INTRODUCTION

An interesting problem that is amenable to solution by digital computer is posed by the following questions. How late can a shipment be detained at city A so that it arrives at city B by a given time? By what route should it be sent? The available routes consist of those provided by scheduled common carriers such as the airlines.

In some situations, no single scheduled carrier trip satisfactorily connects the two cities involved. In such a case it might be necessary to use two vehicles and to transfer the mail between the two at a third city. Conceivably, it might be necessary to use three or more vehicles and two or more transfer points.

The problem may have more meaning if it is posed by the following more personal questions. How late can I stay in my home town and still get to an appointment in another city on time? What route should I take?

An important application of a solution to this problem can be found in the Post Office Department. The Department tries to process mail received in the afternoon so that it will be delivered the following morning at distant cities.

This problem has been studied at the National Bureau of Standards under the sponsorship of the Post Office Department. This paper describes and discusses some solutions that have been obtained. These solutions are related to and stem from published literature regarding the Shortest Route Problem.

THE PROBLEM

Relationship to the Shortest Route Problem

Given a network of points and lines, the latter numbered by the distance between the points they connect, it may be of interest to know the shortest path between any two points. This is the shortest route problem and many solutions to this problem appear in the literature.1 2 5 7

If the points correspond to cities and the values of the lines correspond to the travel times between the cities, solutions to the shortest route problem yield the route with the least time in transit. These solutions do not necessarily give optimum answers to the questions posed in the introduction. For example, suppose we live in Boston, Mass. and must get to Binghamton, N.Y., for a luncheon meeting. There is a plane that leaves Boston at 6:30 a.m. and which arrives in Binghamton at 8:37—a total flight time of 127 minutes. Getting up in time to catch a 6:30 a.m. plane is not a very satisfying thought, so we investigate the possibility of going another way. We find we can take a plane from Boston to New York's Idlewild Airport, leaving Boston at 7:45 a.m. and arriving at Idlewild at 8:38. We can then transfer to another plane leaving Idlewild at 9:30 and arriving Binghamton at 10:37. This routing requires 172 minutes from the time the first plane is scheduled to leave Boston to the time the second plane arrives at Binghamton; 52 minutes of the elapsed time is involved in transferring from the first plane to the second. This
is not the shortest route, but it may be the desired one.

Factors to be Considered

If our only consideration is to leave Boston as late as possible in order to make our luncheon appointment, then the routing by way of Idlewild is superior because it leaves 75 minutes later than the direct flight. There are, however, other factors that must be taken into consideration in deciding which routing is superior, in addition to flight time and departure time.

1. Cost: It costs $21.80 to fly directly from Boston to Binghamton. It costs $28.80 to fly from Boston to Idlewild to Binghamton. In other words, we must pay seven dollars plus tax for the extra 75 minutes of sleep.

2. Reliability: The routing by way of Idlewild allows 52 minutes for transferring from the Boston-Idlewild plane to the Idlewild-Binghamton plane. If the Boston-Idlewild plane were late, the connection might be missed, causing us to be stranded at Idlewild and, therefore, to miss our luncheon. And further, if the Idlewild-Binghamton plane were late, we might also miss our luncheon. Also, any scheduled airline flight is subject to cancellation due to weather or equipment problems. The direct flight from Boston to Binghamton involves only one plane and therefore such problems are less likely to occur than in the two plane routing by way of Idlewild. In other words, to get 75 minutes more sleep, we would increase the probability of not arriving by the required time.

3. Transfer time: We have considered the problem of missing connections due to the late arrival of the first plane in a two-plane routing. Even if the first plane is on time, some time must be allowed for transferring from it to another plane. This time varies according to the size of the airport and according to whether or not the transfer is between two planes of the same airline or between planes of two different airlines. The recommended minimum time for transferring at Idlewild from the Boston-Idlewild flight to the Idlewild-Binghamton flight is 30 minutes, which is less than the available 52 minutes. In addition to the 7:45 a.m. flight, there is a plane from Boston to Idlewild leaving Boston at 8:30 a.m. arriving at Idlewild at 9:22 a.m., only 8 minutes before the departure from Idlewild to Binghamton. We would not select this flight because eight minutes is not sufficient transfer time.

In summary, we have stated five factors that are of importance in selecting routes: (1) departure time, (2) cost, (3) reliability, (4) transfer time, and (5) arrival time. The only required property of the arrival time is that it be before a specified time.

