# Time-Varying Shortest Path Problems with Constraints

**X. Cai,[1] T. Kloks,[2] C. K. Wong[3,*]**

[1] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

[2] Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

[3] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong

*Received 28 November 1995; accepted 20 November 1996*

**Abstract:** We study a new version of the shortest path problem. Let $G = (V, E)$ be a directed graph. Each arc $e \in E$ has two numbers attached to it: a transit time $b(e, u)$ and a cost $c(e, u)$, which are functions of the departure time $u$ at the beginning vertex of the arc. Moreover, postponement of departure (i.e., waiting) at a vertex may be allowed. The problem is to find the shortest path, i.e., the path with the least possible cost, subject to the constraint that the total traverse time is at most some number $T$. Three variants of the problem are examined. In the first one, we assume arbitrary waiting times, where waiting at a vertex without any restriction is allowed. In the second variant, we assume zero waiting times, namely, waiting at any vertex is strictly prohibited. Finally, we consider the general case where there is a vertex-dependent upper bound on the waiting time at each vertex. Several algorithms with pseudopolynomial time complexity are proposed to optimally solve the problems. First, we assume that all transit times $b(e, u)$ are positive integers. In the last section, we show how to include zero transit times.
 Networks **29:** 141–149, 1997

## 1. INTRODUCTION

One of the most studied problems in graph algorithms is the shortest path problem. A natural extension is the problem where one would like to find the shortest path subject to some constraint, such as the total time required to traverse the path being at most some number $T$. This kind of problem is also often studied (see, e.g., [2, 6–9]) and is known to be NP-complete [1, 5].

In this article, we study yet a new extension of the problem. We address the situations where the transit time and the cost to traverse an arc are varying over time, which depend upon the departure time at the beginning vertex of the arc. Moreover, waiting times at vertices are considered as decision variables. Our problem is to

*Correspondence to:* C. K. Wong

* On leave from IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.

Contract grant sponsor: Research Grants Council of Hong Kong

Contract grant number: CUHK 278/94E

Contract grant sponsor: Strategic Research Program at The Chinese University of Hong Kong

Contract grant number: SRP 9505



CCC 0028-3045/97/030141-09

determine an optimal path as well as the optimal waiting times at the vertices along the path, subject to the constraint that the total traverse time of the path is at most $T$. Applications of our model are varied. One example is the data transmission problem. Suppose that a data packet has to be sent between two specified nodes in a network as cheaply as possible within a time limit $T$. As the transit time and the cost needed to send the packet on an arc vary over different periods, there exists an optimal departure time to traverse an arc. Thus, an optimal solution to the problem should not only provide the best path connecting the two nodes, but also specify the optimal duration for the data packet to stay at each node to wait for the best departure time. Another example is the freight transport problem. Suppose that some freight is to be sent from a source to a sink in a network before a deadline $T$. Between two neighboring cities, several types of freight services are available, which, however, have different costs and transport times, depending upon the seasons. An optimal solution should thus specify the route as well as the waiting times of the freight at each city so that the overall cost is minimum while the freight arrives at the destination no later than the deadline $T$.

We address three variants of the problem: The first one assumes no constraints on the waiting times, namely, waiting at any vertex arbitrarily is allowed. The second variant assumes zero waiting time, where waiting at any vertex is strictly prohibited. Finally, we consider the situation where each vertex has a waiting time limit. Several algorithms are derived to compute optimal solutions. We first examine algorithms under the assumption that transit times on all arcs are strictly positive. Three algorithms are proposed to optimally solve the three variants of the problem under this assumption, with time complexity of $O(T(m + n))$, $O(T(m + n))$, and $O(T(m + n \log T))$, respectively, where $m$ and $n$ are, respectively, the numbers of arcs and vertices. We then generalize the algorithms to cases with nonnegative transit times. We show that the three variants of the problem can be optimally solved by algorithms with running time $O(T(m + n \log n))$, $O(T(m + n \log n))$, and $O(T(m + n \log n + n \log T))$, respectively. Note that the various versions of the problem that we consider are NP-complete, since a simpler version of the problem (where transit times and costs are constant) has been known to be NP-complete [1, 5]. Nevertheless, the pseudopolynomial time complexity of the algorithms that we propose indicates that they are NP-complete in the ordinary sense only.

The remainder of the article is organized as follows: In Section 2, we introduce the notations and the basic model. Then, Sections 3, 4, and 5 are devoted to problems with arbitrary waiting times, zero waiting times, and bounded waiting times, respectively, all under the assumption of strictly positive transit times. The results are generalized to cases with nonnegative transit times in Section 6. Finally, some concluding remarks are given in Section 7.

## 2. PROBLEM FORMULATION

Let $G = (V, E)$ be a directed graph, without loops or multiple arcs. Let $b(e, u)$ be the *transit time* needed to traverse an arc $e \in E$ and $c(e, u)$ be the *length*, or *cost*, to traverse the arc.[†] Both the transit time $b(e, u)$ and the length $c(e, u)$ are functions of the departure time $u$ at the beginning vertex of $e$, where $u$ is an integer in $[0, T]$ and $T > 0$ is a given integer, which is the maximum time that is allowed to traverse a whole path.[‡] We assume that $b(e, u)$ are nonnegative integers and $c(e, u)$ are nonnegative. Throughout the article, we let $n = |V|$ and $m = |E|$. We also write $c(x, y, u)$ and $b(x, y, u)$ if $e = (x, y)$.

