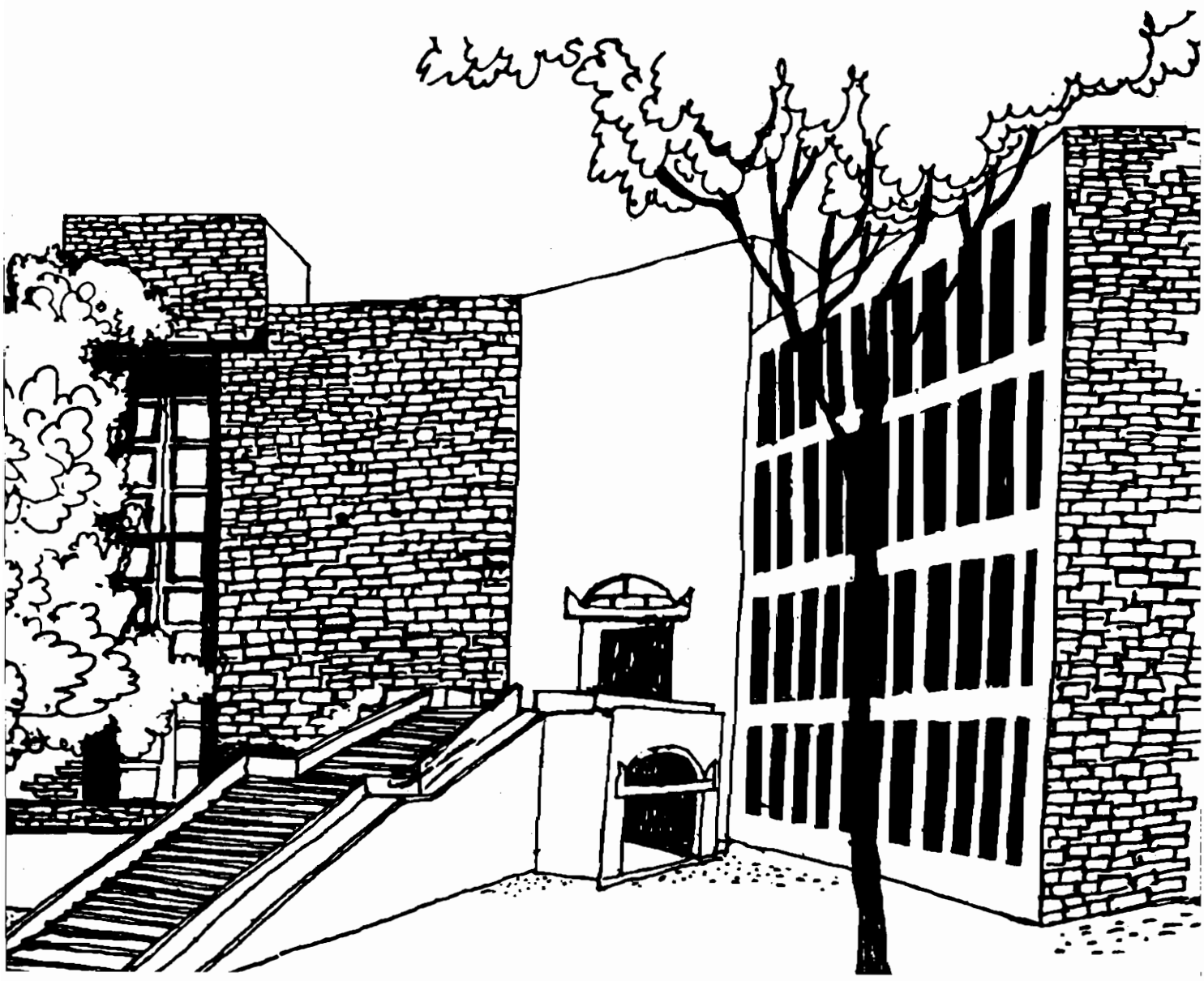
विद्याविनियोगाद्विकासः
IIM
AHMEDABAD

Working Paper



COMPUTER BASED ITINERARY PLANNING ON TRANSPORTATION SYSTEMS

BY

G. RAGHURAM
R. SHOBANA

W.P.NO. 967
SEPTEMBER 1991

The main objective of the working paper series of the IIMA is to help faculty members to test out their research findings at the pre-publication stage.

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

ABSTRACT

Customers seek to know the best way of getting from an origin to a destination on a transportation system. Knowing the best way, i.e., itinerary planning, becomes all the more important when there are varied choices depending on the desired starting day and time at the origin or a required arrival day and time at the destination. Other considerations like cost, number of connections, etc., also play a role in the planning of an itinerary. A computer based system could enhance a customer's ability to make better choices in itinerary planning. No such system exists in either the Airlines, Railways, or Road Transport Corporations in India.

This paper describes an interactive PC - based computer system developed by the authors, for itinerary planning on airlines (Indian Airlines and Vayudoot). The paper is supplemented by four annexures to facilitate further development covering system flow charts, structure of the database for maintenance and the implementation of the shortest and the k-shortest path algorithms. A users manual for the package is also provided.

ACKNOWLEDGEMENT

We acknowledge the support provided by the Computer Aided Management Fund.

CONTENTS

COMPUTER BASED ITINERARY PLANNING ON TRANSPORTATION SYSTEMS

G. Raghuram and R. Shobana

Indian Institute of Management, Ahmedabad - 380015.

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Customers seek to know the best way of getting from an origin to a destination on a transportation system. Knowing the best way, i.e., itinerary planning, becomes all the more important when there are varied choices depending on the desired starting day and time at the origin or a required arrival day and time at the destination. Other considerations like fare, number of connections, etc., also play a role in the planning of an itinerary. A computer based system could enhance a customer's ability to make better choices in itinerary planning. No such system exists in either the Airlines, Railways, or Road Transport Corporations in India.

This paper describes an interactive PC - based computer system developed by the authors, for itinerary planning. The paper focusses on an airline application. The attempt in this paper is to describe the data structure requirements and the modification to well known algorithms for application on space-time transport networks.

1.2 OBJECTIVES IN ITINERARY PLANNING

Customers usually desire itineraries that optimise on the following criteria:

1) Time 2) Fare 3) Number of Connections

A customer may have the primary criteria as fare and secondary criteria as the number of connections. On the other hand, a customer may have the primary criteria as time, with an upper bound specification on the fare.

2.1 MODEL SPECIFICATION

We describe three computer based models for itinerary planning. The model specifications have been arrived at through discussions with travel agents and executives who fly frequently.

Model A: To give the customer all feasible itineraries, in **ascending order of fare** upto a specified limit, given the origin city, the destination city, and upper bound on number of connections. The objective of this model is to give a set of reasonable options for a travel agent/traveller to choose from.

Model B: To give the customer the requested number of feasible itineraries, in **ascending order of total travel time**, given the origin city, the destination city, via city - if any, **desired time and day of departure from the origin** and upper bound on the fare. The total travel time is defined as the difference between the time of arrival at the destination and desired (not actual) time of departure from the origin. The objective of this model is to provide options to a travel agent/traveller who is primarily sensitive to time and secondarily to the fare.

Model C: To give the customer the requested number of feasible itineraries, in **ascending order of total travel time**, given the origin city, the destination city, via city - if any, **desired time and day of arrival at the destination**, and upper bound on the fare. In this model the total travel time is defined as the difference between the desired time of arrival at the destination and the actual time of departure from the origin. The objective of this model is the same as Model B.

2.2 PROCESSING

Maintenance of the Database: The one time inputs namely, time table and the fare information keep changing rather continually. Hence a user friendly database maintenance, in terms of deletions, additions or revisions is provided. The changes could be in terms of airports (nodes), routes, flight days, timings and fare.

Generation of the Itineraries: The algorithm for generation of itineraries involves generating shortest paths on space-time networks for the three models. The k-shortest path algorithm [1] is used to generate upto k distinct itineraries, in ascending order of total fare (Model A) or total travel time (Models B and C), from the origin to the destination city. All the paths must satisfy the permitted fare and connection requirements.

The algorithm is executed differently between Models B and C, since in one case the desired departure time and day at the origin is specified and in the other the desired arrival time

and day at the destination is specified. We call the first case as a forward pass and the latter as a backward pass.

2.3 OUTPUT

Model A: The itineraries are displayed in the order of increasing fare. A different fare would invariably mean a different route. For a given route, there would usually be more than one itinerary.

Model B: The itineraries are displayed in the order of increasing total travel time, which automatically translates into the order of increasing arrival time at the destination. If two itineraries have the same arrival time at the destination, then that itinerary which has lesser number of connections is displayed first. If the number of connections is also equal, then that which has a later actual departure time from the origin is given the priority.

Model C: The itineraries are displayed in the order of increasing total travel time, which automatically translates into the order of decreasing departure time from the origin. If two itineraries have the same departure time from the origin, then that itinerary which has lesser number of connections is displayed first. If the number of connections is also equal, then that which has an earlier actual arrival time at the destination is given the priority.

Format for Itinerary Display: An itinerary may consist of more than one leg depending on the number of connections in the itinerary. Each leg of an itinerary is displayed in one line, as shown in figure 1.

3. DATA STRUCTURE

This section describes the internal representation of routes and related information required by the underlying models. The representation required by models A and B being the same, is first described, followed by the changes required for Model C.

3.1 REPRESENTATION OF A SERVICE

To explain the representation of a service, we focus on airline flights. A flight may consist of more than one segment, depending on stops enroute. Consequently, separate records would be needed to reflect

a) each consecutive flight segment (direct graph) or
b) each 'direct' connection provided between city pairs (perfect graph).

The former structure, though using lesser storage space, requires additional checks to determine if two or more consecutive flight segments are part of the same flight, offering a direct service. More importantly, the Bellman's principle would be violated when we impose minimum connection times which are in general greater than halting times at

```
Available for departure from TRIVANDRUM  MON   8:00    Fare limit : Rs.5500
    FLIGHT TYPE FROM           DAY DEP  ARR  TO                FLT TIME COST
  1 IC168  300  TRIVANDRUM     MON 1455 1940 DELHI              4h45m Rs. 3379

    IC483  737  DELHI          TUE  630  745 LEH               1h15m Rs. 1898

         Fare Rs.4477 (Min fare Rs.4477)

  2 IC530  737  TRIVANDRUM     MON 1125 1225 BANGALORE          1h 0m Rs.  967

    IC404  300  BANGALORE      MON 1430 1700 DELHI              2h30m Rs. 2485

    IC483  737  DELHI          TUE  630  745 LEH               1h15m Rs. 1898

         Fare Rs.4550 (Min fare Rs.4477)



Press any key to view more

Available for departure from TRIVANDRUM  MON   8:00    Fare limit : Rs.5500
    FLIGHT TYPE FROM           DAY DEP  ARR  TO                FLT TIME COST
  3 IC168  300  TRIVANDRUM     MON 1455 1650 BOMBAY             1h55m Rs. 1918

    IC446  737  BOMBAY         MON 1830 2000 JAIPUR             1h30m Rs. 1518

    IC494  737  JAIPUR         MON 2125 2205 DELHI                40m Rs.  517

    IC483  737  DELHI          TUE  630  745 LEH               1h15m Rs. 1898

         Fare Rs.5835 (Min fare Rs.4477)

  4 IC530  737  TRIVANDRUM     MON 1125 1225 BANGALORE          1h 0m Rs.  967

    IC686  320  BANGALORE      MON 1400 1530 BOMBAY             1h30m Rs. 1367

    IC185  737  BOMBAY         MON 1700 1850 DELHI              1h50m Rs. 1779

    IC483  737  DELHI          TUE  630  745 LEH               1h15m Rs. 1898

         Fare Rs.5211 (Min fare Rs.4477)


Press any key to view more
```

Figure 1

airports (explained in section 5.4 of the paper). The latter structure overcomes the above problems through its perfect graph representation. We give below two situations of this representation.

Straight Flights: These are the usual type of flights where the origin and final destinations are different nodes. This could be from A to D with halts at B and C. This flight will be represented as being between A-B, A-C, A-D, B-C, B-D and C-D (though with the same flight number). This is shown graphically in figure 2.

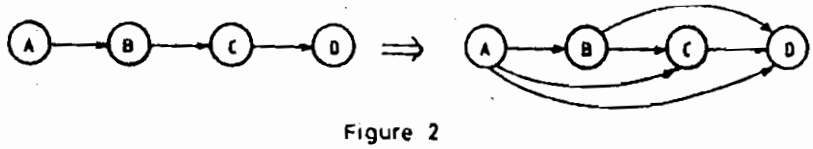


Figure 2

Direct Graph Perfect Graph

Thus, if a flight has n segments, there will be

$$n(n+1)/2$$

i.e., $O(n^2)$ separate city-pair representations.

Circular Flights: These flights originate and terminate in the same node, touching atleast two nodes in between. (There are no circular flights touching only one node in between). These flights are represented as a perfect graph except that no 'direct' connections would be offered between the city-pairs requiring travel via the origin node. For example, a flight from A to A with halts at B and C will be represented as being between A-B, A-C, B-C, B-A and C-A (no connection from C to B). This is shown graphically in figure 3.

