Interactive Graphics Model for Testing Stochastic Traffic Assignment Algorithms

JEROME M. LUTIN

This paper reports on the use of interactive computer graphics to investigate properties of a stochastic traffic assignment algorithm used to model pedestrian flow in the Urban Mass Transportation Administration's transit station simulation model (USS). The use of an interactive computer language (APL) to transform networks and models into easily manipulable matrix forms is discussed. Graphical calibration of model parameters is discussed and illustrated by using computer plots. The development of an interactive network builder and stochastic traffic assignment model is reported. The exploration of various model forms through interactive graphics is discussed and illustrated by using computer graphics plots of typical networks.

There are many instances in the design of transportation facilities and pedestrian networks in which one would like to estimate the volume of pedestrians who will pass a given point. In recent years, this question has most often been asked by transit station planners and designers who have traditionally had few analytical tools to aid in estimating pedestrian flow within stations. Recently, however, the Urban Mass Transportation Administration (UMTA) has sponsored development of a transit station simulation model (USS) to model pedestrian flow and to determine at what point queues will form. An innovative multiple-path traffic assignment model is incorporated into the USS program to simulate pedestrian flow. This paper reports on interactive computer research that is leading to improvements in the model.

Most traffic assignment models use minimum-path algorithms to simulate the choice of route by the driver or shipper. However, these models do not appear to be very realistic in their replication of human behavior, especially that of pedestrians. First, they typically yield routes structured only as trees that have one path per origin-destination (O-D) pair. Second, they apply the same cost-minimizing rule at each decision point regardless of the choice set. Third, they always select the minimum-cost route, assuming perfect knowledge of costs and perfect rational decision making by the driver or shipper.

In many instances, however, a pedestrian does not have perfect information about the route ahead. He or she must anticipate congestion and queueing and attempt to guess at the best available route. In addition, there may be several routes that appear to have equal costs between O-D pairs. Empirical work has indicated that for many types of flows there tends to be a set of routes used between O-D pairs (1,2). As noted by several researchers (1,3,4), distance estimation is subject to distortion and the minimum-time path is not always chosen by travelers.

MULTIPLE-PATH MODELS

In recent years, a number of researchers have developed methods of allocating traffic flows to competing routes (5-7). Of these so-called "stochastic traffic assignment" algorithms, Dial's analytical method, which is based on a multinomial logit formulation, has attracted a considerable amount of attention. Although a number of criticisms have been brought forth concerning the ability of this model to produce reasonable results (8-10), it lends itself to some interesting behavioral interpretations. The model has been implemented in UMTA's recently released USS model to simulate pedestrian path choice in transit stations. Dial's multiple-route traffic assignment model takes the form shown below:

$$P_i = \left[1 + \exp(\theta t_i) \cdot \sum_{j=1}^{n} \exp(-\theta t_j)\right]^{-1} \quad (1)$$

where

P_i = probability path i will be chosen,
exp = base of Naperian logarithms,
θ = parameter,
t = travel time,
j = family of competing paths (j = 1, ... , n), and
$i \neq j$.

Basically, the model allocates a percentage of all trips between an O-D pair to each efficient route based on the travel time for that route. Efficient routes are based on the calculation of an unconditional arc likelihood A(e) that a given link in the network will be traversed, as shown in Equation 2:

$$A(e) = \begin{cases} \exp\theta[P(j) - P(i) - T(i,j)] & \text{if } P(j) > P(i) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where

A(e) = arc likelihood,
(e) = link index for (i,j) pair,
P(j) = shortest path distance from origin to node j,
P(i) = shortest path distance from origin to node i, and
T(i,j) = length of link from i to j.

MODELING PEDESTRIAN FLOWS IN TRANSIT STATIONS

Research by Lutin (11) on the UMTA USS model showed that the use of Dial's stochastic traffic assignment algorithm was very sensitive to the value of θ used. If θ was set too high, only one path--the minimum path--was used by each individual who moved through the station. This led to the embarrassing result in which all passengers were modeled as leaving a subway train from a single door. Similarly, passengers all tended to queue at only one location on a platform. On the other hand, setting θ too low permitted individuals to wander through the station without much regard to signs or walk time to reach the exit. These effects were perplexing. While the multiple-path model seemed appropriate to model pedestrian paths, it needed to be fine-tuned to model spreading of flows on platforms and concentration of flows elsewhere in the station. In order to test various modifications to the multiple-path model, it was decided to develop an interactive graphics version of the model by using APL, an interactive programming language based on tensor algebra.

INTERACTIVE GRAPHICS IN MODEL RESEARCH

An entire interactive package was developed that

Figure 1. Multipath assignment probabilities as a function of path length and θ.

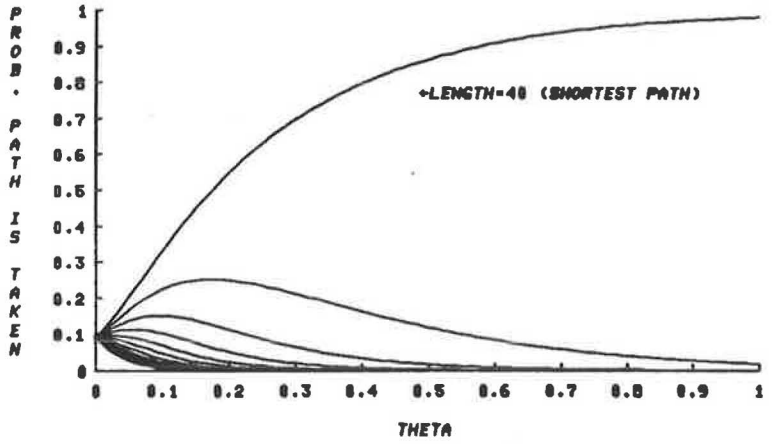
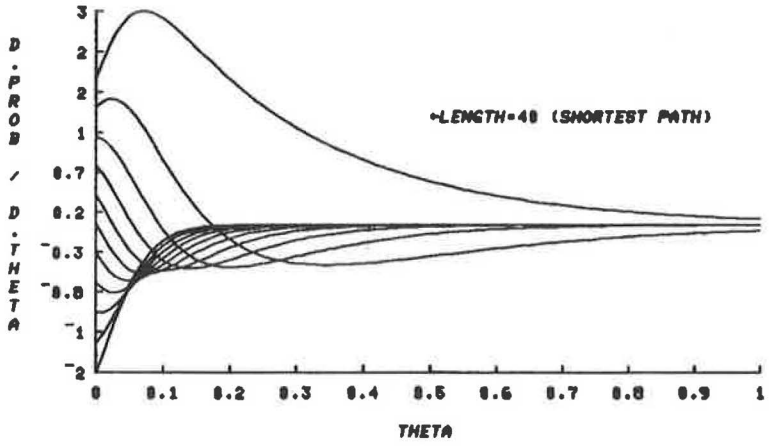