**Definition 1.** *A* waiting time $w(x)$ *at a vertex $x$ is a nonnegative integer, and $\mathcal{U}_x \geq 0$ is its upper bound.*

**Definition 2.** *Let $P = (s = x_1, \ldots, x_r = x)$ be a path from $s$ to $x$. Let $w(x_i)$ $(i = 1, \ldots, r)$ be waiting times at vertices $x_1, \ldots, x_r$. Let $\mathcal{T}(x_1) = w(x_1)$ and define recursively $\mathcal{T}(x_i) = w(x_i) + \mathcal{T}(x_{i-1}) + b(x_{i-1}, x_i, \mathcal{T}(x_{i-1}))$ for $i = 2, \ldots, r$. The departure time of a vertex $x_i$ on $P$, $1 \leq i < r$, is defined as $\mathcal{T}(x_i)$.*

**Definition 3.** *Let $P = (s = x_1, \ldots, x_r = x)$ be a path from $s$ to $x$. The time of $P$, given the waiting times, is defined as $\mathcal{T}(x_r)$. A path has time at most $t$, if there are waiting times for the vertices on the path such that the time of the path with these waiting times is at most $t$.*

**Definition 4.** *Let $P = (s = x_1, \ldots, x_r = x)$ be a path from $s$ to $x$. Let $\mathcal{T}(x_i)$ be the departure time at $x_i$ for $1 \leq i < r$, given the waiting times of the vertices on the path. Let $\mathcal{L}(x_1) = 0$ and define recursively $\mathcal{L}(x_i) = \mathcal{L}(x_{i-1}) + c(x_{i-1}, x_i, \mathcal{T}(x_{i-1}))$ for $i = 2, \ldots, r$. The* length *of $P$, with the given waiting times, is defined as $\mathcal{L}(x_r)$.*

We are interested in the problem of finding a path $P^*$ from a given vertex $s$ to another vertex $y$, $y \in V \setminus \{s\}$ and determining the optimal values of the waiting times at the vertices on the path, so that the length of the path is minimal subject to the following constraints:

[†] For consistency, we shall use the term *length* in the rest of the article.

[‡] The case with $T = 0$ reduces to the ordinary shortest path problem without time-varying constraints (see, e.g., [1]) and is not the concern of this article.

C1. The time of $P^*$ is at most $T$.
C2. The waiting time at each vertex $x$ along $P^*$ is at most $\mathcal{U}_x$.

For completeness, we adopt the following convention:

**Definition 5.** *For a vertex $x \in V \setminus \{s\}$ and a given number $t \leq T$, the length of a shortest path from $s$ to $x$ with time $t$ is said to be $\infty$ if $(i)$ there does not exist any path from $s$ to $x$ or $(ii)$ all paths from $s$ to $x$ either have times greater than $t$ or violate the constraint $C2$ above.*

## 3. ARBITRARY WAITING TIMES

We examine, in this section, the problem when there is no constraint imposed on the waiting times. We refer to this problem as the *Time-Varying Constrained Shortest Path with Arbitrary Waiting Times problem* (TCSP-AWT problem). In this section, we assume that all $b(x, y, u)$ are positive integers.

**Definition 6.** *$d_A(y, t)$ is the length of a shortest path from $s$ to $y$ of time at most $t$, where waiting at any vertex is not restricted.*

The following lemma gives us a recursive relation to compute $d_A(x, t)$. Note that the optimal waiting times can be obtained implicitly by the recursive computations. This will be elaborated on later in Remark 1.

**Lemma 1.** *$d_A(s, t) = 0$ for all $t$ and $d_A(y, 0) = \infty$ for all $y \neq s$. For $t > 0$ and $y \neq s$, we have*

$$d_A(y, t) = \min\{d_A(y, t-1), \min_{\{x|(x,y)\in E\}} \min_{\{u|u+b(x,y,u)=t\}} \{d_A(x, u) + c(x, y, u)\}\}.$$

*Proof.* It is easy to see that $d_A(s, t) = 0$ for all $0 \leq t \leq T$ and $d_A(y, 0) = \infty$ for all $y \neq s$, since all transit times are positive.

We now prove the formula by induction. Consider $t = 1$. The only vertices for which there can exist a path of time at most one are $s$ and neighbors of $s$. For $y = s$, the formula clearly holds. Assume that $y \neq s$. Consider first the case where $(s, y) \in E$ and $b(s, y, 0) = 1$. In this case, the formula holds with $d_A(y, t) = d_A(x, u) + c(x, y, u)$, where $u = 0$ and $x = s$. In any other cases, the formula holds with $d_A(y, t) = d_A(y, t-1) = \infty$ as there is no feasible solution for a path from $s$ to $y$ of time $t = 1$.

Assume that the formula is correct for all $t' < t$. Consider a vertex $y$. Again, if $y = s$, there is nothing to prove. So, assume that $y \neq s$. First, we prove the claim that *there exists a path of time at most $t$ and length $d_A(y, t)$.*

If $d_A(y, t) = \infty$, there is nothing to prove. So, assume that $d_A(y, t)$ is finite. If $d_A(y, t) = d_A(y, t-1)$, then, by induction, there is a path from $s$ to $y$ of length $d_A(y, t)$. The time of the path is at most $t-1$ and, of course, at most $t$.

