JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS **166**, 192–198 (1992)

A Dynamic Programming Solution of a Shortest Path Problem with Time Constraints on Movement and Parking

N. G. F. SANCHO

Department of Mathematics and Statistics, McGill University, Montreal, H3A 2K6 Quebec, Canada

Submitted by E. Stanley Lee

Received September 7, 1990

In this paper we present a dynamic programming formulation of a shortest path problem when some arcs are open for travelling during specified periods of time. We consider the case when parking is allowed on the nodes whenever it is necessary to wait for an adjacent arc to be opened. We will also assume the possibility of no parking at the nodes during the time when the nodes are occupied. © 1992 Academic Press, Inc.

INTRODUCTION

The shortest path problem has been studied before and an appraisal and survey of a dynamic programming solution have been given by Dreyfus [1]. In this type of problem, finding the shortest path from source node to terminal node with no restriction of movement along the arc or on the node is normally required. Recently a shortest path problem with restriction on time windows on the node was studied by Desrochers and Soumis [2]. In this type of problem it is permitted to enter a node only during a certain time interval or time window. It is not permitted to enter a node after the time window has closed. However, it is permitted to enter a node before the time window has opened but one has to wait at the node for the time window to open.

In a paper by Halpern and Priess [3] an algorithm is presented for finding a shortest path problem with time constraints on movement and parking. It is assumed that some arcs are closed for travelling during specified periods of time and parking in the vertices of the network is permitted, when it is necessary to wait for an arc to be opened. The possibility of "no-parking" or "occupied periods" in the nodes is also considered. Since there are many types of routing problems which have been formulated by dynamic programming [4], a dynamic programming formulation is given for the shortest path problem presented by Halpern and Priess [3]. This type of problem emerges in the management of railway system or a network of narrow roads with convoys moving on it.

A dynamic programming formulation of this type of shortest path problem would widen the scope of problems which can effectively be solved by dynamic programming.

Formulation of the Problem

In a network of single lane roads or railways there is always movement of trains or convoys along the arcs of the network. Hence if a train leaves station i at time t and moves along an arc $a = (i, j)$ to an adjacent station j, for $d(a)$ units of time, then the arc a is closed for travel in the opposite direction during the period $[t, t + d(a)]$. There is also a safety requirement of a time interval Δ for which departing trains can leave a station. Hence if a train leaves a station i at time t, the station i will be closed for travel until $t + \Delta$. If a train waits at a station or parking area, then the station is considered to be occupied and no other train may stay in the parking area. It is however possible that another train may pass through a station while the parking facilities of that station are occupied.

Let $G = (V, A)$ be an N-vertex graph, where V denotes the set of all vertices of G and A denotes the set of all arcs. For convenience an undirected arc (i, j) may be changed to a directed arc by replacing (i, j) by a pair (i, j) and (j, i) respectively, i.e., one leading from i to j and one from j to i. We write 1 as the origin node and N as the destination node. Hence a feasible journey would be a path $P = \{1, i_1, i_2, ..., N\}$ provided all arcs fall within the constraint of movement along the arcs and all parking at nodes occurs within the required time constraint.

In the problem we consider we will normally assume that if a feasible path exists then at each node one has the choice of two decisions. The first is that parking for a period of time is permitted and then it is permissible to continue along an arc when the time period for movement along the arc is open. The second is that no parking at the node is allowed but it is permissible to continue along an arc and be able to park at another node. If these decisions cannot be made then an infeasible path will exist. In the network we consider we will normally assume that a feasible path will exist from source to sink.

Dynamic Programming Formulation

For convenience we assume that the starting time at the origin node 1 is zero. Parking at a vertex i is permitted during time interval $[\alpha_i, \beta_i]$ for a period $p_i \geqslant 0$, where $\beta_i > \alpha_i > 0$ and $p_i \leqslant \beta_i - \alpha_i$. If $\beta_i = \alpha_i$ then no parking is allowed at vertex i.

Departure at vertex i along arc (i, j) is permitted during time interval $[\gamma_{ij}, \delta_{ij}]$, where $\delta_{ij} > \gamma_{ij} > 0$. The time to travel from i to j is then given by $t_{ij} > 0$.

If $i \neq j$ then let $f(i, j)$ be defined as the earliest feasible arrival time at i provided one can depart from i along arc (i, j) and that the route ahead is not blocked. A feasible arrival time at i is defined as the time of arrival at i where one can park for a period $p_i \geqslant 0$ and then depart along (i, j) within the permitted time interval.