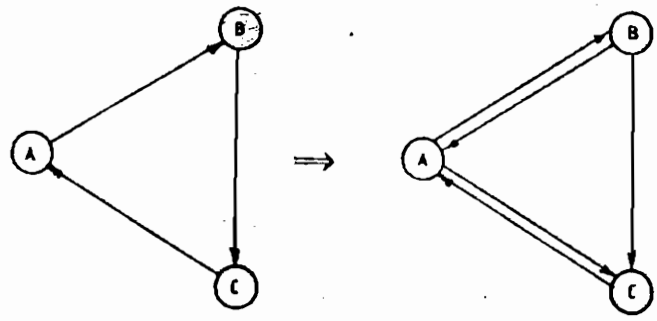


Figure 3

Direct Graph Perfect Graph

Thus, if a flight has n segments, there will be

$$2(n-1) + (n-2)(n-1)/2$$

i.e., $O(n^2)$ separate city-pair representations. The first term represents the direct connections to and from the origin and the second term represents the direct connections between the halts.

A given flight may not have the same set of timings on all the days of the week that it operates. There are even situations where the route of the flight is different on the different days that it operates. To uniquely identify a flight segment and the timings for each of the city-pair representations, a record is required with flight number and days of operation on which the timings are identical. Thus, for each city-pair of a flight, there would be as many records as the different set of timings operated. Each such record is called a **'service'**. The entire network is stored as a file of 'services'.

3.2 FILE ORGANISATION

Models A and B: In this file of services, all services

originating from the same node called link records are grouped together under a header record. This header record contains the **number** of the origin node followed by the letter 'H'. The data elements of a link record start with the **number** of the destination node, followed by the letter 'L' and the service details. For example, the representation of services **1-2, 1-8, and 1-9** along with the interpretation is given below :

```
1 H
2 L  D1  A1  F1  T1 1 1 0 1 1 1 1
8 L  D2  A2  F2  T2 1 0 1 0 0 0 1
9 L  D3  A3  F3  T3 1 1 1 1 1 1 1
```

D1, D2, D3 - departure time of services 1-2, 1-8, and 1-9 from node 1.

A1, A2, A3 - arrival time of services 1-2, 1-8, and 1-9 at nodes 2, 8 and 9 respectively.

F1, F2, F3 - flight numbers of the services 1-2, 1-8, and 1-9.

T1, T2, T3 - type of aircraft used in services 1-2, 1-8, and 1-9 respectively.

The days of the week that a service is available/not available is represented by a seven element string of 1's and 0's in the sequence Monday-Sunday. 1 indicates availability and 0 indicates non-availability of the service on that day. A node number can appear in a header record only once and all the services starting from this node should be listed prior to the next header record. The file organisation described above is suitable only for Models A and B where the shortest path is constructed from the origin towards the destination.

Model C: In this model the shortest path is to be constructed in the backward direction i.e., from the destination towards the origin. In this file organisation the services having common destination nodes are grouped together as opposed to the grouping of services having common origin nodes in Models A and B. For example, services **1-4, 6-4, 8-4** would be represented as below:

```
4 H
1 L  D1  A1  F1  T1 1 1 0 1 1 1 1
6 L  D2  A2  F2  T2 1 0 1 0 0 0 1
8 L  D3  A3  F3  T3 1 1 1 1 1 1 1
```

Though this file organisation implies that a separate data file has to be maintained for Model C, it is essential so that the backward pass can have an execution time equal to the forward pass.

4. DATABASE MAINTENANCE

The database consists of the following three files:

CITIES - Names of all cities and their numbers

(assigned by the system) in the network.

TIME TABLE - Contains details about flight timings, aircraft type and days of operation of flight services in the system.

FARE TABLE - Fare for every direct service offered.

Addition, revision and deletion of cities, flight details, a fare records are permitted. These three files have to l compatible with each other for proper itinerary generation, fc which certain interfile checks are applied on them afte updation. For example, there should be a fare record for ever city pair served by a flight and vice versa. Similarly, ther should be at least one flight for every city in the data base and vice versa.

For maintenance purposes the flight services are represented as a direct graph. After updation of the database, the direct graph representation is converted to the perfect graph representation which is used by the itinerary generation module. Also the fare information is used to compute the least possible fare between every pair of cities in the network, using the shortest path algorithm. This information is required for fare based itinerary genration (model A) and for checking with the upper bound on fare specified by the user.

5. ALGORITHMS

5.1 MODEL A

Generation of Routes: Routes are generated in increasing order of fare. The maximum number of routes to be generated is user-specified. The routes are also subject to upper bounds on fare and number of connections. The route generation is done using the k-shortest path algorithm on a network whose

nodes represent airports
arcs represent existence of 'services'
arc costs represent the given fares.

Consider a 5 airport system with straight flights A-C-E and E-C-A, B-C and C-B and a circular flight A-B-D-A. The services would then be represented in a network as shown in figure 4. Each edge has a return service except between D and B, which are the intermediate nodes of the circular flight. All costs Cij are equal to Cji, if they exist.

Generation of Itineraries: Based on each of the routes generated in the above step, itineraries are constructed. The number of such itineraries would be combinatorially high. These are pruned down to 'reasonable' itineraries using a three step procedure described below. The procedure is illustrated by the example of a sample of trips on route A-B-C-D, shown in figure 5.

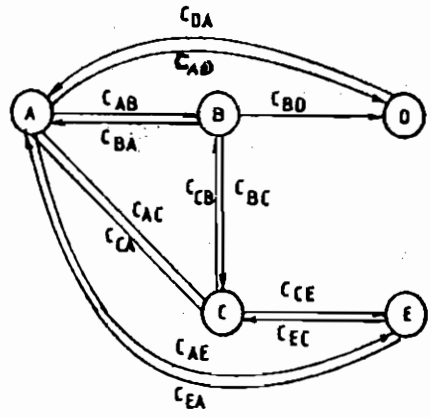


Figure 4

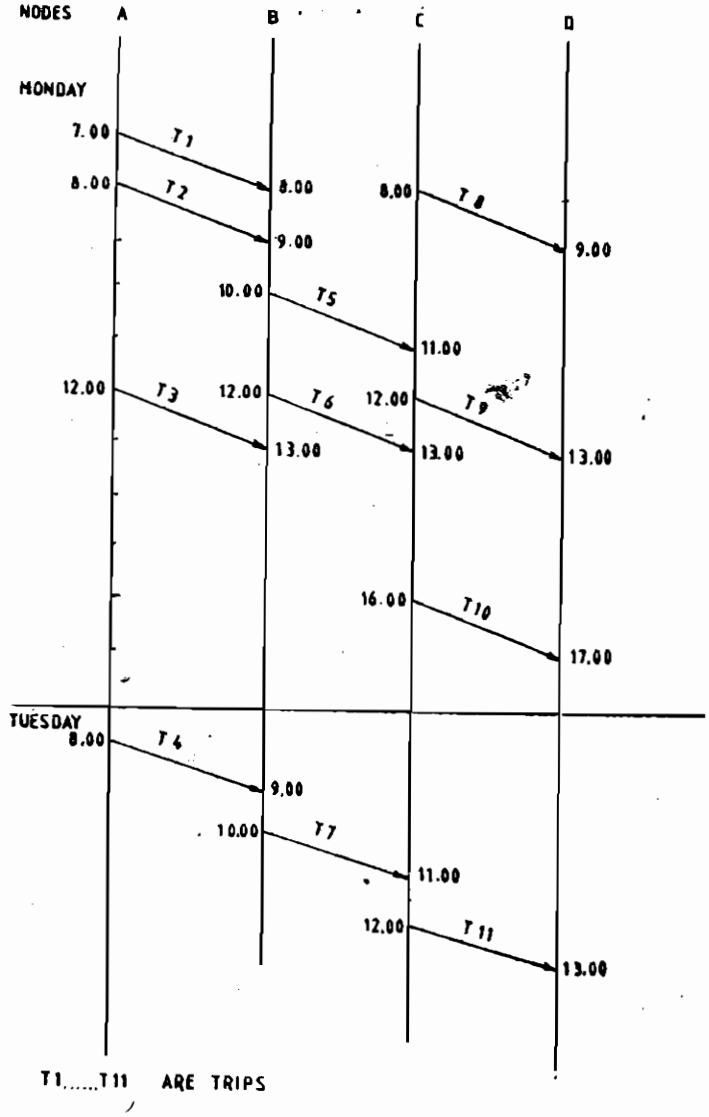


Figure 5

1. Minimum Travel Time Itinerary Construction: Let R = A-B-C-D be the physical route upon which itineraries are to be drawn. The edge which has the least number of services operating on it

is identified. In this example the minimal edge is B-C. For every trip T available on B-C, an itinerary is constructed on R by proceeding in the forward and backward directions from B-C in a greedy manner. This would normally result in an itinerary whose total travel time is the least possible, given the trip T. (The minimum travel time may not occur if there are say, two trips between a city pair such that one trip starts earlier but reaches later than the other. Such a schedule is a rare occurence in a flight time table.) Thus the following three itineraries get generated:

T2 - T5 - T9
T2 - T6 - T10
T4 - T7 - T11

(T1 - T5 - T9 would not be an itinerary since the total travel time is greater then that for T2 - T5 - T9).

2. Comparison of Itineraries for Identical Timings: The itineraries generated thus are compared with each other to identify identical itineraries. Two itineraries are identical, if the set of trips used have the same timings, but operate on different days of the week. In this example the itineraries T2 - T5 - T9 & T4 - T7 - T11 are identical. In case of such itineraries, they are merged as one itinerary, and the available days of the itinerary are updated as Monday and Tuesday.

3. Comparison Of Itineraries for Dominance: Each itinerary is compared against other itineraries for 'dominance' as follows: If two itineraries start from the origin at the same time and day, then the itinerary which arrives at the destination earlier is the dominant one. Similarly, if two itineraries arrive at the destination at the same time and day, then the itinerary which leaves the origin later is the dominant one. The dominant itinerary is considered while the other is not. In the above example T2 - T6 - T10 is dropped since it is dominated by T2 - T5 - T9.

At the end of the three step procedure no two itineraries will have even one trip in common.

5.2 MODEL B

Itineraries are generated using the k-shortest path algorithm, in increasing order of arrival time at the destination. At nodes where connections have to be made between flights, a minimum connection time is required. The shortest path as required at each stage of the k-shortest path algorithm is generated using Dijkstra's algorithm [2], on a network

- whose nodes represent arrival and departure events at airports,
- arcs represent services and inter-event waits between successive events,

arc costs represent flying time plus connection time service arcs and waiting time on inter-event arcs.

Consider a timetable between three cities A, B, and C:

	MON	THU				MON	THU
Flt #	111	111				222	222
	800	800	dep	A	arr	1500	1400
	900	900	arr	B	dep	1400	
	930	930	dep	B	arr	1330	
	1100	1100	arr	C	dep	1200	1200

With a required connection time of 75 minutes added to the arrival times and the perfect graph file organisation, the time table can be recast as follows. (The letters below each time represent the nodes.)

FLT #	MON 111	MON 111	MON 111	THU 111	THU 111	THU 111
A	8.00 (AD1)	8.00 (AD1)		8.00 (AD2)	8.00 (AD2)	
B	10.15 (BA1)		9.30 (BD1)	10.15 (BA3)		9.30 (BD3)
C		12.15 (CA1)	12.15 (CA1)		12.15 (CA2)	12.15 (CA2)

FLT #	MON 222	MON 222	MON 222	THU 222
A		16.15 (AA1)	16.15 (AA1)	15.15 (AA1)
B	14.45 (BA2)		14.00 (BD2)	
C	12.00 (CD1)	12.00 (CD1)		12.00 (CD2)

ADi represents the i th departure from A and BAi represents the i th arrival at B. The graphical representation is shown in figure 6. If the desired time of departure is 10 am on Monday from A, then a start node for the algorithm is inserted between nodes AD1 and AA1, 2 hours after AD1.

Dijkstra's algorithm is implemented in such a way that the resulting shortest path has a total path fare less than or equal to p times the cheapest fare possible between the given origin and destination, where p is user-specified. The example of the fare structure between nodes A, B, C and D, as shown in figure 7 illustrates this.

