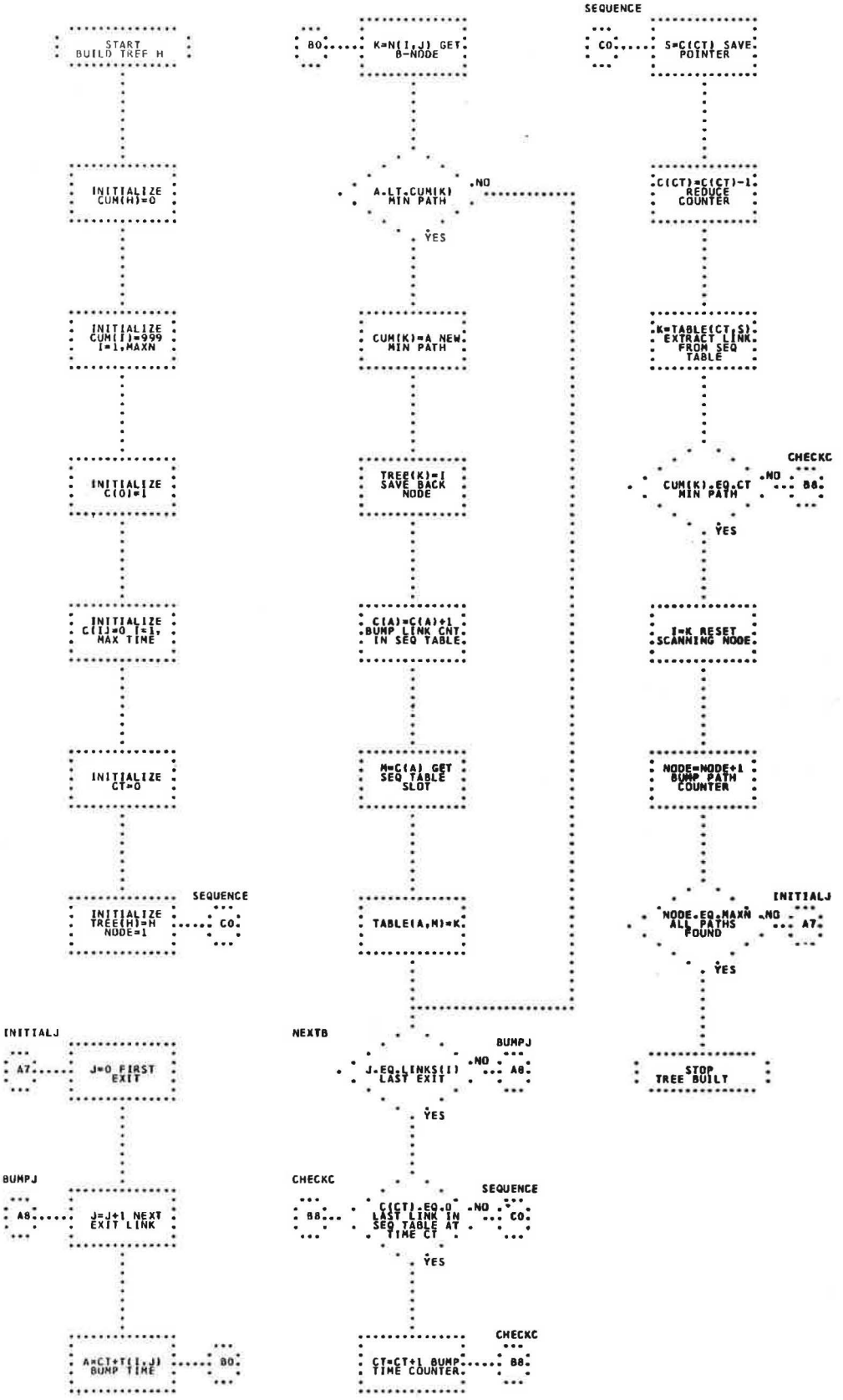


Figure 1. Moore algorithm (continued).

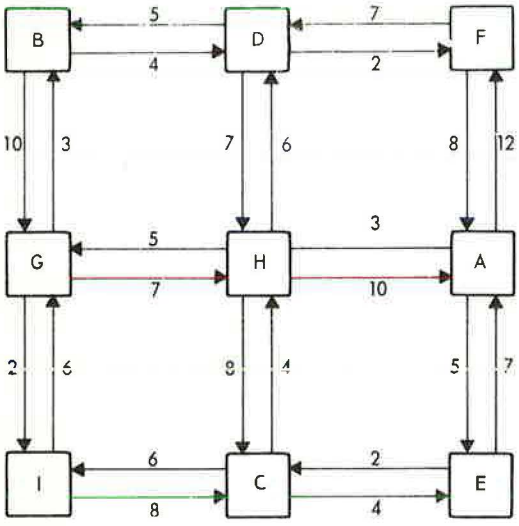


Figure 2. Sample network showing node labels and traverse times.

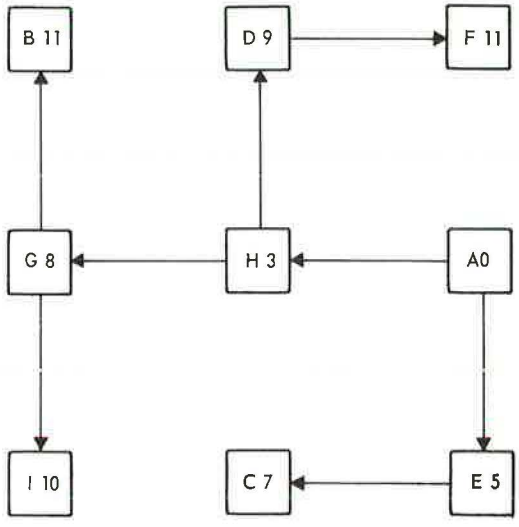


Figure 3. Tree (with cumulative times).

and the "distances" (i.e., cumulative times) to each node via the minimum path appear in Figure 3 and Table 3. Tables 4 and 5 are "scratch" tables. Tables 1 and 2 describe the network in a computer-oriented fashion. (This same example with slightly different notation and algorithm coding is found in Ref. 3 with a detailed discussion.)