If now $\alpha_i \leqslant \gamma_{ij}$ then it is possible to depart from i in a feasible time if the following conditions are met:

$$\begin{gathered}\alpha_i \leqslant f(i, j) \leqslant \beta_i \quad \text{and} \quad \gamma_{ij} \leqslant f(i, j) + p_i \leqslant \delta_{i,j}, \quad \text{where} \\ 0 \leqslant p_i \leqslant \min[\beta_i - f(i, j), \delta_{i,j} - f(i, j)]\end{gathered} \tag{1}$$

or

$$\text{if } \beta_i - f(i, j) \leqslant 0 \quad \text{and} \quad \gamma_{ij} \leqslant f(i, j) \leqslant \delta_{ij} \text{ then } p_i = 0. \tag{2}$$

If

$$\begin{gathered}f(i, j) < \gamma_{ij} < \alpha_i \quad \text{or} \quad f(i, j) < \alpha_i < \gamma_{ij} < \beta_i \quad \text{or} \\ f(i, j) < \alpha_i < \beta_i < \gamma_{ij}\end{gathered} \tag{3}$$

then it is not possible to park at i and not possible to depart at i along arc(i, j). This means that the route along arc(i, j) is blocked.

If

$$\gamma_{ij} < f(i, j) < \alpha_i \leqslant \delta_{ij} \quad \text{or} \quad \gamma_{ij} < f(i, j) < \delta_{ij} \leqslant \alpha_i \tag{4}$$

then $p_i = 0$ but it is possible to depart from i along arc(i, j).

If

$$\delta_{ij} < f(i, j) \tag{5}$$

then it is not possible to arrive at j in a feasible time; i.e., the route along (i, j) is blocked.

Hence if $j \neq k$ then $f(j, k)$ is similarly defined as the earliest feasible arrival time at j provided that one can depart from j along arc(j, k) and that the route ahead is not blocked.

In the functional $f(j, k)$, j acts as a state variable and k is a dummy index giving the direction from j along which we can proceed. Hence using the principle of optimality we obtain

$$f(j, k) = \min_{i,\, p_i}[f(i, j) + p_i + t_{ij}], \qquad \text{with} \quad f(1, j) = 0 \tag{6}$$

if there is a direct route from 1 to j and it is possible to arrive at j in a feasible time. In Eq. (6) we minimize the right-hand side with respect to i and p_i. Equation (6) is valid if conditions given by Eqs. (1), (2), or (4) are satisfied for $f(j, k)$ and arc(j, k).

The solution for arriving at N, the destination node, at the earliest time is then given by

$$\min_{i,\, p_i}[f(i, N) + p_i + t_{iN}], \tag{7}$$

provided

$$\alpha_N \leqslant f(i, N) + p_i + t_{iN} \leqslant \beta_N \tag{8}$$

is satisfied.

On the other hand if the condition given by Eq. (3) is satisfied then we know that the route along (i, j) is blocked. However, by bracktracking on the route arrived at, it may be possible to park longer at some previous node and increase $f(i, j)$ such that $f(i, j) = \gamma_{ij}$ or $f(i, j) = \alpha_i$. Hence the route which includes arc(i, j) is no longer blocked. This means that for some previous arc(i_1, i_2) the value of $f(i_1, i_2)$ may not be unique, since for one route which includes (i_1, i_2) the route ahead is not blocked, but for another route which includes arc(i_1, i_2) the route ahead is blocked and condition (3) is satisfied. However, parking at i_1 may be increased such that the route is no longer blocked. In this case $f(i_1, i_2)$ may have more than one value. In practice however this normally does not occur. If arc(i, j) still remains blocked we write $f(i, j) = \infty$.

If Eq. (5) is satisfied then there is no possible way to reduce $f(i, j)$ and we write $f(i, j) = \infty$.

The solution procedure for solving Eq. (6) can be described as follows. Let $f^r(j, k)$ be the earliest feasible arrival time at j, in r or fewer arc lengths $r = 1, 2, 3, \ldots$, provided one can depart from j along arc(j, k) and that the route ahead is not blocked. Hence we have

$$f^{r+1}(j, k) = \min[f^r(j, k), \min_{i,\, p_i}\{f^r(i, j) + p_i + t_{ij}\}] \tag{9}$$

with $f^r(i, j) = 0$ $\forall r$'s for all direct links (i, j) and feasible arrival time at j.

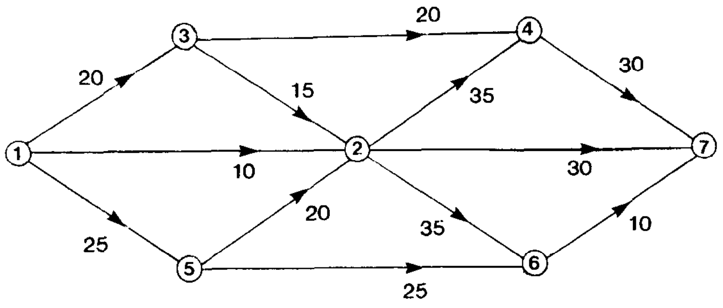

FIG. 1. Network for the worked example.

