CONTINUOUS-TIME SHORTEST PATH PROBLEMS WITH STOPPING AND STARTING COSTS

A. B. PHILPOTT
Department of Engineering Science, University of Auckland, New Zealand

A. I. MEES
Department of Mathematics, University of Western Australia
Nedlands, Australia 6009

(Received April 1992)

Abstract—We describe a general solution method for the problem of finding the shortest path between two vertices of a graph in which each edge has some transit time, costs can vary with time, and stopping and parking (with corresponding costs) are allowed at the vertices.

1. INTRODUCTION

Although least-cost path problems are nowadays considered routine, there has been relatively little work done on the increasingly important case where costs may vary with time and parking is allowed, so that it may be possible to delay transit at various points in order to reduce the overall cost. In this paper, we show how to compute a minimum-cost path for a vehicle traversing the edges of a graph in which each edge has a time-varying cost and a time-varying transit time, and parking with some penalty is allowed at the vertices; stopping and restarting the vehicle incur (possibly different) time-varying costs, and themselves take possibly nonzero lengths of time. Given that the vehicle begins in a specified origin vertex at rest, we seek a set of routing decisions which will allow it to travel through the graph so as to arrive at rest at a specified destination vertex at least travel cost.

A specific example of such a model arises in the routing of trains in a railway network [1], where it is necessary for trains to wait while tracks clear, but the cost of stopping and restarting has to be taken into account, and it may be better to stop for a single longer period than for several shorter periods. Similar situations occur in determining routing strategies in packet switching networks. Here, there are again variations in load over time, and a possibility of storage of packets at nodes for later transmission.

A recent paper by Orda and Rom [2] gives an algorithm and a convergence result for a problem very similar to the one we present, making it possible to omit many details here. The main difference between our work and theirs is that we allow stopping and starting costs and times.

The class of shortest path problems we wish to consider can be posed formally as follows. Given a graph with n vertices, if one leaves vertex j for vertex k at time t, let the transit time between these vertices be denoted by \( d_{jk}(t) \), and let the cost of traversing the corresponding edge be \( c_{jk}(t) \). Suppose for each vertex j that a vehicle arriving at j at time t and stopping will take time \( a_j(t) \) to stop, thus incurring a cost of \( u_j(t) \). It may then park for some time interval, at a cost per unit time of \( s_j(t) \). When it leaves from a park at time \( t \), then suppose it takes \( b_j(t) \) time units to depart, incurring a cost of \( v_j(t) \). A vehicle arriving at j at t may, of course, elect not to stop and may therefore pass through j at t at no cost. We assume that transit costs, parking

We would like to acknowledge K.L. Teo for discussions relating to this paper, and the support of an ARC Small Grant.
costs, and stopping and starting costs are all non-negative, and that the functions \( a, b, \) and \( d \) are continuous and non-negative.

We define a vertex-time pair (VTP) to be a member of \( \{1, 2, \ldots, n\} \times [0, \infty) \). Given an origin vertex 1 and a destination vertex \( n \), a continuous-time path from \((1, 0)\) to \((n, T)\) is a finite sequence of VTPs

\[
(1,0) = (i_0, t_0), \quad (i_1, t_1), \ldots, (i_p, t_p) = (n, T),
\]

in which either \( i_j \neq i_{j+1} \), in which case traffic leaves vertex \( i_j \) for vertex \( i_{j+1} \) at time \( t_j \) and arrives at \( i_{j+1} \) at time \( t_{j+1} = t_j + d_{ij,i_{j+1}}(t_j) \), or \( i_j \) is a member of a subsequence \( S_j \) of at most four consecutive VTPs of the path, all of which have \( i_j \) as the same first element. If \( j \neq 0, p \) then \( S_j \) contains exactly four VTPs with second components, say \( \tau_1, \tau_2, \tau_3, \tau_4 \), defined to be the following times:

\[
\tau_1: \text{time when stopping begins}
\]

\[
\tau_2: \text{time when parking begins} = \tau_1 + a_j(\tau_1)
\]

\[
\tau_3: \text{time when restarting begins}
\]

\[
\tau_4: \text{time when we depart from the vertex} = \tau_3 + b_j(\tau_3).
\]

We denote by \( S \) the set of all such subsequences in the path. Observe that for \( S_0 \), we have \( \tau_1 = \tau_2 = 0 \), and \( u_1(0) = 0 \), and for \( S_p \), we have \( \tau_3 = \tau_4 = T \), and \( v_1(t) = 0 \).

