MINIMUM WEIGHT PATHS
in
TIME-DEPENDENT NETWORKS

Ariel Orda
Raphael Rom

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, Israel 32000

March 1989
(Revised May 1990)

ABSTRACT

We investigate the minimum weight path problem in networks whose link weights and link delays are both functions of time. We demonstrate that in general there exist cases in which no finite path is optimal leading us to define an infinite path (naturally, containing loops) in such a way that the minimum weight problem always has a solution. We also characterize the structure of an infinite optimal path.

In many practical cases, finite optimal paths do exist. We formulate a criterion that guarantees the existence of a finite optimal path and develop an algorithm to find such a path. Some special cases, e.g., optimal loopless paths, are also discussed.

I. INTRODUCTION

Shortest path algorithms have been the subject of extensive research for many years, resulting in a large number of algorithms for various conditions and constraints [1]. While the vast majority of these deal with fixed graphs i.e., fixed topology and fixed link weights, surprisingly few dealt with time dependent graphs.

The quality of a path in a network is affected by several factors. One of them is the traversal cost of links, which is termed *link weight*. Another factor is the *delay* of links namely the time required to traverse the links. A special case of shortest path problems is the minimum-delay problem, in which the weight of a link equals its delay. More generally, in a time-dependent network, the link weight depends on the departure time on the link. Since this in turn depends on the arrival time to the node, it turns out that link delays affect the quality of a path, even when link weights are not directly related to delays. Another interesting property of time-dependent networks is that it may be preferable in some cases to wait (or park) at a node before departing on an outgoing link, in order to keep the total weight low (this may happen if a link weight function decreases during a certain period). On the other hand, parking at a node consumes resources. Hence, a third factor that affects the quality of a path is the cost of parking at a node.

Some works have addressed the problem of finding minimum delay paths in networks with time-dependent link delays. Among the works addressing the problem Klafszky[2] and Cooke & Halsey[3] dealt with discrete delay functions whose domain and range are integers. Arbitrary functions for link delays were briefly treated by Dreyfus[4] and Ling et al.[5] who addressed only limited cases. Halpern[6] dealt with a more general case, in which waiting at nodes may be useful but is allowed only at certain times. The general problem was addressed recently in [7], where algorithms for finding the shortest path under various waiting constraints were presented and investigated. It is shown there that if departure time from the source is unrestricted and delay functions are continuous then a shortest path can be found that is simple and achieves a delay as short as the most unrestricted strategy. On the other hand, it is shown that in more restricted transit, there exist cases where the minimum delay is finite yet the path that achieves it is infinite.

Except for a very brief treatment in [5], time-dependent minimum-weight path problems have not been treated to date. However, such problems are of interest, since the link weight may account for various important factors, such as reliability, transshipment cost, buffer size etc. As is the case with link delays, these factors may vary with time in a way that may be predicted or observed or recorded. Hence the motivation for investigating the problem.

As was noted in [6,7,8], the minimum-delay path problem becomes more complicated in a time-dependent environment (compared to a static one), since in this case such paths may contain loops and even be infinite, and since in general they are not "concatenated" (see [7]). However, as was shown in [7], under quite simple assumptions the problem’s complexity reduces to that of the static one. This is not the case with the general minimum-weight path problem. Here, there is no direct relation between delay and weight, and thus the delay of an optimal path (as well as its length, i.e. number of hops) may grow without bounds. Thus, there is a need to incorporate infinite paths in the analysis of the problem, calling for a rigorous definition of the problem itself. Also, it is worthwhile to identify conditions under which a more tractable solution (e.g., one with finite delay or length) is guaranteed.

The rest of the paper is organized as follows: In section II the time dependent model is formulated. In Section III the concepts of finite and infinite paths are defined, and it is shown that the minimum weight path problem has always a solution in the form of a finite or infinite path. Section IV deals with cases in which a finite path solution is guaranteed, and an algorithm for solving the problem is presented and validated; some special cases for which the algorithm is also applicable are then discussed. Concluding remarks are presented in Section V.

II. THE MODEL

We consider a directed and bidirectional network $G(V, E, W, P, D)$ with $V = \{ 1,2, \ldots, n \}$ being the set of nodes, $E \subseteq V \times V$ the set of links (since G is bidirectional, $(i,k) \in E$ implies $(k,i) \in E$), and W, P, D are sets of *link weights*, *node parking weight densities*, and *link delays*. In the following subsections we dwell on these concepts of weights and delays in more detail.

A. Link and Node Weights

$W = \{ W_{ik}(t) \mid (i, k) \in E \}$ is a set of time-dependent link weights (or costs) i.e., $W_{ik}(t)$ is a nonnegative function of time defined for $[0, \infty)$, describing the cost associated with a commodity leaving node i on link (i, k) at time t. In order to deal with links that are unavailable during some epochs, we let $W_{ik}(t)$ equal infinity in some countable union of open and nonoverlapping intervals of time; outside these intervals $0 \le W_{ik}(t) < \infty$. We make the following assumption regarding link weights (the following holds for all $(i,k) \in E$): for an infinite sequence $\{ t_n \}$ such that $t_n \ge 0$, if $\lim_{n \to \infty} t_n$ and $\lim_{n \to \infty} W_{ik}(t_n)$ exist and are finite, then $W_{ik}(\lim_{n \to \infty} t_n) \le \lim_{n \to \infty} W_{ik}(t_n)$. This assumption holds with equality if $W_{ik}(t)$ is continuous. Also, if it is piecewise continuous, then a sufficient condition for the assumption to hold is that $W_{ik}(t)$ is defined at each noncontinuity point as the minimum of limits from both directions, i.e., $W_{ik}(t) = \min \{ W_{ik}(t^-), W_{ik}(t^+) \}$.

As we indicated before it may sometimes be preferable to wait (or park) at a node before departing on an outgoing link, in order to keep the total weight low. To account for the resources consumed by such parkings we incorporate in our model the set $P = \{ \pi_i(t) \mid i \in V \}$ of *node parking weight densities* representing the infinitesimal cost of parking at node i during an infinitesimally small interval $(t, t+dt)$. We assume that, for each $i \in V$, there is a countable union of open and nonoverlapping intervals, inside which parking is prohibited, and outside of which it is permitted. Thus, parking is permitted in a countable union of closed and nonoverlapping intervals, plus possibly some semi-closed interval $[\hat{t}, \infty)$. $\pi_i(t)$ is nonnegative and integrable inside the permitted intervals of node i, and equals infinity outside them. Given $\pi_i(t)$, we define the *node parking weight function* $P_i(\tau, t)$, for $0 \le \tau \le t < \infty$, as follows: if τ and t belong both to the same permitted interval for node i, then $P_i(\tau, t) \stackrel{\Delta}{=} \int_\tau^t \pi_i(\theta)\, d\theta$; else, $P_i(\tau, t) \stackrel{\Delta}{=} \infty$. The following properties are evident:

- For $0 \le \tau \le t < \infty$: $P_i(\tau, t) \ge 0$;

- $P_i(\tau, t)$ is nondecreasing with respect to t and nonincreasing with respect to τ.
- For $\tau < \tau' < t$ we have: $P_i(\tau, t) = P_i(\tau, \tau') + P_i(\tau', t)$.
- For an infinite sequence $\{ t_n \}$ such that for all n $t_n \geq \tau_o \geq 0$: if $\lim_{n \to \infty} t_n$ and $\lim_{n \to \infty} P_i(\tau_0, t_n)$ exist and are finite, then $P_i(\tau_0, \lim_{n \to \infty} t_n) = \lim_{n \to \infty} P_i(\tau_0, t_n)$.
- For infinite sequences $\{ \tau_n \}$ and $\{ t_n \}$, such that $\forall n$ $t_n \geq \tau_n \geq 0$: if $\lim_{n \to \infty} \tau_n$ and $\lim_{n \to \infty} P_i(\tau_n, t_n)$ exist and are finite then $\lim_{n \to \infty} P_i(\lim_{n \to \infty} \tau_n, t_n) = \lim_{n \to \infty} P_i(\tau_n, t_n)$.
- For infinite sequences $\{ \tau_n \}$ and $\{ t_n \}$ such that $\forall n$ $t_n \geq \tau_n \geq 0$: if $\lim_{n \to \infty} \tau_n$ and $\lim_{n \to \infty} t_n$ exist and are finite and if $\forall n$ $P_i(\tau_n, t_n) < \infty$, then $P_i(\lim_{n \to \infty} \tau_n, \lim_{n \to \infty} t_n) < \infty$.

The above properties shall be used for proving some claims in what follows.

B. Link Delays

$D = \{ d_{ik}(t) \mid (i, k) \in E \}$ is a set of time-dependent link delays i.e., $d_{ik}(t)$ is a continuous and strictly positive function of time defined for $[0, \infty)$ that describes the delay incurred by a commodity that starts traversing link (i, k) at time t. Leaving node i at time t, it arrives at node k at time $t + d_{ik}(t)$. Note that in general non-FIFO behavior of links is possible (i.e., commodities may arrive at the endpoint of a link not in the order they departed).

As was noted in [7], under certain circumstances it is possible for a commodity to traverse a path consisting of an infinite number of links, but arrive at the destination within finite time. The following lemma establishes conditions under which such extreme scenarios are avoided.

Lemma 1: If $\forall\ 0 < T < \infty$, $\exists\ \delta_T > 0$, such that $\forall\ (i, k) \in E$ $\forall\ 0 \leq t \leq T$ $d_{ik}(t) \geq \delta_T$ (i.e., each link delay function has a strictly positive lower bound for every finite interval) then the delay of a path with an infinite number of links is infinite.