Assume that $d_A(y, t) = d_A(x, u) + c(x, y, u)$ for some $x$ such that $(x, y) \in E$ and some $u$ such that $u + b(x, y, u) = t$. Since $b(x, y, u) > 0$, we have $u < t$ and, therefore, by induction, we know there must exist a path $P' = (s = x_1, \ldots, x_r = x)$ from $s$ to $x$ of time at most $u$ and length $d_A(x, u)$. Hence, there are waiting times $w(x_i)$ at $x_i$ such that the time of the path $P'$ with these waiting times is $u' \leq u$. Add $u - u'$ to the waiting time $w(x_r)$. We extend the path to vertex $y$, obtaining a path $P$ with the given waiting times and with waiting time zero at $y$. The time of $P$, with these waiting times, is at most $t$, since $u + b(x, y, u) = t$. The length of $P$ with these waiting times is $d_A(x, u) + c(x, y, u' + (u - u')) = d_A(x, u) + c(x, y, u) = d_A(y, t)$. This proves the claim.

We now prove that *$d_A(y, t)$ is the length of a shortest path from $s$ to $y$ with time at most $t$*. Let $P = (s = x_1, \ldots, x_r = y)$ be a shortest path from $s$ to $y$ of time $t' \leq t$, and let $w(x_i)$ be the waiting time at $x_i$ $(i = 1, \ldots, r)$. Clearly, we may assume that $w(x_r) = 0$. Let $x$ be the predecessor of $y$ on this path. Let $u$ be the time of the subpath $P'$ (with the waiting times) from $s$ to $x$, and let $\mathcal{L}(x)$ be the length of $P'$. By definition, $t' = u + b(x, y, u) + w(x_r) = u + b(x, y, u) \leq t$, implying that $u < t$ since $b(x, y, u) > 0$. Thus, by induction, $\mathcal{L}(x) \geq d_A(x, u)$. By definition, the length of $P$ is $\mathcal{L}(y) = \mathcal{L}(x) + c(x, y, u) \geq d_A(x, u) + c(x, y, u)$. If $u + b(x, y, u) = t$, then $\mathcal{L}(y) \geq d_A(x, u) + c(x, y, u) \geq d_A(y, t)$ due to the computation of the formula. Otherwise, if $u + b(x, y, u) < t$, then there must exist a time $t'' = u + b(x, y, u) < t$. Again, according to the formula, we have $d_A(y, t) \leq d_A(y, t-1) \leq \cdots \leq d_A(y, t'') \leq d_A(x, u) + c(x, y, u) \leq \mathcal{L}(y)$. In both cases, we must have $\mathcal{L}(y) = d_A(y, t)$, since $P$ is a path of shortest length and since there exists a path achieving $d_A(y, t)$, as we showed above. This completes the proof. ■

**Definition 7.** *For every arc $(x, y) \in E$ and for $t = 0, \ldots, T$, let*

$$\gamma_A(x, y, t) = \min_{\{u|u+b(x,y,u)=t\}} \{d_A(x, u) + c(x, y, u)\}.$$

*We adopt the convention that $\gamma_A(x, y, t) = \infty$ whenever $\{u|u + b(x, y, u) = t\} = \varnothing$.*

The result below follows directly from Lemma 1:

**Corollary 1.**

$$d_A(y, t) = \min\{d_A(y, t-1), \min_{\{x|(x,y)\in E\}} \gamma_A(x, y, t)\}.$$

*Algorithm TCSP-AWT*
**begin**
**Initialize** $d_A(s,t) = 0$ and $\forall_{x \neq s} d_A(x,t) = \infty$ for $t = 0, \ldots, T$
**Sort** all values $u + b(x,y,u)$ for $u = 1, \ldots, T$ and for all arcs $(x,y) \in E$
**For** $t = 1, \ldots, T$ **do**
  **For** every arc $(x,y) \in E$ **do** $\gamma_A(x,y,t) := \infty$
  **For** all arcs $(x,y) \in E$ and all $u$ such that $u + b(x,y,u) = t$ **do**
    $\gamma_A(x,y,t) := \min\{\gamma_A(x,y,t), d_A(x,u) + c(x,y,u)\}$
  **For** every vertex $y$ **do** $d_A(y,t) := \min\{d_A(y,t-1), \min_{(x,y)\in E} \gamma_A(x,y,t)\}$
**end.**

**Fig. 1.** Algorithm TCSP-AWT.

Corollary 1 indicates that when $d_A(y, t)$ is to be updated we have to know $\gamma_A(x, y, t)$ for all $(x, y) \in E$. Given $t$ and $(x, y)$, $\gamma_A(x, y, t)$ could be evaluated by a naive approach of enumerating $0 \leq u \leq T$ to find those satisfying $u + b(x, y, u) = t$, according to Definition 7. This would, however, require a worst-case running time of $O(T)$ for every $t$. Clearly, we need some mechanism to make the evaluation of $\gamma_A(x, y, t)$ efficiently. Our idea in the algorithm below is to first sort the values of $u + b(x, y, u)$ for all $u = 1, 2, \ldots, T$ and all arcs $(x, y) \in E$, before the recursive relation as given in Lemma 1 is applied to compute $d_A(y, t)$ for all $y \in V$ and $t = 1, 2, \ldots, T$.

We describe the algorithm in Figure 1, followed by an illustrative example. Then, we show its worst-case running time in Lemma 2. Some implementation details which guarantee its worst-case running time are also discussed in the proof of Lemma 2.

We now consider a simple example. Assume that there is a network as shown in Figure 2, where the two elements in the square brackets along each arc $(x, y)$ represent the transit time $b(x, y, u)$ and the length $c(x, y, u)$ of the arc, respectively. The problem is to find a shortest path connecting the source node $s$ and the sink node $i$ such that the time of the path is at most $T = 12$.