Figure 2. First partial derivative of path choice probability as a function of θ.



permitted one to create test networks, to edit networks, to calculate minimum paths, to input various modifications to the multiple-path model, and to examine output graphically on a cathode-ray tube (CRT) terminal. In this research, graphics was not used solely as pictorial output to display final results. Graphics was used as a means to explore alternatives and to determine whether various program elements were working correctly. Without the use of interactive graphics, this work would have taken considerably longer or might not have been done at all.

The first step in the research was to examine the sensitivity of the model to various settings of θ for a specified set of paths. This was accomplished by creating an interactive plotting program that would permit the user to input a set of paths by length and to specify a range of values for θ. The program would then plot the set of curves and indicate the probability of selecting each path as a function of θ. Figure 1 shows a plot of one set of curves for a set of paths. Each curve represents 1 of 11 paths. Path lengths are 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, and 80 min. In addition, one can find the first partial derivative of the model with respect to θ:

$$\partial P_i/\partial\theta = \left\{1 + \sum_{j=1}^{n} \exp[\theta(t_i - t_j)]\right\}^{-2} \times \left\{\sum_{j=1}^{n} (t_i - t_j) \exp[\theta(t_i - t_j)]\right\} \quad (3)$$

where $i \neq j$.

Figure 2 shows the plot of the first partial derivative of path-selection probability with respect to θ. For very small values of θ ($0 < \theta < 0.2$), one finds local maxima for the path-selection probability of five of the nonminimum paths. For $\theta > 0.2$, all derivatives are negative, except for the shortest path. Thus, in a very small range of θ, it is possible to divert trips away from the longer paths to several of the next-shortest paths. Discovery of the existence of local maxima for the selection probability of nonminimum paths was made possible by interactively plotting the derivatives for a set of paths simultaneously.

USE OF APL IN INTERACTIVE NETWORK MODELING

As noted earlier, the interactive language used for this work was APL, a tensor algebra-based language that permits one to treat and manipulate vectors and matrices as easily as scalars. The power of APL lies in five areas. First, one does not need to deal with matrices on an element-by-element basis, as in most other languages. Most primitive functions (e.g., +, -, x, ÷) can be applied to matrices and vectors as well as to scalars. Second, interactive use is accomplished through an APL interpreter, which permits one to halt programs in the middle and even rewrite them or change data before continuing execution. Both data and programs are accessible simultaneously within a single workspace, and virtual-storage computers permit one to use up to 16 000 000 bytes of direct-access virtual storage on-line for the manipulation of large files of data. Third, the language has developed the concept of the linear operator, or generalized operator, which can be applied to

vectors or matrices by using a two- or three-character symbol. By using these generalized operators, it is possible to manipulate networks in symbolic ways and map scalars, vectors, and matrices onto other vectors and matrices quite simply. Fourth, logical operators are developed in APL in a manner that permits filtering matrices through Boolean algebra. Finally, the data structures generated by APL are amenable to computer graphics display without much transformation.

A discussion of the application of APL linear operators to network modeling follows. Two specific algorithms were developed; one determined minimum costs of travel between nodes, and a second performed the arc-likelihood calculations found in Dial's algorithm. By using these matrix techniques, it became clear that forming a diagonal matrix of parameters was a convenient way of performing the calculations in APL.

By creating a diagonal matrix of θ-parameters, it became obvious that one could insert a unique value of θ for each node. Thus, in areas of the station in which one wished to simulate spreading of flows (such as on platforms), one could supply low values of θ. At key junctions and corridors in which persons might be directed by signs to the shortest paths, one could simulate a greater sensitivity to distance by increasing θ at selected nodes.

Algorithm to Determine Minimum-Path Cost

Lutin developed the following algorithm for finding the minimum-path cost and implemented it interactively in APL computer code on an IBM 370/158 computer (12). Basically, it is derived from the algorithm shown by Aho, Hopcroft, and Ullman (13) for transitive closure of a semiring. It is of interest not because it is efficient, but because it preserves the algebraic properties of the algorithm in its translation to APL.