Proof: Consider an infinite sequence of nodes $(v_0, v_1, v_2, \cdots)$ where $\forall\ i \geq 0$ $(v_i, v_{i+1}) \in E$. Denote by A_i the arrival time at node v_i, and by D_i the departure time from that node. A_0 denotes the starting time i.e., the moment at which our journey starts. We have that $\forall\ i \geq 0$: $D_i \geq A_i$ and $A_{i+1} = D_i + d_{v_i v_{i+1}}(D_i)$. To prove the lemma it suffices to show that $\forall\ T > 0$, $\exists\ N$, such that $A_N \geq T$. Assuming the contrary let $T' > 0$ be such that $\forall\ N$ $A_N < T'$. According to the assumption, there is a $\delta = \delta_{T'} > 0$ such that $\forall\ 0 \leq t \leq T'$ $d_{ik}(t') \geq \delta$. However, $A_N \geq \sum_{i=0}^{N-1} d_{v_i v_{i+1}}(D_i)$, and $\forall i,\ 0 \leq i < N$:

$0 \le D_i < A_N < T'$, thus $d_{v_i v_{i+1}}(D_i) \ge \delta$. It follows that $A_N \ge N \cdot \delta$, and for $N \ge \frac{T'}{\delta}$ we have a contradiction. □

Since delay functions are continuous and strictly positive, they fulfill the condition of Lemma 1, and thus traversals with finite delay are achieved only through finite length paths.

III. THE MINIMUM WEIGHT PATH PROBLEM

Our aim is to formulate a minimum weight problem, that seeks a path between two nodes with minimal total link- and node-weight. In the case of static shortest path problems, and in most time-dependent minimum-delay path problems, such paths are guaranteed to be of finite length. However, Figure 1 depicts a situation in which the minimum-weight path between nodes 1 and 3 is infinite. Let $W_{12}(t) = W_{21}(t) = 2^{-t}$, $W_{13}(t) = 2^{2-t}$; $d_{12}(t) = d_{21}(t) = d_{13}(t) \equiv 1$; $\pi_1(t) = \pi_2(t) = \pi_3(t) \equiv \infty$. Assume that we depart from node 1 at time $t = 0$. Then, it is easy to verify that if we traverse the loop (1,2), (2,1) an infinite number of times "and then" depart on link (1,3) we get a total weight of 2, whereas for any finite length path the weight is higher. In other words, we get lower weights by staying longer in the network. Thus, in order to define the minimum weight path problem, it is necessary to define the concept of infinite paths and their respective weights.

In this section we find it convenient to deal with parkings that are limited to a unit of time. This can be achieved without loss of generality by using the following transformation: a fictitious link (i,i) is added to each node i with delay $d_{ii}(t) \equiv 1$ and $W_{ii}(t) \overset{\Delta}{=} P_i(t, t+1)$. A new parking weight function $\tilde{P}_i(\tau, t)$ is defined by $\tilde{P}_i(\tau, t) \overset{\Delta}{=} \begin{cases} P_i(\tau, t) & t \le \tau+1 \\ \infty & otherwise \end{cases}$. It is easy to verify that the new representation is equivalent to the original one.

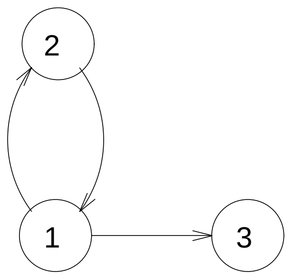

Figure 1: Example for an infinite minimum-weight path

A. Finite and Infinite Paths

We begin with a somewhat uncommon definition of a finite path that suits us better in defining our problem.

Definition 1: Finite path

A finite path fp between a source node s and a destination node r for starting time t_S is a finite sequence of ordered triplets,

$fp = ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_m, \tau_m, t_m))$,

where

1. $v_o = s$.
2. $v_m = r$.
3. $\tau_o = t_S$.
4. $\forall\ 0 \le i \le m-1$: $(v_i, v_{i+1}) \in E$, $t_i \ge \tau_i$, $\tau_{i+1} = t_i + d_{v_i v_{i+1}}(t_i)$; i.e., for $i \ge 0$ t_i is the departure time from node v_i and for $i > 0$ τ_i is the arrival time at that node.
5. $\forall\ 0 \le i \le m$: $[\tau_i, t_i]$ is within a permitted interval for node v_i i.e., $P_{v_i}(\tau_i, t_i) < \infty$.
6. $\forall\ 0 \le i \le m-1$: link (v_i, v_{i+1}) is available at time t_i i.e., $W_{v_i, v_{i+1}}(t_i) < \infty$.

Note that the arrival time at the destination is τ_m, and that t_m, referred to as the *termination time*, denotes an additional stay at the destination node before the travel concludes.

Thus, the definition of a finite path describes the sequence of nodes traversed, the parking periods at the nodes and the delays on links. The number of triplets minus one (equaling the number of edges, or "hops") of a finite path is called the path *length* .

Definition 2: Weight of finite path

The weight $W(fp)$ of a finite path $fp = ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_m, \tau_m, t_m))$ is defined by:

$$W(fp) \triangleq \sum_{i=0}^{m-1} [P_{v_i}(\tau_i, t_i) + W_{v_i v_{i+1}}(t_i)] + P_{v_m}(\tau_m, t_m) \quad .$$

Note that, according to Definition 1, $W(fp) < \infty$. Also, note that to minimize the weight of fp, one should clearly choose $t_m = \tau_m$ i.e., without "extra stays" at the destination. We allow paths to have such extra stays in order to facilitate some proofs.

An assumption is made throughout the paper that between any two nodes in the network and for any starting time $t_S \ge 0$ there exists a finite path; in other words, at all times nodes are reachable from one another through finite paths of finite weights.

We now turn to deal with infinite paths. We intend to define an infinite path as a limit of a sequence of finite paths, so we start by introducing path sequences, as follows. Let $FP = \{ fp_n \}$ be a sequence of finite paths between nodes s and r for starting time t_S, i.e., a sequence each of whose elements is of the form

$$fp_n = ((v_o^{(n)}, \tau_o^{(n)}, t_o^{(n)}), (v_1^{(n)}, \tau_1^{(n)}, t_1^{(n)}), \ldots, (v_{l(n)}^{(n)}, \tau_{l(n)}^{(n)}, t_{l(n)}^{(n)}))$$

$$v_0^{(n)} = s, \quad v_{l(n)}^{(n)} = r, \quad \tau_o^{(n)} = t_S$$

Here $l(n)$ denotes the path length of the n-th path in the sequence.

Definition 3: Invariance

Suppose that $FP = \{ fp_n \}_{n=0}^{\infty}$ is an infinite sequence of finite paths between nodes s and r for starting time t_S. Suppose also that the paths of FP are of increasing length, i.e., $\forall n \quad l(n+1) > l(n) \geq 1$. Finally, suppose that $\forall\, n \geq j$: $(v_j^{(n)}, \tau_j^{(n)}, t_j^{(n)}) \equiv (v_j, \tau_j, t_j)$ i.e., all paths of FP except maybe for the first j have the same triplet in the jth position; if this property holds for all $j \geq 0$ then FP is said to be *invariant* .

For each such sequence there is a rule (referred to as the rule of production) specifying the way to construct the elements of the sequence. As an example, consider again Figure 1. We have seen that additional traversals of the loop $(1,2)$, $(2,1)$ cause a reduction in the total weight incurred. Consider then the sequence of paths $FP = \{ fp_n \}_{n=0}^{\infty}$, where fp_n is generated in the following recursive way:

$$fp_0 = ((1,0,0),(3,1,1))$$

$$\forall\, n \geq 0: fp_{n+1} = fp_n^- \,\&\, (2,2n+1,2n+1) \,\&\, (1,2n+2,2n+2) \,\&\, (3,2n+3,2n+3),$$

where superscript "-" denotes the deletion of the last triplet from a path and "&" denotes appending a triplet to the end of a path (these operations are performed sequentially from left to right). In other words

$fp_1 = ((1,0,0), (2,1,1), (1,2,2), (3,3,3))$,
$fp_2 = ((1,0,0), (2,1,1), (1,2,2), (2,3,3), (1,4,4), (3,5,5))$,
etc.

It is evident that FP is invariant.

Definition 4: Infinite path

The infinite path ip, corresponding to an invariant sequence of finite paths $FP = \{ fp_n \}$ for s, r, t_S, is the infinite sequence of triplets resulting from an infinite application of the rule of production of FP.

Note that since FP is invariant, then for all $n \geq 0$ the first n triplets of ip are equal to those of fp_n. Thus, one can build the prefix of ip for any number of triplets, given the rule of production of FP. Considering again Figure 1 and the sequence FP defined above we note that its corresponding ip stands for an infinite traversal of the loop $(1,2)$, $(2,1)$ and "then" a departure on link $(1,3)$. In other words ip denotes the optimal way to traverse from node 1 to node 3 for starting time $t_S = 0$.

Definition 5: Weight of an infinite path

Let $\{ W_n \}$ be the sequence defined by $W_n = W(fp_n)$. The weight of an infinite path ip corresponding to FP is given by

$$W(ip) = \liminf_{n\to\infty} W(fp_n)$$

(the limit is in the broad sense i.e., $+\infty$ is also included).

Since $W_n = W(fp_n) \geq 0$, the sequence $\{ W_n \}$ is bounded from below (by 0), it follows that $\liminf_{n\to\infty} W_n$ exists and thus $W(ip)$ is well defined. Because the limit exists, there is some sub-sequence $\{ fp_{n_k} \}$ of FP such that $\lim_{k\to\infty} W(fp_{k_n}) = W(ip)$ (limit in the broad sense). Thus (according to infinitesimal calculus), we may delete elements from FP and get a new sequence $FP' = \{ fp'_n \} \subseteq FP$ such that $W(ip) = \lim_{n\to\infty} W(fp'_n)$. We call a sequence such as FP' *proper* and (without loss of generality) from now on assume that all infinite paths correspond to proper sequences of finite paths.

