# A Routing and Scheduling Algorithm for Radioactive Material Shipments

D. Smith

*SCIENTECH, Inc., Rockville, Maryland, United States of America*

## INTRODUCTION

The United States Department of Transportation regulations for routing and scheduling of large quantity shipments of radioactive material acknowledge the relevance of route-specific parameters such as population density and activities, accident rates, time of day, and day of week. On the other hand, review of transportation analysis literature reveals little work in analytical techniques for optimization of routes and schedules based on such characteristics.

To address this deficiency, a model and a corresponding algorithm have been developed to optimize analytically paths and schedules through networks described by route-specific, time-varying parameters. A computer code incorporating the algorithm has been developed and successfully tested. The algorithm and the corresponding computer code consider deterministic time-of-day variation of parameters (e.g., traffic speed and density during rush hours on urban road networks) for individual network arcs. The relative importance of arc parameters can be selected by altering their weighting factors in the objective function. The code is structured to allow selection of additional or alternative optimization parameters and to permit modification for application to other hazardous materials shipments. The results of the analyses describe the optimum route(s) and departure time(s) for selected origin and destination nodes in a network.

## OVERVIEW OF SOLUTION TECHNIQUE

Figure 1 illustrates the major steps of the procedure devised to analyze network models for spent fuel shipping routes and schedules. The objective function selected was a weighted sum of time of travel and general population radiation dose and defines the impedance of an arc or path. The material in this section provides an overview of the primary elements of the analysis technique. Then, described in the following sections are the four key procedures of the technique -- the reaching algorithm, the elimination by bounds procedure, the backtracking procedure, and the Dijkstra shortest path algorithm.

The reaching algorithm (Jensen 1986) is the central element of the transportation analysis developed. Beginning with the initial origin and departure time states defined for the problem, the algorithm generates possible states in a time/place array (Smith 1989) and checks the parameters of the states as they are generated to determine the lowest impedance path to each state from a source state. Each state represents arrival at a specified node n at an allowable time t based on the departure time, estimated speed, and the preceding path through the network.

As each new state Y(t,n) is generated, an elimination by bounds test (Jensen 1986) is performed. The test determines whether the lowest impedance path from a source state to the state Y(t,n) plus the lower bound of the path from Y(t,n) to a sink state can possibly improve the best available source to sink path for the time/place array. The lower bound $Z_{lb}$ associated with each state Y(t,n) is the lowest possible impedance in traversing the network from state Y(t,n) to a sink state. The value of $Z_{lb}$ for each state is determined by relaxing the rush period constraint on the time/place array (ignoring rush period effects) and then analyzing the resulting, less complex place network with a Dijkstra shortest path analysis (Jensen and Barnes, 1980)

A second important test performed during the reaching analysis determines whether the place node n associated with a state Y(t,n) about to be generated has been encountered at some earlier state in the path that leads to Y(t,n). If it has, the state under consideration is not generated. This operation is the backtracking elimination.

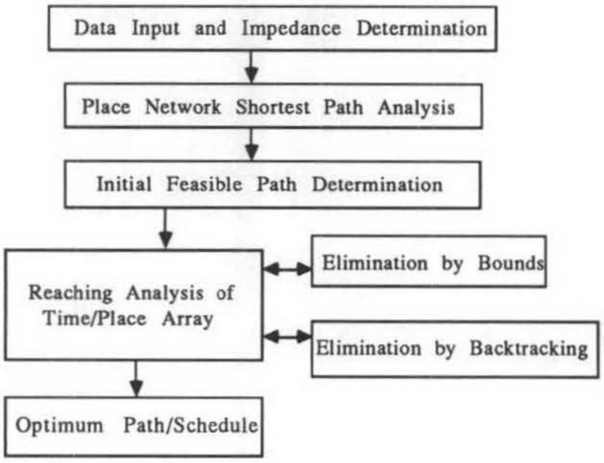
Data Input and Impedance Determination
Place Network Shortest Path Analysis
Initial Feasible Path Determination
Reaching Analysis of Time/Place Array
Elimination by Bounds
Elimination by Backtracking
Optimum Path/Schedule

Figure 1. Basic steps of analysis procedure.