Figure 2 is a graph of a network which employs letters for node labels and arrowheads to indicate the direction of flow. Each (one-way) link has its traverse time shown in arabic numerals. Table 1 describes this network by tabulating each node's exit links. This table, whose elements are called N(I, J) in the flow chart, shows all links exiting any given node. It is entered by finding the column of the node of interest and reading across to find the B-nodes of all links which have the given node as an A-node. N(I, J) is therefore the B-node of the Jth link which has I as an A-node.

Table 2 simply lists the times associated with the links in the corresponding positions of Table 1. The flow chart in Figure 1 refers to this table by the symbolic name of T, and T(I, J) is the time associated with the Jth exit from A-node I.

TABLE 1

N(I, J): LINK DESCRIPTION

a \ i	1	2	3	4
A	E	F	H	–
B	D	G	–	–
C	E	H	I	–
D	B	F	H	–
E	A	C	–	–
F	A	D	–	–
G	B	H	I	–
H	A	C	D	G
I	C	G	–	–

TABLE 2

T(I, J): LINK TRAVERSE TIME

a \ i	1	2	3	4
A	5	12	3	–
B	4	10	–	–
C	4	4	6	–
D	5	2	7	–
E	7	2	–	–
F	8	7	–	–
G	3	7	2	–
H	10	8	6	5
I	8	6	–	–

TABLE 3

CUMULATIVE TIMES VIA MINIMUM PATH

J	CUM(J)	TREE(J)
A	~~99999~~ 0	HOME NODE
B	~~99999~~ 11	G
C	~~99999~~ ~~11~~ 7	~~H~~ E
D	~~99999~~ 9	H
E	~~99999~~ 5	A
F	~~99999~~ ~~12~~ 11	~~A~~ D
G	~~99999~~ 8	H
H	~~99999~~ 3	A
I	~~99999~~ ~~13~~ 10	~~C~~ G

TABLE 4

"SCRATCH" TABLE

CT	TABLE(CT,K)
2	
3	~~H~~
4	
5	~~E~~
6	
7	~~C~~
8	~~G~~
9	~~D~~
10	~~I~~
11	~~C~~, ~~B~~, ~~F~~
12	F

TABLE 5

"SCRATCH" TABLE

I	CT	NODE
~~A~~	~~6~~	~~1~~
~~H~~	~~3~~	~~2~~
~~E~~	~~5~~	~~3~~
~~C~~	~~7~~	~~4~~
~~G~~	~~8~~	~~5~~
~~D~~	~~9~~	~~6~~
~~I~~	~~10~~	~~7~~
~~B~~	~~11~~	~~8~~
F	11	9

Three other tables are used in the flow chart. TREE(I) contains the A-node of the final link in the minimum path from the home node to node I. TABLE (I, J) holds the B-node of the J th link whose distance from the home node is equal to I minutes. C(I) is a count of the number of links in the Ith time slot of TABLE.

Explanation of the Moore Algorithm Flow Chart

1. Initialization consists of setting all minimum path times arbitrarily high except for the home node's, which is set to zero. The link sequencing table is emptied, and one link is entered into it representing the "minimum path" to the home node H. This dummy link is also placed in the tree table at location TREE(H). NODE, which counts the number of nodes for which the program has found a minimum time path, is set equal to one. The cumulative time value CT is reset, and control transfers to step 3 below.

2. After a link has been placed in the tree, all links connected to it are examined for possible entry into the link sequencing table. For each of the exits from the new tree link, a "distance" is calculated by adding the current cumulative time CT to the exit link's traverse time T(I, J). Its B-node is then examined to see if a shorter path to it has already been found, i.e., if CUM (B-node) is less than the distance just calculated. If it is, the link is ignored. If the new distance is less than that in the CUM table, the CUM table entry is updated, CUM (B-node) = CT + T(I, J), and the link's A-node, I, is placed in the tree, while its B-node is entered into the link sequencing

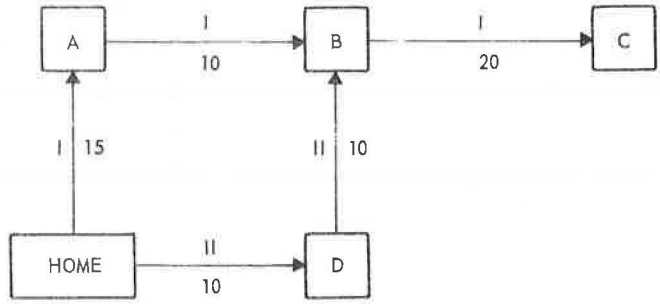


Figure 4. Simple transit network (frequency of Line I is 5/hour, of Line II, 5/hour).

table at a position corresponding to CT + T(I, J). Step 2 is repeated until all exits from node I have been examined, at which time control passes to step 3.

3. This step extracts links from the link sequencing table. Remember that the entities in this table are B-nodes and their position in the table corresponds exactly to the B-node's distance, i. e., cumulative time, from the home node. The program removes the first link at or below position CT in the table. CT is then updated to this node's position in the table. If this B-node's cumulative time is equal to its positional value, i.e., CUM(TABLE(CT, J)) = CT, then this is a minimum path link, in which case control returns to step 4. If CUM(TABLE(CT, J)) is less than CT, then the link is ignored and step 3 is repeated.

4. This step enters a link into the tree. It increases the value of NODE (the number of nodes to which minimum paths have been found) by one, and transfers control to step 3, if all nodes have not been reached. Otherwise, it stops; the tree is complete.