Definition 6: General path

A general path (or just path) between nodes s and r for starting time t_S is either a finite or an infinite path between s and r for t_S.

We denote by $\Pi(s, r, t_S)$ the set of paths between nodes s and r for starting time t_S.

To facilitate the presentation of the next section we shall have to introduce several more terms whose meaning will become clear when they are used. However, in order to keep all definitions in one place these are presented next. (On first passage the reader can skip to sub-section B).

Definition 7: t-termination path

A path $p \in \Pi(s,k,t_S)$ whose termination time at node k is t is referred to as a t-termination path for s,k,t_S.

Definition 8: Hop distance

The length of the minimal-hop path between nodes s and k is denoted $HD(k)$ and is referred to as the hop-distance between these nodes.

Definition 9: Earliest arrival time

The earliest possible arrival time at node $k \in V$ on any of the paths of $\Pi(s,k,t_S)$ whose length is at most N is denotes $EA_N(k)$.

Definition 10: Optimal t-termination path

Assume that for $k \in V$ there exist t-termination paths between nodes s and k. Among these, one with minimum weight is called an optimal t-termination path and its weight is denoted by $MW_k(t)$.

Finally, we denote by $MH_k(t)$ the smallest number of hops of all optimal t-termination paths for s, k, t_S.

B. Time-Dependent Minimum Weight Path (TMWP) Problem Formulation

Problem TMWP: For a network $G(V, E, W, P, D)$, a source node $s \in V$, a destination node $r \in V$ and a starting time $t_S \in [0, \infty)$, find a path $p^* \in \Pi(s, r, t_S)$ such that $\forall\, p \in \Pi(s, r, t_S)$: $W(p^*) \le W(p)$. In other words, we seek a path p^* whose weight is minimal among the set $\Pi(s, r, t_S)$. p^* is referred to as a *minimum weight path* and its weight is referred to as the minimum weight.

Hidden in this formulation is the fact that $\Pi(s, r, t_S)$ may have an infinite number of elements, and even for the restricted case in which parking is prohibited its number of elements may not be countable. To demonstrate this consider the network of Figure 2. If parking is prohibited, a path is completely defined by the sequence of nodes traversed, and the starting time. Attach to the loop (s, v, s) the label '0', and to the loop (s, l, s) the label '1'. Then, there is a 1-1 correspondence between the (finite and infinite) paths of $\Pi(s, r, t_S)$ and the (finite and infinite) binary sequences; for example, the path composed of the sequence of nodes (s, v, s, r) corresponds to '0', that composed of (s, v, s, l, s, r) to '01', etc. Since the cardinality of the set of binary sequences is equal to that of the continuum, it follows that $\Pi(s, r, t_S)$ is noncountable. Hence, the existence of a minimal path as described in Problem TMWP should be proved rigorously.

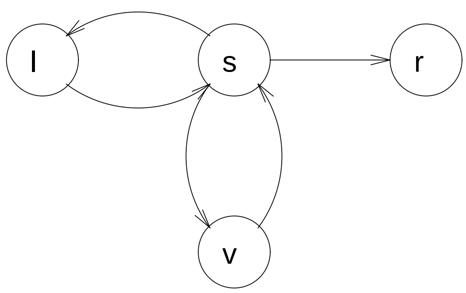

Figure 2: Example of an uncountable number of paths

Theorem 1: For s, r, t_S, there is a $p^* \in \Pi(s, r, t_S)$ such that $\forall\ p \in \Pi(s, r, t_S):\ W(p^*) \leq W(p)$.

Proof: See Appendix A.

□

From the theorem it follows that Problem TMWP is well defined. The theorem also shows that when the optimal path is infinite, one can build it to any desired length by using the rule of production of the corresponding finite path sequence. However, this result is only conceptual, since it seems impossible to build an algorithm for finding that rule in a general case. In the next section we deal with the solution of the problem for the simpler and more practical case of finite paths.

IV. AN ALGORITHM FOR FINITE PATHS

Although we have seen that for some cases the minimum weight path may be infinite, for a large class of practical problems a finite path solution, with an a priori known upper bound on the number of edges, is guaranteed. In this section we concentrate on this class of problems. First, we present a criterion that guarantees the existence of a finite solution, and demonstrate its generality for a number of practical examples.

A. Finiteness Criterion

Lemma 2: If (i) $\exists\ \eta > 0$, $T \geq 0$ such that $\forall\ i, k, t \geq T$ holds $W_{ik}(t) \geq \eta$ (i.e., for $t \geq T$ link weights have a strictly positive lower bound), and (ii) $\forall\ i, \tau\ \lim_{t \to \infty} P_i(\tau, t) = \infty$ (i.e., the cost of parking grows without bound with increasing time of parking), then there is a finite path fp^* with finite parking times at all nodes that solves problem TMWP (i.e., $\forall\ p \in \Pi(s, r, t_S)\ W(fp^*) \leq W(p)$).

Proof. By adding self loops, transform the given network to one in which parking is restricted to a single unit of time (as explained in section III). We refer to the original links as regular links and to the newly added ones as fictitious. According to Theorem 1 there is some path $p^* \in \Pi(s, r, t_S)$ that solves problem TMWP. Assume that p^* is infinite. Then, since delay functions are continuous, Lemma 1 holds and it follows that the arrival time at the destination for path p^* is infinite. Also, it follows from that lemma that there is only a finite number of triplets in p^* for which the corresponding times are not greater than T. Consider then the infinite sequence of triplets in p^* whose corresponding arrival times are greater than T. Each successive pair of triplets stands for a traversal of some link; because each traversal of a regular link after time T incurs a weight of at least η and because $W(p^*) < \infty$, it follows that the number of triplet pairs that stands for regular link traversals is finite. We thus conclude that there is an infinite sequence of triplets in p^* that stand for an infinite successive traversal of some fictitious link (i, i). Because $d_{i,i}(t) = 1$ it will take infinitely long to traverse these links, which, in the original network, means infinite parking at node i which cannot be done in a finite cost. p^* must therefore be finite. □

The Finiteness Criterion presented in lemma 2 applies in a large number of practical problems such as the following.

- *Restricted Destination Arrival Time:* Suppose that there is some bound T, $t_S < T < \infty$, on the arrival time at the destination i.e., we are not interested in paths whose arrival time is later, even if their weights are smaller. Defining for all i and for $t > T$ $P_i(\tau, t) \overset{\Delta}{=} \infty$, and for all i, k and for all t for which $t + d_{ik}(t) > T$ $W_{ik}(t) \overset{\Delta}{=} \infty$, we accommodate the Finiteness Criterion for this case.

- *Restricted Length with Bounded Parking:* Suppose that there are bounds on the length (i.e., number of triplets) of the requested path and on the maximal allowable parking time at any node. Since link delays are always finite it is easy to see that this is a special case of the former and thus the Finiteness Criterion can be applied to it as well.
- *Network with Transient Epochs:* Assume that the dynamic behavior of the network is transient, so that for some $T>0$ we have that $\forall\ i,k,t>T\ \ W_{ik}(t)\equiv W_{ik}>0$. Since parking at nodes is of no benefit for $t\geq T$, it turns out that the Finiteness Criterion holds for this case as well. If infinite parkings are excluded i.e., if $\forall\ i,\tau\ \ P_i(\tau,t)\underset{t\to\infty}{\rightarrow}\infty$, then the above holds also for asymptotic convergence, that is, for the case that $\forall\ \varepsilon\geq 0,\ \exists\ T>0$, such that $\forall\ i,k,t>T$: $|W_{ik}(t)-W_{ik}|<\varepsilon$.

An important property of a network for which the Finiteness Criterion holds is that the time domain of interest is finite. Its extent can be calculated in the following way.

As we have seen, the Finiteness Criterion gives an upper bound for the length of optimal paths. An upper bound for parking times in such paths is also easily obtained. The above, together with the fact that delay functions are bounded in finite domains, enables a recursive calculation of upper bounds for arrival and departure times in optimal paths. This reasoning is now discussed formally.

Let N be such that the arrival times at the destination of paths with at least N triplets is not before T. Let W' be the weight of some path from s to r (for starting time t_S). Since $\exists\ \eta>0$ such that $\forall\ i,k,t\geq T$ holds $W_{ik}(t)\geq\eta$, it follows that $N^*=N+\lceil\frac{W'}{\eta}\rceil$ is an upper bound on the length of an optimal path. Since $\forall\ i,\tau\ \lim_{t\to\infty}P_i(\tau,t)=\infty$, then $\forall\ \tau,\ \exists\ \theta(\tau)>0$ such that $\forall\ i,t\geq\tau+\theta(\tau)\ \ P_i(\tau,t)>W'$, and we denote by $\delta(\tau)$ the infimal such value for each τ. Denote $\Delta(t)\stackrel{\Delta}{=}\max_{\substack{(i,k)\in E\\ \tau\in[t_s,t]}} d_{ik}(\tau)$. We obtain an upper bound t^* on the arrival time at the destination of an optimal path by the following recursive procedure:

1. $\hat{\tau}_0=t_S$.
2. $\forall\ 0\leq n\leq N^*-1\ \ \hat{\tau}_{n+1}=\hat{\tau}_n+\delta(\hat{\tau}_n)+\Delta(\hat{\tau}_n+\delta(\hat{\tau}_n))$.

$\hat{\tau}_n$ is the latest possible arrival time to the n'th node of an optimal path. We start with the trivial assignment to $\hat{\tau}_0$. Then, having calculated $\hat{\tau}_n$, we calculate the latest possible departure time from the n'th node by adding to $\hat{\tau}_n$ the maximal possible parking time $\delta(\hat{\tau}_n)$; we then obtain $\hat{\tau}_{n+1}$ by adding to this sum the maximal possible delay on a link for a departure which is not later than $\hat{\tau}_n+\delta(\hat{\tau}_n)$. This discussion is concluded by the following lemma.