PRELIMINARY SOLUTIONS

Selection of Flights Along a Known Route

A digital computer can be programmed to select flights along a known path. For example, the known path could be Boston to Idlewild to Binghamton. There are several flights each day between both pairs of airports. The computer can be programmed to select the set of flights that best meets the criteria discussed in the previous section. Only the criterion of time will be discussed in this section.

To simplify matters, the procedures in this paper will be described by means of terminology peculiar to airline flights. The procedures can be used for air transportation, surface transportation, and for combinations of surface-air transportation.

The selection of the flights is not as simple a task as it may at first appear. One solution would be to check all possible combinations of flights. While this is simple conceptually, it involves an inefficient use of the computer. Another obvious approach would be to select a desirable departure time from Boston and choose the flight to Idlewild with the departure time closest to this desired time. The first flight leaving Idlewild for Binghamton after the arrival of the selected Boston-to-Idlewild flight would be the selected second link flight. This solution is efficient but has two important shortcomings which are easy to overcome:

1. It overlooks some problems involved in making transfers.
2. It is not designed to answer one of the specific questions asked, namely, that of leaving as late as possible.

In making transfers from one flight to another, it is necessary to consider whether or not the transfer is between two planes of the same airline or between two planes of different airlines. A transfer between two planes of the same airline is called an intra-line transfer; a transfer between planes of different airlines is
called an inter-line transfer. The minimum time allowed to make a transfer is usually less in the first case than in the latter. There is no problem in having two minimum transfer times for each airport: one for intra-line transfers and one for inter-line transfers. The problem arises when the first flight to arrive requires an inter-line transfer and a second flight, which arrives soon afterward, involves an intra-line transfer. It is possible that the earlier flight cannot make connections with the plane which departs at the desired time, while the later flight can. For example, suppose there is a flight from city A to city B leaving at 7:00 and arriving at 8:00 on airline one. Suppose there is also a flight between the same two cities leaving at 7:05 and arriving at 8:05 on airline two, and there is a flight on airline two departing city B for city C at 8:45. If intra-line transfers require 30 minutes and inter-line transfers require 60 minutes, then the flight which arrives at 8:05 can make connections, while the 8:00 flight cannot. The solution to this problem of transfer times is to consider as potential first-link flights all flights whose arrival time at city B is within X minutes after the earliest arrival time at city B, where X is the difference between the inter-line and intra-line transfer times at city B. In cases of routes having more than one transfer point, this factor must be considered at each transfer point.

The problem posed in Section 1 was to select a routing that permitted the shipment to be detained at the originating city as late as possible. The solution described so far uses a prescribed earliest departure time and, therefore, does not really solve the problem posed. This can easily be corrected by tracing the route backwards. The last leg of the trip would be selected first, based on its arrival and departure times, and so forth. The basic nature of the solution would be unchanged. In overcoming the transfer problem described in the previous paragraph, we would consider as potential last link flights all flights whose departure time at city B is within X minutes before the latest departure time at city B.

Selection of Promising Routes

The backward tracing of the path with proper correction for the transfer problem will select the optimum path on a basis of the criterion of latest departure. However, it requires the routing to be predetermined. In order to find the best set of flights, it will often be necessary to trace out numerous routes. These routes can be determined by another computer program or by someone making decisions regarding the routes to be tested. An algorithm has been developed which finds the N shortest routes between two given cities where \( N_1 < N_2 < N_3 \) and \( N_1 \) and \( N_2 \) are variables. This algorithm considers only the travel time among the cities and does not consider delays involved in transferring. It will be the subject of a separate paper now being prepared. (It would have been possible also to have used other procedures for finding the N shortest routes.)

For each of the N routes, a set of flights is selected and compared with the sets of flights for the other routes, and the set of flights with the latest departure time selected. A computer program has been written which selects the N shortest routes neglecting transfer times, selects a set of flights for each route considering transfer times, and rank orders them on the basis of latest departure time. This program uses as input data IBM cards containing the name of the airline, the flight number, the departure airport and time, and the arrival airport and time. It can handle 2,000 trip segments and 45 transfer points. To save space in the computer memory, a trip that has several stops is considered as a series of nonstop trip segments. For example, a plane trip from Boston to New York to Washington is treated as two trip segments: Boston to New York and New York to Washington.