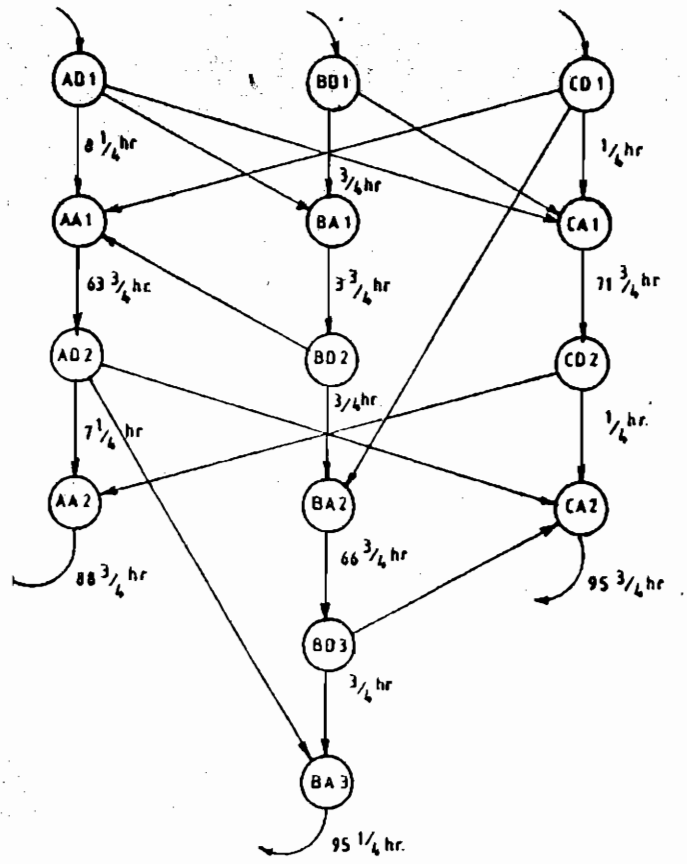


Figure 6

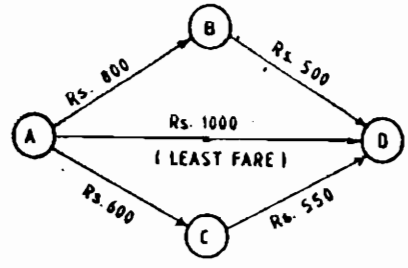


Figure 7

If p = 1.2 for a traveller from A to D, then while executing the algorithm, the route via B will not be considered. Among the paths A-D and A-C-D which satisfy the upper limit on total fare (Rs.1200), the one with the least total travel time, depending on the desired time of departure from the origin, will be selected as the shortest time path from A to D.

The way this algorithm is implemented in this package is described in Annexure 3.

5.3 MODEL C

Itineraries are generated in a manner similar to Model B, except that the backward pass of the shortest path algorithm is used. This is applied on a network representation suitably modified for this procedure, described in section 3.2 under Model C.

5.4 DIRECT GRAPH REPRESENTATION - VIOLATION OF BELLMAN'S PRINCIPLE

Bellman's principle as applied to the shortest path problem states that if the shortest path P from I to K is via J, then the segment of path P between I and J is the shortest path from I to J. The example shown in figure 8 illustrates how the direct graph representation violates this principle.

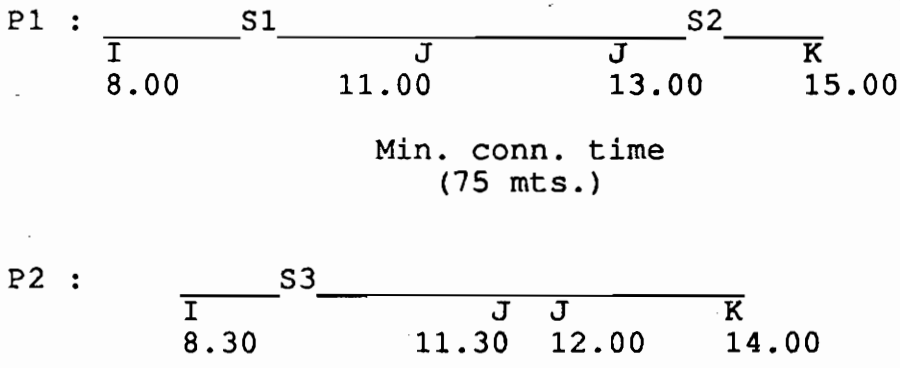


Figure 8

For a person ready to travel from I at 8.00 am, path P1 is the shortest path from I to J but path P2 is the shortest path from I to K.

In the direct graph representation, service S3 from I to K is representated as two segments I-J and J-K only. A shortest path algorithm (which follows Bellman's principle) having recognised P1 as the shortest path from I to J would continue its search to K from J along P1 only, since the J-K segment of S3 would not be available to any arrival at J along P1 because of the connection time. This would result in P1 as the shortest path from I to K, which is obviously incorrect.

In the perfect graph representation, service S3 is represented as three segments I-J, J-K and I-K. The shortest path algorithm would now choose P2 as the shortest path from I to K since segment I-K is available.

6. EXTENSIONS

6.1 CONCEPTUAL DESIGN FOR OTHER MODES

Rail System[1]*: The main difference in a rail-based system from airlines is

a) number of nodes and services are large
b) fare is a pure function of distance of travel, independant of connections.
c) concept of a direct service is 'fuzzier', with through coaches and through reservation quotas.
d) since trains travel over days, the time of arrival should also indicate the number of days after which it occurs.

While the objectives, inputs and processing would be similar to what is already described, (except for the larger size and different fare structure) the output could be reoriented substantially.

All the itineraries generated can be displayed one by one in the order specified below. Each itinerary would contain information about the route, the timings of the services along with the service number, the total distance to be travelled along this route and the corresponding fare chargeable. The itineraries would be listed in the following order:

1) Direct services between the origin and destination
2) Through coach services available between the two cities
3) Services on which through quota is available
4) If an itinerary has a break in journey, then that itinerary which has the first leg as the longest or the first leg as a night journey is given priority over the others.
5) If there exist two itineraries having equal priority on all the above conditions, then which reaches the destination earlier is given the higher priority.

Bus System: The itinerary planning system for bus systems would be relatively easier since 'connections' from one service to another are not as important as knowing the range of options on direct services. This is primarily because the travelling public attribute a 'high' cost to connections. Further, the frequency of direct services provided is usually high.

---

1. * A similar system for a rail network is available for the Netherland Railways. This is supplied by the railway along with the time table, at an extra cost. The Centre for Railway Information Systems, New Delhi is currently developing a system for itinerary planning on the Indian Railways, to be extended to include air services also. This uses a heuristic search procedure, using artificial intelligence concepts [3].

6.2 OTHER EXTENSIONS

Since this is an interactive system, the response time will have to be very quick. In rail and road based itinerary planning systems, the number of boarding/alighting points is so high that if enquiries are permitted for all of them, the response time will also increase substantially. One way of overcoming this disadvantage is by identifying a limited number of major nodes and enquiries from/to a non-major node being linked to the closest major node.

Customers may also want to know about the facilities available on each leg of the itinerary. General information about amenities at nodes, like tourist information, hotels and related facilities may also be desired. These can be easily incorporated in the model.

Itinerary planning on international airlines can also be done based on these models, with all local time references converted to a standard time like GMT.

Such systems would be useful for travellers, institutions with a high level of executive travel, travel agents, airlines, railways, bus corporations and front offices of hotels etc. Individual travellers would instead seek updated versions with each new time table.

References

[1] Horowitz, E., and S. Sahni, "Fundamentals of Data Structures in Pascal", Computer Science Press Inc., 1984.

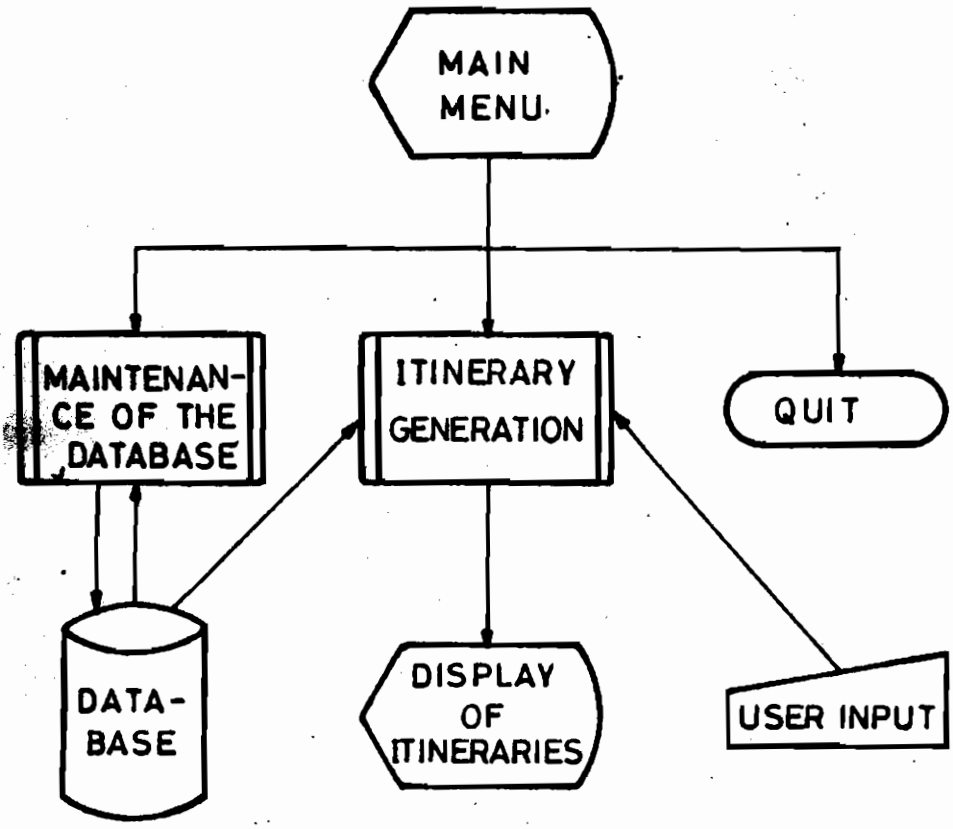
[2] Larson, R. C., and A. R. Odoni, "Urban Operations Research", Prentice-Hall Inc., 1981.

[3] Dhawan, V., V.S.Rajput and R.G.S.Asthana, "Intelligent Rail-Air Travel Planner", Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications, 1991, Miami Beach, Florida, U.S.A.