PECULIARITIES OF TRANSIT TREES

The Moore algorithm is an efficient means to build a tree, thanks to the fact that once a minimum time path to a node is found, the node can be ignored for the remainder of the tree-building process. Any minimum path crossing that node will use the same (minimum) path up to the node, regardless of the path's ultimate destination.

Transit trees lack this property. A node can lie on two or more minimum paths, none of which uses the same path from the home node to the common node. The simple network of Figure 4 illustrates this fact. (In this example, as in all subsequent graphs of a transit network, the following conventions are used: boxed letters represent nodes, arabic numerals represent link traverse time in the direction of the arrow, and the roman numerals represent line numbers of routes which traverse the trunk line link in the same time.) Assume that the headways of lines I and II are equal, and each is 12

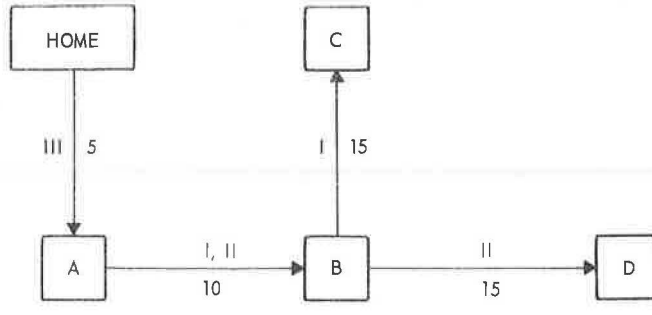


Figure 5. Entire transit network (frequency of Line I is 4/hour, of Line II, 2/hour, of Line I and Line II, 6/hour).

minutes. The transfer penalty to change from line II to line I at node B is therefore 6 minutes, and the total time to go from Home to C via this path is 46 minutes (plus the initial wait time for line II). On the other hand, by getting on line I at home, the trip to C takes 45 minutes (plus the initial wait time for line I, which is the same as line II's). Thus the minimum to C passes through B via a different path than the minimum time path to B, which is to get on line II at Home.

While transfers to a line require the pathfinder algorithm to examine reentries to a node after the minimum path has been found, the algorithm must also cope with another nuisance, namely the penalty for "transferring from" a line. This occurs with the diminishing of a line selection group. Assume that Figure 5 is an entire transit network. Then obviously the minimum path to any node except A requires a transfer at A. If I and II's headways are respectively 15 and 30 minutes, then the minimum paths to B, C and D will each arrive at B in a different cumulative time. But any Moore-type algorithm, including the proposed version, will arrive at B exactly once. It will find a path to B requiring a total of 15 + 5 = 20 minutes. The 5 minutes are for the transfer at node A, where 6 buses go by each hour.

However, for any trip destined for C, only two usable buses go by B each hour, since the minimum time to go to C is obviously realized by transferring to line I at node A. Thus the time to get to B for any trip heading for C is 15 + 7.5 = 22.5 minutes. Similarly, the minimum path to D crosses B 15 + 15 = 30 minutes after leaving home. Therefore each of the three paths requires a different time to B, although the algorithm gets them there at the "same" time of 25 minutes.

An easy solution exists; the algorithm simply adjusts the cumulative time to B for each exiting path at the time it is generating the exit links from B to place in the link sequencing table. Because it knows on what lines it arrived at B, it knows the value of the current transfer penalty. It subtracts that penalty and adds the penalty for "transferring" to the through lines as it adds the link's traverse time to the cumulative time. The actual computation is described later.

Although the above problems are easily solved theoretically, it is important to realize that there are practical repercussions. The pathfinder has to work much harder than the Moore algorithm. While it does nearly the same actions, it performs them with much greater frequency. Many more links have to be examined. It will always have a fuller link sequencing table, and will have to analyze twice as many links for each link drawn from the table. The effect is significant in terms of running time.

MECHANICS OF THE PATHFINDER ALGORITHM

Differences between the Moore algorithm and the pathfinder are results of transfer penalties. For this reason, the way the program uses them will be elaborated on here. A transfer penalty measures the effect of waiting time, and it can seriously affect a path. Waiting time can occur at the start of a path and also at transfer points en route, whenever a minimum path requires line changing.

To the algorithm a transfer penalty X, as discussed earlier, is merely a function of a trunk-line link or, more simply, a function of the frequencies of the lines on the trunk-line link. In particular, it is a certain number of minutes, equal to one-half the inverse of the sum of the frequencies of the link's lines. For example, if link k has line numbers II, VII and XII whose frequencies are three, five and one bus per hour, the link k's transfer penalty is

$$\begin{aligned} X(k) &= 0.5/(F(II) + F(VII) + F(XII) \\ &= 0.5/(3 + 5 + 1) \\ &= 1/18 \text{ hours} \\ &\cong 3.3 \text{ minutes} \end{aligned}$$

The algorithm incorporates the transfer penalty function X as it puts links into the link sequencing table. For each link extracted from the table, it analyzes all links exiting from its B-node. For each of these exit links it enters in the sequencing table one or both of the following links (assume that L is the link extracted from the sequencing table, and E is a link exiting from L's B-node):

1. Link 1 is (almost) always generated. It represents "through" paths and is comprised of (a) the A-node of L; (b) the B-node of E; (c) the lines which are common to L and E; and (d) the time equal to the cumulative time plus E's link time minus L's transfer penalty plus E's transfer penalty, i.e., CT + T(E) - X(L) + X(E).

2. Link 2 is only generated the first time a link with L's B-node is extracted from the table, and then only if T(Link 1) is greater than T(Link 2). Link 2 represents "pure transfer paths," and has the following elements: (a) the A-node of E; (b) the B-node of E; (c) the lines on E but not on L; and (d) the time equal to the cumulative time plus the time to traverse E plus the transfer penalty for E, i.e., CT + T(E) + X(E).