The cardinality of a continuous-time path \( P \) is defined as the number of times that \( i_j \neq i_{j+1} \) in \( P \). The cost of a continuous-time path is defined by

\[
C(P) = \sum_{i_j \neq i_{j+1}} c_{i_j,i_{j+1}}(t_j) + \sum_{s} \left\{ u_{i_j}(\tau_1) + \int_{\tau_2}^{\tau_3} s_{i_j}(t) \, dt + v_{i_j}(\tau_3) \right\}.
\]

The continuous-time shortest path problem \( \text{BP}(T) \) seeks a continuous-time path which minimises \( C(P) \) over all paths from \((1, 0)\) to \((n, T)\). We denote by \( \text{CSP}(\infty) \) the problem of determining a continuous-time path which minimises \( C(P) \) over \( \{P(T) : P(T) \text{ solves } \text{CSP}(T), T \geq 0\} \).

Problems similar to \( \text{BP}(T) \) and \( \text{CSP}(\infty) \) have been discussed by a number of authors, most of whom treat the case where the edge distances are the same as the (possibly time-varying) transit times, and a path is sought which minimises the arrival time at the destination vertex (see [3–6]). For the case where edge distances and traversal times are distinct, most work prior to [2] has concentrated on the multi-objective problem in which one seeks a Pareto optimal path with respect to time, distance, and possibly other objectives. Continuous-time dynamic programming procedures are presented in [7] for this problem in the absence of parking, under some monotonicity assumptions regarding transit times and edge distances.

We proceed to give an algorithm for \( \text{CSP}(T) \) as posed above and use the approach of [2] to derive conditions under which this algorithm converges. We leave the time-complexity of the algorithm (when the edge distances are assumed to be piecewise linear and the parking penalties are assumed to be piecewise constant) and a numerical example to a subsequent paper.

2. THE CSP ALGORITHM

As posed above, the problem \( \text{CSP}(\infty) \) is formulated in an interval with infinite length. As observed in [2], this raises the possibility of an infinite sequence of paths with increasing cardinality and decreasing cost. Indeed, such sequences exist even for \( \text{CSP}(T) \). To ensure that an optimal solution to \( \text{CSP}(\infty) \) exists, we make the following assumptions:

A1: The edge transit times are strictly positive (for all \( t, d_{jk}(t) > 0 \))

A2: For each \( j \), either \( a_j(t) = b_j(t) = 0 \) for all \( t \), or \( a_j(t) > 0 \) for all \( t \), or \( b_j(t) > 0 \), for all \( t \)

A3: There is some \( \epsilon > 0 \), and a time \( \tau_c \), with \( c_{jk}(t), u_j(t), v_j(t) \geq \epsilon \), for all \( t \geq \tau_c \)

A4: For every \( K > 0 \), there is some time \( t \), with \( \int_{\tau_c}^{t} s_j(t) \, dt \geq K \).

Under these assumptions, it is straightforward to show using the argument of [2] that there exists a (finite) path \( P^* \) which solves \( \text{CSP}(\infty) \), and that there is some \( N^* \), a function of \( \tau_c, \epsilon \), and \( C(P^*) \), which gives an upper bound on the cardinality of \( P^* \). Using the continuity of the transit times, it is possible then to construct a time \( T^* \) which gives an upper bound on the arrival time of \( P^* \) at vertex \( n \). Thus, the solution to \( \text{CSP}(\infty) \) may be obtained by solving \( \text{CSP}(T^*) \).
We remark that in most practical applications we lose no generality in assuming the transit times to be strictly positive. In practice, one also is likely to regard the planning period to be some finite interval \([0, T]\). If we know that there is at least one path from \((1, 0)\) to \((n, T)\) then by making the costs \(c_{jk}(t)\), \(u_j(t)\), \(v_j(t)\), and \(s_j(t)\) bounded away from zero for \(t \geq T\), we can ensure that a (finite) path exists which solves \(\text{CSP}(\infty)\). Furthermore, if \(c_{jk}(t)\) is chosen to be large for \(t > T\), we can ensure that this path arrives at \(n\) in the interval \([0, T]\).

We now proceed to formulate an algorithm for \(\text{CSP}(T)\) under the above assumptions. We require the following definitions:

\[
A_j(t) = \{\tau: \tau + a_j(\tau) = t\}, \quad \text{(stopping)},
\]

\[
B_j(t) = \{\tau: \tau + b_j(\tau) = t\}, \quad \text{(starting)},
\]

\[
P_{kj}(t) = \{\tau: \tau + d_{kj}(\tau) = t\}, \quad \text{(in transit)}.
\]

Here, given some time \(t\), \(A_j(t)\) is the set of times that we could initiate a stopping action at vertex \(j\) so as to come to rest at \(t\), \(B_j(t)\) is the set of times that we could initiate a starting action at vertex \(j\) and be moving at \(t\), and \(P_{kj}(t)\) is the set of times at which we can leave vertex \(k\) and arrive at vertex \(j\) at time \(t\).