The algorithm begins with the definition of a generalized linear operator $@_1 \cdot @_2$ so that for vectors V and V' of length n,

$$V(@_1 \cdot @_2)V' = S^* \tag{4}$$

where S* is a scalar. Likewise, for square (n x n) matrices, M and M' are as follows:

$$M(@_1 \cdot @_2)M' = M^* \tag{5}$$

where M* is a square matrix of order n. The operation shown in Equation 5 is termed a generalized inner product (14). Expanding on the operation of $@_1 \cdot @_2$, $@_2$ is specified as +, the element sum of two vectors V and V^T of length n, so that

$$\begin{bmatrix} V_1 @_2 V_1^T \\ V_2 @_2 V_1^T \\ \cdot \\ \cdot \\ V_n @_2 V_n^T \end{bmatrix} = \begin{bmatrix} V_1 + V_1^T \\ V_2 + V_2^T \\ \cdot \\ \cdot \\ V_n + V_n^T \end{bmatrix} = V^* \tag{6}$$

The first element of the linear operator $@_1$ is specified as a monadic operation (L) on vector V* so that

$$\min V^* = L(V^*) \tag{7}$$

where L(V*) is known as the floor of V* and represents the smallest element of V*. Thus the operator L.+ is called the minimum-cost operator. Given a matrix of costs (C), the algorithm establishes a vector of sums that represents the costs (i,k) + (k,j) of travel from node i to node j via each node k and selects the minimum cost, which is placed in the i,jth cell of C*. The operation is expressed below:

$$CL. + C = C^* \tag{8}$$

The operation shown in Equation 5 is repeated and C is replaced by C* in subsequent iterations:

$$CL. + C_{i-1}^* = C_i^* \tag{9}$$

It can be shown that C* is closed under the operation L.+, as in Equation 9. By using this property of closure, Equation 9 is repeated until $C_{i-1}^* = C_i^*$; that is, the matrix C* is unchanged for further iterations. C* is said to be idempotent. In APL, the entire algorithm is implemented in only six lines of code; Equation 9 appears in the code as a single line and is virtually identical to the algorithmic form presented here.

Dial's Algorithm

Turning to Dial's multiple-path algorithm, it too was transformed to matrix form by using linear operators as found in APL. In this case, it was necessary to use an outer-product operator.

Define : @ as a generalized outer product operator (14) so that, for vectors V and V' of length n,

$$V :@ V' = \begin{bmatrix} V_1 @ V_1' & \cdots & V_1 @ V_n' \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ V_n @ V_1' & \cdots & V_n @ V_n' \end{bmatrix} \tag{10}$$

Let V be a vector of distances from the destination node d to all other nodes j. Let @ be a dyadic operator (-). Then

$$D_d = [V : -V] = [q(i) - q(j)] \quad \text{for all } i, j \in N \tag{11}$$

where

D_d = matrix of cost differences with respect the destination node d,
q(i) = shortest-path distance from node i to destination node d,
q(j) = shortest-path distance from node j to destination node d, and
N = set of all nodes, N = 1, ..., i,j,n.

Let C be the cost matrix for the network so that

$$[c_{ij}] = \begin{cases} c_{ij} & \text{if } (i,j) \in E \text{ and } i \neq j \\ 0 & \text{if } (i,j) \in E \text{ and } i = j \\ \infty & \text{if } (i,j) \notin E \end{cases} \tag{12}$$

where

c_{ij} = cost or time of travel between nodes i and j,
i,j = node numbers,
(i,j) = link that connects node i to node j, and
E = set of all links (i,j) in the network.

Further, let θ be a diagonal matrix of the sensitivity parameters for each node i so that

$$\theta_{ij} = \begin{cases} \theta_{ij} & \text{when } i = j \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

Then θ is shown as follows:

$$\theta = \begin{bmatrix} \theta_{1,1} & 0 & \cdots & 0 \\ 0 & \theta_{2,2} & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \theta_{n,n} \end{bmatrix} \quad (14)$$

The arc-likelihood calculation can now be restated:

$$A(E) = \exp [\theta \cdot (D_d - C)] \quad (15)$$

The calculation $(D_d - C)$ represents the elementwise subtraction of the two matrices and is equivalent to $\{[q(i) - q(j)] - T(i,j)\}$, the arc likelihoods in Equation 2. A(E) represents the matrix of arc likelihoods and is the matrix product of θ and $(D_d - C)$. To test for the necessary condition $q(i) > q(j)$, a Boolean matrix, $D_d > 0$, is created. Likewise, another Boolean matrix is created to test for connectedness $(C \neq \infty)$. These matrices are used to form the elementwise product of A(E)*, as follows:

$$A(E)^* = (C \neq \infty) \times (D_d > 0) \times A(E) \quad (16)$$

GRAPHICAL TESTING OF MODEL RESULTS

In order to see how well the model worked and to determine how difficult it would be to calibrate, it was decided to create a test network by using an interactive-network building program. The interactive-network builder permitted one to enter links and nodes graphically by using a crosshair cursor controlled by thumb wheels on a Tektronix 4013 CRT terminal. Link costs were automatically calculated as the Euclidean distance between points on the Tektronix screen or entered separately as a vector of link costs. One-way links could be specified if desired.

An interactive program was created in which the user was asked to specify an O-D node pair. The user was queried to supply the value of θ desired. One could input a single value of θ for the entire network or a vector of θ-parameters that assigned individual values to each node. The program permitted the user to display various link data, such as minimum-path costs, A(e) values, or the percentage of travelers that used a given link. The user was queried to supply a title for each graphic display.

After receiving optional link-cost data from the user, the program calculated the minimum-cost matrix by following the algorithm described above. When possible, the program alternated between executing various functions and querying the user for information. This technique permitted us to cut down the length of the periods when the machine was processing data and appeared dead to the user. In addition, it made the execution seem faster, since the user was thinking about responses to program queries while the program was being executed.

After calculating the minimum paths and obtaining the desired O-D node pair and θ-value, the program calculated the probability of using each link in all feasible paths from origin to destination and stored the results in a vector. Prior to the display of the results, the user was asked to specify the title. Thus, immediately after receiving this last user input, the program began to draw the results on the screen; this emphasized the rapidity of response.

In the display, nodes were indicated by circles; the node number was displayed in the circle. Links were drawn as lines that connected each node. Passenger flow volumes were posted at the center of each link. To aid in visual interpretation of results, rectangles were drawn to the right of each link so that the width of the rectangle was proportional to the expected passenger flow volume. Parameter values were posted above and to the right of each node. To calibrate the model, one examined the display from a run and changed the vector of θ-values that had been used in the previous run.