Explanation of the Pathfinder Algorithm Flow Chart, Figure 6

1. Initialization is exactly analogous to that of the Moore algorithm, wherein a dummy link is placed in the zeroth position of the link sequencing table. The remainder of the table is emptied. Minimum path times to all nodes is set arbitrarily high except for the home node, whose time is set to zero. The cumulative time pointer CT is set to zero, and control goes to step 3.

2. This step merely finds the uppermost link in the link sequencing table. The search begins at the CTth slot and continues downward (upward in minutes) until a non-empty slot is found. CT is reset to the slot value containing the first link found and control transfers to step 4. If no link is found this means some nodes are not connected to the home node, and tree-building stops.

3. This step extracts a link, called Y, from the link sequencing table. The variable I represents this link's B-node, and its line number set is called R. All links exiting from I, except for U-turns, are examined for line numbers in L. A link is generated

DICTIONARY

VARIABLE	EXPLANATION
CUM(I)	TIME FROM HOME NODE TO NODE I VIA MINIMUM PATH
SEQ(I,J)	J'TH LINK IN TIME SLOT I OF SEQUENCING TABLE
C(I)	NUMBER OF LINKS IN SLOT I OF SEQUENCING TABLE
LINK(I,J)	J'TH EXIT LINK FROM NODE I (LINK TABLE)
LINKS(I)	NUMBER OF EXITS FROM NODE I
MAX(I,J)	MAXIMUM FUNCTION SELECTS LARGEST VALUE OF I,J
CT	CUMULATIVE TIME POINTER
AND(I,J)	LOGICAL AND FUNCTION
COMP(I)	LOGICAL COMPLEMENTATION FUNCTION
DIFF(I,J)	LOGICAL DIFFERENCE FUNCTION,AND(I,COMP(J))
NULL	THE EMPTY SET
TRFE(I)	LAST LINK(S) IN MINIMUM PATH(S) FROM HOME TO I
MAXN	HIGHEST NODE NUMBER
MAXT	HIGHEST TIME PERMITTED IN LINK SEQUENCE TABLE
X(I)	TRANSFER TIME FUNCTION (PENALTY TO LOAD)
H	HOME NODE
A(I)	A-NODE FUNCTION FETCHES A LINKS ANODE
B(I)	B-NODE FUNCTION FETCHES LINK I'S B-NODE
L(I)	LINES FUNCTION FETCHES LINK I'S LINES
T(I)	TIME FUNCTION FETCHES LINK I'S TIME

Figure 6. Flow chart of pathfinder algorithm implementation.

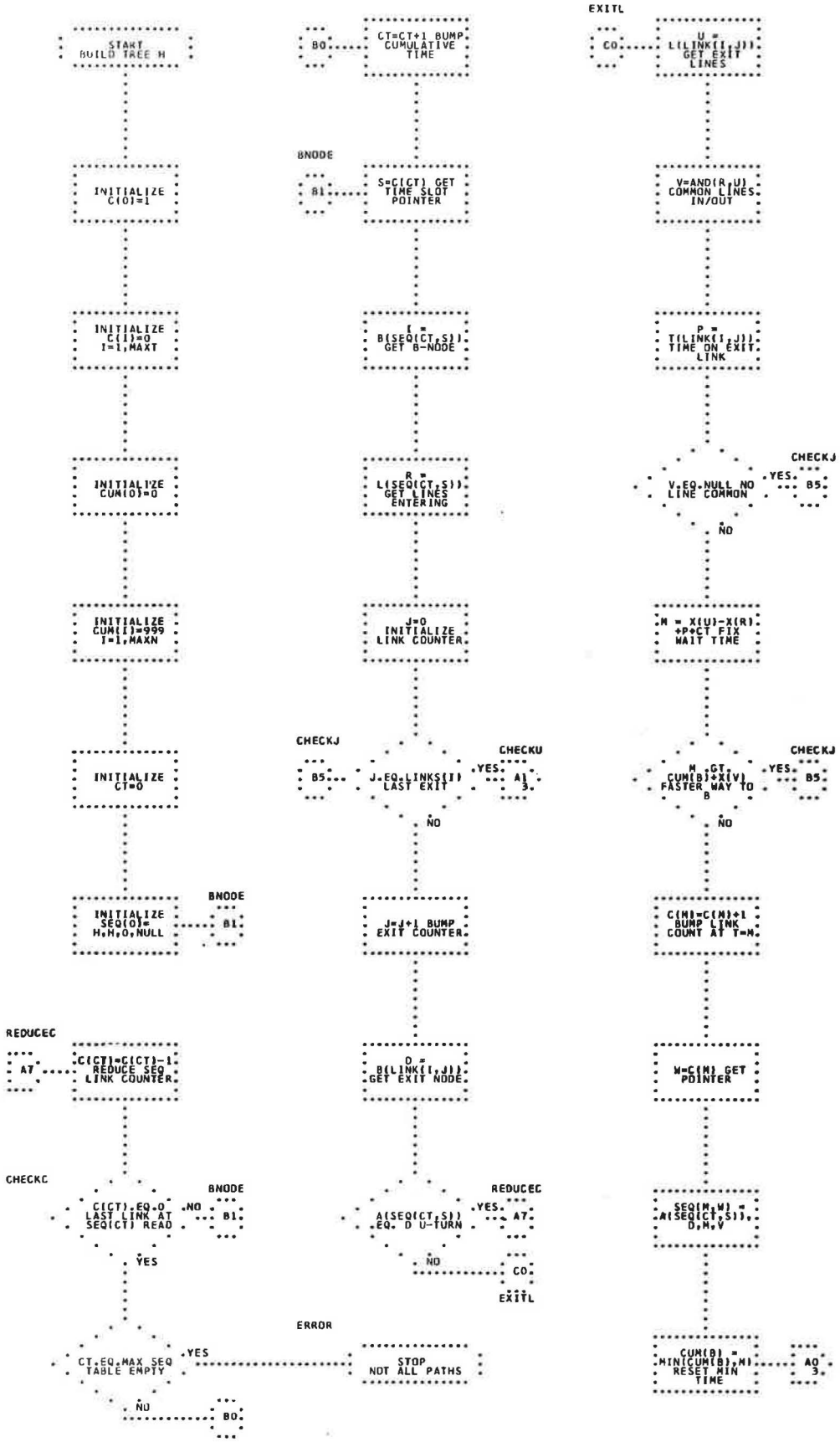


Figure 6. Pathfinder algorithm (continued).