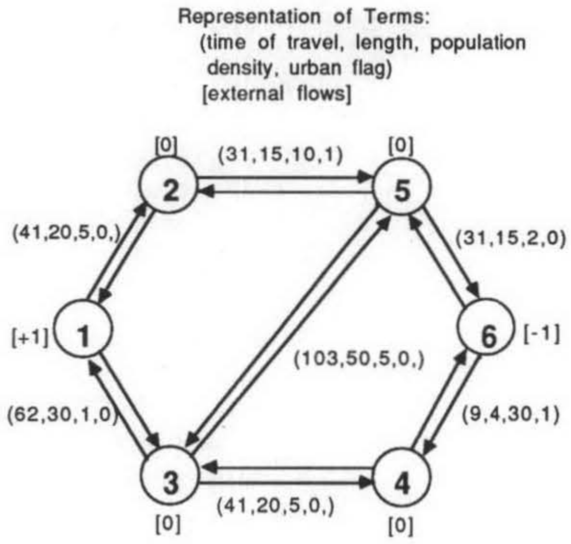

Figure 2. SIMPNET sample network.

The primary comparisons and eliminations described provide powerful mechanisms to limit the number of states generated for time/place state arrays by the reaching algorithm. Without the eliminations, analysis of the time/place array would amount to enumeration of all possible states followed by identification of the sink states and associated paths with the lowest accumulated impedance.

A simple six node, seven arc network (SIMPNET) described in Fig. 2 was devised to provide simple illustrative examples and to test the logic of the algorithm. Application of the reaching algorithm (with elimination procedures) to the SIMPNET example with three departure times yielded a total of nine states. Without the elimination procedures, at least thirty-three states would have been generated (see Fig. 4).

## REACHING ALGORITHM

The following section describes the reaching algorithm and is simplified by the following notation:

$Y(t,n)$ = a state of time t and place node n in the time/place array
$Y(t,S)$ = a source state (associated with place network node S and a time t)
$Y(t,T)$ = a sink state (associated with place network node T and a time t)
$S[Y(t,n)]$ = optimum path impedance from state $Y(t,n)$ to a sink state
$g'[Y(t,n)]$ = optimum path impedance from an initial state to state $Y(t,n)$
$r_{t1}, r_{t2}, \ldots$ = rush period indicators; equal to 1 for rush periods, equal to 0 for non-rush periods
$td_1, td_2, \ldots$ = departure time from source node S; note that $td_i < td_{i+1}$
$A(Y)$ = the set of operations that generate all the time/place states that can be reached from state Y with no intermediate states
$A_1(Y), A_2(Y), \ldots$ = elements of the set $A(Y)$
$P(Y)$ = a pointer that identifies the state preceding Y through which the lowest impedance path to Y proceeds

$h(Y,A_i)$ = non-rush period impedance for operation $A_i$ from state Y
$h'(Y,A_i)$ = non-rush period impedance for operation $A_i$ to state Y
$H(Y,A_i)$ = rush period impedance increment for operation $A_i$ from state Y at any rush period (for which $r_{t1}, r_{t2}, \ldots = 1$)
$H(Y,A_i)$ = rush period impedance increment for operation $A_i$ to state Y at any rush period (for which $r_{tl}, r_{t2}, \ldots = 1$).

The "lexicographically smallest state" in a state array is the smallest place node value among the set of states with the smallest time value that has not yet been examined to determine whether it can lead to the generation of adjacent states. States adjacent to state Y are those that can be reached directly from Y without encountering any other states. Finally, the set of operations A(Y) defined above describes the logic for generating possible adjacent time/place states and considers rush periods.

Application of the reaching algorithm begins with identification of the initial states $Y(td_i,S)$ of the time/place array. For the spent fuel transportation problem treated here, the time values for the initial states are the selected departure times. The place value for all initial states is the source node S for the associated place network.

The steps of the reaching algorithm are as follows:

(1) Select the lexicographically smallest state in the state array. [For the first iteration of the reaching algorithm, the smallest time value will be exhibited by state $Y(td_1,S)$.]
(2) Determine the set of operations $A(Y)$ by which all states adjacent to Y are generated.
(3) Perform one of the operations $A_i(Y)$ to determine the time and place of a potential state Y' as well as the impedance in reaching that state.
(4) If the state Y' has already been generated, proceed to step 5. If Y' has not yet been generated, add the state to the time/place array, determine and save the value of $g'[Y'(t,n)]$, and save the value of $P(Y')$. Proceed to step 6.
(5) If $g'(Y') \leq h(y,A_i) + (r_{tl} + r_{t2})H(y,A_i) + g'(Y)$, make no changes in state values and proceed to step 6. Otherwise set $g'(Y') = h(Y,A_i) + (r_{t1} + r_{t2})H(Y,A_i) + g'(Y)$ and $P(Y')=Y$.
(6) If all operations in the set $A(Y)$ have not yet been performed, select an operation $A_i(Y)$ that has not been performed (i is less than or equal to the number of arcs originating at the place node associated with state Y) and return to step 3. Otherwise proceed to step 7.
(7) If all states have not yet been subjected to their associated operations $A(Y)$, return to step 1. Otherwise, the lowest impedance path from each initial state has been determined and the reaching process completed.