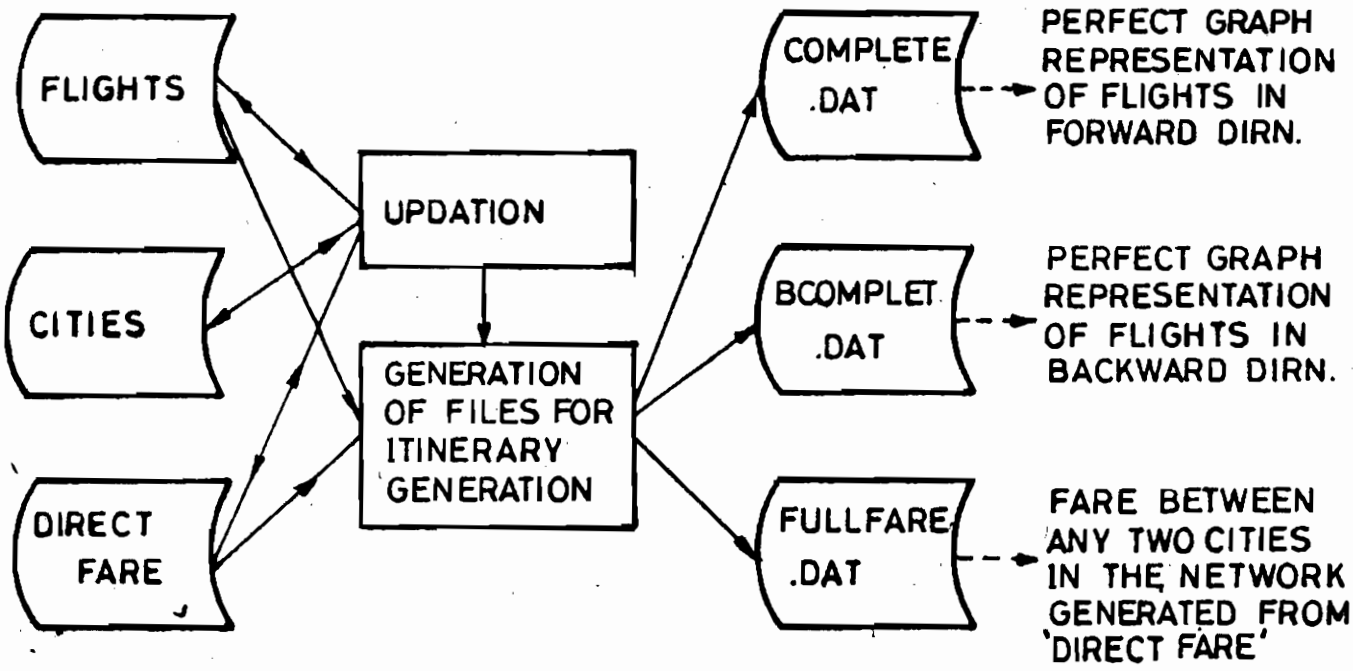
ANNEXURE 1

SYSTEM FLOWCHARTS

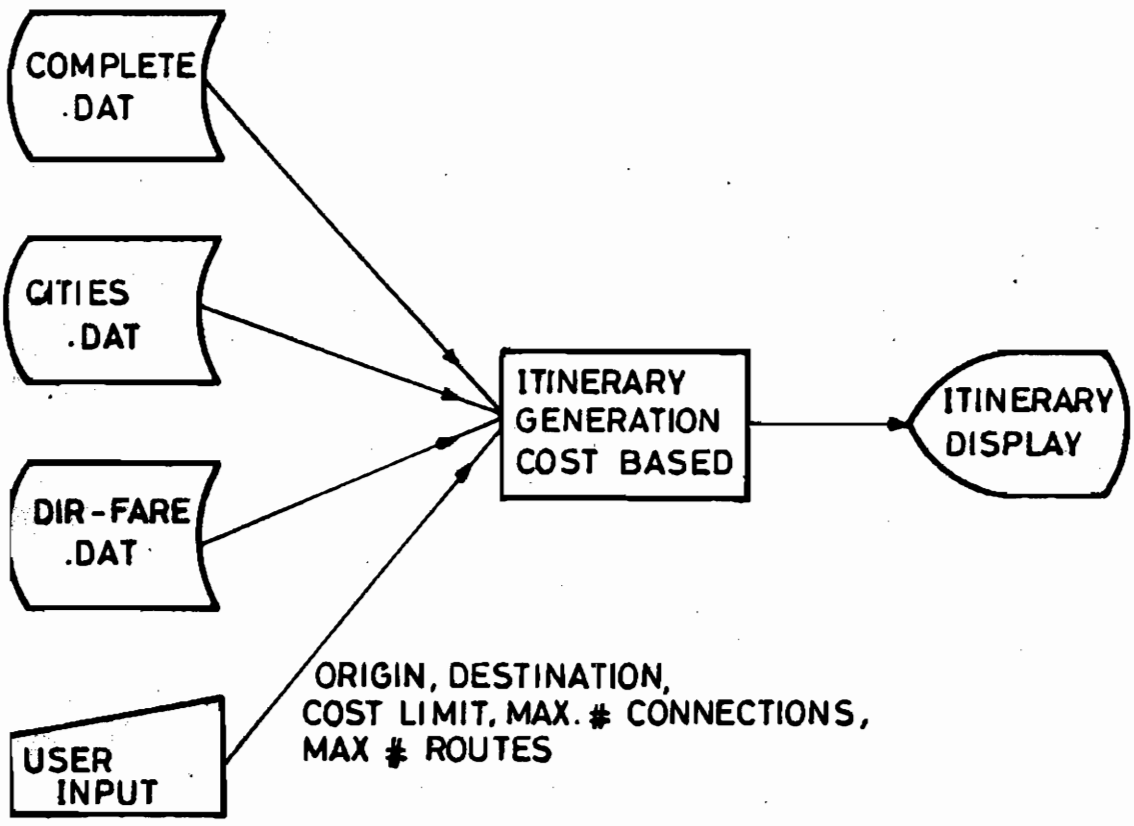
1. SYSTEM OVERVIEW



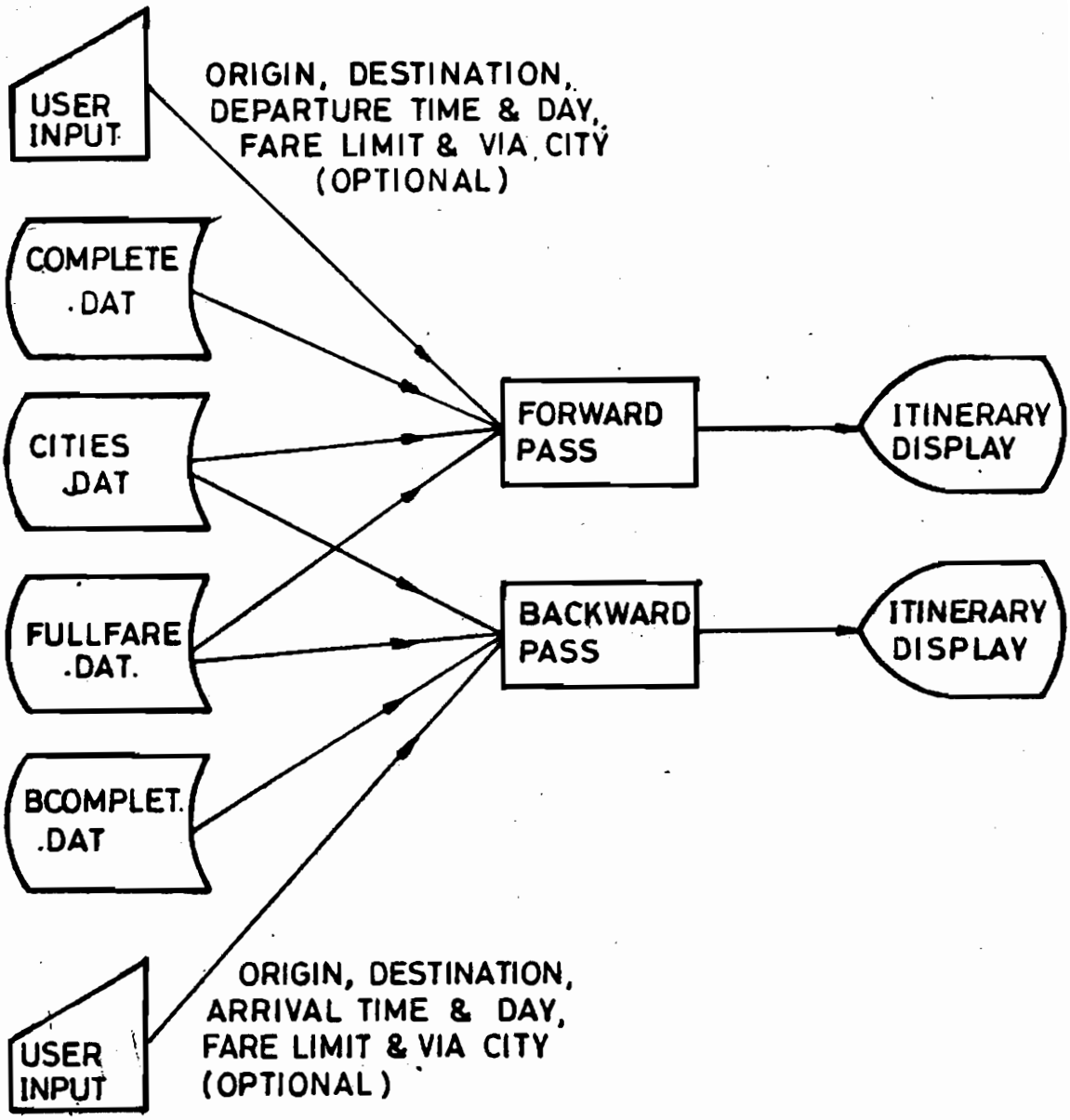
2. MAINTENANCE OF THE DATABASE



3. ITINERARY GENERATION (COST-BASED) (MODEL A)



4. ITINERARY GENERATION (TIME-BASED) (MODELS B & C)



ANNEXURE 2

DATABASE MAINTENANCE

The database for this system consists of six files. The following three files are the main files. Other three files are generated from these files.

FLIGHT DETAILS (TABLE.DAT)

CITIES (CITIES.DAT)

FARE DETAILS (DIR-FARE.DAT)

The maintenance of these three files can be done either

a) by creating an index list in memory and update the file on the disk through the index list

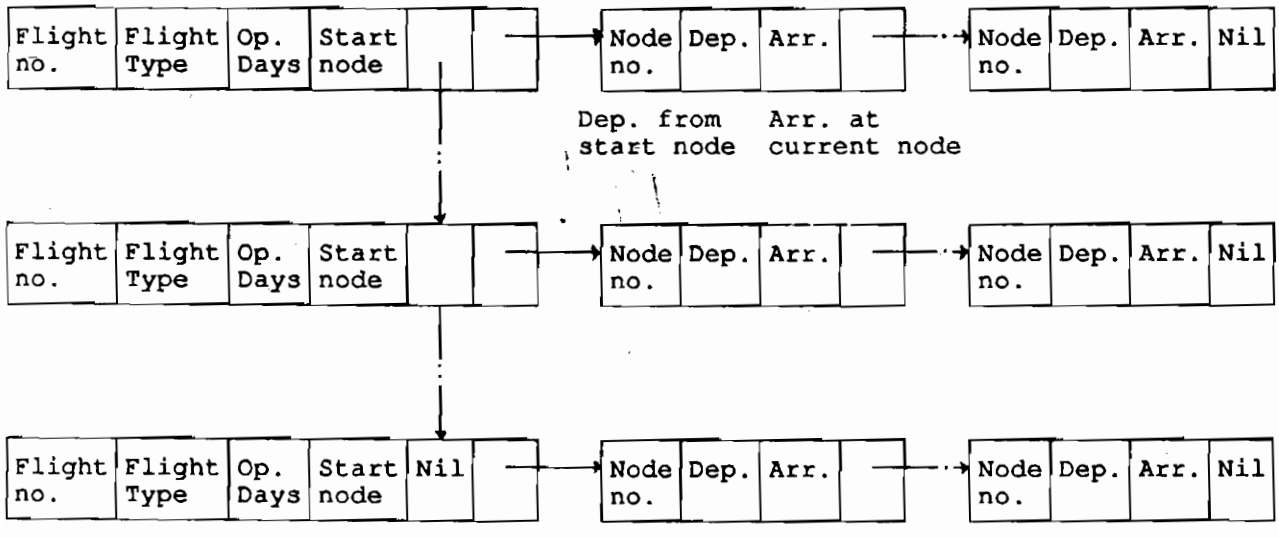
(or)

b) by loading the entire file onto memory, update the memory-copy and then write back to the disk file.

Method (b) will be faster than (a) since retrieving data from primary memory is faster than retrieving the same from hard disk. But method (b) is feasible only if the amount of memory needed to load the entire file is available. In this package, method (b) has been adopted since the memory required is very small.

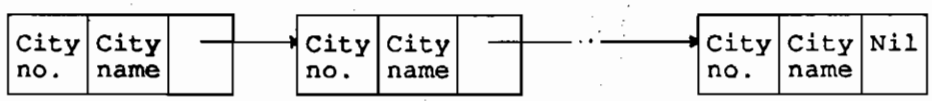
The three main files are maintained as linked lists in the memory. The structure of each of those lists is given below.

FLIGHT DETAILS

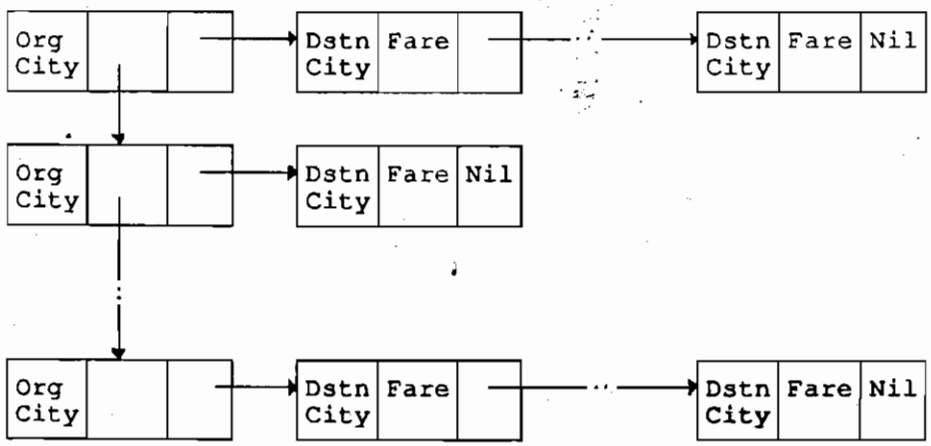


This list is maintained in ascending order of flight number.

CITIES



FARE DETAILS



ANNEXURE 3

MODIFIED SHORTEST PATH ALGORITHM

Notations:

Specified by the user:

ORG	-	origin node
DSTN	-	destination node
P	-	ratio of actual fare to the least possible fare

Pre-calculated from the Database:

Fare [U , V]	-	cheapest fare from U to V (This is derived for all node pairs by executing Dijkstra's algorithm on the fare table provided for the various service legs).

Determined iteratively during the algorithm:

Total Travel Time [V]	-	shortest time taken to travel from ORG to node V .
Predecessor [V]	-	the node which is the immediate predecessor of V along the shortest path from ORG to V.
Path fare [V]	-	fare along the shortest path from ORG to V.
SET	-	set of vertices to which the shortest path from ORG have already been determined.
Delay (C, S)	-	connection time for service S at C, given the shortest time path to C from ORG.

Initial conditions:

Total Travel Time [V]	=	infinity (or a very large no.) for any node V other than ORG.
Total Travel Time [ORG]	=	0 .
Path fare [V]	=	0 for all nodes V.
SET	=	{ }

Algorithm:

Step I:

Among those nodes not in SET pick up that node which has the minimum Total Travel Time. Call it C, the current node. Add C to SET.