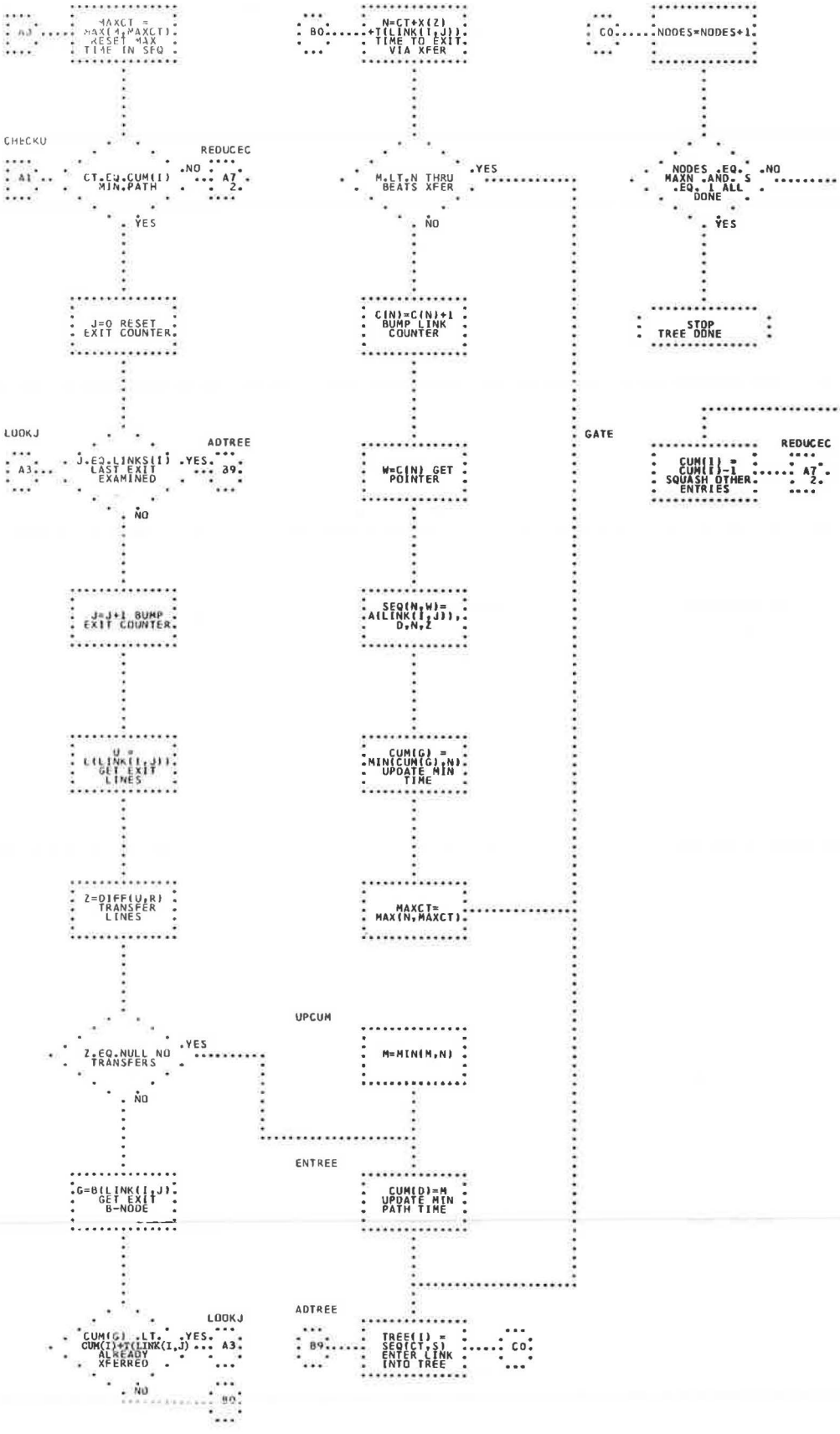


Figure 6. Pathfinder algorithm (continued).

for each exit and placed in the cumulative time table. The description of this through link appears earlier, where it is called "Link 1." (The algorithm only generates this link whenever there are line numbers common to both Y and the exit link under examination.) The algorithm next examines the minimum path time to I, stored in CUM (I). If this is the same as CT, the slot in the link sequencing table at which Y was stored, then Y comprises the last leg of the minimum path to I. In this case transfer out of I must be examined, and control goes to step 4. Otherwise step 4 is skipped and control goes to step 2.

4. Step 4 generates the transfer paths out of node I. It is reached only when Y is on the minimum path to I. The transfer path link's description appears earlier, where it is called "Link 2." (Step 4 is not executed if Y is the last link in the tree, i.e., the number of links found equals the number of nodes in the network. In this case the tree is built and the program stops.) A transfer link to I is only generated whenever the cumulative time to its B-node, CUM (B-node), is greater than the time to get to I via the transfer link. When all exiting transfers from I have been examined, the procedure continues at step 5.

5. This step merely adds the link Y to the tree. It increments the total number of links in the tree by one and checks if it has found a link for every node. If so, it is finished with the tree, and the program ends. If not, CUM (I) is set to a value below the minimum path time to preclude any other link being put in the tree for this node. The program then returns to step 2 to extract the next ranking link from the sequencing table.