The reaching algorithm as described above provides the basic structure of the analysis technique developed. The algorithm was, however, enhanced in several ways which render it more efficient.

## ELIMINATION BY BOUNDS

Although the reaching algorithm described above is logically valid, in reality it would generate many states in the time/place array that could not possibly be part of a minimum impedance path. A key enhancement to the reaching algorithm is a technique for comparing the impedance $Z(P)$ of the best possible path through each time/place state considered to the impedance $Z_b$ of the best available path from a source state to sink state at the time of the comparison. (The value of $Z_b$ may be improved during the reaching analysis.)

The reaching analysis is begun with a best available path impedance $Z(Py)$ derived from a preliminary shortest path analysis of the place network. The familiar Dijkstra algorithm is used for the preliminary analysis. The impedance for the best available path, Py, is denoted $Z_b$ and is obtained by determining the impedance of traversing the path Py <u>including all additional impedances resulting from rush period encounters</u>.

The path P through a state $Y(t,n)$ is evaluated in two parts: (1) the partial path P' from source state $Y(td_i,S)$ to the state $Y(t,n)$ where $td_i$ is any allowable departure time and (2) the partial path P" from state Y(t,n) to a sink state Y(ta,T) where ta is the time of arrival at the sink state. The previous section describes the comparison in the reaching algorithm for values of g'(Y) and the elimination of all paths to Y except the lowest impedance path. These values for g'(Y) are the impedance Z(P') of the best path <u>to</u> Y from a source state. To establish the lower bound for the impedance $Z_{lb}$ of partial path P", the assumption is made that no rush periods are encountered from Y(t,n) to a sink state.

If $Z(p') + Z_{lb} > Z_b$, then the best path through state P of the time/place array (which will exhibit an even higher impedance if any rush periods are actually encountered) cannot possibly be better than the best available path and the new state will not be generated. On the other hand, if $Z(P') + Z_{lb} \leq Z_b$, the path through P merits further consideration. The further consideration first requires determination of any impedance increments to $Z_{lb}$ which result from rush period encounters along P" to yield the impedance Z(P") of a feasible path (rush periods included). If $Z(P") + Z(P') < Z_b$, then $Z_b$ is set equal to this smaller value, the new state Y(t,n) is saved, and the reaching algorithm will be applied to the state Y(t,n). If $Z(P") + Z(P') \geq Z_b$; the value of $Z_b$ remains the same, but the state Y(t,n) will remain among the states to which the reaching algorithm will be applied.

The elimination by bounds test is applied to each state generated by the reaching analysis in order to eliminate, as soon as possible, any path that cannot yield the minimum impedance for the state array. The procedure may also identify an improved (smaller) value for $Z_b$ which can improve the elimination process.

## ELIMINATION BY BACKTRACKING

A final, relatively straightforward elimination test was devised for the state array analysis. Each time a new time/place state Y(t,n) is identified, all of the preceding states in the path to Y(t,n) are examined. If place node n is associated with any previous state on the path, adding the state Y(t,n) would either create a circuit in which the path exits and subsequently returns to the same place node or produce backtracking in which the path oscillates between two adjacent nodes. Neither situation is acceptable and in either case the state is not added to the time/place array.

## SHORTEST PATH ALGORITHM

The Dijkstra shortest path algorithm is used to determine the lowest impedance path from a source node to all other nodes in the place network. The use of these preliminary shortest paths in the elimination by bounds procedure was described in a preceding section. The impedance values for all arcs must be non-negative, and the links between nodes must be directed arcs. The elimination by bounds technique requires lower bound impedance values for the paths from intermediate nodes to the sink node. Therefore, the Dijkstra shortest path analysis is initiated from the actual sink node which yields the shortest paths from all initial and intermediate nodes to the sink.

## SHORTEST PATH ANALYSIS

The shortest path is used to determine the initial best available path impedance $Z_b$ for the reaching analysis. Beginning at the actual source node, 1, and any one of the arbitrary departure times, the path is traversed while monitoring time of arrival at each node so that rush period increments can be added where appropriate.