Lemma 3: $t^* \overset{\Delta}{=} \hat{\tau}_{N^*}$ is an upper bound on the arrival time at the destination of an optimal path.

Proof: The proof is rather straightforward and appears in [9] (an extended version of this paper). □

Thus, when the Finiteness Criterion holds, we can restrict our attention to the domain $[0, t^*]$. In the following subsection we present an algorithm for solving problem TMWP for a network that satisfies that criterion.

B. The Algorithm

We present an algorithm to solve problem TMWP for a graph $G(V, E, W, P, D)$, a source node s, a destination node r and a starting time t_S. It is assumed that an upper bound t^* on arrivals is precalculated, as explained in the previous subsection.

There is a clear distinction between our algorithm and shortest path algorithms for the static case: In the latter case, one can rely on the "concatenation" property of shortest paths namely that any subpath of a shortest path is also a shortest path; in the time-dependent case, however, a shortest path may well be non-concatenated, as depicted in Figure 3: let all delay functions be identically 1 and all parking functions equal ∞; let $W_{si}(t) = W_{sl}(t) = W_{li}(t) \equiv 1$ and $W_{ir}(t) = 3/t$. Then, for starting time $t_S = 0$, the minimum weight path between s and i is (s, i), whereas the one for s and r is (s, l, i, r). Nonetheless, minimum weight paths in the time-dependent case have a different concatenation property, as shall be described in the next section, of which the algorithm makes use.

Next we present a specification of the algorithm, and then prove its main properties. For brevity we divide the algorithm into two: the first and main one, called Algorithm Weight, computes the weight of an optimal path while the second (and much simpler one), called Algorithm Path, calculates the path itself.

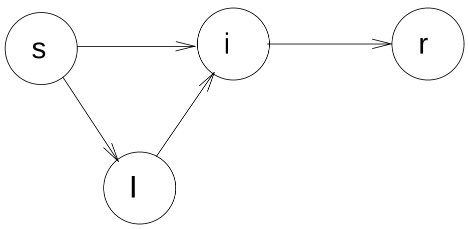

Figure 3: Example of a nonconcatenated path

1. Specification

Algorithm Weight finds the weight of an optimal path between node s and each other node in $G(V, E, W, P, D)$ for starting time t_S. Although we restrict our attention to a solution for a single starting time, it should be noted that the algorithm can be used to find the minimum weight function (for each destination node) for the whole range of $t_S \in [0, \infty)$ – this is done by considering the functions introduced by the algorithm (see below) as being two-variable functions.

Notably different from typical algorithms, our algorithms operate on functions i.e., assignment, addition, or any other operation is performed on entire functions. In particular, we need to be able to detect functions that change during the execution of an algorithm step. We use the term "just changed" to mean that there exists at least one time instance t for which a function changed its value in the most recent operation performed on that function.

Algorithm Weight makes use of two internal functions of time defined for $[t_S, t^*]$ as follows:

- For each node $k \in V$, $X_k(t)$ is the lowest known weight (at some stage of algorithm execution) of a t-termination path between s and k (for starting time t_S).
- For each pair of neighbors k, l $Y_{kl}(t)$ is the lowest known weight (at that stage of algorithm execution) of a t-termination path between s and l (for starting time t_S) for which the last hop is on link (k, l), and arrival to node l is at time t.

Also, the algorithm makes use of the variable X_k (for each $k \in V$) to denote the best known weight (at that stage of algorithm execution) of a path between s and k for starting time t_S. Finally, we denote by N_k the set of neighbors of node k, i.e., $N_k \triangleq \{ l \mid (k, l) \in E \}$.

Algorithm Weight

1. (a) $\forall\, k \in V \;\; k \neq s$: $X_k(t) \leftarrow \infty$, $\forall\, l \in N_k \;\; Y_{kl}(t) \leftarrow \infty$.

   (b) $X_s(t) \leftarrow P_s(t_S, t)$, $\forall l \in N_s$: $Y_{sl}(t) \leftarrow \min\limits_{\tau,\, \tau + d_{sl}(\tau) = t} (X_s(\tau) + W_{sl}(\tau))$

2. $\forall\, (k, l) \in E$:

   $X_l(t) \leftarrow \min\limits_{\tau \le t, k \in N_l} [Y_{kl}(\tau) + P_l(\tau, t)]$

3. $\forall\, l \in V$:

   (a) $X_l \leftarrow \min\limits_{t_S \le \tau \le t^*} (X_l(\tau))$

   (b) $\forall\, k \in N_l \quad Y_{lk}(t) \leftarrow \min\limits_{\tau,\, \tau + d_{lk}(\tau) = t} (X_l(\tau) + W_{lk}(\tau))$

4. If for some $(l, k) \in E$ $Y_{lk}(t)$ just changed then go to step 2; else stop.

Some explanation of the computation is called for. In step 1(a) we assume that all minimal weights are infinite; in step 1(b) we calculate the minimal weights of direct paths from s to its neighbors. In step 2 we try to improve the values of $X_l(t)$ by using a possibly better t-termination path through neighbor k. In step 3 we use the improved values of $X_l(t)$ to improve the value of X_l and also in order to try to find a better path for each neighbor of k. In steps 1.b, 2 and 3 a minimization is performed. In Appendix B it is shown that the corresponding minima exist and thus those operations are valid. Note that in steps 1.b and 3.b the minimization is over some set $\{ \tau | \tau + d_{lk}(\tau) = t \}$. That set is empty if t is smaller than $\min_{t_s \leq \tau} [\tau + d_{lk}(\tau)]$; for such values of t we regard the "min" operation as returning ∞. From the continuity of delay functions it follows that for larger values of t that set is nonempty. Finally, we point out that the computational complexity of function operations, such as those of Algorithm Weight, is clearly higher than that of simple operations (on numbers). The reader is referred to [7] for a discussion on that issue (the above statement is especially true regarding the "min" operations). As is the case for other algorithms that work on functions (e.g., [10]), the complexity of such operations depends heavily on the "well behaving" nature of the functions considered.

Having computed the functions $X_k(t)$ and $Y_{kl}(t)$ and the values X_k, Algorithm Path calculates the corresponding path for destination node r, i.e., it finds a path between s and r whose weight (for starting time t_S) is X_r. Algorithm Path may be used for any destination node $k \in V$. In the description we use the operator "&" to signify concatenation i.e., prepending a component to a sequence.

Algorithm Path

1. $k \leftarrow r$, $p \leftarrow \emptyset$, choose a $t \in [t_S, t^*]$ such that $X_r = X_r(t)$.
2. If $k = s$ and $X_s(t) = P_s(t_S, t)$ then
   a. $p \leftarrow (s, t_S, t)\ \&\ p$.
   b. Stop.
3. a. Find a node l and a time $\tau < t$ such that $X_k(t) = Y_{lk}(\tau + d_{lk}(\tau)) + P_k(\tau + d_{lk}(\tau), t)$.
   b. $p \leftarrow (k, \tau + d_{lk}(\tau), t)\ \&\ p$.
   c. $k \leftarrow l$, $t \leftarrow \tau$.
   d. Go to step 2.

The algorithm is actually trivial. It builds the path backwards starting with an empty sequence. We know the arrival time at the destination of the optimal path. This is a time τ for which $X_r = X_r(\tau)$. Given that at time t the path reaches node k we look, in step 3, for a node l which could serve as the predecessor in the required path and a departure time τ from that node (step 3.a). Having identified these, we prepend the component $(k, \tau + d_{lk}(\tau), t)$ to the path. Step 2 assures termination and the addition of the first component to the sequence.

2. Validation

Observing the execution of Algorithm Weight we note that it executes step 1 once and then steps 2 through 4 repeatedly. We refer to each passage through step 2 as an *iteration* of the algorithm, and the execution of step 1 (the initialization) as the zeroth iteration.

From the structure of Algorithm Weight it is clear that at any stage of execution, for all $k \in V$ and each time $\hat{t} \in [t_S, t^*]$ if $X_k(\hat{t}) < \infty$ then there is at least one path

$$p = ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_m, \tau_m, t_m))$$

such that $v_o = s$, $\tau_o = t_S$, $v_m = k$, $t_m = \hat{t}$, whose weight equals $X_k(\hat{t})$. In other words, there is a $\hat{t}$-termination path from s to k for starting time t_S, whose weight is $X_k(\hat{t})$.

Shortest paths in static networks have the property that every subpath is a shortest path itself; this is the famous "concatenation" property. The following lemma demonstrates the existence of a similar property for the minimum weight path in dynamic networks, but in a much more restricted sense.

Lemma 4: For a time $\hat{t} \in [0, \infty)$ and a node $j \in V$ with $HD(j) \le M$ and $EA_M(j) \le \hat{t}$, let $p_m = ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_m, \tau_m, t_m))$ (where $m \le M$, $v_o = s$, $v_m = j$, $\tau_o = t_S$, and $t_m = \hat{t}$) be an optimal $\hat{t}$-termination path between s and j (for starting time t_S), among all paths of $\Pi(s, j, t_S)$ of length at most M. Then, for all i, $1 \le i \le m-1$, $p_i \stackrel{\Delta}{=} ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_i, \tau_i, t_i))$ is an optimal t_i-termination path between s and v_i (for starting time t_S), among all paths of $\Pi(s, v_i, t_S)$ of length at most i.