It should be noted that a different method of storing the tree has to be used in the pathfinder program than was used for the Moore algorithm. The reason is that the transit network requires a far larger description and link sequencing table. The sizes of these tables are such that no room is left in core storage to contain the entire tree, and its links must be "written out" as they are found.

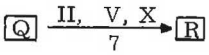
Notation

Because the pathfinder algorithm is more complicated in its detail than the Moore algorithm, an extended notation is used in its flow chart to keep the number of boxes down and still provide a thorough idea of the processing. While the basic element of the Moore algorithm was a B-node, the pathfinder handles cumbersome trunk-line links. It fetches them, stores them, creates them, and discards them. To facilitate their symbolic manipulation, the following notation is employed in the flow chart:

If K is a trunk-line link it is often written as

$$K = A, B, T, L$$

where A is an A-node number, B is a B-node number, T is a time, and L is a line number set. For example, if the graph of the link P is



then this link is written as

$$P = Q, R, 7, S$$

where S is a set comprised of lines II, V and X.

The individual elements of a trunk line link are extracted with the "functions" A, B, T and L where A(K) is K's A-node number, e.g., A(P) = Q; B(K) is K's B-node number, e.g., B(P) = R; T(K) is K's time, e.g., T(P) = 7; and L(K) is K's line number set, e.g., L(P) = S = II, V, X.

As line number sets have to be compared and created, the following notation is useful: If A, B and C are sets of line numbers, then

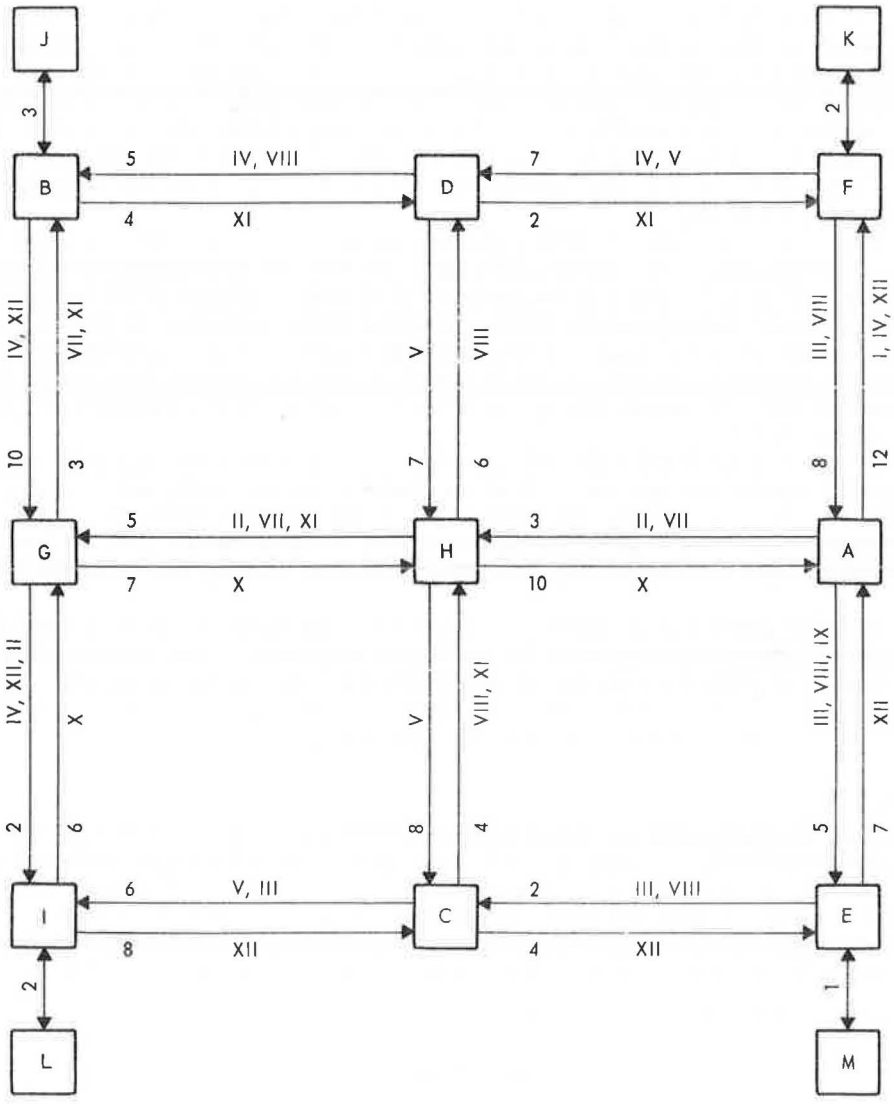


Figure 7. Hypothetical transit network (letters are node identifiers, arabic numbers are link times, roman numerals are bus line (route) numbers, and double-headed arrows are walk links).