For the SIMPNET example, a time cycle that repeats every 60 minutes was selected and the rush period ranges (in units of minutes after the beginning of each cycle) were

$20 + (60)i \leq t \leq 25 + (60)i$ and
$40 + (60)i \leq t \leq 45 + (60)i$ where $i = 0,1,2,....$

The arcs (2,5),(5,2),(4,6), and (6,4) in Fig. 2 represent urban areas. For the sample case the initial departure time arbitrarily selected was 0 minutes -- the beginning of the first cycle.

## REACHING ANALYSIS WITH ELIMINATIONS

The reaching procedure described begins with the initial state having the earliest departure time $Y(td_1,S)$. The departure times chosen for this example were 0, 20, and 40 minutes, so the initial state with the earliest departure time was (0,1). That state (0,1) is designated "a" in Fig. 3 . The first operation $A_1(O,1)$ yields state (41,2).

The lowest impedance source-to-sink path through the array is found by examining all sink states generated and selecting the one with the smallest accumulated impedance. The only sink state generated was (123,6) shown in Fig. 3 with an accumulated impedance of 0.192. The path is found by successively returning to the state that generated each following state on the path. The shortest path through the time/place array is (20,1), (61,2), (92,5), and (123,6).

The seven eliminations that occurred in the reaching analysis resulted when the impedance Z(P) of the best possible path through the state under consideration exceeded $Z_b$. States that would have resulted in circuits or backtracking were not considered.

Figure 4 illustrates the proliferation of states and paths that would have been generated by the reaching algorithm without using the elimination by bounds procedure. No paths were allowed to return to the source node at any time. Actually, the paths that would have been generated without the backtracking elimination would have continued indefinitely as a result of oscillations between node 3 and node 5.

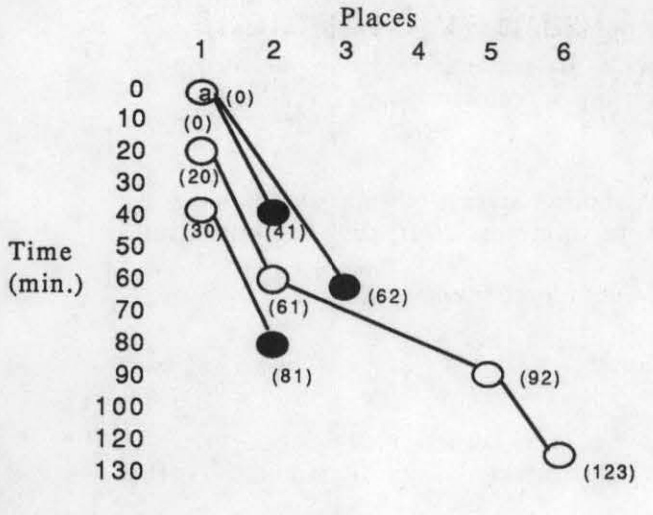

Figure 3. Reaching analysis.

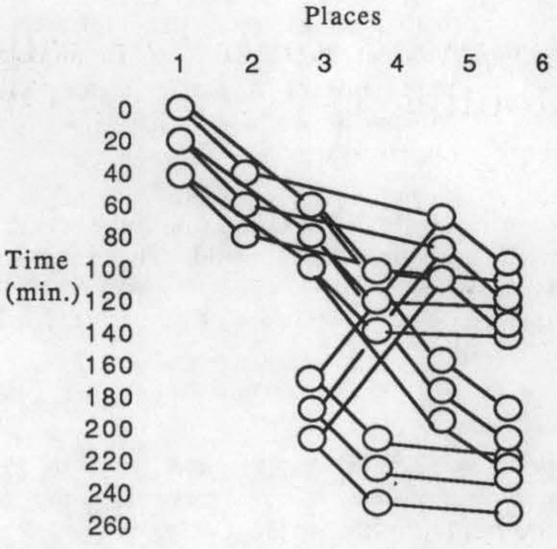

Figure 4. Time/place array for analysis without elimination procedures.

The results presented here for manual application of the transportation analysis technique to SIMPNET agree in every respect with the results from the computer program DANTRAN which was written to use the algorithm described. Minor round-off differences occurred for some values, but the states generated, the states eliminated, and the minimum impedance path were all identical.

## REFERENCES

Jensen, P.A. and Barnes, J.W. *Network Flow Programming*, John Wiley and Sons, Inc., New York, New York (1980).

Jensen, P.A. *Notes on Network Analysis.* Photocopy (1986).

Smith, D. *Multi-Objective Optimization for Routing and Scheduling of Radioactive Material Shipments*, Waste Management '89 Symposium, Tucson, Arizona (1989).