Step II:

Selection of W (node):

a) adjacent to the current node C (i.e., there is atleast one service from C to W)
b) not a member of SET

Selection of S (among services from C to W):

a) If service S and the immediately preceding service on the shortest path from ORG to C are the same, then S has to satisfy the following inequality,

Path fare [M] + fare [M, W] + fare [W, DSTN]
< = P * fare [ORG, DSTN]

where M is the starting node of the service S. (This is because on a flight a-b-c, fare [a, c] < fare [a, b] + fare [b, c])

b) If S is different from the preceding service on the path from ORG to C, then S has to satisfy the inequality,

Path fare [C] + fare [C, W] + fare [W, DSTN]
< = P * fare [ORG, DSTN]

If S satisfies (a) or (b), then the Total Travel Time [W] is updated as below.

Total Travel Time [W] = min (Total Travel Time [W], Total Travel Time [C] + delay at C to connect to S + travel time from C to W on S)

If Total Travel Time [W] is revised through C, then

Predecessor [W] = C
Path fare [W] = Path fare [M] + fare [M, W] if S satisfies (a) or
Path fare [C] + fare [C, W] otherwise

Step II is repeatedly applied for all services available from C. Steps I and II are repeatedly applied until the current node is DSTN itself.

ANNEXURE 4

K-SHORTEST PATH ALGORITHM

Notations:

ORG	-	origin node
DSTN	-	destination node
$e(V_i, V_j)$	-	represents an edge where V_i, V_j are the starting and ending nodes of the edge
K	-	required no. of paths in the increasing order of time starting with the shortest time path
S	-	shortest time path from ORG to DSTN
C_s	-	conditions associated with the generation of path S i.e., S is the shortest time path from ORG to DSTN satisfying C_s. Conditions are the set of edges to be excluded or included while generating the shortest path
Q	-	a dynamic set of time paths from ORG to DSTN from which the Kth shortest path is selected
P	-	parent path - the path in Q with the least total travel time
C_P	-	conditions associated with the generation of the parent path

Initial conditions:

Q = { S } where S is the shortest time path from ORG to DSTN

C_s = ϕ

Algorithm:

Step I:

Select the current parent path P from Q.
Initially, $C_p = C_s = \phi$

Step II:

Let P be $e_1 \ldots e_k \ldots e_m$ where e_i is an edge. From P at the most 'm' child paths ($S_1 \ldots S_k \ldots S_m$) can be generated by excluding each of the m edges.

Let edge e_k be (V_i, V_j). A child path S_k is generated as below:

a) Generate a shortest path T from node V_i to DSTN, excluding edge e_k and satisfying C_p.

b) $S_k = e_1 \ldots e_{k-1} + T$

and $C_k = C_P$ + Include edges $e_1 \ldots e_{k-1}$ + Exclude edge e_k

Thus from P, paths $S_1 \ldots S_m$ are generated using the above procedure. All the new paths are added to Q.

Step III:

Print the parent path P and delete it from Q. At any point of time Q contains only those paths which have not been selected as parent path so far.

Steps I, II and III are repeated K times or until Q is empty.

USERS MANUAL

FOR

COMPUTER BASED ITINERARY PLANNING ON TRANSPORTATION SYSTEMS

1. INTRODUCTION

ITIPLAN is an interactive PC - based software for itinerary planning on a transportation system given the timetable and fare. This software package generates all possible itineraries satisfying user specified cost,time and connection requirements.

This users manual takes the reader through the specific application of the software to the domestic air transportation network (Indian Airlines and Vayudoot).

This software is aimed at travel agents and users of Indian Airlines and Vayudoot. It also provides facilities for easy updation of time table and fare information.

HARDWARE REQUIREMENTS

PC/XT / PC/AT with atleast 400k of usable RAM is required. Colour monitor is preferable.

For faster execution, install this software in your hard disk. This package will require approximately 360K of disk space. It may need more space as the timetable grows.

SOFTWARE REQUIREMENTS

MS-DOS operating system version 3.0 or above.

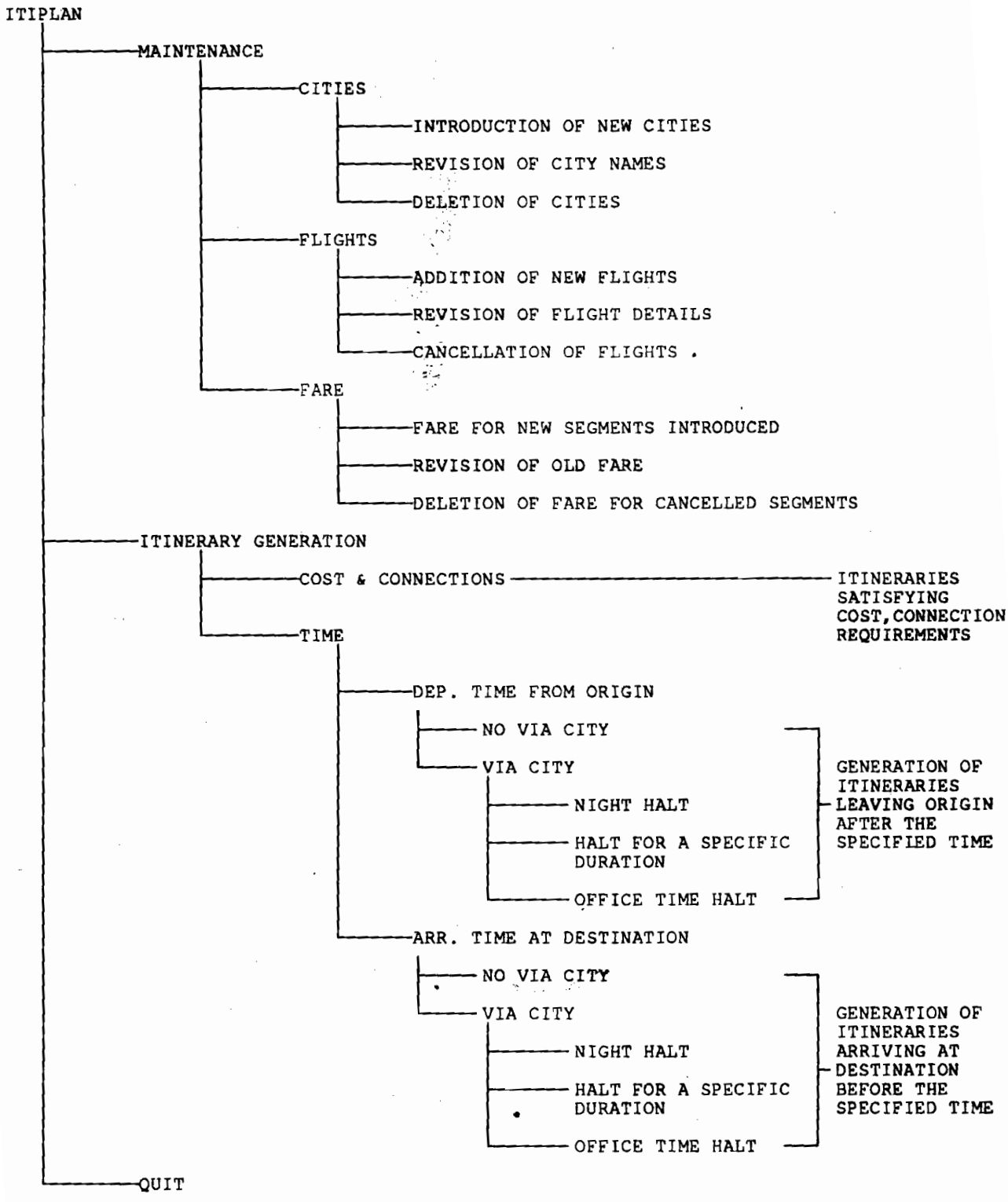
THE PACKAGE

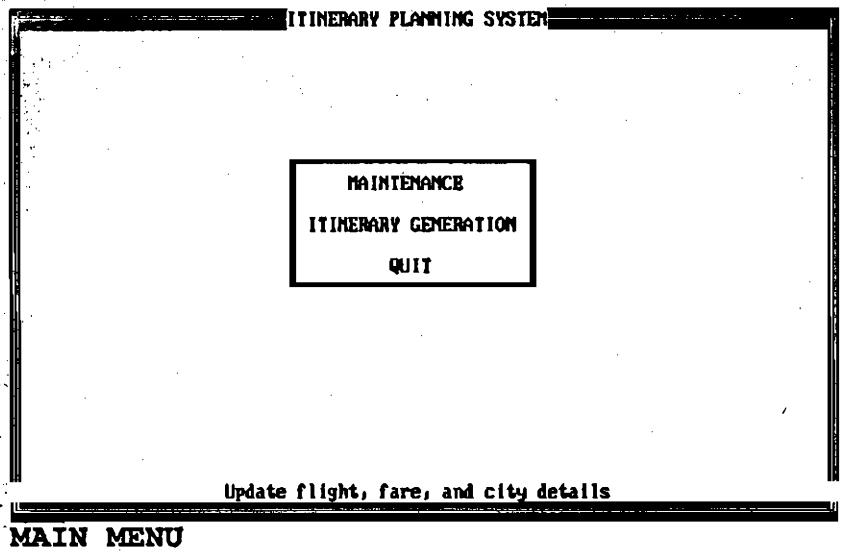
Executable file	-	**ITIPLAN.EXE**
Data files	-	**TIMETABL.DAT**
		FARETABL.DAT
		CITIES.DAT
		COMPLETE.DAT
		BCOMPLET.DAT
		DIR_FARE.DAT

None of these files are readable except CITIES.DAT. Any attempt to edit these files will make them unusable. The file CITIES.DAT though in readable form should not be edited.

An overview of the user features of the system is presented as a menutree in the following page.

MENUTREE


ITIPLAN
MAINTENANCE
CITIES
INTRODUCTION OF NEW CITIES
REVISION OF CITY NAMES
DELETION OF CITIES
FLIGHTS
ADDITION OF NEW FLIGHTS
REVISION OF FLIGHT DETAILS
CANCELLATION OF FLIGHTS .
FARE
FARE FOR NEW SEGMENTS INTRODUCED
REVISION OF OLD FARE
DELETION OF FARE FOR CANCELLED SEGMENTS
ITINERARY GENERATION
COST & CONNECTIONS
ITINERARIES SATISFYING COST,CONNECTION REQUIREMENTS
TIME
DEP. TIME FROM ORIGIN
NO VIA CITY
VIA CITY
NIGHT HALT
HALT FOR A SPECIFIC DURATION
OFFICE TIME HALT
GENERATION OF ITINERARIES LEAVING ORIGIN AFTER THE SPECIFIED TIME
ARR. TIME AT DESTINATION
NO VIA CITY
VIA CITY
NIGHT HALT
HALT FOR A SPECIFIC DURATION
OFFICE TIME HALT
GENERATION OF ITINERARIES ARRIVING AT DESTINATION BEFORE THE SPECIFIED TIME
QUIT



MAIN MENU

This menu contains the following three options

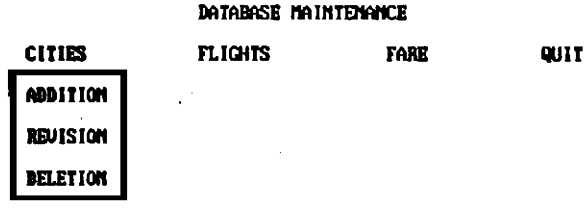
a) MAINTENANCE - For updating the Flight details, Cities, and Fare information.

b) ITINERARY GENERATION- For generating itineraries satisfying user specifications.

c) QUIT - Quit from the system.

The following sections will describe in detail all the options available.

2. MAINTENANCE



This module presents the following options:

CITIES FLIGHTS FARE QUIT

CITIES - Add new cities, Modify city names, Delete old cities.

FLIGHTS - Add new flights, Revise existing flight details, Delete/Cancel old flights.

FARE - Add new fare records, Revise old fare details, Delete old fare records.

QUIT - Quit the maintenance module. <ESC> key can also be used to quit from this menu.

The three files consisting of flights, cities and fare records are related to each other through consistency checks. These checks are performed during addition, deletion and at the end while quitting the maintenance module.

2.1 CITIES

This provides a vertical menu with the following options.