Applying Algorithm TCSP-AWT, one may obtain the results in Table I. Thus, when $T = 12$, the length of the shortest path connecting $s$ and $i$ is $d_A(i, 12) = 18$. By a backtracking procedure, it is easy to find that the shortest path is $P^* = (s, g, h, i)$, where the departure times at the vertices $s$, $g$, and $h$ are, respectively, 0, 10, and 11, while the arrival times at the vertices $g$, $h$, and $i$ are, respectively, 4, 11, and 12. There is a waiting time 6 at the vertex $g$ to achieve the shortest length 18.

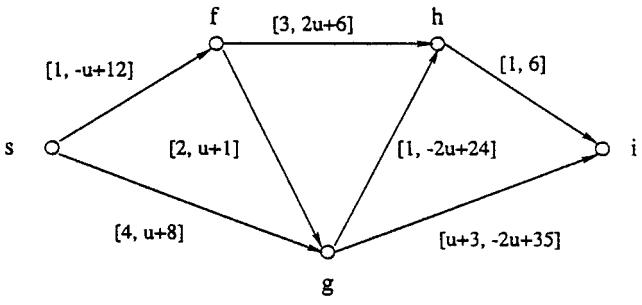


**Fig. 2.** An example.

**Remark 1.** In general, the algorithm TCSP-AWT computes the length $d_A(x, T)$ of the shortest path from the source $s$ to a vertex $x$ with time at most $T$. Let the shortest path be $P^* = (s = x_1^*, \ldots, x_i^*, x_j^*, \ldots, x_r^* = x)$. Note that this path and the optimal departure time at each vertex $x_i^*$ on the path can be identified by a standard backtracking procedure of dynamic programming. Then, the waiting times at the vertices on $P^*$ can be obtained using the departure times. For example, if $\mathcal{T}(x_i^*)$ and $\mathcal{T}(x_j^*)$ are the optimal departure times at the two vertices $x_i^*$ and $x_j^*$ of an arc $(x_i^*, x_j^*)$ on $P^*$, then the optimal waiting time at the vertex $x_j^*$ is $w(x_j^*) = \mathcal{T}(x_j^*) - \mathcal{T}(x_i^*) - b(x_i^*, x_j^*, \mathcal{T}(x_i^*))$.

**Lemma 2.** *The algorithm TCSP-AWT can be implemented such that it runs in $O(T(n + m))$ time.*

**TABLE I. The length of shortest path from $s$ to $y$: $d_A(y, t)$**

| $t$ | $d_A(s, t)$ | $d_A(f, t)$ | $d_A(g, t)$ | $d_A(h, t)$ | $d_A(i, t)$ |
|---|---|---|---|---|---|
| 0 | 0 | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| 1 | 0 | 12 | $\infty$ | $\infty$ | $\infty$ |
| 2 | 0 | 11 | $\infty$ | $\infty$ | $\infty$ |
| 3 | 0 | 10 | 14 | $\infty$ | $\infty$ |
| 4 | 0 | 9 | 8 | 20 | $\infty$ |
| 5 | 0 | 8 | 8 | 20 | 26 |
| 6 | 0 | 7 | 8 | 20 | 26 |
| 7 | 0 | 6 | 8 | 20 | 26 |
| 8 | 0 | 5 | 8 | 18 | 26 |
| 9 | 0 | 4 | 8 | 16 | 24 |
| 10 | 0 | 3 | 8 | 14 | 22 |
| 11 | 0 | 2 | 8 | 12 | 20 |
| 12 | 0 | 1 | 8 | 10 | 18 |

*Proof.* It is easy to check that the initialization can be done in $O(Tn)$ time.

For the sorting in step 2, we can use bucketsort, with $T$ buckets. Since there are $Tm$ values to be sorted, this step can then be performed in $O(Tm)$ time.

Since the values $u + b(x, y, u)$ are now sorted, the overall time needed to update the values $\gamma_A(x, y, t)$ can be done in $O(Tm)$ time.

Finally, the overall time to update the values $d_A(y, t)$ in the last line is proportional to $T$ times the number of arcs, i.e., $O(Tm)$.

It follows that the running time of the algorithm is bounded by $O(T(n + m))$. ■

From Lemma 1 and Corollary 1, one can easily see that, after the termination of the algorithm TCSP-AWT, each computed value $d_A(x, t)$ is the length of a shortest path from $s$ to $x$ of time at most $t$. This, together with Lemma 2, gives us

**Theorem 1.** *The TCSP-AWT problem with positive transit times can be optimally solved in $O(T(n + m))$ time.*

## 4. ZERO WAITING TIMES

In this section, we consider the case in which no waiting times are allowed at any vertices. We refer to this problem as the *Time-Varying Constrained Shortest Path with Zero Waiting Times problem* (TCSP-ZWT problem). We also assume, in this section, that all times $b(x, y, u)$ are positive integers.

Recall Definitions 2, 3, and 4 on departure times at vertices, time of path, and length of path. Note that the waiting times in these definitions should be set to zero for the TCSP-ZWT problem.

**Definition 8.** *$d_Z(y, t)$ is the length of a shortest path from s to y of time exactly t. If such a path does not exist, then $d_Z(y, t) = \infty$.*

Note that the definition of $d_Z(y, t)$ is different from that of $d_A(y, t)$. This is because of the constraint that no waiting is allowed at any vertex in the present problem. Note also that $d_Z^*(y) = \min_{0 \le t \le T} d_Z(y, t)$ is the length of a shortest path from $s$ to $y$ of time at most $T$.