STATION NETWORK CALIBRATION EXAMPLE

Figure 3 (11) shows a computer-drawn plan of a typical transit station; a platform, a concourse, and several possible exit corridors are shown. In Figure 4 (11), this station layout is shown as a computer-drawn network representation that has 22 nodes that are required for analysis by the simulation model. Node numbers are circled. The train is represented by node 1; train doors are indicated by nodes 2-9. Links that connect the train node 1 to the door nodes have been given dummy lengths of zero. In this example, only the flow of passengers from the train to the exit (node 21) is modeled. Figure 5 (11) shows an assignment of passenger flows by using a single parameter value, θ = 1, at all nodes. Flow volumes are shown by rectangles of proportional widths, and the percentage of flow is posted at the midpoint of each link. This figure shows a minimum-path all-or-nothing assignment; all passengers are represented as leaving the train via one door. No spreading of flow among train doors occurs.

In Figure 6 (11), the simulation was rerun, again with a single parameter (θ = 0). All efficient paths from node 1 to node 21 are included. Passenger flows are distributed fairly uniformly across all train doors and the platform. However, in the corridors that lead out of the station, one sees that the main diagonal corridor, which is the shortest route from the concourse to the exit for most passengers, receives only 33 percent of the flow. In addition, nodes 10 and 11, which represent train doors, receive smaller flows than the other door nodes.

In Figure 7, multiple-parameter assignment has been run. Parameter values are set to θ = 1 at nodes 12, 18, 19, 20, 21, and 22 and to θ = 0 at all other nodes. In this example, most detraining passengers are distributed uniformly across the platform and use the main corridor to exit the station. However, the distribution of passengers who leave the train is still not uniform, and passengers detraining at node 10 are shown backtracking to nodes 19 and 20. In Figure 8, parameters were increasd to θ = 1 for nodes 10 and 11 and decreased to θ = 0 for node 19. In this example, an equal volume of passengers leaves from each train door. Those detraining from doors at nodes 4 through 9 are all routed through the main concourse and the major diagonal corridor. Passengers detraining at nodes 10 and 11 are all routed out the side corridor via node 20. Thus, by selectively changing parameter values, one can fine-tune the station simulation to more accurately model passenger flows.

One of the major problems with the original Dial model is the fact that the parameter θ produces the same probabilities for a set of paths that differ by similar increments of length regardless of the lengths of the paths. For example, the selection probabilities for a set of three paths with

Figure 3. Transit station plan.

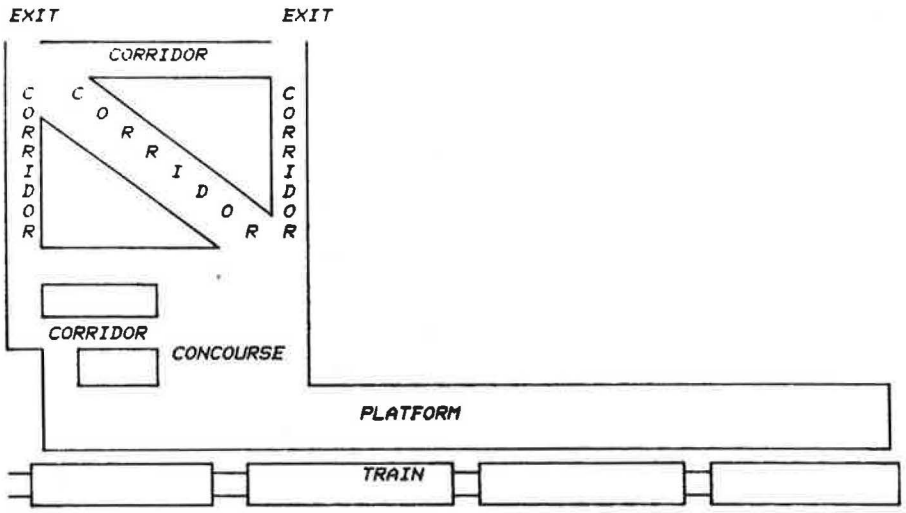
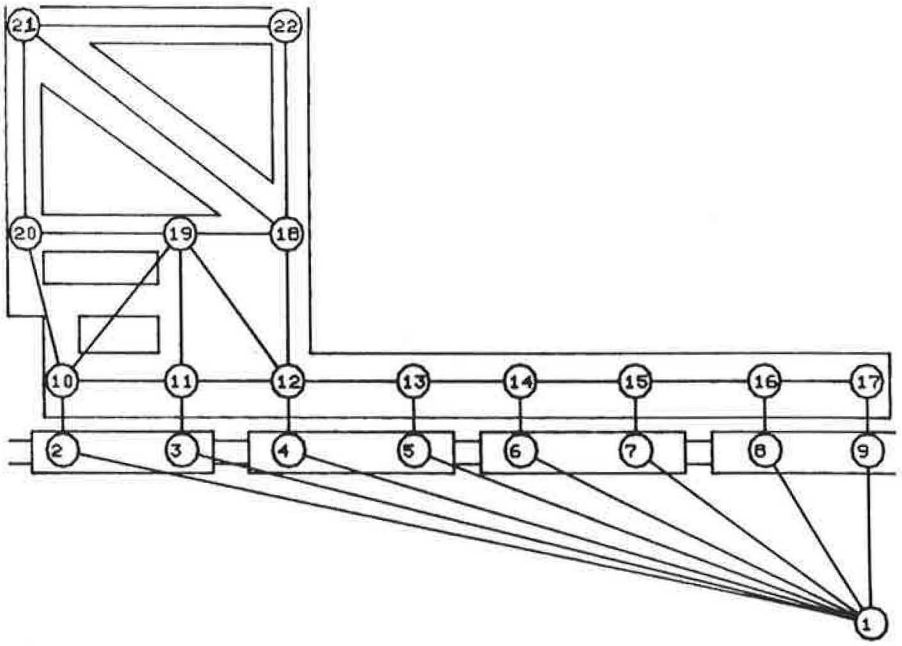