ADDITION REVISION DELETION

ADDITION

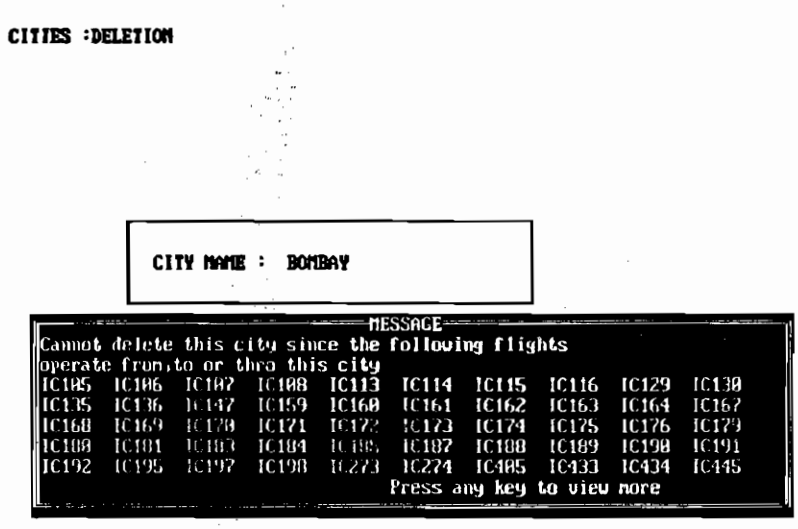
The new city name should not contain more than 15 characters. Only alphabets are accepted. The new city name is inserted at the appropiate place (alphabetical order) in the city window and displayed immediately. If the new city name is already present in the database, then it is not added to the database but the system displays a message informing the user that it is already present.

REVISION

This option is for revising the name of an existing city in the database. First a city has to be selected from the city window and then the new name has to be input. The new name should not contain more than 15 characters and should not be same as any other city in the database.

DELETION

From the city window the user has to select the city to be deleted. After selection the system checks whether there exists one or more flights from/to this city. If so, then the city is not deleted from the database. As an additional information to the user, all flights operating through the selected city are displayed in a window.



Before deleting a city, all flights which have this city as part of their route, must be revised so that there is no flight from/to that city.

2.2 FLIGHTS

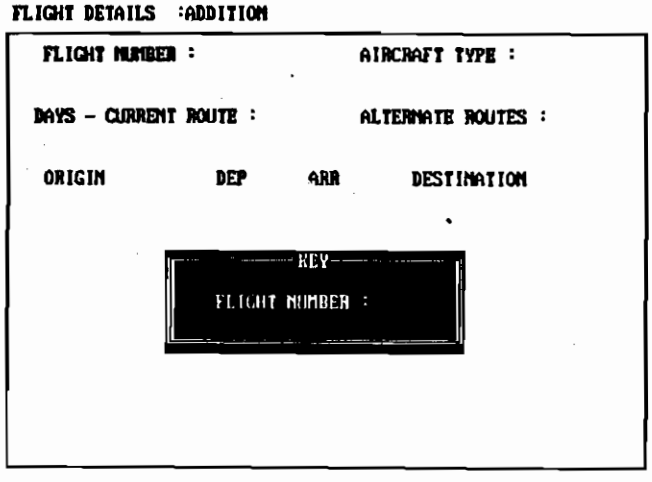
This provides a vertical menu with the following options.

ADDITION REVISION DELETION

Press <ESC> to get back to the previous menu.

ADDITION

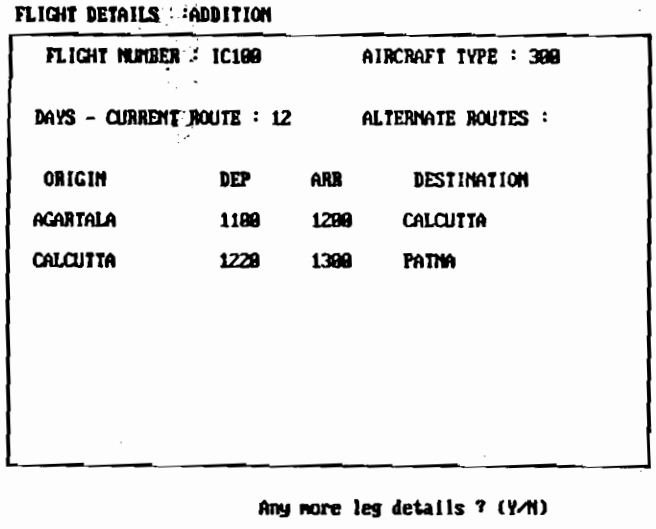
On selection of this option, the flight number of the flight to be added is requested.



FLIGHT NO. : A string of 6 characters is accepted. The first two charcters of the flight no. must be its IATA code. Rest 4 characters can comprise of alphabets and digits only. If the user attempts to enter a seventh character as part of

the key the system generates a beep sound. The seventh character is not accepted. The system waits until the user presses <ENTER>.

It checks whether a flight with the specified flight number is already present in the database. If present then details of that flight is displayed else a fresh screen is displayed to allow the user to add the new flight's details. In the display of flight's details, if the flight operates on different time schedules/ routes on different days of the week, then the user can view all of them by using the <PgUp> and <PgDn> keys.



A screen display of a newly added flight is given above. For every field, the kind of data to be input and the validations applied on the data are described below.

AIRCRAFT TYPE : Any string of 3 characters is accepted as the type of aircraft. This is not part of the key.

OPERATING DAYS : Press D for Daily. If the flight is not operating daily then use digits 1-7 to enter the days of operation.
(1-MON 2-TUE 3-WED 4-THU 5-FRI 6-SAT 7-SUN)

LEG DETAILS : A leg is displayed as below

City name(1)	Departure from City1	Arrival at City2	City name(2)
(eg.) AHMEDABAD	600	650	BOMBAY

The user has to input the origin city, the timings and the destination city to complete a leg.

CITY SELECTION : A window with a list of all cities in the database is displayed. The city names are displayed in the alphabetical order. In one screen 19 city names are

displayed. Use <PgUp> and <PgDn> keys to view more. <HOME> key positions the highlighted cursor on the top of the list and the <END> key positions it on the last city in the list. Use and to move within the same screen. Press <ENTER> to select the city on which the highlighted bar is positioned currently.

TIME INPUT : 24 hour clock. The format is HHMM where HH indicate the hours and MM indicate the minutes. Validations are applied based on the 24 hour clock.

Eg. 1210 & 1559 are valid

1260 & 2400 are invalid.

0000 is a valid time input.

The arrival time has to be atleast a minute greater than the departure time.

ORIGIN-DESTINATION : While entering a leg detail user cannot select the destination city same as the origin city. Until a city other than the origin is selected the user cannot proceed further.

END OF INPUT : After the completion of each leg detail, the system asks the following question

'Any more leg details ? (Y/N)'.

It does not accept any character other than 'Y' or 'N'.

MULTILPLE LEGS : If there is more than one leg in the flight's route,the destination city of the preceding leg is repeated as the origin city of the succeding leg. Also the departure time of the succeding leg has to be greater than the arrival time of the immediately preceding leg. It is possible to enter **circular routes.**

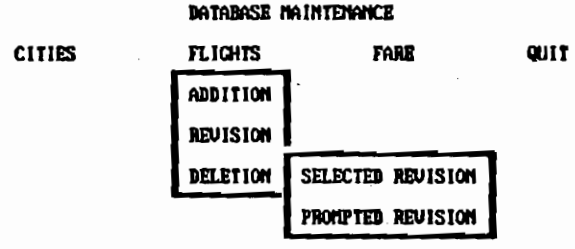
ALTERNATE ROUTES : A flight number can have more than one route. On different days of the week it can have either different timings or it can cover different cities. The route which is being displayed on the screen currently, is called the current route and all other routes of the same flight number are called alternate routes.

ADDING ALTERNATE ROUTES : If the flight is not operating daily then press 'A' to add another route. The operating days of this new route should be only those days on which the flight is not operating on any of the other routes.

EXIT : Press 'N' for next new flight to add or <ESC> to get back to the flights menu.

REVISION

On selection of this option, another menu is displayed with the options, SELECTED REVISION & PROMPTED REVISION. These two options provide the user with two different modes for revision of flight details.

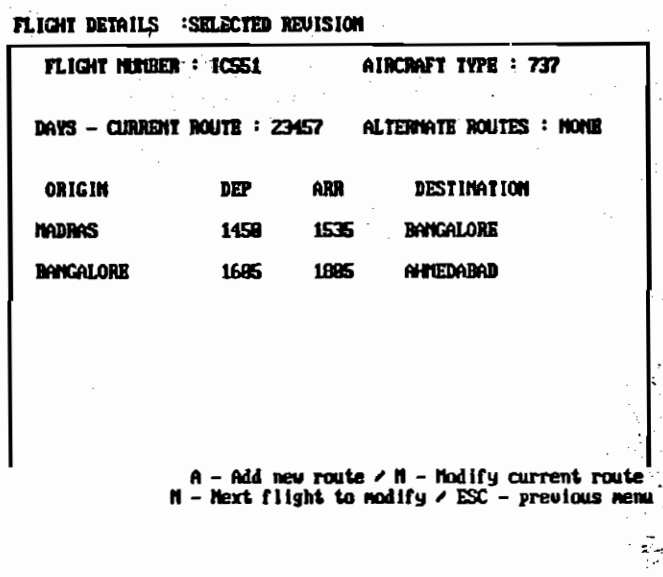


Revise details of selected flights
ESC to previous menu

SELECTED REVISION : The system prompts the user to enter the flight number (key) of every record that is to be revised. This mode is useful when there are few records to be revised and the user knows the flight number of all those records.

PROMPTED REVISION : The system requests a starting flight number from where the revision should start. It presents the record with the starting flight number and all subsequent records for revision, one by one. The user is allowed to end the revision at any point by pressing the <ESC> key. In this mode the system does not ask the user for the flight number of every record that is to be revised. It automatically presents the records in ascending order of flight numbers. The user is given the option to revise or skip any record. This mode is useful when there is a large scale revision to be done. .
The user can start revision from the first flight in the list, by entering 0 as starting flight number.

To revise any field, position the cursor on that field, enter the new value and press <ENTER>. To skip a field without revising, just press <ENTER>. The validations which are specific to revision are explained below.



OPERATING DAYS : If the revised days of operation includes a day on which an alternate route exists then the new operating days is not accepted.

For eg., Route Operating Days

Route 1 (IC 101) 127

Route 2 (IC 101) 346

If the above schedule is to be changed to,

Route 1 (IC 101) 1234

Route 2 (IC 101) 567

then revision will have to be done in the following order

1) Revise Operating Days of Route 2

346 ---> 6 { Days 3 & 4 are released by Route 2}

2) Revise Operating Days of Route 1

127 ---> 1234 {Day 7 is released and 3,4 are included since they are free}

3) Revise Operating Days of Route 2

6 ---> 567 { Days 5,7 are included. 7 is free since it was released in the above step}

CHANGING CITY NAMES : When the cursor is on the city name press 'C' to change city name or <ENTER> to skip without changing. Only destination names and first origin name can be changed. The new city name for a destination should be different from its origin.

REMOVING A LEG : When the cursor is positioned on the beginning of a leg, press 'R' to remove that leg. The cursor is positioned on the succeeding leg to revise the timings if necessary. The example below illustrates it.

Eg.,	AHMEDABAD	1300	1500	LUCKNOW
	LUCKNOW	1530	1600	PATNA
	PATNA	1630	1715	CALCUTTA

a) If the first leg is to be removed position the cursor on AHMEDABAD and press 'R'. The resulting route will be

LUCKNOW	1530	1600	PATNA
PATNA	1630	1715	CALCUTTA

b) If the second leg is to be removed, then position the cursor on LUCKNOW in the second line. Press 'R' and the resulting route will be

AHMEDABAD	1300	1600	PATNA
PATNA	1630	1715	CALCUTTA

c) If the third leg is to be removed, then position the cursor on PATNA in the third line. Press 'R' and the resulting route will be

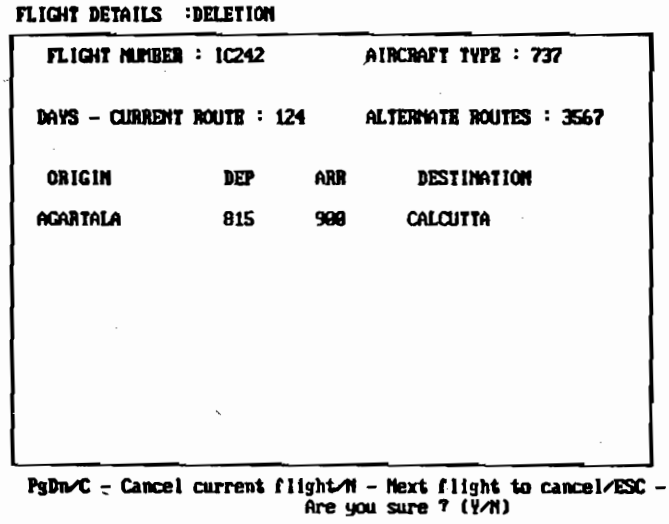
AHMEDABAD	1300	1500	LUCKNOW
LUCKNOW	1530	1715	CALCUTTA

DELETING A ROUTE : This option is available only when the flight has more than one route. The route which is being displayed currently can be deleted by pressing 'D'.

ADDING A ROUTE : Similar to adding a route through the ADDITION mode. This option is availble only if the flight is not operating on alteast one of the week days.

EXIT : Press 'N' for next new flight to revise or <ESC> to get back to the flights menu.

DELETION



Use this option to cancel an existing flight. The system requires the flight number to access the details of the flight. The flight details are displayed and only after confirming with the user, the flight is deleted from the database. All routes of the current flight are deleted.