The algorithm outlined below terminates with a function \(\pi_j(t)\) whose values give the cost of a shortest path from \((1, 0)\) to \((n, t)\) for every \(t \in [0, T]\). The values of \(\pi_j(t)\) on termination give the minimum cost of being in motion at vertex \(j\) at time \(t\), and the values of \(\alpha_j(t)\) on termination give the minimum cost of being in motion at vertex \(j\) at time \(t\) having just arrived from some other vertex. The minimum cost path \(P^*\) is easy to recover from these functions by tracing back through the sequence of VTPs yielding the minimum values.

**CSP Algorithm.** Initialise: Set

\[
\pi_j(t) = \min_{t \in B_j(t)} \left\{ \int_0^t s_1(\tau) d\tau + v_j(t) \right\}, \quad \pi_j(t) = \infty, \quad j \neq 1.
\]

Iterate: Apply the following sequence of steps to every \(j\) until for every \(j\), \(\pi_j(t)\) does not change from one iteration to the next.

1. Set

\[
\alpha_j(t) = \min_{k \neq j} \min_{\tau \in P_{kj}(t)} \{ \pi_k(\tau) + c_{kj}(\tau) \},
\]

2. Set

\[
\lambda_j(t) = \min_{t \in A_j(t)} \{ \alpha_j(t_u) + u_j(t_u) \},
\]

\[
\mu_j(t) = \min_{t \leq t} \left\{ \lambda_j(t_0) + \int_{t_0}^t s_j(\tau) d\tau \right\},
\]

\[
\beta_j(t) = \min_{t \in B_j(t)} \{ \mu_j(t_v) + v_j(t_v) \},
\]

3. Replace \(\pi_j(t)\) by \(\min \{ \pi_j(t), \alpha_j(t), \beta_j(t) \} \).

In order for the minimisation operations in each iteration to be well-defined, we require some continuity conditions on \(c\), \(v\), and \(u\). We assume that each component of these functions is lower semi-continuous. By virtue of the continuity of \(d_{jk}(t)\), the set \(P_{jk}(t)\) is compact, and so the minimum in Step 1 is well defined for each \(j\) and can be easily shown to yield a lower semi-continuous function \(\alpha_j\). (Observe that if \(d_{jk}\) is not continuous then even though \(P_{jk}\) may be compact for each \(t\), we have no guarantee that Step 1 will yield a lower semi-continuous function.) A similar argument shows that in Step 2, each of \(\lambda_j\), \(\mu_j\), and \(\beta_j\) is well-defined and lower semi-continuous. Thus, lower semi-continuity is inherited in each step of the iteration which guarantees that each iteration is well-defined throughout the course of the algorithm.

To show that the CSP Algorithm converges to a solution to \(\text{CSP}(T)\), we appeal to [2, Theorem 2], which establishes the convergence of algorithm Weight described in the same paper. We
observe that the CSP Algorithm applied to a graph $G$ is essentially the same as Weight applied to a graph obtained by adding at most $n$ dummy vertices to $G$ in the following way. For each vertex $j$ in $G$, if $a_j(t) = b_j(t) = 0$ for all $t$, we make the cost of parking in $j$ equal to

$$P_j(\tau, t) = u_j(\tau) + \int_\tau^t s_j(x) \, dx + v_j(t).$$

Otherwise, $P_j(\tau, t) = \infty$ for all $\tau$ and $t$, and $j$ is made adjacent to a corresponding dummy vertex $p(j)$, with $P_{p(j)}(\tau, t) = \int_\tau^t s_j(t_0) \, dt_0$, and transit times $d_{p(j)}(t) = a_j(t)$, and $d_{p(j)}(t) = b_j(t)$, and transit costs $c_{p(j)}(t) = u_j(t)$, and $c_{p(j)}(t) = v_j(t)$. In the cases where our assumptions are equivalent to those in [2], Theorem 2 of this paper can now be applied directly. The only exception occurs when for some $j$ either $a_j$ or $b_j$ is zero at some times in $[0, T]$, thus admitting the possibility of an infinite sequence of updates to $\alpha_j$ and $\alpha_{p(j)}$ in the CSP Algorithm. However, it is easy to see that this cannot occur because $p(j)$ is adjacent only to $j$, and since $a_j + b_j$ is bounded away from zero on $[0, T]$, it follows that $\alpha_{p(j)}$ may be updated only a finite number of times.

References