Figure 4. Network representation of transit station.



respective lengths of 5, 6, and 7 min are identical to the path-selection probabilities for paths 15, 16, and 17 min long, respectively. According to Dial's model, for any value of θ, the same proportion of individuals who choose the 17-min path, which is 13 percent longer than the shortest path, will choose the 7-min path, which is 40 percent longer than the shortest path in that set of choices. However, in the multiple-parameter model, each set of links that emanates from a node can be evaluated by using a parameter value appropriate to the set of choices at that node. Although this multiple-parameter formulation still contains some of the undesirable properties of the original model, it can be used more flexibly and calibrated much more easily.

REDUCING THE IMPACT OF ROUTE OVERLAP

One of the most severe criticisms of Dial's model is that unreasonable assignments are obtained when routes overlap, especially when these competing routes have identical or nearly identical travel times (7). By using a well-worn example (Figure 9), one would expect flow that emanates from node 1 to be split equally between link 1-7 and link 1-2. However, Dial's model produces twice as much flow on link 1-7 because this link appears in two distinct paths (1-7-6-5-4 and 1-7-6-8-4) as shown in Figure 10 (flow volumes may not total 100 due to rounding).

Essentially, in calculating the link weights, the model is calculating the binomial distribution of all possible paths between origin and destination, and the weight calculated for each link represents the number of feasible paths in which that link participates. This effect can be ameliorated by introducing a weight (λ_{ij}) for each link that is proportional to the number of paths that cross that link. Let Ψ^p be an n x n matrix (where n is the number of nodes) that represents the set of feasible links (i.e., "efficient links" in Dial's terminology) for the set of possible paths between O-D pair p. The elements ($\psi_{i,j}p$) of Ψ^p are as follows:

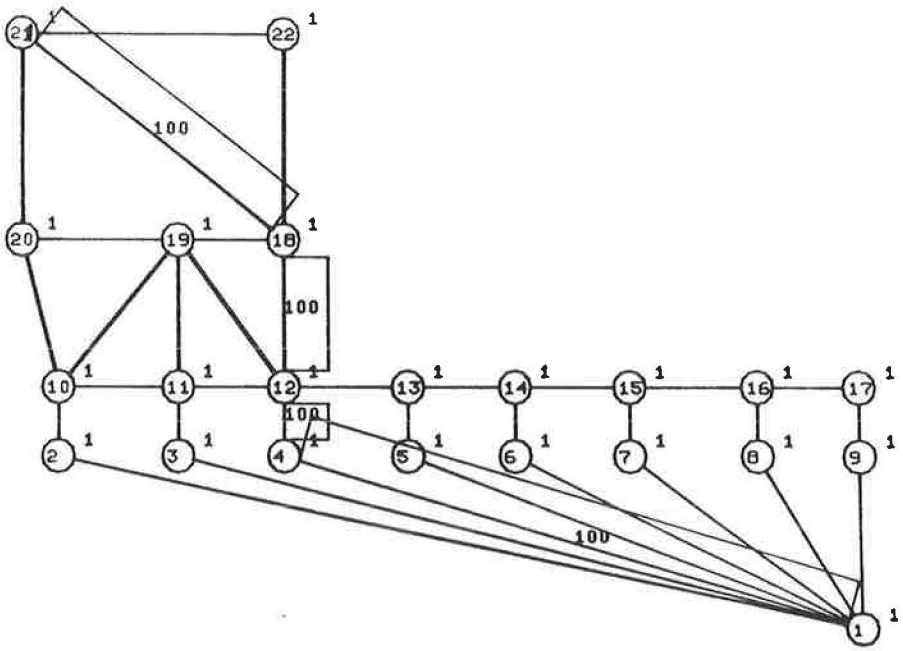


Figure 5. Station network with minimum-path flow from node 1 to node 21.

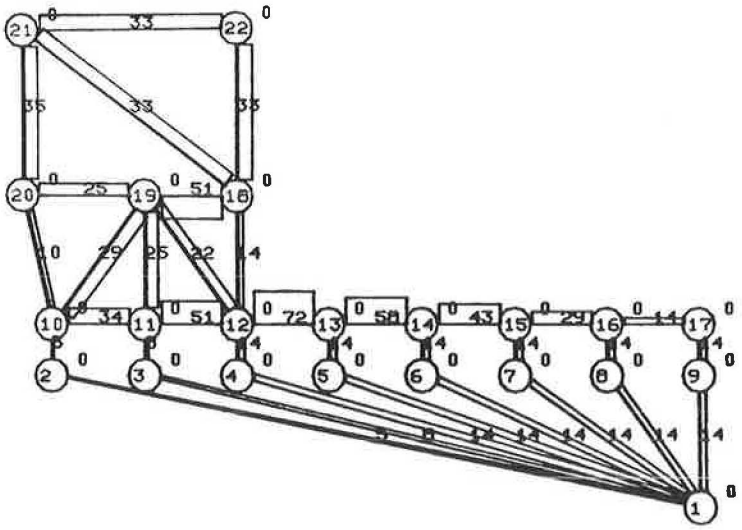


Figure 6. Station network with loading on all efficient paths.