The program was written in FORTRAN for the IBM 704 and 7090 computers. On the IBM 7090, it took two minutes to find paths between 30 pairs of cities. This involved determining and tracing 265 paths.

There is no assurance that the best route will be one of the N shortest routes as determined by the algorithm. All of the N shortest routes may involve poor connections at the transfer points while a longer routing may involve good connections at the transfer points. This is a weakness of the procedures. Also, cost is not considered.

Until now, the only consideration given to the arrival time has been that it be before the prescribed arrival time. After the latest possible
departure time at the city of origin is determined, consideration can be given to selecting the earliest arrival time. The airline passenger may wish to depart as late as possible, but he would prefer having idle time in the city of destination rather than at a transfer point. In the same way, we may want to have the departure of the last dispatch of mail as late as possible. (Mail ready for an early dispatch would be sent on the early one.) However, once the dispatch time is set, it would be desirable to get the mail to its destination as early as possible so that there would be more time to process it at the destination post office. This can be accomplished by retracing the path in a forward direction after the departure time has been determined by the backward tracing. For example, if we wish to leave Minneapolis and arrive at Cleveland by 7:00 p.m., we would find, by tracing backwards, that we could take a 3:45 plane from Chicago, arriving in Cleveland at 6:20. We could take a 1:45 plane from Minneapolis to Chicago and connect with the 3:45 plane. Tracing forward, we find we could take the 1:45 plane from Minneapolis and make connections in Chicago with a plane arriving at Cleveland at 6:05, fifteen minutes earlier than the trip selected by the backward tracing.

Selection of Optimum Routes

A computer program has been written that does not have the limitations just described. It always finds the fastest route; it includes cost as a factor; and it does not require both a forward and backward tracing. It is based on the algorithm presented in the Appendix (with modifications indicated below) and is similar to a procedure suggested by Minty. The basic idea is to find the best direct flights given a starting time from the originating airport to all other airports being considered. The direct flights are the links of the network. Each of these selected direct flights is considered as a possible first link of a two-link route to each of the other airports. The airport at which the direct flight lands is considered as a potential transfer point. Every flight leaving that airport is paired with the direct flight. Each resulting two-link route is compared with the best previously determined route to the arrival city of the two-link route. If it is better, it is stored in place of the previously determined best. After all the stored one-link routes are tested, the stored two-link routes are tested to see if they can be used as the first two links of a usable three-link route. Whenever the three-link route is better than the previously stored route, it is stored in place of the previous best. This process is continued until for some \( m \), all of the \( m \)-link stored routes are tested as the first \( m \) links of a route having \( m+1 \) links, and none of the \((m+1)\)-link routes warrant retention. This algorithm is similar to the “Moore Algorithm”.

The transfer problem described in the previous section arises here also. It is solved by storing more than one route between a pair of cities whenever conditions warrant, and using each of the stored routes to see if it can be the first part of a route to another city. The criteria for storing additional routes are the same as before. That is, consider as potential first-link flights all sets of flights along any route whose arrival time at city \( B \) is within \( X \) minutes after the earliest arrival time found so far at city \( B \), where \( X \) is the difference between inter-line and intra-line transfer times at city \( B \).

It should be noted that this solution finds routes not only from the origin city to the destination city but also from the origin city to all other cities. Hence, some routes are retained only because they are promising routes to potential transfer points; other routes are retained because they are tentatively the best routes to a city as well as because they are promising routes to potential transfer points.

As described above, the solution is geared to an originating time rather than to a desired arrival time. Another approach would have the program work from a desired arrival time and find the latest possible departure time from each of the other cities to that city. After the departure times are found, the earliest arrival times to the destination city based on each of the computed departure times could be determined. However, the approach described below should be more desirable because it finds good routes (fastest for some departure time) for all times of day in an efficient manner. If good routes for all times of day are known, the desired route for a given arrival time can easily be selected. The best routes for all times of day can be computed by means of the approach that follows.

The best routing for a given departure time, say 11:30 p.m., could be determined. Then the
best routes for a new departure time, say 10:30 p.m., could be considered in the following manner: Using as first links only flights which depart between 10:30 and 11:30, new potentially useful multiple-link routes would be computed. In deciding whether or not to save a computed route, it must be compared with the best route found so far, which includes those routes based on the 11:30 departure time. The computations involved would be less time-consuming for this second departure time because a set of flights would be saved only if it is better than the retained 11:30 departure routes. A third time, say 9:30, could be selected and the process continued by selection of earlier times throughout the day. The final results will be the fastest routes from the origin city to all other cities for each departure time used. (Duplicate routes can be suppressed.) From this mass of data, the routing that best answers the question originally posed can be selected. In addition, data are available to answer many similar questions.