Proof: Let $p'_i \stackrel{\Delta}{=} ((v'_o, \tau'_o, t'_o), (v'_1, \tau'_1, t'_1), \ldots, (v'_l, \tau'_l, t'_l))$, where $v'_o = s$, $v'_l = v_i$, $\tau'_o = t_S$, $t'_l = t_i$ and $l \le i$, be a path such that $W(p'_i) < W(p_i)$, i.e., p'_i contradicts the lemma. Let

$$p'_m \stackrel{\Delta}{=} ((v'_o, \tau'_o, t'_o), (v'_1, \tau'_1, t'_1), \ldots, (v'_l = v_i, \tau'_l, t'_l = t_i), (v_{i+1}, \tau_{i+1}, t_{i+1}), \ldots, (v_m, \tau_m, t_m)) .$$

We have:

$$W(p'_m) = \sum_{q=0}^{l-1}[P_{v'_q}(\tau'_q , t'_q)+W_{v'_q , v'_{q+1}}(t'_q)]+P_{v'_l}(\tau'_l , t_i)+\sum_{q=i}^{m-1}[W_{v_q , v_{q+1}}(t_q)+P_{v_{q+1}}(\tau_{q+1} , t_{q+1})] =$$

$$= W(p'_i) + \sum_{q=i}^{m-1}[W_{v_q , v_{q+1}}(t_q) + P_{v_{q+1}}(\tau_{q+1} , t_{q+1})] <$$

$$< W(p_i) + \sum_{q=i}^{m-1}[W_{v_q , v_{q+1}}(t_q) + P_{v_{q+1}}(\tau_{q+1} , t_{q+1})] = W(p_m)$$

which is a contradiction. □

In the following lemmas, when making reference to the "Nth iteration" of the algorithm it is implicitly assumed that such an iteration is indeed performed before the algorithm stops.

Lemma 5: After N iterations of the algorithm, for every node $i \in V$ with $HD(j) \le N$ and for any time $\hat{t} \in [EA_N(j) , t^*]$, $X_j(\hat{t})$ is not higher than the weight of an optimal path between s and j (for starting time t_S), among all $\hat{t}$-termination paths of $\Pi(s , j , t_S)$ of length at most N.

Proof: We begin by noting that (for all k , l) the functions $X_k(t)$, $Y_{kl}(t)$ cannot increase for any value of t from one iteration to the next. We prove the lemma by induction on N. The claim is clearly true for $N = 0$. Assuming truth for $N - 1$ we prove for N. Let j be a node for which $HD(j) \le N$ and $\hat{t}$ be a time such that $EA_N(j) \le \hat{t} \le t^*$. Let $p_N = ((v_o , \tau_o , t_o) , (v_1 , \tau_1 , t_1) , \ldots , (v_m , \tau_m , t_m))$ (where $v_o = s$, $v_m = j$, $\tau_o = t_S$, $t_m = \hat{t}$ and $m \le N$) be an optimal path between s and j for starting time t_S, among all $\hat{t}$-termination paths of $\Pi(s , j , t_S)$ of length at most N. $p_{N-1} \triangleq ((v_o , \tau_o , t_o) , (v_1 , \tau_1 , t_1) , \ldots , (v_{m-1} , \tau_{m-1} , t_{m-1}))$ is a t_{m-1}-termination path for s , v_{m-1} , t_S whose length is $m-1$. From the inductive assumption it follows that after $N-1$ iterations $X_{v_{m-1}}(t_{m-1}) \le W(p_{N-1})$. Thus, after executing step 3(b) at the $N-1$-st iteration we have

$$Y_{v_{m-1} v_m}(\tau_m) \le X_{v_{m-1}}(t_{m-1}) + W_{v_{m-1} v_m}(t_{m-1}) \le$$

$$\le W(p_{N-1}) + W_{v_{m-1} v_m}(t_{m-1}) \quad .$$

Thus, after executing step 2 at the N-th iteration we have

$$X_{v_m}(\hat{t}) = X_{v_m}(t_m) \le Y_{v_{m-1}v_m}(\tau_m) + P_{v_m}(\tau_m , t_m) \le W(p_{N-1}) + W_{v_{m-1} , v_m}(t_{m-1}) + P_{v_m}(\tau_m , t_m) = W(p_N) ,$$

completing the inductive step. □

Lemma 6: Suppose that for $j \in V$ and $\hat{t} \in [t_S , t^*]$ we have $EA_N(j) \le \hat{t}$ and $MH_j(\hat{t}) = N$; then, after the N-th iteration we have $X_j(\hat{t}) = MW_j(\hat{t})$, but prior to that iteration $X_j(\hat{t}) > MW_j(\hat{t})$.

Proof: Since $N = MH_j(\hat{t}) \geq HD(j)$, it follows from Lemma 5 that after the N-th iteration $X_j(\hat{t}) \leq MW_j(\hat{t})$. On the other hand, to $X_j(\hat{t})$ corresponds some $\hat{t}$-termination path for s, j, t_s, and thus (at all times) $X_j(\hat{t}) \geq MW_j(\hat{t})$. We conclude that after the N-th iteration $X_j(\hat{t}) = MW_j(\hat{t})$.

We now prove the second part of the lemma by induction on N: the inductive claim is that, for $j \in V$ and $\hat{t} \in [t_S, t^*]$ such that $MH_j(\hat{t}) > N$, after the N-th iteration we have $X_j(\hat{t}) > MW_j(\hat{t})$. For $N = 0$ the claim is trivial. Assuming truth for $N - 1$, we prove for N. Let $j \in V$ and $\hat{t} \in [t_S, t^*]$ be such that $MH_j(\hat{t}) > N$. According to the inductive assumption, prior to the N-th iteration we have $X_j(\hat{t}) > MW_j(\hat{t})$. Contradicting the inductive claim, suppose that after the N-th iteration: $X_j(\hat{t}) = MW_j(\hat{t})$. To that value of $X_j(\hat{t})$ corresponds some $\hat{t}$-termination path (for s, j, t_S), say:

$p_N = ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_{m-1}, \tau_{m-1}, t_{m-1}), (v_m, \tau_m, t_m))$, where $v_o = s$, $\tau_o = t_S$, $v_m = j$, $t_m = \hat{t}$ and $W(p_N) = MW_j(\hat{t})$.

Denote: $p_{N-1} = ((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_{m-1}, \tau_{m-1}, t_{m-1}))$. Obviously, $m \geq MH_j(\hat{t})$. Since $X_j(\hat{t})$ was assigned $MW_j(\hat{t})$ during the N-th iteration, there must be a path p_N such as the one above for which $X_{v_{m-1}}(t_{m-1})$ was assigned the value of $W(p_{N-1})$ during the $N-1$-st iteration; among all such p_N's we choose one of minimal length. From Lemma 4 follows that p_{N-1} is an optimal t_{m-1}-termination path for s, v_{m-1}, t_S, among those paths having at most $m-1$ hops. This, and the fact that p_N was chosen to be of minimal length, lead to the conclusion that $MH_{v_{m-1}}(t_{m-1}) \geq m-1$. On the other hand, if $MH_{v_{m-1}}(t_{m-1}) > m-1$, then there is an optimal t_{m-1}-termination path for s, v_{m-1}, t_s whose weight is strictly lower than $W(p_{N-1})$, and it follows that there is a $\hat{t}$-termination path for s, j, t_S whose weight is strictly less than $W(p_N)$; but since $W(p_N) = MW_j(\hat{t})$ this is a contradiction. Thus, $MH_{v_{m-1}}(t_{m-1}) = m-1$. We note that after the $N-1$-st iteration we had $X_{v_{m-1}}(t_{m-1}) = MW_{v_{m-1}}(t_{m-1})$; on the other hand

$$MH_{v_{m-1}}(t_{m-1}) = m-1 \geq MH_j(\hat{t}) - 1 > N-1$$

which contradicts the inductive assumption. □

Theorem 2: Algorithm Weight stops after a finite number of iterations. When it does, for all $j \in V$ the value of X_j is the minimum weight for s, j, t_S.

Proof: Since, $\forall (i,k) \in E$, $d_{ik}(t) > 0$ and $d_{ik}(t)$ is continuous, there is some $\delta > 0$ such that $\forall (i,k) \in E \;\; \forall t \in [t_S, t^*]: d_{ik}(t) > \delta$. Thus, $\lceil \frac{t^*}{\delta} \rceil$ is an upper-bound on the length of an optimal $\hat{t}$-termination path for all $j \in V$ and $\hat{t} \in [t_S, t^*]$ i.e., $\forall j \in V, \; \hat{t} \in [t_S, t^*] : MH_j(\hat{t}) \leq \lceil \frac{t^*}{\delta} \rceil$. This, together with the first part of Lemma 6, leads to the conclusion that the functions $X_l(t)$ stop changing (for all $l \in V$) after a finite number of iterations. Thus, so do the functions $Y_{lk}(t)$, and hence the first part of the theorem.

Denote $MH \stackrel{\Delta}{=} \max_{j \in V, t \in [t_S, t^*]} MH_j(\hat{t})$, and consider node j for which $MH_j(\hat{t}) = MH$. Denote the corresponding optimal $\hat{t}$-termination path for s, j, t_S by:
$((v_o, \tau_o, t_o), (v_1, \tau_1, t_1), \ldots, (v_{MH}, \tau_{MH}, t_{MH}))$, where $v_o = s$, $\tau_o = t_S$, $v_{MH} = j$ and $t_{MH} = \hat{t}$. From the discussion in the proof of Lemma 6 follows that, for $1 \le N \le MH$, $MH_{v_N}(t_N) = N$. We now prove by induction on N, that the algorithm does not stop before the N-th iteration is performed. For $N = 0$ and $N = 1$ the claim is trivial. Assuming truth for N, we prove for $N+1$. We know that the N-th iteration was performed, and we also know that for some $i \in V$ and $t' \in [t_S, t^*]$ holds $MH_i(t') = N$. Thus, from Lemma 6 follows that at the N-th iteration $X_i(t')$ changes its value (to that of $MW_i(t')$), and thus $Y_{ik}(t)$ (for all $k \in N_i$) changes during that iteration. Thus, the algorithm does not stop at the N-th iteration, and the $N+1$-st one is indeed performed. Thus, the algorithm runs for at least MH iterations (in fact, for $MH+1$ iterations), and thus, according to Lemma 6, when it stops we have: $\forall j \in V, \hat{t} \in [t_S, t^*] : X_j(t) = MW_j(\hat{t})$. Since for each $j \in V$ there is some $\hat{t} \in [t_S, t^*]$ □

which is the arrival time of an optimal path for s, j, t_S, the second part of the theorem follows.