A = NULL means that A has no line numbers
C = A means C has all and only those line numbers in A
C = COMP (A) means C has only those line numbers not in A
C = AND(A, B) means C has only those line numbers in both A and B
C = DIFF(A, B) means C has only those line numbers in A but not in B

The minimum or maximum of a pair of numbers is often desired. The following notation is used in the flow chart:

A = MIN(T, S) means A equals the smaller of T and S
A = MAX(T, S) means A equals the larger of T and S

Transfer penalties are frequently calculated in the algorithm. A shorthand method of indicating this computation is used in the flow chart. The penalty for transferring to lines contained in the set A is denoted as X(A). Its value is equal to one-half the

TABLE 6

TRANSIT NETWORK DESCRIPTION (NET) COMBINED LINK-TIME-LINE TABLE

From node
To node
Link Time
Line number set

A	E, 5(III, VIII, IX)	F, 12(I, IV, XII)	H, 3(II, VII)	
B	D, 4(XI)	G, 10(IV, XII)	J, 3(0)	
C	E, 4(XII)	H, 4(VIII, XI)	I, 6(III, V)	
D	B, 5(IV, VIII)	F, 2(XI)	H, 7(V)	
E	A, 7(XII)	C, 2(III, VIII)	M, 1(0)	
F	A, 8(III, VIII)	D, 7(IV, V)	K, 2(0)	
G	B, 3(VII, XI)	H, 7(X)	I, 2(IV, XII)	
H	A, 10(X)	C, 8(V)	D, 6(VIII)	G, 5(II, VII, XI)
I	C, 8(XII)	G, 6(II, X)	L, 2(0)	
J	B, 3(0)			
K	F, 2(0)			
L	I, 2(0)			
M	E, 1(0)			

inverse of the sum of the frequencies of all lines in A. For example, the penalty for transferring to the lines on link P above is

$$\begin{aligned} X(L(P)) &= X(S) \\ &= X(II, V, X) \\ &= 0.5/(F(II) + F(V) + F(X)) \end{aligned}$$

where F(II) is line II's frequency, etc.

EXAMPLE

The transit network shown in Figure 7 is tabulated in Table 6. In the network, nodes J, K, L are assumed centroids, and the links connecting them to the system are walk links. As above, roman numerals represent line numbers and arabic numerals are time. There are no line numbers associated with the walk links. The line frequencies listed in Table 9 complete the system's description.

The pathfinder algorithm is used to build a tree from node J. A worksheet describing the principal calculations comprises Table 11, which tabulates the computations in generating candidates for the link sequencing table. Tables 7, 8 and 10 are the working tables used in the tree-building process. Tree-building consists of adding links in the table called TREE until one link is present for each node. Figure 8 portrays the final tree.

TABLE 7

WORKING TABLE

DEST. NODE	CUMULATIVE TIME
I	CUM(I)
J	0
K	~~16~~ 15
L	~~23~~ 22
M	~~32~~ 31
B	~~3~~ 2
D	~~12~~ 11
F	~~14~~ 13
G	~~19~~ 18
H	~~24~~ 23
A	~~26~~ 25
I	~~21~~ 20
C	~~33~~ ~~32~~ 31
E	~~31~~ 30

TABLE 8

WORKING TABLE

LINK NO.	LAST LEG ON MINIMUM PATH
I	TREE(I)
1	J–J(0) 0
2	J–B(0) 3
3	B–D(XI) 12
4	B–F(XI) 14
5	F–K(0) 16
6	B–G(IV, XII) 19
7	B–I(IV, XII) 21
8	I–L(0) 23
9	D–H(V) 24
10	F–A(III, VIII) 26
11	F–E(III, VIII) 31
12	D–C(V) 32
13	E–M(0) 32

TABLE 10

WORKING TABLE

TIME SLOT	LINK SEQUENCING TABLE
T	SEQ(T)
0	~~J–J(0)~~
~~1~~ 21	~~B–I(IV, XII)~~
2	
~~3~~ 23	~~J–B(0), I–L(0)~~
~~4~~ 24	D–H(V)
5	
~~6~~ 26	~~E–A(III, VIII)~~
7	
8	
9	
10	
~~11~~ 31	~~E–E(III, VIII)~~
~~12~~ 32	~~B–D(XII), D–C(V), E–M(0)~~
~~13~~ 33	B–C(12), F–C(III, VIII)
14	~~B–F(XII)~~
15	
16	~~7–2(0)~~
17	
18	
19	~~B–G(IV, XII)~~
20	

TABLE 9

LINE FREQUENCIES

LINE NO.	FREQUENCY (BUSES HR.)
I	6
II	10
III	5
IV	2
V	6
VI	5
VII	3
VIII	3
IX	10
X	6
XI	6
XII	3

TABLE 11

WORKSHEET FOR TRANSIT TREE-BUILDING EXAMPLE

Step	Extracted Link	Generated Link	Time Calculation	Entered into Sequencing table?
0	-	J-J(0)	0	Yes
1	J-J(0)	J-B(0)	3	Yes
2	J-B(0)	B-G(IV, XII)	3+10+X(IV, XII) = 13+6= 19	Yes
		B-D(XI)	3+4+X(XI) = 7+5 = 12	Yes
3	B-D(XI)	D-H(V)	12+7+X(V) = 19+5 = 24	Yes
		B-F(XI)	12+2 = 14	Yes
		D-B(IV, VII)	12+5 =17 >2	No
4	B-F(XI)	F-K(0)	14+2 = 16	Yes
		F-A(III, VIII)	14+8+X(III, VIII=22+4 = 26	Yes
		F-D(IV, V)	14+7 = 21> 12	No
5	F-K(0)	None	-	-
6	B-G(IV, XII)	B-I(IV, XII)	19+2 = 21	Yes
		G-I(II)	19+2+X(II)> 21	No
		G-H(X)	19+7+X(X)=26+5=31>24	No
7	B-I(IV, XII)	I-L(0)	21+2 = 23	Yes
		B-C(XII)	21+8+X(IV, XII)+X(XII)=29-6+10=33	Yes
8	I-L(0)	None	-	-
9	D-H(V)	D-C(V)	24+8 = 32	Yes
		H-A(X)	24+10+X(X) = 25 + X(X)> 25	No
		H-G(II, VII, XI)	24+5+X(II, VII, XI)=21+X(II, VII, XI) > 18	No
		H-D(VIII)	24+6+X(VIII)> 12	No
10	F-A(III, VIII)	A-H(II, VII)	26+3+X(II, VII)>24	No
		A-F(I, IV, XII)	26+12+X(I, IV, XII)> 14	No
		F-E(III, VIII)	26+5 =31	Yes
		A-F(IX)	26+5+X(II)>13)	No
11	F-E(III, VIII)	E-M(0)	31+1=32	Yes
		F-C(III, VIII)	31+2 = 33	Yes
		E-A(XII)	31+7+X(XII) > 25	No
12	D-C(V)	D-H(V)	32+6=38 >24+X(V)	No
		C-H(III)	32+6+X(III) > 24+X(III)	No
		C-E(XII)	32+4+X(XII) > 31+X(XII)	No
		C-H(VIII, XI)	32+4+X(VIII, XI)> 24+X(VIII, XI)	No
13	E-M(0)	None	-	-