Following similar arguments in the proof for Lemma 1, one can show

**Lemma 3.** *$d_Z(s, 0) = 0$ and $d_Z(y, 0) = \infty$ for all $y \neq s$. For $t > 0$ and $y \neq s$, we have*

$$d_Z(y, t) = \min_{\{x \mid (x,y) \in E\}} \min_{\{u \mid u + b(x,y,u) = t\}} \{d_Z(x, u) + c(x, y, u)\}.$$

Let us now further introduce the following definition:

**Definition 9.** *For each arc $(x, y)$ and each $1 \le t \le T$, define*

$$\gamma_Z(x, y, t) = \min_{\{u \mid u + b(x,y,u) = t\}} \{d_Z(x, u) + c(x, y, u)\}$$

*and adopt the convention that $\gamma_Z(x, y, t) = \infty$ whenever the set $\{u \mid u + b(x, y, u) = t\}$ is empty.*

The result below follows directly from Lemma 3:

**Corollary 2.** *For $1 \le t \le T$, and for each vertex y,*

$$d_Z(y, t) = \min_{\{x \mid (x,y) \in E\}} \gamma_Z(x, y, t).$$

We describe our algorithm for solving the TCSP-ZWT problem in Figure 3. Note that we have to evaluate $\gamma_Z(x, y, t)$ for all $(x, y) \in E$ when $d_Z(y, t)$ is to be updated. Again, to compute $\gamma_Z(x, y, t)$ as efficiently as possible, our idea in the algorithm TCSP-ZWT is to use sorting. We sort in advance the values of $u + b(x, y, u) = t$ for all $u = 1, 2, \ldots, T$ and all arcs $(x, y) \in E$.

The following lemma gives the worst-case running time of the algorithm TCSP-ZWT. The proof of the lemma is similar to that for Lemma 2 and is omitted here.

**Lemma 4.** *The algorithm TCSP-ZWT can be implemented such that it runs in $O(T(n + m))$ time.*

Combining Lemma 3, Corollary 2, and Lemma 4, we have

**Theorem 2.** *The TCSP-ZWT problem with positive transit times can be optimally solved in $O(T(n + m))$ time.*

## 5. BOUNDED WAITING TIMES

In this section, we consider the case where waiting at a vertex is allowed, but there is an upper-bound $\mathcal{U}_x$ on the waiting time at a vertex $x$. We refer to this problem as the *Time-Varying Constrained Shortest Path with Constrained Waiting Times problem* (TCSP-CWT problem). Again, we assume that all $b(x, y, u)$ are positive integers. Recall the definition of $\mathcal{U}_x$ (Definition 1). Clearly, we may assume that $\mathcal{U}_x \le T$ for all $x \in V$. Recall also Definitions 2, 3, 4, and 5 on departure times at vertices, time of path, and length of path.

*Algorithm TCSP-ZWT*
**begin**
**Initialize** $d_Z(s,0) = 0$, and $d_Z(x,0) = \infty$ for all $x \neq s$ and $t = 0, 1, ..., T$
**Sort** all values $u + b(x,y,u)$ for all $u = 1, \ldots, T$ and for all arcs $(x,y) \in E$
**For** $t = 1, \ldots, T$ **do**
    **For** every arc $(x,y) \in E$ **do** $\Upsilon_Z(x,y,t) := \infty$
    **For** all arcs $(x,y) \in E$ and all $u$ such that $u + b(x,y,u) = t$ **do**
        $\Upsilon_Z(x,y,t) := \min\{\Upsilon_Z(x,y,t), d_Z(x,u) + c(x,y,u)\}$
    **For** every vertex $y$ **do** $d_Z(y,t) := \min_{(x,y)\in E} \Upsilon_Z(x,y,t)$
**For** every vertex $y$ **do** $d_Z^*(y) = \min_{0\leq t\leq T} d_Z(y,t)$
**end.**

**Fig. 3.** Algorithm TCSP-ZWT.

**Definition 10.** *$d_C(x, t)$ is the length of a shortest feasible path from $s$ to $x$ of time exactly $t$ and with waiting time zero at $x$, subject to the constraint that the waiting time at any other vertex $y$ on the path is not greater than $\mathcal{U}_y$. If such a feasible path does not exist, then $d_C(x, t) = \infty$.*

**Definition 11.** *$d_C^*(x)$ is the length of a shortest feasible path from $s$ to $x$ of time at most $T$.*

**Lemma 5.**

$$d_C^*(x) = \min_{0\leq t\leq T} d_C(x, t).$$

*Proof.* Consider a shortest feasible path $P$ of time at most $T$. Let $t \leq T$ be the time of $P$, with waiting time zero at $x$. Then, $d_C^*(x) = d_C(x, t)$. ■

**Lemma 6.** *$d_C(s, 0) = 0$ and $d_C(x, 0) = \infty$ for all $x \neq s$. For $t > 0$ and $y \neq s$, we have*

$$d_C(y, t) = \min_{\{x\,|\,(x,y)\in E\}} \min_{(u_A,u_D)\in\mathcal{F}(x,y,t)} \{d_C(x, u_A) + c(x, y, u_D)\},$$

*where $\mathcal{F}(x, y, t) = \{(u_A, u_D) | u_D + b(x, y, u_D) = t \wedge 0 \leq u_D - u_A \leq \mathcal{U}_x\}$.*

*Proof.* It is easy to see that $d_C(s, 0) = 0$ and $d_C(y, 0) = \infty$ for all $y \neq s$, since all transit times are positive. Thus, in the following, we need only examine $t > 0$ and $y \neq s$.