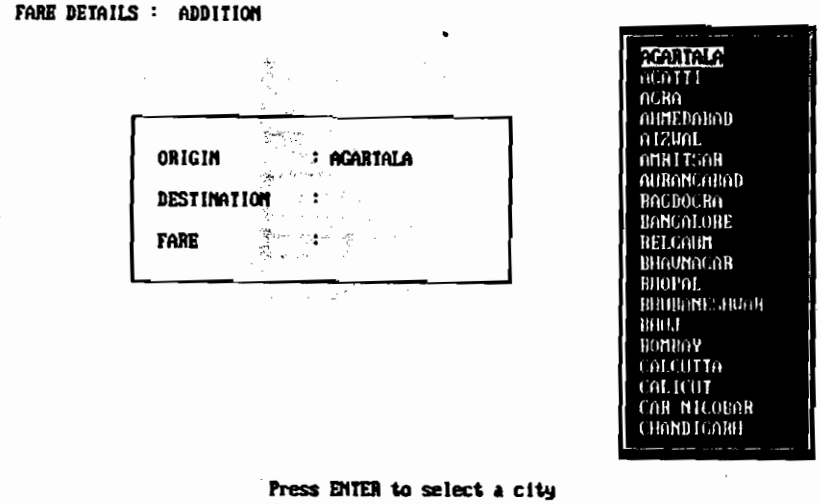
EXIT : Press 'N' for next new flight to delete or <ESC> to get back to the flights menu.

2.3 FARE

This provides a vertical menu with the following options for updating fare records.

ADDITION REVISION DELETION

A fare record consists of a pair of cities (Origin,Destination) and the direct fare from the Origin to the Destination.

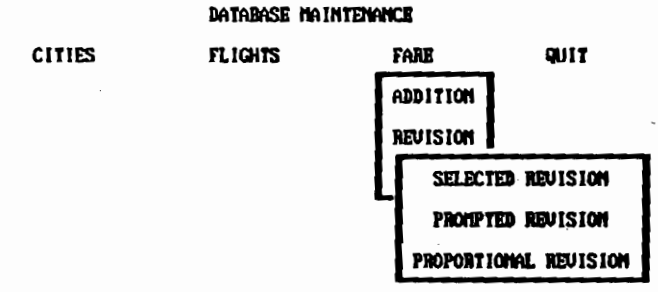


ADDITION

To add a new fare record the user has to select a pair of cities from the city window. The system accepts a new fare only if
a) there is alteast one flight operating between the specified pair and
b) there is no fare record present in the database for the specified pair. Also the fare should not exceed 65536.

REVISION

This option provides three modes of revision.



Revise details of selected fare records
ESC to previous menu

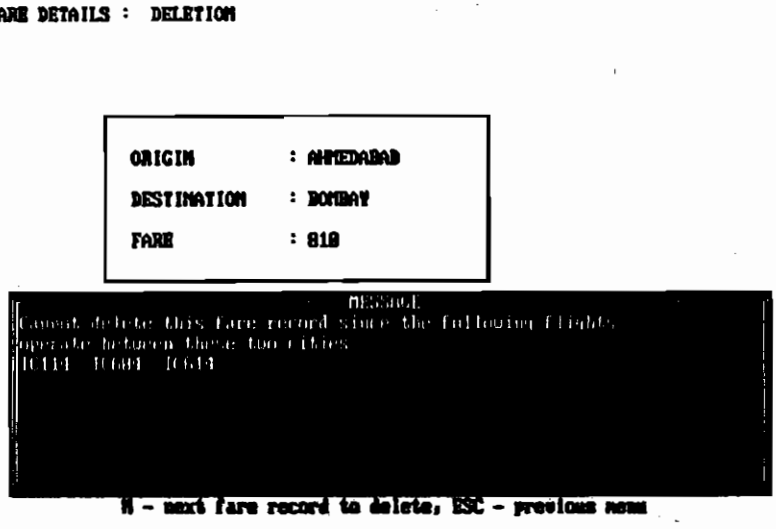
SELECTED REVISION : This option is for selective updation of fare records. The user has to select the pair of cities. If the fare record of the specified pair is present in the database, the system displays the old fare and allows the user to update it. The fare record has to be present already.
This mode is useful when there are few records to be updated.

PROMPTED REVISION : Under this mode the system accepts from the user the starting city name and presents all fare records with the selected city as origin, for revision. The user can revise or skip any of these records.
The above step is repeated for all subsequent cities in the alphabetical order. At any point the user can end revision by pressing the <ESC> key.
This mode is useful for large scale revisions.

PROPORTIONAL REVISION : If there is a proportional change in the fare for the entire system, then this option accepts the proportion of increase/decrease and revises all fare records in the database.

DELETION

The user has to specify the pair of cities whose fare record is to be deleted. After selection of the cities, the system checks whether there are one or more flights operating between the specified pair. If so, then the fare record is not deleted from the database. As an additional information to the user, all flights operating between the specified pair of cities are displayed in a window.



Before deleting a fare record, the details of all flights which have the specified pair of cities as part of their route, must be revised so that there is no flight operating between the two cities.

2.4 QUIT

On selecting this option, the system asks the user whether to save the changes made during the current session.

If the answer is yes then the system does certain interfile compatibility checks before saving onto the files. Each of those checks are explained below.

a) For every pair of cities connected by a direct service there must exist a corresponding fare record. If not present, the following screen will be displayed.

DATABASE MAINTENANCE

CITIES FLIGHTS FARE QUIT

C to Cancel the flight and continue checking, ESC to go back to main menu
ESC to previous menu

b) For every city in the cities file, there must be atleast one flight from/to that city. The following screen will be displayed for every city that fails this check.

DATABASE MAINTENANCE

CITIES FLIGHTS FARE QUIT

D to Delete the city, ESC to go back to main menu
ESC to previous menu

c) For every fare record present in the database there must be atleast one flight operating between the corresponding pair of cities. Any record that fails this check will be displayed as below.

D to Delete this record, ESC to go back to main menu

When any of these checks fail, the system provides the following two options.
- to get back to the maintenance module to make necessary changes
- to delete the incompatible record and continue with the interfile checks.

OUTPUT FILES

The system saves the changes to the following files after the interfile checks are over.

TIMETABL.DAT - Contains complete details of all flights.

DIR_FARE.DAT - Fare record for every pair of cities between which a direct service exists.

CITIES.DAT - City names and the numbers given to them by the system

After saving the changes to the disk, the system generates the following output files.

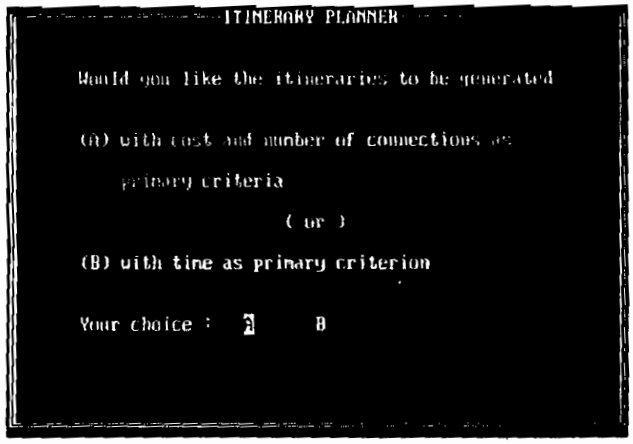
COMPLETE.DAT - Generated from TIMETABL.DAT. Required for both cost-based and time-based itinerary generation.

BCOMPLET.DAT - Generated from COMPLETE.DAT. Required for the backward pass of time-based itinerary generation.

FULLFARE.DAT - Generated form DIR_FARE.DAT. Least possible fare between every pair of cities in the system is computed. Necessary for all modules of itinerary generation. The system takes 20-25 seconds to generate this file.

3. ITINERARY GENERATION

On selection of this option, the following screen is displayed.



COST BASED ITINERARY GENERATION

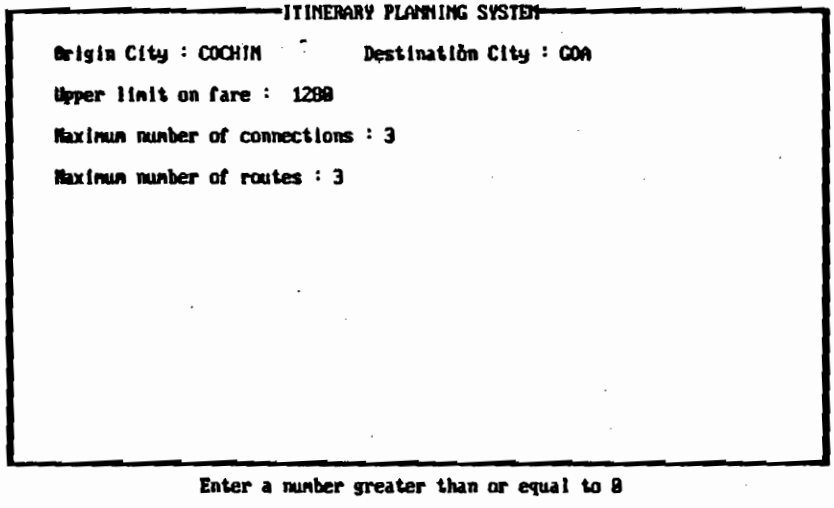
Under this option itineraries are generated in ascending order of the total cost(fare). These itineraries also satisfy upper limits on number of connections and cost which are user-specified.

TIME BASED ITINERARY GENERATION

Under this option itineraries are generated in ascending order of total travel time. All of them satisfy the upper limit on total cost specified by the user.

The following sections describe the inputs requested and the validations applied on them under both these options.

3.1 ITINERARY GENERATION (COST BASED)



ORIGIN CITY : The user can select any city from those displayed in the window. The cities are displayed in the alphabetical order. The user can use , , <HOME>, <END>, <PGUP> & <PGDN> keys to select the origin city.

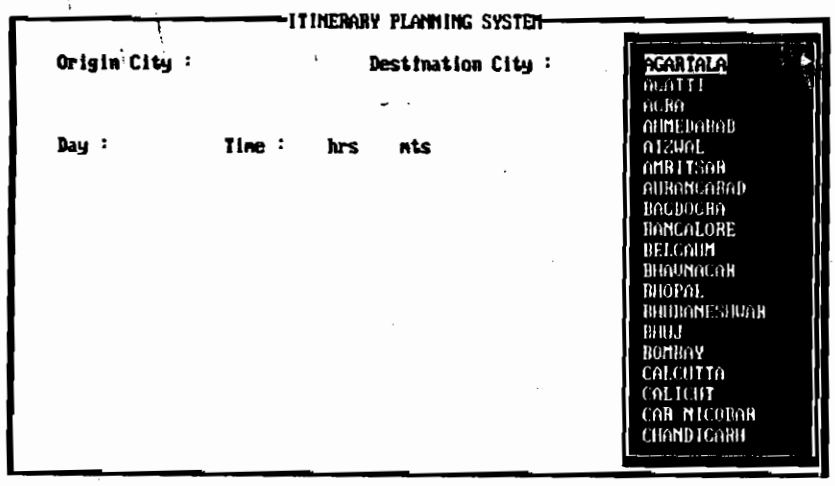
DESTINATION CITY : From the same window of city names, the user can select the destination city. The destination city should not be same as the origin city.

MAXIMUM NO. OF CONNECTIONS : Number of connections is defined here as the number of flights to be changed while travelling from origin to destination. The system requests the user to specify the maximum number of connections he is willing to permit in the itineraries displayed. If he specifies a number which is less than the least possible, then the system displays no itineraries. This input is used to control the total number of itinearies generated and also to avoid displaying cockade routes.

FARE UPPER LIMIT : The upper limit on the total cost of each itinerary generated, is accepted either as a value or as a proportion of the least fare possible to fly from the origin to the destination. To help the user to specify a reasonable amount, the least fare from the origin to the destination is also displayed. The fare upper limit should not be less than the least fare possible.

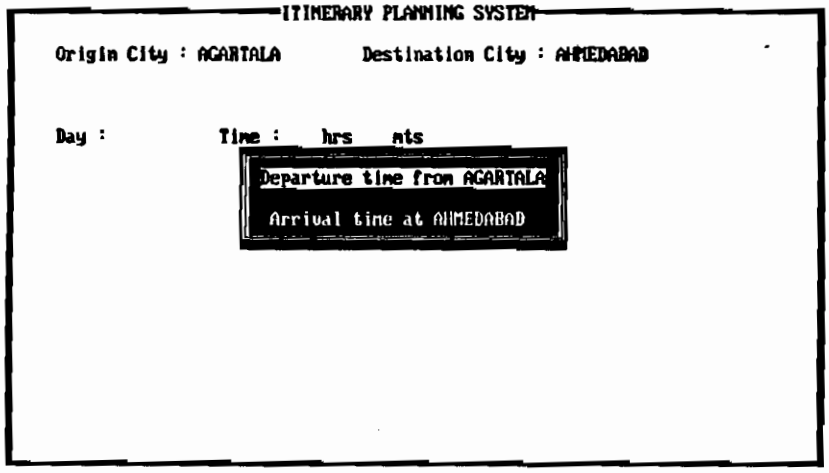
NO. OF ROUTES : This input is also requested to restrict the total number of itineraries displayed. If the fare upper limit is high, then the total number of itineraries satisfying that limit will also be high. Hence the system requests the user to specify the total no. of routes he would like to look at and compare. But if the user specifies a high figure as no. of routes, then the number of itineraries generated and the amount of time taken to generate them will increase substantially.