This solution involves one minor problem. This can best be described by an example. Suppose we wish to arrive at a given city by midnight. If there are two direct flights, one which leaves at 10:30 p.m. and the other at 10:40 p.m. and both take an hour, the algorithm as described above would pick the flight with the earlier departure and arrival times, the flight leaving at 10:40. The 10:50 flight leaves later and would be the better flight according to the criteria described earlier in this paper. However, the selected flight is so similar to the best flight that this cannot be considered an important problem, especially since the interval between the departure times considered is subject to control.

An advantage of this approach is that a large amount of useful data is obtained in a systematic fashion.

LATEST SOLUTION

Routes to Be Stored

There is one property of the algorithm in the Appendix that warrants emphasis. This property can be best be described using the network in figure one. Assume node A is the origin point. The shortest routes to node B and node C are the direct links of four and three units length, respectively. We then try to develop two-link routes to C, D, E, and F, using the link AB as the first link. The two-link route ABC is nine units in length, which is longer than the direct link. Although we compute the link ABC, we do not retain it. The links ABD and ABE, on the other hand, are retained. We then try using as a first link AC. The route ACB is computed but rejected. The route ACE (length 6) is computed and retained in place of the route ABE (length 8). In going through these operations we systematically consider all possible ways of extending each M link route to M + 1 link routes. In developing three-link routes, the route ABD is extended to make the route ABDF and the route ACE is extended to make the route ACDF, which is longer than the stored route ABDF and is therefore ignored. The route ABDF is extended to the route ABDFE which is longer than the route ACE, and therefore is not retained. In finding the route ABDF, the rule of using an M link route to find an M + 1 link route still held. There was no need to use any other procedure to find this route, such as extending the route AB by two links at one time.

Listing Procedure

In adapting the algorithm to find optimum transfer airports and flights, it is sometimes necessary to retain several routings from the origin airport to a given transfer airport. If time is the only criterion for the selection of an optimum route, the only criterion for retaining non-optimum routes to potential transfer points is also time, where the amount of time is a function of the required minimum transfer times at that city. We need consider only the arrival times at that one point along the route; we need not worry about transfer problems at previous cities along the route nor at cities yet to be added to the route.

As indicated above, it is sometimes necessary to save more than one set of flights between the origin city and another city, simultaneously. To save space in the computer memory, the retained flights are stored in a list format. There is a limit on the space reserved for this list. However, there is no limit on the number of routes that can be stored between the origin city and a potential transfer point so long as there is space remaining in the list to store information regarding these routes.
Additional Criteria

The introduction of the list procedure permits additional route-selection criteria to be introduced. In introducing additional criteria, it is necessary to be very specific. Additional criteria that have been introduced and programmed are those of specific interest to the Post Office Department in routing air mail.

The Post Office Department pays for air transportation according to the following rules.

1. If only one airline is involved, the Post Office pays a loading charge based on the size of the airport plus a transportation cost based on the "short line distance" between the origin and destination airport. The short line distances are the shortest distances between the two airports involved, using a single carrier. For example, suppose we wish to get from airport A in Figure 1 to airport E and the routing we wish to consider is from A to B to E on airline 1. If airline 2 has planes between airports A and C and also has planes between C and E, airline one will be paid for only one loading charge and only six units of distance (the distance of the ACE route) rather than the eight units of distance that the mail was transported.

2. If more than one airline is involved, the short line distance is paid each airline for any continuous portion of the route handled by that airline. An additional loading cost is added each time the mail is transferred from one airline to another, based on the size of the airport at which the transfer is made.

It should be noted that when these rules apply, a straightforward application of the algorithm cannot answer questions regarding cost. However, as actual sets of flights are selected, costs can be computed. Whenever the fastest route found can also be saved. Hence, the criterion of cost, as well as that of speed, can be taken into account.