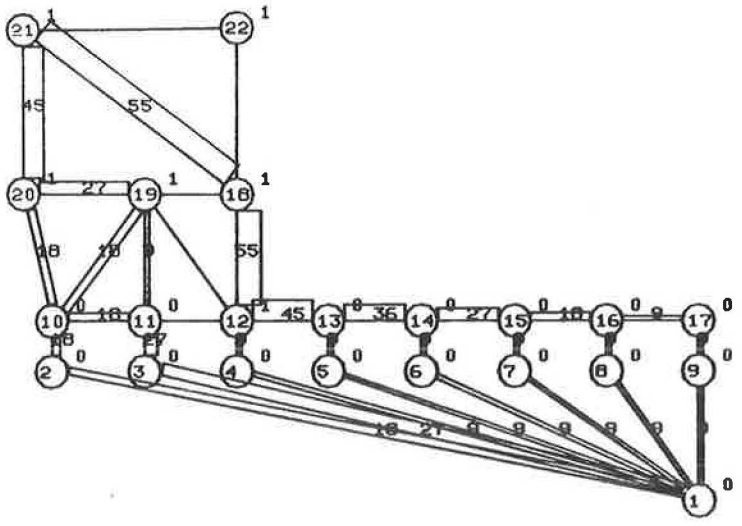


Figure 7. Multiple-parameter assignment (first run).

Figure 8. Multiple-parameter assignment (second run).

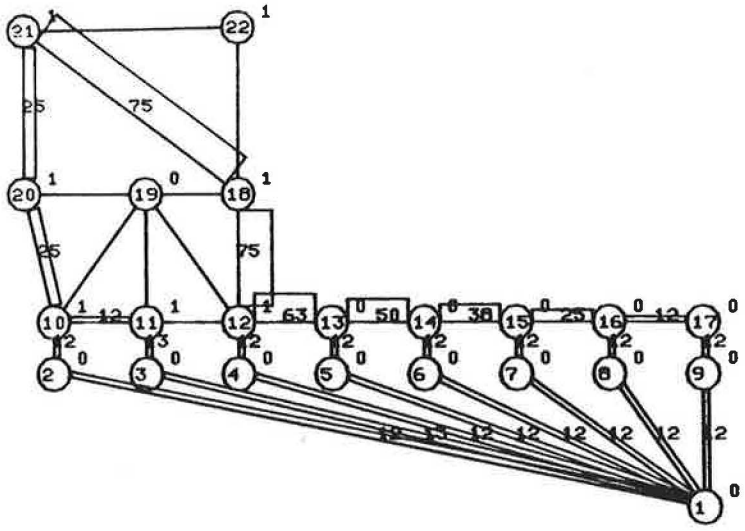


Figure 9. Expected flows from node 1 to node 4.

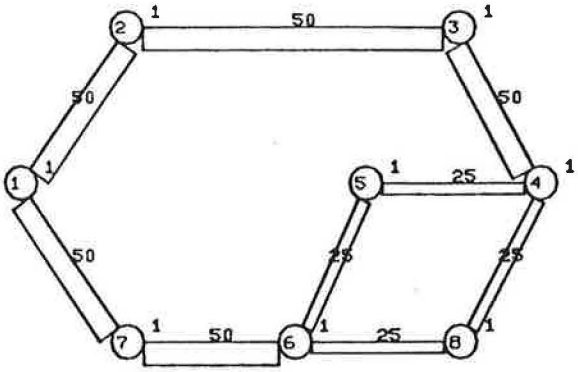
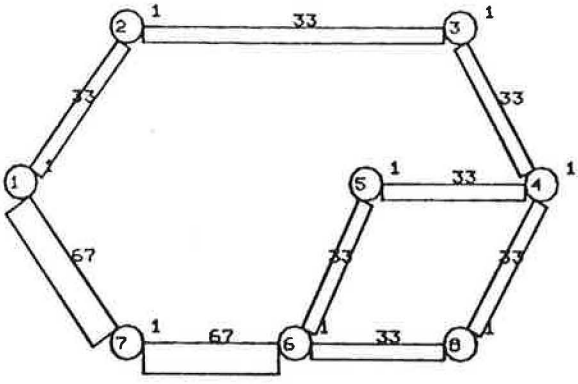


Figure 10. Flows generated by Dial's algorithm from node 1 to node 4.



$$\psi_{i,j}^{p} = \begin{cases} 1 & \text{if } A(E)_{i,j}^{*} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

where A(E)* is the n x n matrix of arc likelihoods as in Equation 9. Then, for each node i in N, let

$$\lambda_i = \sum_{j=1}^{n} \psi_{i,j}^{p} \quad \text{for all } i,j \in N \quad (18)$$

A new set of weighted arc likelihoods $\Gamma(E)$ that has elements $\gamma_{i,j}$ can be calculated as follows:

$$\gamma_{i,j} = A(E)_{i,j}^{*} / \lambda_j \quad (19)$$

The remaining calculations follow Dial's algorithm; weighted arc likelihoods $\Gamma(E)$ are substituted for A(E), the conventional likelihoods.

Figure 11 shows a network that contains three diversion points; the loadings were calculated by using Dial's algorithm. One would expect the upper route (1-8-7) to attract 50 percent of the trips, given that two paths of equal length emanate from node 1. Dial's model, however, finds four distinct paths through the lower set of links and allocates 80 percent of the trips to link 1-2. Figure 12 shows the results of the modified version in which both upper and lower routes share equally in the traffic.

Figure 13 shows a slight modification of the network in Figures 11 and 12. Here, a shortcut, link 9-10, has been added. The value of θ has been reduced to 0.02 to permit multiple-route simulation. (If $\theta = 1$ in this example, only the minimum path would be selected.) Note that only 33 percent of the trips are assigned to the upper route, and 42 percent take the shortcut. For the sake of a contrived example, we assume that 50 percent of the observed flow travels the upper route, whereas everything out of node 2 follows the minimum path, but at node 9 traffic splits equally among all paths to the destination. To achieve flow concentration at node 2, θ_2 is increased to 1, and, to achieve equal dispersion at node 9, θ_9 is set to zero. Through iterative analysis (i.e., trial and error by using interactive graphics), it is found that adjusting θ_1 downward to 0.0085 will balance the arc weights of all paths to achieve the desired flow (Figure 14).