We prove the formula by induction. Consider $t = 1$. The only vertices for which there exists a feasible path of time one are neighbors of $s$. Assume that $y$ is a neighbor of $s$. We must have $b(s, y, 0) = 1$ and all waiting times must be zero. In that case, the formula holds with $u_A = u_D = 0$ and $x = s$.

Assume that the formula is correct for all $t' < t$. Consider a vertex $y$. First, let us prove the claim that *there exists a feasible path of time $t$ and length $d_C(y, t)$, with waiting time zero at vertex $y$.*

If $d_C(y, t) = \infty$, there is nothing to prove. So, assume that $d_C(y, t)$ is finite. Assume that $d_C(y, t) = d_C(x, u_A) + c(x, y, u_D)$ for some $x$ such that $(x, y) \in E$ and some $(u_A, u_D) \in \mathcal{F}(x, y, t)$.

By induction, we know that there is a feasible path $P' = (s = x_1, \ldots, x_r = x)$ from $s$ to $x$ of time exactly $u_A$, with length $d_C(x, u_A)$ and with zero waiting time at $x$. We let $u_D - u_A$ be the new waiting time at $x$. Since $0 \leq u_D - u_A \leq \mathcal{U}_x$, the new path is again feasible. We extend the path with vertex $y$, obtaining a path $P$ with the given waiting times and with waiting time zero at $y$. The time of $P$, with these waiting times, is exactly $t$, since $u_D + b(x, y, u_D) = t$, which is the arrival time at $y$. The length of $P$ with these waiting times is $d_C(x, u_A) + c(x, y, u_D) = d_C(y, t)$. This proves the claim.

We now prove that $d_C(y, t)$ is the length of a shortest feasible path from $s$ to $y$ with time $t$ and with waiting time zero at $y$. Let $P = (s = x_1, \ldots, x_r = y)$ be a shortest feasible path from $s$ to $y$ of time $t$ with waiting time zero at $y$. Let $w(x_i)$ be the waiting time at $x_i$ $(i = 1, \ldots, r)$. So, we have $w(x_r) = 0$. Let $x$ be the predecessor of $y$ on this path. Let $u_D$ be the time of the subpath $P'$ (with the waiting times) from $s$ to $x$, let $u_A = u_D - w(x)$ be the arrival time at $x$ along $P'$, and let $\mathcal{L}(x)$ be the length of $P'$. By definition, $t = u_D + b(x, y, u_D)$. By induction, $\mathcal{L}(x) \geq d_C(x, u_D)$. By definition, the length of $P$ is $\mathcal{L}(x) + c(x, y, u_D) \geq d_C(x, u_D) + c(x, y, u_D) \geq d_C(y, t)$, where the last inequality comes from the formula on the computation of $d_C(y, t)$. This length must be equal to $d_C(y, t)$, since $P$ is a path of the shortest possible length and since there exists a path that achieves $d_C(y, t)$, as we showed above. This completes the proof. ■

**Definition 12.** *For each arc $(x, y) \in E$ and each $1 \leq t \leq T$, define*

$$\gamma_C(x, y, t) = \min_{(u_A,u_D)\in\mathcal{F}(x,y,t)} \{d_C(x, u_A) + c(x, y, u_D)\},$$

*and adopt the convention that $\gamma_C(x, y, t) = \infty$ whenever the set $\mathcal{F}(x, y, t)$ is empty.*

*Algorithm TCSP-CWT*
**begin**
**Initialize** $d_C(s,0) := 0$ and $\forall_{x \neq s} d_C(x,0) = \infty$; $\forall_x \forall_{t>0}$ **Initialize** $d_C(x,t) := \infty$;
$\forall_x$ **Initialize** $Heap_x := \{d_C(x,0)\}$ and $d_C^m(x,0) := d_C(x,0)$
**Sort** all values $u + b(x,y,u)$ for all $u = 1, \ldots, T$ and for all arcs $(x,y)$
**For** $t = 1, \ldots, T$ **do**
  **For** every arc $(x,y)$ **do** $\Upsilon_C(x,y,t) := \infty$
  **For** all arcs $(x,y)$ and all $u_D$ such that $u_D + b(x,y,u_D) = t$ **do**
    $\Upsilon_C(x,y,t) := \min\{\Upsilon_C(x,y,t), d_C^m(x,u_D) + c(x,y,u_D)\}$
  **For** every vertex $y$ **do** $d_C(y,t) = \min_{\{x|(x,y)\in E\}} \Upsilon_C(x,y,t)$
  **For** every vertex $y$ **update** the heap as follows
    **Insert-heap**$_{(y)}$ $d_C(y,t)$
    **If** $t > \mathcal{U}_y$ **then delete-heap**$_{(y)}$ $d_C(y, t - \mathcal{U}_y - 1)$
  **For** every vertex $y$ **do**
    $u_A :=$**Minimum-heap**$_{(y)}$
    $d_C^m(y,t) := d_C(y,u_A)$
**For** every vertex $y$ **do** $d_C^*(y) = \min_{0 \le t \le T} d_C(y,t)$
**end.**