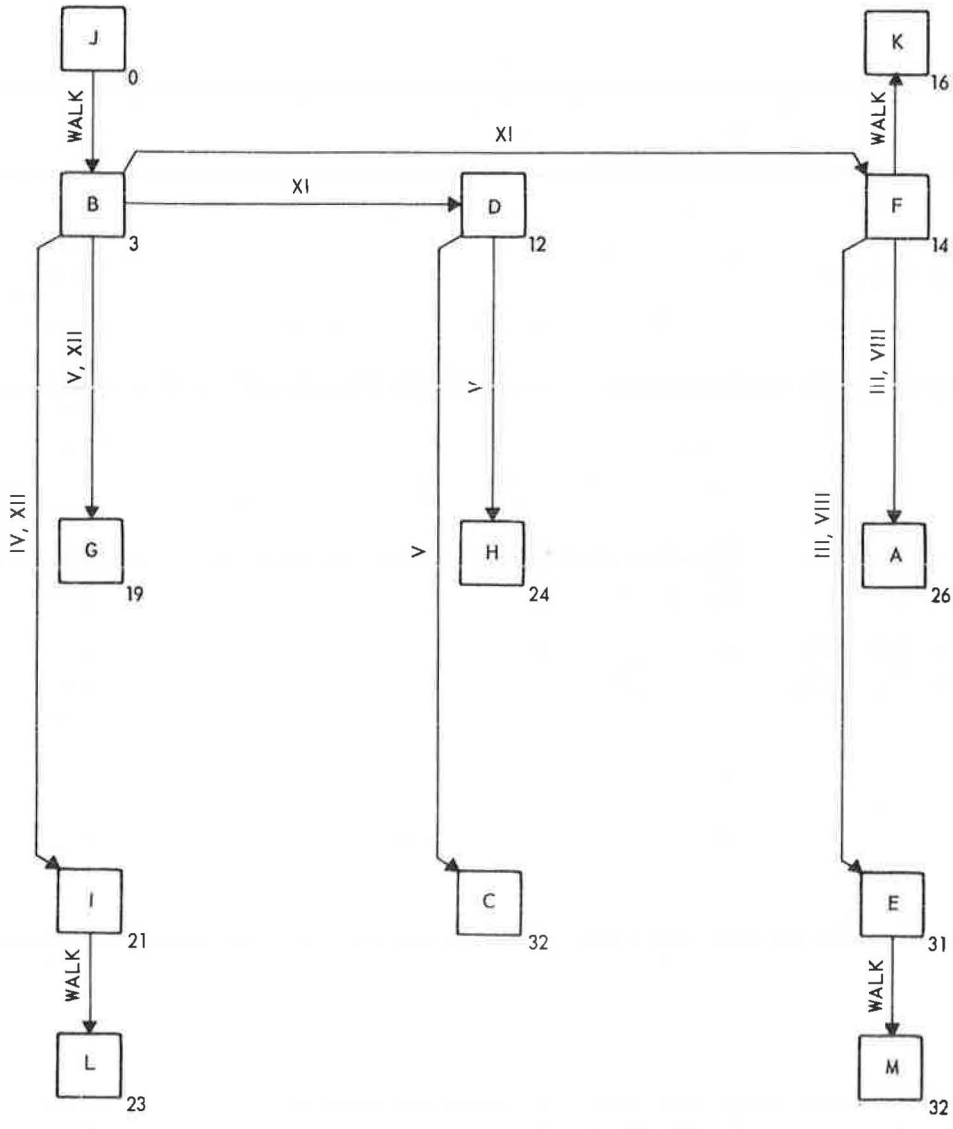


Figure 8. Minimum paths from home node J (letters are node identifiers, roman numerals are line numbers, and arabic numerals are total times from home node).

To follow the example, the reader should read Table 11 from top to bottom. The first column tabulates the links extracted from the link sequencing table in the order they are extracted. Each is entered into the tree. The second column lists the transfer links and through links exiting the B-node of the link extracted from the link sequencing table. Note that there is usually more than one link generated for each taken out. The third column shows the "distance" (time) calculation. First is posted the cumulative time to the extracted link's B-node. Next is shown the link traversing time, followed by a symbolic representation of the transfer penalty. A yes or no in the last column indicates whether or not the generated link was entered into the link sequencing table.

ACKNOWLEDGMENTS

This report comprises part of a contract with the U. S. Department of Housing and Urban Development. Messrs Walter Hansen, Howard Bevis and Richard Bunyan formed a responsive audience during the development of the algorithm, and Mr. Vince Calvert wrote the program (in MAP). The author thanks them all.

REFERENCES

1. Ford, L. R., and Fulkersen, D. R. Flows in Networks. Princeton University Press, 1962.
2. Hebetnemi, S., and Leven, B. M. Determining Fastest Routes Using Fixed Schedules. 1963 Spring Joint Computer Conference Proceedings. Spartan Book Co., Baltimore, 1963.
3. Moore, E. F. The Shortest Path Through a Maze. International Symposium on the Theory of Switching Proceedings, Harvard University, April 2-5, 1957.
4. Traffic Assignment Manual. U. S. Bureau of Public Roads, Office of Planning, Urban Planning Division, June 1964.