CONCLUSIONS AND DISCUSSION

Although many researchers have cast doubt on the ability of Dial's model to predict flows accurately or even to behave reasonably, there still seems to be life left in the model. This paper has shown that several modifications are possible that improve its behavior. First, the link-efficiency criterion was changed to select links closer to the destination. Second, it was shown that it was possible to include a unique parameter for each node. Third, the overallocation of trips to overlapping paths was reduced by weighting the arc likelihoods.

Others have suggested that path enumeration, simu-

Figure 11. Flows generated by Dial's algorithm on network with three cycles.

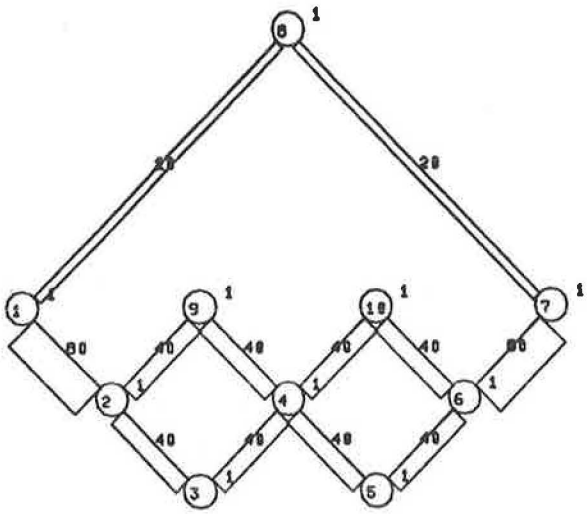


Figure 12. Flows generated by modified algorithm by using gamma weights.

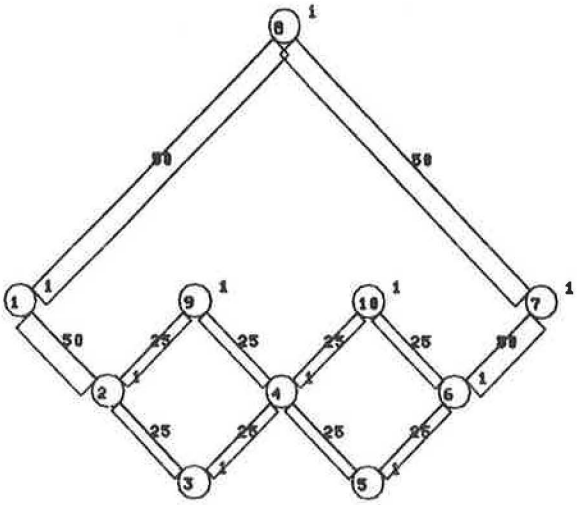


Figure 13. Single-parameter assignment, modified algorithm.

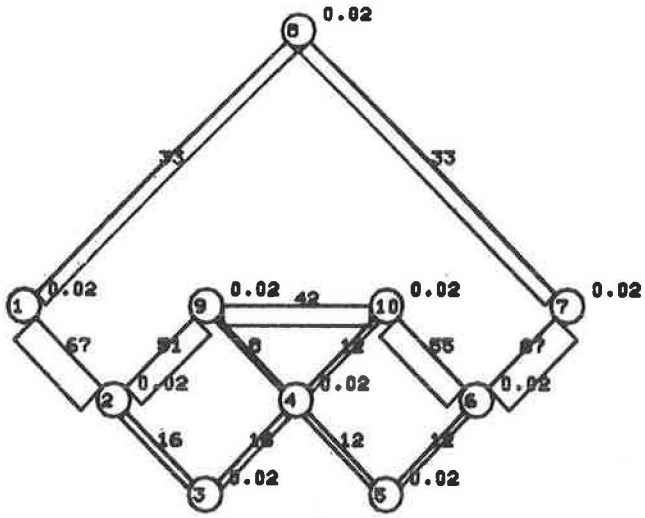
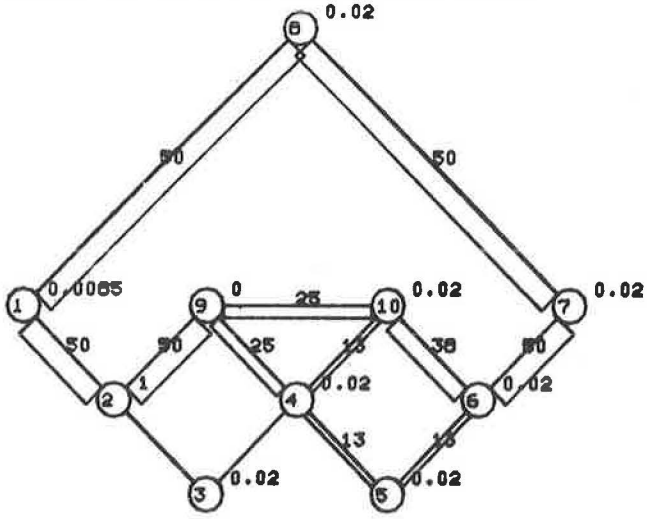


Figure 14. Flow balancing and concentration, modified algorithm.



lation, or interactive editing combined with a stochastic model may yield a more-fruitful approach to multiple-route traffic assignment. Daganzo and Sheffi (7) showed that Dial's logit model implied that route error terms are independent and identically distributed. Their approach involved developing the actual variance-covariance matrix of route travel times. Although this seems theoretically more appealing, it nevertheless requires calculation of the link-route incidence matrix for all feasible paths for each O-D pair. For most networks, this is now computationally infeasible. For example, a simple square grid network that has 10 nodes and 9 links on a side has 48 620 possible paths that connect the extreme nodes on the diagonal. It is questionable whether a simulation approach can be substituted for path enumeration and can still achieve reasonable results for large networks. In spite of all its shortcomings, the modified version of Dial's model still holds promise of becoming an efficient, accurate, and easily implemented multiple-route traffic assignment algorithm.