The introduction of the criterion of cost with the costing rules described above creates some problems. If we consider the problem of getting from A to B by way of T, the following could happen. Let us say that the best flight from A to T is on airline one, while airline two provides a flight which arrives much later. However, airline two provides the only service between T and B; therefore, the more time-consuming connections from A to T and B on airline two are cheaper, because there is no inter-line transfer cost. In other words, many poor flights and sets of flights must be retained and tested if it is required that the cheapest routing be found. This would greatly increase the computer time necessary. In order to make solutions practical, we have programmed the computer to select the cheapest set of flights that arrives at a transfer point less than X minutes after the fastest, where X is a variable set equal to, say, 120 minutes. Each of these selected sets is tested as the first m links of a route having m+1 links.

The problem of finding alternate routes when the fastest route does not operate every day is straightforward. Each computed routing can be tested to see if it is the fastest for any day of the week and if it is the fastest, it is retained.

The Program

A computer program has been written and debugged using the techniques described in this section which does the following:

1. It finds the fastest route from an origin city or airport to all other cities or airports.
2. It finds "cheapest" routes, using the rules described above.
3. It finds alternate routes when the fastest route does not operate every day.
4. It finds routes for all times of the day, using the procedure of finding the best flights for each of 24 different desirable departure times throughout the day, as described in section 3.
5. It can handle 50 cities and 2,000 non-stop flight segments.
The program was written in FORTRAN, with FAP function subprograms used to pack and unpack data for the IBM 7090. It takes about one second to compute the “best” routes from one airport to the other nine airports in a ten airport network with 200 trip segments for a specified earliest departure time. This computation time does not include “set-up” time nor the time required to enter the data.

APPLICATIONS

The procedures described in this paper have many potential areas of application. Two such applications related to the Post Office problems will suffice as examples.

The Post Office Department prepares lists of multiple-link routes for the routing of airmail. It is anticipated that the procedures described in this paper will be used instead of a hand operation to develop these routes.

The Post Office Department schedules many mail trucks to supplement service provided by common carriers. The techniques described in this paper can be used to evaluate a proposed revision of schedules.

There are, of course, many other areas of potential use. In most of these cases, a corresponding hand operation is now being used. It is anticipated that computer procedures will be more efficient and more economical in many situations.

APPENDIX

A Computational Algorithm for Obtaining the Shortest Path From One Point to Every Other Point in a Network

Given a network of points $p_1$, $p_2$, ..., $p_n$ and lines between them, construct a distance matrix $A$, with elements $a_{ij}$ representing the length of the line between points $p_i$ and $p_j$. If no line exists between the points, let $a_{ij} = \infty$.

The algorithm also applies to the situation where the lines are directional. The value of $a_{ij}$ would be the length of the line going from $p_i$ to $p_j$. It would not be necessary that $a_{ij} = a_{ji}$.

Let $e_i$ contain the ordered sequence of points of the shortest path found so far from $p_1$ to $p_i$.

Let $b_i$ be the length of the shortest path found so far from 1 to $i$. The original values will be the direct distances, i.e., the first row of matrix $A$.

Let $d_i$ indicate if the path $e_i$ has been used in an attempt to create improved paths to other points. If $d_i = 0$, it means it has been used, otherwise $d_i = 1$.

Let $f = 1$ if any $d_i$ has been set equal to 1 since the last test of $f$, otherwise let $f = 0$.

Steps

1. Set $b_i = a_{i1}$ $i = 2, 3, \ldots, m$
2. Set $d_i = 1$ $i = 2, 3, \ldots, m$
3. Set $e_i = 1, i$ $i = 2, 3, \ldots, m$
4. Set $i = 2$
5. Set $f = 0$
6. If $d_i = 1$, go to step 7
7. If $d_i = 0$, go to step 4
8. Compute $c = b_i + a_{ij}$
9. If $c \geq b_i$, go to step 11
10. Go to step 8
11. If $c < b_i$, go to step 10
12. Go to step 13
13. Set $d_i = 0$

When the algorithm is finished, the contents of $e_i$, $i = 2, 3, \ldots, m$, will be the points through which a shortest path (more than one may exist) from $p_1$ to $p_i$ passes. The values of $b_i$, $i = 2, 3, \ldots, m$, will be the length of the shortest paths.

It should be noted that the values of $a_{ij}$ are used only in step 8. At that time, trial value of $b_i$ is known and $a_{ij}$ can be a function of that value of $b_i$. If $b_i$ is the time required to get to $i$ and $a_{ij}$ is the time from the arrival at $i$ to the arrival at $j$, then $a_{ij}$ can be determined from published schedules using $b_i$ in determining the earliest possible departure time at $i$.

REFERENCES