If

$$f^{r+1}(j,k)=f^{r}(j,k) \qquad \forall \operatorname{arc}(j,k) \tag{10}$$

then

$$f(j,k)=f^{r}(j,k). \tag{11}$$

Under fairly general conditions this procedure terminates [4].

Worked Example

We consider the example given by Halpern and Priess [3] of a directed network with 7 nodes and 12 arcs as illustrated in Fig. 1. The numbers along the arcs indicate the time which is required to travel from the initial node to the end node. The permissible departure time intervals for each arc are presented in Table I. The feasible parking periods for each node are

TABLE I

Feasible Departure Times on Arcs

Initial Node	End Note	Feasible departure times
1	2	10–15
1	3	10–25
1	5	15–25
2	4	28–35
2	6	26–34
2	7	32–60
3	2	35–50
3	4	43–50
4	7	70–99
5	2	30–80
5	6	48–55
6	7	72–80

TABLE II

Feasible Parking Times in Nodes

Note	Feasible parking times
1	0–∞
2	18–30
3	35–45
4	64–75
5	35–50
6	64–75
7	0–∞

presented in Table II. Using Eq. (9) and satisfying the set of Eqs. given by (1), (2), (3), (4), or (5), the following values are obtained:

$$f(1,3)=0; \quad f(1,2)=0; \quad f(1,5)=0$$

$$\begin{array}{lll} f(3,2)=35; & f(3,4)=35; & f(2,4)=25 \\ \quad \text{opt } i=1,\ p_1=15; & \quad \text{opt } i=1,\ p_1=15; & \quad \text{opt } i=1,\ p_1=15 \\ f(5,2)=40; & f(5,6)=48; & f(2,7)=50 \\ \quad \text{opt } i=1,\ p_1=15; & \quad \text{opt } i=1,\ p_1=23; & \quad \text{opt } i=3,\ p_3=0 \\ f(2,6)=20; & f(4,7)=64; & f(6,7)=64 \\ \quad \text{opt } i=1,\ p_1=10; & \quad \text{opt } i=2,\ p_2=4; & \quad \text{opt } i=2,\ p_2=9. \end{array}$$

Hence earliest arrival at

$$7=\min\begin{cases} f(4,7)+p_4+t_{47}=64+6+30=100 \\ f(2,7)+p_2+t_{27}=50+0+30=80 \\ f(6,7)+p_6+t_{67}=64+8+10=82 \end{cases}$$

$$=80, \qquad \text{opt } i=2,\ p_2=0.$$

Hence the best route is 1–3–2–7 with the time schedule given in Table III Part (a).

If the permissible parking period at source node 1 is restricted to time 0–12, instead of free parking, the earliest arrival time is increased to 82. The corresponding path is 1–2–6–7 given in part (b) of Table III.

Note that by delaying the departure from node 1 until 12, another solution is given in part (c). Hence there may be many solutions to the problem since one can delay the time of departure from node 1 for any time from 10 to 12 and still arrive at node 7 at time 82.

TABLE III

Earliest Arrival Paths

	Node	Arrival time	Departure time	Lt. of parking period
(a)	1	0	15	15
	3	35	35	—
	2	50	50	—
	7	80	—	—
(b)	1	0	10	10
	2	20	29	9
	6	64	72	8
	7	82	—	—
(c)	1	0	12	12
	2	22	29	7
	6	64	72	8
	7	82	—	—

CONCLUSION

The result of Halpern and Priess [3] gives a Dykstras-type solution to the problem and it requires 3–4 times more computer time than Dykstra's procedure. However, their results just give one optimal solution. In the method presented the results are similar to the Bellman–Ford algorithm [4], but sightly more computer time is required because of all the constraints to be satisfied. However, as shown in the worked example, by delaying the departure time from the source node it may be possible to obtain different optimal solutions. It may sometimes be required to find the earliest and latest arrival at each node for the same problem, in which case all possible solutions would have to be obtained. This will be discussed at a later time.

REFERENCES

1. S. E. DREYFUS, An appraisal of some shortest-path algorithms, *Oper. Res.* **17** (1969), 395–412.
2. M. DESROCHERS AND F. A. SOUMIS, Reoptimization algorithm for the shortest path problem with time windows, *European J. Oper. Res.* **35** (1988), 242–254.
3. J. HALPERN AND I. PRIESS, Shortest path with time constraints on movement and parking, *Networks* **4** (1974), 241–253.
4. L. LAWLER, "Combinatorial Optimization," Holt, Reinhart & Winston, New York, 1976.