**Fig. 4.** Algorithm TCSP-CWT.

From Lemma 6, we have

**Corollary 3.** *For $1 \le t \le T$, and for each vertex $y$,*

$$d_C(y,t) = \min_{\{x|(x,y)\in E\}} \gamma_C(x,y,t).$$

In addition to the idea of sorting the values of $u + b(x,y,u)$ as discussed previously, our key idea in the algorithm to be presented below is the use of a binary heap. For every vertex $x$, we maintain a binary heap, which contains the values of $d_C(x,u_A)$ for all $\max\{0, t - \mathcal{U}_x\} \le u_A \le t$. Using this data structure, initialization and finding the minimum take constant time. Each insertion and each deletion take $O(\log \mathcal{U}_x) = O(\log T)$ time [3]. For convenience, we introduce the following notation:

**Definition 13.** *$d_C^m(x,t)$ is the minimum in the heap.*

We need $d_C^m(x,t)$ when evaluating $\gamma_C(x,y,t)$. We see from Definition 12 that we have to solve an optimization problem of minimizing $\{d_C(x,u_A) + c(x,y,u_D)\}$ subject to $(u_A,u_D) \in \mathcal{F}(x,y,t)$ to obtain $\gamma_C(x,y,t)$. Clearly, given $(x,y)$ and $t$, a value of $u_D$ that satisfies $u_D + b(x,y,u_D) = t$ is known and, consequently, the corresponding value of $c(x,y,u_D)$ is known. Thus, solving the optimization problem reduces to solving a problem of minimizing $\{d_C(x,u_A)\}$ subject to $\max\{0, u_D - \mathcal{U}_x\} \le u_A \le u_D$ [recall the definition of $\mathcal{F}(x,y,t)$ in Lemma 6]. Therefore, if $d_C^m(x,u_D)$ is known, we can obtain, for every $(x,y) \in E$ and $t$, $\gamma_C(x,y,t)$, which is equal to the minimum of $d_C^m(x,u_D) + c(x,y,u_D)$ over all $u_D$ satisfying $u_D + b(x,y,u_D) = t$.

We describe our algorithm for the TCSP-CWT problem in Figure 4. Note that Algorithm TCSP-CWT computes iteratively $d_C(y,t)$ for all $y$ at $t = 0, 1, \ldots, T$. At any time $t$, the algorithm keeps all $d_C^m(y,u)$ for all vertices $y$ and all $u \le t - 1$. Nevertheless, for each vertex $y$, the algorithm maintains only one heap $Heap_y$. After $d_C(y,t)$ is obtained, the new $Heap_y$ at time $t$ is obtained by deleting $d_C(y, t - \mathcal{U}_y - 1)$ from the heap (if $t - \mathcal{U}_y - 1 \ge 0$) and inserting $d_C(y,t)$.

The following lemma is needed to show the correctness of the algorithm:

**Lemma 7.** *After the termination of the algorithm TCSP-CWT, $d_C(y,t)$ is the length of a shortest feasible path from $s$ to $y$ of time exactly $t$ and with waiting time zero at the vertex $y$.*

*Proof.* We show that the formula given in Lemma 6 is correctly computed. Clearly, it suffices to show that $d_C^m(x,u)$, for all $0 \le u \le t$, computed by the algorithm is the minimum value of $d_C(x,u_A)$, for $u - \mathcal{U}_x \le u_A \le u$. We use induction on $t$.

The argument holds for $t = 0$ because of the initialization. Now assume that the argument holds for any $0 \le u \le t - 1$, and we consider $u = t$. Note that $u$ subject to $u + b(x,y,u) = t$ in line 7 of the algorithm must satisfy $u \le t - 1$ since $b(x,y,u)$ is positive. Thus, line 8 of the algorithm gives the correct value for $\gamma_C(x,y,t)$ because of the assumption that $d_C^m(x,u)$ are correct for all $u \le t - 1$. In addition, when $\gamma_C(x,y,t)$ is correct, then $d_C(y,t)$ is correct according to Corollary 3. Consequently, lines 11 and 12 of the algorithm generate the correct $Heap_y$ at time $t$, and therefore $d_C^m(x,t)$ is correct. This completes the proof. ■

**Lemma 8.** *The algorithm TCSP-CWT can be implemented such that it runs in $O(T(m + n \log T))$ time.*

*Proof.* It is easy to see that the initialization can be done in $O(Tn)$ time.

For the sorting, we can use bucketsort, with $T$ buckets. Since there are $Tm$ values to be sorted, this step can then be implemented in $O(Tm)$ time.

Since the values $u + b(x, y, u)$ are now sorted, the overall time needed to update the values $\gamma_C(x, y, t)$ is $O(Tm)$.

The two steps of inserting $d_C(y, t)$ to $Heap_y$ and deleting $d_C(y, t - \ell_y - 1)$ (if $t > \ell_y$) from $Heap_y$ take $O(\log \ell_y) = O(\log T)$ time. Since the algorithm has to perform these two steps for all $t = 1, 2, \ldots, T$ and all vertex $y \in V$, it takes in total $O(Tn \log T)$ time to maintain the heaps.

The step of finding $d_C^m(y, t)$ takes $O(1)$ time. Finally, the last step of computing $d^*(y)$ for all $y \in V$ takes $O(Tn)$ time.

It follows that the overall running time of the algorithm is bounded above by $O(T(m + n \log T))$. ■

Combining Lemma 7 with Lemma 8, we obtain

**Theorem 3.** *The TCSP-CWT problem with positive transit times can be optimally solved in $O(T(m + n \log T))$ time.*

## 6. TAKING CARE OF THE ZEROS

In this section, we propose an approach to handle zero transit times. The approach holds for all problems, TCSP-AWT, TCSP-ZWT, and TCSP-CWT. In the following, we describe it in detail for the TCSP-AWT problem. The particulars of the approach for other problems can be similarly derived following the same idea.