Following the discussion in the previous subsection regarding Algorithm Path, it is easy to see that after Algorithm Weight stops, one can find a (finite) path p for which $W(p) = X_r$. Thus, Theorem 2 guarantees that an optimal solution to Problem TMWP is found by running Algorithm Weight followed by Algorithm Path.

C. Special Cases

In this subsection we discuss some cases of special interest, that are solved by the algorithm presented above.

1. Optimal Loopless Paths

Even when optimal paths are finite they may well contain loops. An interesting question is whether one can devise an algorithm to find the optimal *loopless* path i.e., the minimum-weight path among those that do not contain loops, and whether the algorithm is efficient. At this point we can state that such an algorithm cannot belong to the family investigated in this paper, i.e., an iterative algorithm, that finds at each iteration an additional triplet of the requested path and that makes $O(|E|)$ operations per iteration. This can be shown by a reduction from the Hamiltonian Path Problem (known to be NP-Complete [11]). Since the optimal path is loopless, the number of triplets is at most $|V|-1$ and hence the total number of iterations performed by the algorithm is bounded by a polynomial of $|V|$. Given a network $G(V, E)$, two nodes $s, r \in V$, the Hamiltonian Path Problem is to determine whether there is a loopless path between s and r passing through all nodes; we transform this problem into an instance of ours in the following way:

$$\forall (i, k) \in E: \ d_{ik}(t) \equiv 1 \ ,$$

$$\forall (i, k) \in E, k \neq r: \ W_{ik}(t) \equiv 0 \ ,$$

$$\forall i \in N_r: \ W_{ir}(t) = \begin{cases} 0 & t = |V| - 2 \\ 1 & otherwise \end{cases} .$$

$$\forall i \in V: \ P_i(\tau, t) \equiv \infty \ .$$

It is easy to verify that a Hamiltonian path between s and r exists iff for $t_S = 0$ the optimal loopless path has zero weight.

However, there are cases in which the existence of a solution to problem TMWP in form of a loopless path is guaranteed (i.e., there is a loopless p^* such that $\forall p \in \Pi(s,r,t_S)$: $W(p^*) \leq W(p)$). One such case is when the Finiteness Criterion is satisfied, and in addition there is a "homogeneous" penalty for delays throughout the network that is the only cost for parking weights; in other words:

a. $\forall \ i, k, \tau, t \quad P_i(\tau, t) = P_k(\tau, t)$, i.e.: parking costs are identical in all nodes;

b. $\forall \ i, k, t \quad P_i(t, t + d_{ik}(t)) \leq W_{ik}(t)$, i.e., it is not more expensive to "spend time" in nodes than on links (on a link there is a penalty for the delay and also for using the link).

It is easy to verify that in this case there is an optimal path which is loop free, and thus its number of edges is at most $|V| - 1$. Thus, running Algorithm Weight for $|V| - 1$ iterations gives a solution to Problem TMWP, which, in this particular case, is also a solution to the optimal loopless path problem.

2. Minimum Delay Problem

Let $W_{ik}(t) = d_{ik}(t)$ (for all $(i, k) \in E$). Then, by assigning proper values to $P_i(\tau, t)$ we may define within our model various time-dependent minimum-delay problems such as those investigated in [6] and [7]:

- If $\forall \ i, \tau, t \quad P_i(\tau, t) = t - \tau$, then this is the "Unrestricted Waiting Model" of [7]; similarly, if $\forall \ i, \tau, t \quad P_i(\tau, t) = \infty$, this is the "Forbidden Waiting Model", and if $\forall \ i \neq s, \tau, t \quad P_i(\tau, t) = \infty$ but $\forall \ \tau, t \quad P_s(\tau, t) = t - \tau$, then this is the "Source Waiting Model" (see [7]). Note that the Unrestricted Waiting problem is a special case of the homogeneous delay penalty problem discussed above.

- If $\forall\ i\ ,\tau\ ,t \quad P_i(\tau\ ,t)\ =\ \begin{cases} t-\tau & if \quad parking \quad is \quad allowed \quad during\ [\tau\ ,t] \\ \infty & otherwise \end{cases}$, then this is the model investigated in [6].

Note that the above problems fulfill the Finiteness Criterion.

3. A Dynamic Location Problem

The model and algorithms presented in this paper are used in[12] as building blocks for solving the following dynamic network location problem: a facility is to be located at each moment of the interval $[0\ ,T]$ at a node of a given network. There is a performance measure $C_i(t)$ that captures the cost of having the facility located at node i at time t. Also, for each pair of nodes $i\ ,k$, there is a "switching cost function" $S_{ik}(\tau\ ,t)$, that accounts for the cost of moving the facility from i to k during the interval $(\tau\ ,t)$. A dynamic location problem is formulated, and an optimal algorithm is presented and validated.

V. CONCLUSION

We have shown that the time-dependent minimum-weight path problem always has a solution, either in the form of a finite path or in that of an infinite one. Moreover, optimal infinite paths have an invariance property and build themselves nicely out of an infinite sequence of finite paths with increasing lengths. We believe, however, that generally it is hard to identify the rule for producing such a sequence.

For a large class of problems, namely those for which a finite solution is guaranteed, we presented an algorithm that solves the problem in a finite number of steps. In contrast to the algorithms of [7] (but similarly to that of [6]), the number of operations performed by this algorithm depend not only on the network topology (i.e., the sets V, E) but also on the sets of functions D, W, P. In [8], an instance of Problem TMWP is shown to be NP-hard, and thus it is unlikely that there exists an algorithm for TMWP whose number of operations is bounded by a polynomial of $|V|$.

Finally, some words regarding problems with *infinite* path solutions. These problems describe extreme situations in which it pays to stay longer in the network, given that one knows how to ‘‘plan the stay’’ (i.e., by traveling through an optimal infinite path). We have indicated that in general it it hard to produce an optimal infinite path. However, if the algorithm of Section IV is made to work on an infinite domain of time ($[t_S, \infty)$), it can be shown that the solutions obtained in successive iterations (in the form of weights given by $X_i(t)$) approach the minimum weight asymptotically. Thus, one may get paths whose weights approach the minimal one infinitesimally. However, the algorithm gives no indication as to the distance between the solution obtained in any stage and the optimal one, and thus one cannot know when the execution was performed sufficiently long.

APPENDIX A

Proof of Theorem 1

Consider the set $W_{s,r,t_s} \stackrel{\Delta}{=} \{ W(p) \mid p \in \Pi(s,r,t_S) \}$. Since $\forall\, p \in \Pi(s,r,t_S)\ \ W(p) \geq 0$ and $\Pi(s,r,t_S)$ is not empty, it follows that $W_{inf} \stackrel{\Delta}{=} inf\ \ W_{s,r,t_s}$ exists (and is finite). Thus, either there is an infinite sequence $\{ p_n \}_{n=0}^{\infty}$, $p_n \in \Pi(s,r,t_S)$, such that $\lim_{n \to \infty} W(p_n) = W_{inf}$, or a finite one $\{ p_n \}_{n=0}^{N}$ such that $W(p_N) = W_{inf}$. If $\{ p_n \}$ is a finite sequence $(n \leq N)$, then we are done, since p_N is the required path. Assume then that $\{ p_n \}$ is an infinite sequence.

Each p_n is either a finite path, or an infinite one corresponding to a (proper and invariant) sequence $FP^{(n)} = \{ fp_m^{(n)} \}_{m=0}^{\infty}$ such that $\lim_{m \to \infty} W(fp_m^{(n)}) = W(p_n)$. Define a sequence of finite paths $\tilde{FP} = \{ \tilde{fp}_n \}$ by the following rule: if p_n is finite then $\tilde{fp}_n = p_n$, else (p_n is infinite) $\tilde{fp}_n = fp_n^{(n)}$. Note that $\tilde{FP}$ is not necessarily invariant. Since $\lim_{n \to \infty} W(p_n) = W_{inf}$, and since for an infinite p_n $\lim_{m \to \infty} W(fp_m^{(n)}) = W(p_n)$, it follows that $\lim_{n \to \infty} W(\tilde{fp}_n) = W_{inf}$.