3.2 ITINERARY GENERATION (TIME BASED)



ORIGIN CITY : The user can select any city from those displayed in the window. The cities are displayed in the alphabetical order. The user can use , ,<HOME>,<END>, <PGUP> & <PGDN> keys to select the origin city.

DESTINATION CITY : From the same window of city names, the user can select the destination city. The destination city should not be same as the origin city.



DEPARTURE TIME FROM ORIGIN (FORWARD PASS) : If the user selects this option the system requests the user to input the time and day he is available for departure from the origin. The itineraries generated will leave the origin after the time and day specified by the user.

ARRIVAL TIME AT DESTINATION (BACKWARD PASS) : If this option is selected the system requests the user to input the desired arrival time and day at the destination. The itineraries generated will reach the destination before the day and time specified by the user.

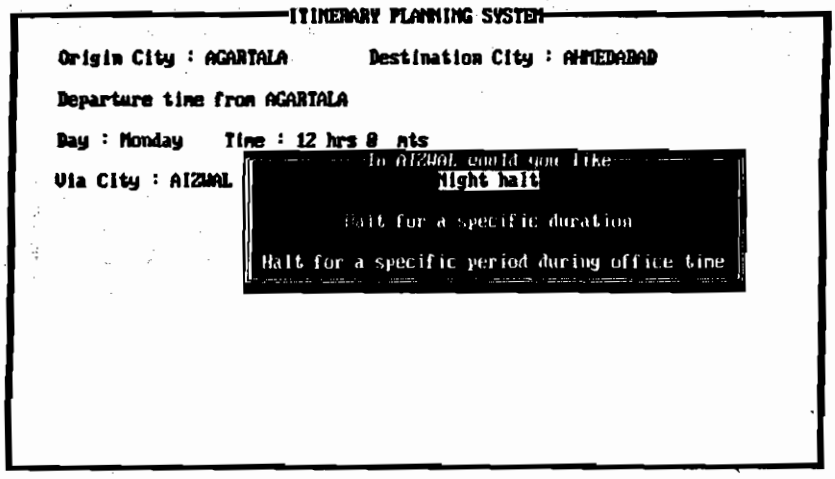
DAY OF TRAVEL : A menu with all the week days is displayed.

TIME : Time is accepted as a two part input of hours and minutes. 24 hour clock is adopted. Normal time validations are applied on the input.

The day and time are treated as departure or arrival day and time according to the option choosen earlier.

VIA CITY : This is an optional input. The system prompts the user whether he would like to specify a via city. If the answer is positive, it allows him to select the via city from the city window. The via city should not be same as the origin city or the destination city choosen earlier.

HALT AT VIA CITY : After selection of the via city, a menu of halt options is displayed.

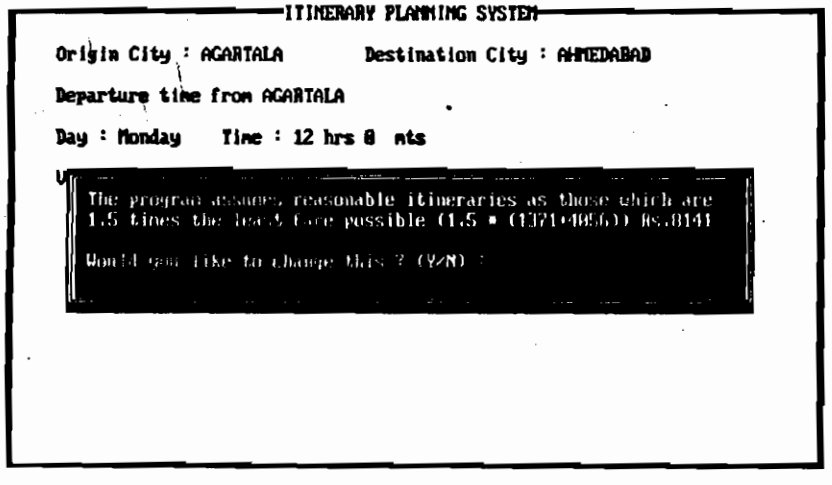


You would be available for the first flight next morning

NIGHT HALT : The itinerary is generated in such a way that there is a night halt at the via city. If the first leg of the journey ends at the via city on Monday, then irrespective of the arrival time at via city, the second leg of the journey starts on Tuesday morning. After the night halt, the system assumes that the user is available for the first flight in the morning.

HALT FOR A SPECIFIC DURATION : If the user selects this option, he also has to input the number of hours and minutes he would like to spend at the via city.

HALT DURING OFFICE TIME : The system assumes the office timings as 10.00 A.M - 1.00 P.M and 2.00 P.M - 5.00 P.M. The halting time accepted is taken to be the number of office hours and minutes. The departure time of the second leg of the journey is computed accordingly.



FARE UPPER LIMIT : The upper limit on the total cost of

each itinerary generated, is accepted either as a value or as a proportion of the least fare possible to fly from the origin to the destination. To help the user to specify a reasonable amount, the least fare from the origin to the destination is also displayed. The upper limit should not be less than the least fare possible.

NO. OF ITINERARIES : The number of itineraries to be displayed is also accepted from the user. Only that many itineraries are generated by the system. There is no validation on this input except that it has to be a number. Higher the number the more time the system takes to generate them. If the fare upper limit is very low, it may not be possible to generate the requested no. of itineraries. Only those which satisfy the cost requirement are generated and displayed.

3.3 ITINERARY DISPLAY

COST-BASED ITINERARY GENERATION

Itineraries are displayed in the order of increasing fare. All itineraries based on a common route are displayed together. For every change in route a fresh screen is displayed.

```
Max. no. of connections 3 Total routes 3 Fare limit 3797
   FLIGHT TYPE FROM          DEP  ARR  TO          DAYS    COST
   IC162  737  COCHIN         755  940 BOMBAY      1234567 Rs. 1654

   IC492  737  BOMBAY        1210 1540 JAIPUR      1234567 Rs. 1510

        Fare Rs.3164 (Min fare Rs.3164)

   IC192  737  COCHIN        1115 1300 BOMBAY      1234567 Rs. 1654

   IC494  737  BOMBAY        1730 2055 JAIPUR      1234567 Rs. 1510

        Fare Rs.3164 (Min fare Rs.3164)


Press any key to view more
```

```
Max. no. of connections 3 Total routes 3 Fare limit 3797
    FLIGHT TYPE FROM          DEP  ARR  TO              DAYS     COST
    IC468  737  COCHIN        1010 1115 GOA             1234567 Rs. 1067

    IC164  300  GOA           1325 1420 BOMBAY          1234567 Rs.  767

    IC494  737  BOMBAY        1730 2055 JAIPUR          1234567 Rs. 1510

         Fare Rs.3344 (Min fare Rs.3164)
```

```
Press any key to view more
```

```
Max. no. of connections 3 Total routes 3 Fare limit 3797
    FLIGHT TYPE FROM          DEP  ARR  TO              DAYS     COST
    IC162  737  COCHIN         755  940 BOMBAY          1234567 Rs. 1654

    IC492  737  BOMBAY        1210 1425 UDAIPUR         1234567 Rs. 1090

    IC494  737  UDAIPUR       1910 2055 JAIPUR          1234567 Rs.  617

         Fare Rs.3369 (Min fare Rs.3164)

    IC192  737  COCHIN        1115 1300 BOMBAY          1234567 Rs. 1654

    IC494  737  BOMBAY        1730 1840 UDAIPUR         1234567 Rs. 1090

    IC492  737  UDAIPUR       1455 1540 JAIPUR          2345671 Rs.  617

         Fare Rs.3369 (Min fare Rs.3164)
```

```
No more itineraries. Press any key to continue.
```

TIME-BASED ITINERARY GENERATION (FORWARD PASS)

```
Available for departure from TRIVANDRUM  MON   8:00    Fare limit : Rs.5500
    FLIGHT TYPE FROM          DAY DEP  ARR  TO              FLT TIME COST
  1 IC168  300  TRIVANDRUM    MON 1455 1940 DELHI              4h45m Rs. 3379

    IC483  737  DELHI         TUE  630  745 LEH               1h15m Rs. 1090

         Fare Rs.4477 (Min fare Rs.4477)

  2 IC530  737  TRIVANDRUM    MON 1125 1225 BANGALORE          1h 0m Rs.  967

    IC404  300  BANGALORE     MON 1430 1700 DELHI              2h30m Rs. 2485

    IC483  737  DELHI         TUE  630  745 LEH               1h15m Rs. 1090

         Fare Rs.4550 (Min fare Rs.4477)
```

```
Press any key to view more
```

```
Available for departure from TRIVANDRUM  MON   8:00    Fare limit : Rs.5500
    FLIGHT TYPE FROM          DAY DEP  ARR  TO              FLT TIME COST
  3 IC168  300  TRIVANDRUM    MON 1455 1650 BOMBAY            1h55m Rs. 1910

    IC446  737  BOMBAY       MON 1830 2000 JAIPUR            1h30m Rs. 1510

    IC494  737  JAIPUR       MON 2125 2205 DELHI               40m Rs.  517

    IC483  737  DELHI        TUE  630  745 LEH               1h15m Rs. 1090

         Fare Rs.5035 (Min fare Rs.4477)

  4 IC530  737  TRIVANDRUM    MON 1125 1225 BANGALORE         1h 0m Rs.  967

    IC606  320  BANGALORE     MON 1400 1530 BOMBAY            1h30m Rs. 1367

    IC185  737  BOMBAY       MON 1700 1850 DELHI             1h50m Rs. 1779

    IC483  737  DELHI        TUE  630  745 LEH               1h15m Rs. 1090

         Fare Rs.5211 (Min fare Rs.4477)


Press any key to view more
```

Itineraries are displayed in the order of increasing arrival time at the destination. If two itineraries arrive at the destination at the same time, then that which has lesser number of breaks in journey is displayed first. If the number of breaks in journey is also equal then that itinerary which leaves the origin later is displayed first.

TIME-BASED ITINERARY GENERATION (BACKWARD PASS)

```
Expected arrival at BAGDOGRA  WED  16:00    Fare limit : Rs.4794
    FLIGHT TYPE FROM          DAY DEP  ARR  TO              FLT TIME COST
  1 IC462  737  AHMEDABAD     TUE 2015 2135 DELHI             1h20m Rs. 1229

    IC489  737  DELHI        WED  605  755 BAGDOGRA           1h50m Rs. 1967

         Fare Rs.3196 (Min fare Rs.3196)

  2 IC462  737  AHMEDABAD     TUE 2015 2135 DELHI             1h20m Rs. 1229

    IC889  320  DELHI        WED  845 1115 GUWAHATI          2h30m Rs. 2292

    IC498  737  GUWAHATI     WED 1250 1335 BAGDOGRA            45m Rs.  504

         Fare Rs.4025 (Min fare Rs.3196)

  3 IC462  737  AHMEDABAD     TUE 2015 2135 DELHI             1h20m Rs. 1229

    IC489  737  DELHI        WED  605  910 GUWAHATI          3h 5m Rs. 2292

    IC498  737  GUWAHATI     WED 1250 1335 BAGDOGRA            45m Rs.  504

         Fare Rs.4025 (Min fare Rs.3196)

Press any key to view more
```

```
Expected arrival at BAGDOGRA  WED  16:00    Fare limit : Rs.4794
    FLIGHT TYPE FROM          DAY DEP  ARR  TO          FLT TIME COST
 4 IC504  320  AHMEDABAD     TUE 1855 1950 BOMBAY          55m Rs.  818

   IC175  300  BOMBAY         WED  540  750 CALCUTTA       2h10m Rs. 2484

   IC221  737  CALCUTTA      WED 1100 1155 BAGDOGRA         55m Rs.  768

        Fare Rs.3974 (Min fare Rs.3196)

 5 IC417  737  AHMEDABAD     TUE  740 1000 DELHI          2h20m Rs. 1229

   IC809  320  DELHI         WED  845 1115 GUWAHATI       2h30m Rs. 2292

   IC490  737  GUWAHATI      WED 1250 1335 BAGDOGRA         45m Rs.  584

        Fare Rs.4825 (Min fare Rs.3196)


No more itineraries. Press any key to continue.
```

Itineraries are displayed in the order of decreasing departure time from the origin. If two itineraries leave the origin at the same time, then that which has lesser number of breaks in journey is displayed first. If the number of breaks in journey is also equal then that itinerary which reaches the destination early is displayed first.