By introducing multiple parameters, a number of degrees of freedom have been incorporated into the model. With the multiple-parameter model, calibration to fit actual route itineraries for individual travelers may be possible. In addition, it is possible to provide a better, more-appropriate means of calibrating traffic assignment models, since one now has much better control over modeling the decision-making process. Previously, in order to make traffic assignment models agree more closely with reality, one had to adjust link travel times and costs, essentially distorting the network to account for behavior. When the multiple-parameter model is used, one can use measured link times and there is no need to tinker with the network itself.

In the model developed in this work, the graphic capabilities of the computer system have been used to provide visual output. Since the subject of the investigation is a network, which is a representation of a spatially determined transport system, the use of graphic output enables the analyst to link numerical results to the spatial structure. Looking at a picture of the network annotated with numerical results provides an easier means of interpreting data than scanning through tables of numbers. In fact, the graphic display capability proved valuable in the development of the model itself, since one

could directly assess several aspects of the model's performance by inspecting the graphic output. Numerous errors were detected in this way that might not otherwise have been caught.

In developing the algorithms, it was found that the ability to program interactively gave greater flexibility in modifying the models and encouraged more experimentation. It was not necessary to submit a program and let the machine grind away. Execution could be interrupted at any time, intermediate output could be examined, and data or programs could even be modified before operation continued. Many ideas could be tested in the time needed for a single run of a conventional model. In addition, the power of APL enabled dealing with networks as matrices and vectors without having to program tedious intermediate steps and do-loops. The algebra of the problem could be dealt with almost directly rather than having to translate the mathematics into a computer language.

ACKNOWLEDGMENT

All model development work was accomplished on an IBM 370/158 computer at the Interactive Computer Graphics Laboratory (ICGL), Princeton University. All programming was done interactively by using VSAPL computer language under a virtual machine configuration that had the conversational monitor system. All figures were drawn by using assembler language drawing functions developed by the ICGL staff. Curve plots used functions of the LINPLOT package developed by Tom Hahn and Yehonathan Hazony of the ICGL. I wish to thank Granville Paules, Linda Moore, and Robert B. Dial of the Office of Planning Methods and Support, Urban Mass Transportation Administration, for their assistance in obtaining support for this research. This research was performed under a research, development, and demonstration grant from the Urban Mass Transportation Administration, U.S. Department of Transportation.

REFERENCES

1. B. Marchand. Pedestrian Traffic Planning and the Perception of the Urban Environment: A French Example. Environment and Planning, Vol. 6, 1974.
2. A. L. Kornhauser and S. E. Still. Analysis of the Flow of Freight Traffic on the U.S. Railroad System for the Year 1974: Final Report. Transportation Program, Princeton Univ., Princeton, NJ, 1977.
3. T. Lee. Perceived Distance as a Function of Direction in the City. Environment and Behavior, Vol. 2, 1970, pp. 40-51.
4. M. Wachs. Relationships Between Drivers' Attitudes Toward Alternative Routes and Driver and Route Characteristics. HRB, Highway Research Record 197, 1967, pp. 70-84.
5. J. E. Burrell. Multipath Route Assignment and Its Application to Capacity Restraint. Proc., Fourth International Symposium on the Theory of Traffic Flow (Karlsruhe, Germany, 1968).
6. R. B. Dial. Probabilistic Assignment: A Multipath Traffic Assignment Model Which Obviates Path Enumeration. Univ. of Washington, Seattle, Ph.D. dissertation, 1970
7. C. F. Daganzo and Y. Sheffi. On Stochastic Models of Traffic Assignment. Transportation Science, Vol. 11, 1977, pp. 253-274.
8. M. Schneider. Probability Maximization in Networks. Proc., International Conference on Transportation Research (Bruges, Belgium, 1973).
9. R. L. Tobin. An Extension of Dial's Algorithm Utilizing a Model of Tripmakers' Perceptions. Transportation Research, Vol. 11, 1977, pp. 337-342.
10. M. Florian and B. Fox. On the Probabilistic Origin of Dial's Multipath Traffic Assignment Model. Transportation Research, Vol. 10, 1976, pp. 339-341.
11. J. M. Lutin. Transit Station Simulation Model (USS) Testing and Evaluation. Princeton Univ., Princeton, NJ, Transportation Rept. 76-TR-9, 1976.
12. J. M. Lutin. A Perceptual Model of Path Choice in Urban Transportation Networks. Princeton Univ., Princeton, NJ, Ph.D. dissertation, 1977.
13. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, MA, 1974.
14. R. P. Polivka and S. Pakin. APL: The Language and Its Usage. Prentice-Hall, Englewood Cliffs, NJ, 1975.

Use of Computer Graphics to Analyze Navigational Performance and Jet-Route Separation

DALE LIVINGSTON, EDWARD J. KOBIALKA, AND NEIL W. POLHEMUS

Formulation of jet-route separation standards to achieve adequate levels of safety requires careful analysis of aircraft navigational performance. In the processing and analysis of extensive data obtained from the continental U.S. air traffic control (ATC) system, computer graphics techniques were used to develop a clean data base, to provide descriptive summaries of achieved navigational performance, and to portray the relationship between collision risk and track-system geometry. This paper describes these applications and their importance in ATC systems analysis.

As part of a general study of aircraft separation standards, the Federal Aviation Administration (FAA) has collected a large amount of radar data on aircraft navigational performance in the high-altitude continental U.S. jet-route system. These data were collected to serve as the empirical base for assessing the adequacy of current route-spacing criteria. In all, more than 10 000 flight records were collected along route segments in the Cleveland,