We shall now generate an infinite and *invariant* subsequence of $\tilde{FP}$ by pruning $\tilde{FP}$. We prove by induction on j that, for all j, there are v_j, τ_j, t_j such that there is an infinite subsequence $\{ \tilde{fp}_{n_k} \}$ of $\tilde{FP}$ for which, for all $n_k \geq j$: $v_j^{(n_k)} \equiv v_j$, $\tau_j^{(n_k)} \equiv \tau_j$, $t_j^{(n_k)} \equiv t_j$. We begin with $j = 0$. Obviously, for all n, $v_o^{(n)} \equiv s$, $\tau_o^{(n)} \equiv t_s$. Assume that we have an infinite sequence $\tilde{FP}$ (which is a subsequence of the original one), for which $\lim_{n \to \infty} W(\tilde{fp}_n) = W_{inf}$, and for $0 \leq i < j$: $\forall n \geq i\ \ v_i^{(n)} \equiv v_i$, $\tau_i^{(n)} \equiv \tau_i$, $t_i^{(n)} \equiv t_i$ and also, for $n \geq j$: $v_j^{(n)} \equiv v_j$, $\tau_j^{(n)} \equiv \tau_j$. Since v_j has a finite number of neighbors and $\tilde{FP}$ is infinite, there is a neighbor v_{j+1} of v_j, such that, for infinitely many n's: $v_{j+1}^{(n)} \equiv v_{j+1}$. We throw out of $\tilde{FP}$ those paths for which the last equation does not hold, and are left with an infinite sequence $\tilde{FP}$ for which $\lim_{n \to \infty} W(\tilde{fp}_n) = W_{inf}$, and for all $n \geq j+1$: $v_{j+1}^{(n)} \equiv v_{j+1}$. Next we observe that the infinite sequences $\{ t_j^{(n)} \}_{n=j}^{\infty}$, $\{ \tau_{j+1}^{(n)} \}_{n=j+1}^{\infty}$, $\{ t_{j+1}^{(n)} \}_{n=j+1}^{\infty}$ are upper-bounded. This follows from the following facts:

1. $\forall_{n \geq j} : t_j^{(n)} \leq \tau_j^{(n)} + 1 = \tau_j + 1$;
2. $\forall_{n \geq j+1} : \tau_{j+1}^{(n)} = t_j^{(n)} + d_{v_j v_{j+1}}(t_j^{(n)})$;
3. $t_{j+1}^{(n)} \leq \tau_{j+1}^{(n)} + 1$;

4. $d_{v_j v_{j+1}}(t)$ is continuous.

Therefore, because $\lim_{n\to\infty} W(\tilde{fp}_n) = W_{inf}$, $\tilde{FP}$ can be further pruned so that the resulting sequence remains infinite, and for it the sequences $P_{v_j}(\tau_j, t_j^{(n)})$, $W_{v_j v_{j+1}}(t_j^{(n)})$, $P_{v_{j+1}}(\tau_{j+1}^{(n)}, t_{j+1}^{(n)})$ are upper-bounded as well. The remaining (infinite) sequence is further pruned to yield a new infinite sequence for which the following finite limits exist:

$$\lim_{n\to\infty} t_j^{(n)}, \quad \lim_{n\to\infty} P_{v_j}(\tau_j, t_j^{(n)}), \quad \lim_{n\to\infty} W_{v_j v_{j+1}}(t_j^{(n)}), \quad \lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}^{(n)}, t_{j+1}^{(n)})$$

Denote $t_j \stackrel{\Delta}{=} \lim_{n\to\infty} t_j^{(n)}$; from (1) above it follows that $t_j \le \tau_j + 1$. Also, since $d_{v_j v_{j+1}}(t)$ is continuous:

$$\tau_{j+1} \stackrel{\Delta}{=} t_j + d_{v_j v_{j+1}}(t_j) =$$

$$= \lim_{n\to\infty} t_j^{(n)} + d_{v_j v_{j+1}}(\lim_{n\to\infty} t_j^{(n)}) = \lim_{n\to\infty} t_j^{(n)} + \lim_{n\to\infty} d_{v_j v_{j+1}}(t_j^{(n)}) = \lim_{n\to\infty} [t_j^{(n)} + d_{v_j v_{j+1}}(t_j^{(n)})] =$$

$$= \lim_{n\to\infty} \tau_{j+1}^{(n)} \quad .$$

Consider now the sequence $\hat{FP} = \{\hat{fp}_n\}$, which is identical to $\tilde{FP}$, except that $\forall n \ge j : t_j^{(n)} \equiv t_j$, and $\forall n \ge j+1 : \tau_{j+1}^{(n)} \equiv \tau_{j+1}$. Since $\lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}^{(n)}, t_{j+1}^{(n)})$ exist and is finite, it follows that $\lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}, t_{j+1}^{(n)})$ exists and is finite. We have that:

$$\lim_{n\to\infty} W(\hat{fp}_n) = \lim_{n\to\infty} \{ W(\tilde{fp}_n) + [P_{v_j}(\tau_j, t_j) + W_{v_j v_{j+1}}(t_j) + P_{v_{j+1}}(\tau_{j+1}, t_{j+1}^{(n)})] -$$

$$- [P_{v_j}(\tau_j, t_j^{(n)}) + W_{v_j v_{j+1}}(t_j^{(n)}) + P_{v_{j+1}}(\tau_{j+1}^{(n)}, t_{j+1}^{(n)})] \} =$$

$$= \lim_{n\to\infty} W(\tilde{fp}_n) + [P_{v_j}(\tau_j, t_j) - \lim_{n\to\infty} P_{v_j}(\tau_j, t_j^{(n)})] +$$

$$+ [W_{v_j v_{j+1}}(t_j) - \lim_{n\to\infty} W_{v_j v_{j+1}}(t_j^{(n)})] + [\lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}, t_{j+1}^{(n)}) - \lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}^{(n)}, t_{j+1}^{(n)})]$$

Since:

1. $P_{v_j}(\tau_j, t_j) = P_{v_j}(\tau_j, \lim_{n\to\infty} t_j^{(n)}) = \lim_{n\to\infty} P_{v_j}(\tau_j, t_j^{(n)})$,
2. $W_{v_j v_{j+1}}(t_j) = W_{v_j v_{j+1}}(\lim_{n\to\infty} t_j^{(n)}) \le \lim_{n\to\infty} W_{v_j v_{j+1}}(t_j^{(n)})$,
3. $\lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}, t_{j+1}^{(n)}) = \lim_{n\to\infty} P_{v_{j+1}}(\tau_{j+1}^{(n)}, t_{j+1}^{(n)})$,

it follows that $\lim_{n\to\infty} W(\hat{fp}_n) \le \lim_{n\to\infty} W(\tilde{fp}_n) = W_{inf}$, and thus $\lim_{n\to\infty} W(\hat{fp}_n) = W_{inf}$.

Replacing $\tilde{FP}$ by $\hat{FP}$, we complete the inductive step. □

APPENDIX B

In this appendix we prove the existence of the minima considered in Algorithm Weight. As noted before, the 'min' operation returns an $'\infty'$ value if the set considered is empty. Thus, we restrict our attention to the case that the corresponding set is nonempty. Throughout the appendix we use the notation $D_{l,k,t} \overset{\Delta}{=} \{ \tau \mid \tau + d_{lk}(\tau) = t \}$.

Lemma 7: At any stage of execution of Algorithm Weight, all the following hold:

(a) $\forall (l, k) \in E \quad \forall t \in [t_s, t^*]$: if $D_{l,k,t} \neq \emptyset$ then $\min\limits_{t \in D_{l,k,t}} (X_l(\tau) + W_{lk}(\tau))$ exists (i.e., there is a $\tau^* \in D_{l,k,t}$ such that for any other $\tau \in D_{l,k,t}$ holds $X_l(\tau^*) + W_{lk}(\tau^*) \leq X_l(\tau) + W_{lk}(\tau)$).

(b) $\forall k, l \in E \quad \forall t \in [t_s, t^*]$: $\min\limits_{\tau \leq t,\, k \in N_l} (Y_{kl}(\tau) + P_l(\tau, t))$ exists.

(c) $\forall l \in V$: $\min\limits_{t_s \leq \tau \leq t^*} (X_l(\tau))$ exists.

Proof: We add to the lemma the following auxiliary claims (related to any stage of execution):

(d) For $l \in V$ and for a converging sequence $\{ t_n \}$, if $\lim\limits_{n \to \infty} X_l(t_n)$ exists then $X_l(\lim\limits_{n \to \infty} t_n) \leq \lim\limits_{n \to \infty} X_l(t_n)$.

(e) For $(k, l) \in E$ and for a converging sequence $\{ t_n \}$, if $\lim\limits_{n \to \infty} Y_{kl}(t_n)$ exists then $Y_{kl}(\lim\limits_{n \to \infty} t_n) \leq \lim\limits_{n \to \infty} Y_{kl}(t_n)$.

Claim (d) is used to prove claim (a), and claim (e) serves to prove claim (b). We prove claims (a)–(e) by induction on the iteration number. Without affecting the proof, we may assume that step 1(a) is applied to *all* $k \in V$ (including s). Thus, before step 1(b) (of the zeroth iteration) is executed, all claims trivially hold. Assuming they hold up to the beginning of the N-th iteration, we prove that they hold after it (given that the N-th iteration is performed). We prove for $N \geq 1$; the proof for $N = 0$ follows immediately.

We begin with claim (b). Consider the execution of step 2 for some $(k, l) \in E$ at the N-th iteration and consider some $t \in [t_s, t^*]$. If for all $\tau \leq t$ hold $Y_{kl}(\tau) + P_l(\tau, t) = \infty$ then claim (b) (for time t and link (k, l)) trivially holds. Otherwise, denote $\psi \overset{\Delta}{=} \{ \tau \mid Y_{kl}(\tau) + P_l(\tau, t) < \infty \}$. Since link and node weight functions may equal infinity only in open intervals, it follows that ψ is closed. There is some converging sequence $\{ \tau_n \}$, $\{ \tau_n \} \subseteq \psi$, such that

$\lim_{n\to\infty}[Y_{kl}(\tau_n)+P_l(\tau_n\ ,t)] = \inf_{\tau\in\psi}[Y_{kl}(\tau)+P_l(\tau\ ,t)]$. Denote $\tau^* = \lim_{n\to\infty}\tau_n$; since ψ is closed, $\tau^*\in\psi$. From the definitions of $P_l(\tau\ ,\ \cdot\ )$ and ψ follows that ψ is a subset of a permitted interval for parking at l. Thus, and since $P_l(\tau\ ,\ \cdot\ )$ is continuous within such intervals, it follows that

$$P_l(\tau^*\ ,t) = P_l(\lim_{n\to\infty}\tau_n\ ,t) = \lim_{n\to\infty}P_l(\tau_n\ ,t) \quad .$$

By the inductive assumption claim (e) holds for $Y_{kl}(t)$, and thus $Y_{kl}(\tau^*) \le \lim_{n\to\infty}Y_{kl}(\tau_n)$. Combining the above, we have

$$Y_{kl}(\tau^*)+P_l(\tau^*\ ,t) \le \lim_{n\to\infty}[Y_{kl}(\tau_n)+P_l(\tau_n\ ,t)] = \inf_{\tau\in\psi}[Y_{kl}(\tau^*)+P_l(\tau^*\ ,t)] \quad ,$$

proving the inductive step for claim (b).

We consider now claim (d). Denote by $X_l^{(N-1)}(t)$ and $X_l^{(N)}(t)$ the value of $X_l(t)$ after the $N-1$-st and the N-th iterations, respectively; we use similar notations for $Y_{kl}(t)$. Consider a converging sequence $\{t_n\}$ ($\lim_{n\to\infty}t_n \triangleq \hat{t}$), such that $\lim_{n\to\infty}X_l^{(N)}(t_n)$ exists. Since we assume that claim (b) holds for the $N-1$-st iteration, we may define sequences $\{\tau_n^{(k)}\}$, such that

$$\forall\ k,n \quad \forall\ \tau\le t_n:\ \ Y_{kl}^{(N-1)}(\tau_n^{(k)})+P_l(\tau_n^{(k)},t_n) \le Y_{kl}^{(N-1)}(\tau)+P_l(\tau,t_n) \tag{1}$$

We prune $\{t_n\}$ so that $\{\tau_n^{(k)}\}$ converges for each k, and we denote $\hat{\tau}^{(k)} \triangleq \lim_{n\to\infty}\tau_n^{(k)}$. We have:

$$\lim_{n\to\infty}X_l^{(N)}(t_n) = \lim \min_{K\in N_l,\ \tau\le t}[Y_{kl}^{(N-1)}(\tau)+P_l(\tau\ ,t_n)] = \min_k[Y_{kl}^{(N-1)}(\hat{\tau}^{(k)},\hat{t})+P_l(\hat{\tau}^{(k)},\hat{t})] \quad .$$

Assume that for some k there is a $\tilde{\tau}\ne\hat{\tau}^{(k)}$, $\tilde{\tau}<\hat{t}$, such that $Y_{kl}^{(N-1)}(\tilde{\tau})+P_l(\tilde{\tau},\hat{t})<Y_{kl}^{(N-1)}(\hat{\tau}^{(k)})+P_l(\hat{\tau}^{(k)},\hat{t})$. Using the inductive assumption on claim (e), we have

$$Y_{kl}^{(N-1)}(\tilde{\tau})+P_l(\tilde{\tau},\hat{t}) < \lim_{n\to\infty}[Y_{kl}^{(N-1)}(\tau_n^{(k)})+P_l(\tau_n^{(k)},t_n)] \tag{2}$$

Since $\tilde{\tau}<\hat{t}$ and $t_n \xrightarrow[n\to\infty]{} \hat{t}$, there is an N_0 such that $\forall\ n>N_0\ \ \tilde{\tau}<t_n$. From (2) follows that there is an $N_1>N_0$ such that

$$\forall\ n>N_1\ \ Y_{kl}^{(N-1)}(\tilde{\tau})+P_l(\tilde{\tau},\hat{t}) < Y_{kl}^{(N-1)}(\tau_n^{(k)})+P_l(\tau_n^{(k)},t_n) \tag{3}$$

Suppose there is an $n_1>N_1$ such that $t_{n_1}\le\hat{t}$. Then $P_l(\tilde{\tau},t_{n_1})\le P_l(\tilde{\tau},\hat{t})$, and from (3) we get

$$Y_{kl}^{(N-1)}(\tilde{\tau})+P_l(\tilde{\tau},t_{n_1}) < Y_{kl}^{(N-1)}(\tau_{n_1}^{(k)})+P_l(\tau_{n_1}^{(k)},t_{n_1})$$

contradicting (1) . Thus, we conclude that $\forall\ n>N_1\ \ t_n>\hat{t}$. Denote by τ^* a minimizer of $[Y_{kl}^{(N-1)}(\tau)+P_l(\tau,\hat{t})]$ for $\tau<\hat{t}$ (which, according to the inductive assumption on claim (b), does exist). From (3) we conclude that for all $n > N_1$ we have $\tau^* \neq \tau_n^{(k)}$. If for some $n_1 > N_1$ $\tau_{n_1}^{(k)} < \hat{t}$, then

$$Y_{kl}^{(N-1)}(\tau_{n_1}^{(k)}) + P_l(\tau_{n_1}^{(k)}, \hat{t}) \; > \; Y_{kl}^{(N-1)}(\tau^*) + P_l(\tau^*, \hat{t})$$

Since $t_{n_1} > \hat{t}$, by the additivity of $P_l(\cdot, t)$ we get

$$Y_{kl}^{(N-1)}(\tau_{n_1}^{(k)}) + P_l(\tau_{n_1}^{(k)}, t_{n_1}) \; > \; Y_{kl}^{(N-1)}(\tau^*) + P_l(\tau^*, t_{n_1})$$

contradicting (1) . We thus conclude that for all $n > N_1$ $\tau_n^{(k)} > \hat{t}$. Since $k \in V$ and $|V| < \infty$, the last relation contradicts the fact that $\tau_n^{(k)} \underset{n \to \infty}{\to} \hat{\tau}^{(k)} < \hat{t}$.

We conclude that, for all k, $\hat{\tau}^{(k)}$ minimizes the expression $[Y_{kl}^{(N-1)}(\tau) + P_l(\tau, \hat{t})]$ for $\tau < \hat{t}$. Thus:

$$X_l^{(N)}(\lim_{n \to \infty} t_n) \; = \; X_l^{(N)}(\hat{t}) \; = \; \min_k \; \min_{\tau < \hat{t}} [Y_{kl}^{(N-1)}(\tau) + P_l(\tau, \hat{t})] \; = \; \min_k \; [Y_{kl}^{(N-1)}(\hat{\tau}^{(k)}) + P_l(\hat{\tau}^{(k)}, \hat{t})] \; =$$

$$= \; \min_k \; [Y_{kl}^{(N-1)}(\lim_{n \to \infty} \tau_n^{(k)}) + P_l(\lim_{n \to \infty} \tau_n^{(k)}, \hat{t})] \; \leq \; \min_k \; \lim_{n \to \infty} [Y_{kl}^{(N-1)}(\tau_n^{(k)}) + P_l(\tau_n^{(k)}, t_n)] \; =$$

$$= \; \lim_{n \to \infty} \; \min_k \; \min_{\tau < t_n} [Y_{kl}^{(N-1)}(\tau) + P_l(\tau, t_n)] \; = \; \lim_{n \to \infty} X_l^{(N)}(t_n)$$

proving the inductive step for claim (d), for case (2) as well.

The proof of the inductive step for claims (a) and (e) is similar to the above, only that step 3(b) is considered and the assumption on link weight functions used.

Finally, we prove the inductive step for claim (c). If $X_l(\tau) \; = \; \infty$ for all $t_s \; \leq \; \tau \; \leq \; t^*$, then the inductive step follows trivially. Otherwise, denote $\psi \; \overset{\Delta}{=} \; \{ \; \tau \; | \; t_s \; \leq \; \tau \; \leq \; t^* \, , \; X_l(\tau) \; < \; \infty \; \}$. It is easy to see that ψ is closed (at any stage of execution), and thus the rest of the proof is similar to that of claim (b). □

REFERENCES

1. N. Deo and C.Y. Pang, ''Shortest Path Algorithms: Taxonomy and Annotation,'' *Networks* **14** pp. 275-323 (1984).
2. E. Klafszky, ''Determination of Shortest Path in a Network with Time-Dependent Edge-Lengths,'' *Mathematische Operationsforschung u. Statistik* **3** pp. 255-257 (1972).
3. K.L. Cooke and E. Halsey, ''The Shortest Route Through a Network with Time Dependent Internodal Transit Times,'' *Journal Math. Analysis Applications* **14** pp. 493-498 (1966).
4. S.E. Dreyfus, ''An Appraisal of some Shortest Path Algorithms,'' *Operations Research* **17** pp. 395-412 (1969).
5. S.T. Ling, K. Furuno, and Y. Tezuka, ''Optimal Path in Networks with Time-Varying Traverse Time and Expenses Branches,'' Technological Reports of Osaka University, Japan (1972).
6. J. Halpern, ''The Shortest Route with Time Dependent Length of Edges and Limited Delay Possibilities in Nodes,'' *Zeitschrift fuer Operations Research* **21** pp. 117-124 (1977).
7. A. Orda and R. Rom, ''Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length,'' *Journal of the ACM* **37**(3) pp. 607-625 (July 1990).
8. A. Orda and R. Rom, *Traveling without Waiting in Time-Dependent Networks is NP-Hard* . March 1989.
9. A. Orda and R. Rom, ''Minimum Weight Paths in Time-Dependent Networks,'' *Networks* **21**(3) pp. 295-319 (May 1991).
10. E.J. Anderson, P. Nash, and A.B. Philpott, ''A Class of Continuous Network Flow Problems,'' *Mathematics of Operations Research* **7**(4) pp. 501-514 (November 1982).
11. M.R. Garey and D.S. Johnson, *Computers and Intractability,* W.H. Freeman and Company, San Francisco (1979).
12. A. Orda and R. Rom, ''Location of Central Nodes in Time Varying Computer Networks,'' pp. 193-199 in *Proceedings of IEEE Infocom 89*, Ottawa, Canada (April 1989).