Consider a network $G = (V, E)$. At the $t$th step of the algorithm TCSP-AWT, we first apply, as usual, the algorithm to a subgraph $G' = (V, E')$. This subgraph $G'$ has the same vertex set $V$ as $G$, but its edge set $E' = \{e \mid e \in E \wedge b(e, t) > 0\}$. Then, after the values of $d_A(y, t)$, namely, the lengths of the shortest paths from $s$ to each vertex $y$, $y \in V$, have been obtained by the algorithm TCSP-AWT, we create, for each $y \in V$, an artificial arc from $s$ to $y$. Call this arc $e'_y = (s, y)$. The length of $e'_y = (s, y)$ is set to $d_A(y, t)$, namely, $c(e'_y, t) = d_A(y, t)$, and the transit time on $e'_y = (s, y)$ is assumed to be $t$. Then, we construct a new subgraph $G'' = (V, E'')$. The vertex set $V$ of the subgraph $G''$ is the same as that of $G$. The edge set $E''$ consists of those edges $e$ for which $b(e, t) = 0$ and those edges $e'_y$ for all $y \in V \setminus \{s\}$. If there are double arcs from $s$ to $y$, delete the arc from $E''$ which has the larger length (or break up a tie arbitrarily if they have equal lengths).

When the subgraph $G''$ is created, we apply a ''common'' shortest path algorithm (say, Dijkstra's algorithm, see [1, 4]) to $G''$ to find the shortest path from $s$ to each $y \in V$. In applying such an algorithm, we ignore the transit times and the problem is thus a classical shortest path problem. For completeness, we describe the application of Dijkstra's algorithm below and refer to it as **SP**.

The algorithm maintains two sets $S$ and $S'$. The set $S$ contains vertices for which the final shortest path lengths have been determined, while the set $S'$ contains vertices for which upper bounds on the final shortest path lengths are known. Initially, $S$ contains only the source $s$, and the lengths of the vertices in $S'$ are set to $d_A(y, t)$. Repeatedly, select the vertex $x \notin S$, for which the distance from $s$ is the shortest. Put $x$ in $S$, and for all outgoing arcs $e = (x, y) \in E''$, update $d_A(y, t) := \min\{d_A(y, t), d_A(x, t) + c(e, t)\}$. The algorithm terminates if all vertices are in $S$.

We are going to show that our approach is correct in terms of finding an optimal solution at each time $t$ for the original problem TCSP-AWT. For any vertex $y \in V$ and time $t > 0$, we can see that any path from $s$ to $y$ with time at most $t$ must be one of the paths of the following type:

1. A path from $s$ to $y$ of time at most $t - 1$,
2. A path from $s$ to $y$ of time exactly $t$, which must pass an arc $(x, y) \in E'$ with $b(x, y, t) > 0$, or
3. A path from $s$ to $y$ of time exactly $t$, which must pass an arc $(x, y) \in E''$ with $b(x, y, t) = 0$.

In fact, for each vertex $y \in V$, our approach first uses the algorithm TCSP-AWT to determine the shortest path among those of types 1 and 2. After this is done, it creates an artificial arc $e'_y = (s, y)$ to represent this shortest path. Then, it uses the procedure SP to further determine the shortest length of all possible paths. The shortest path can be one with only the artificial arc $e'_y$ (in this case the algorithm TCSP-AWT had, in fact, found the optimum) or a path of type 3 (in this case, the procedure SP has found a shorter length than that obtained by the algorithm TCSP-AWT). Since any vertex $y$ can only be reached by one of the paths of the three types, the approach has considered all possible paths and is thus optimal. Formally, we have

**Lemma 9.** *Consider the approach: At each $t = 0, 1, \ldots, T$, apply the algorithm TCSP-AWT to $G'$, then apply the procedure SP to $G''$ to update $d_A(y, t)$ for all $y \in V$. After the $t^{th}$ iteration, $d_A(y, t)$ is the length of a shortest path from $s$ to $y$ of time at most $t$.*

*Proof.* With induction on $t$, we now show that $d_A(x, t)$ is the length of a shortest path from $s$ to $x$ of time at most $t$.

When $t = 0$, the algorithm TCSP-AWT first initializes $d_A(s, 0) = 0$ and $d_A(x, 0) = \infty$ for all $x \neq s$. Then, a subgraph $G''$ is created, and the procedure SP is applied to this graph. Clearly, this procedure can correctly obtain, for each $x \in V$, the length of a shortest path from $s$ to $x$ in the graph $G''$. Hence, the values for $d_A(x, t)$ are correct for $t = 0$.

Now assume that the values of $d_A(x, t')$ are correct for all $x \in V$ and $t' < t$. Under this assumption, it is easy to show that the values for $d_A(x, t)$ obtained by the algorithm TCSP-AWT are the lengths of shortest paths of types 1 and 2. Now consider the subgraph $G''$ created with artificial arcs $e'_y = (s, y)$ associated with these lengths. As the procedure SP is, in fact, the algorithm of Dijkstra, it can find the length of a shortest path $P$ from $s$ to $y$ in the graph $G''$, for each $y \in V$. Moreover, the time of this path is at most $t$, since all arcs except those artificial arcs in $G''$ have zero transit times. The artificial arcs in $G''$ have a transit time $t$, but all of them originate from $s$ and thus any path from $s$ to $y$ in $G''$ can contain at most one such arc. By the notation of our approach, $d_A(y, t)$ (updated by the procedure SP) is the length of this shortest path $P$.

We now claim that $d_A(y, t)$ is also the length of the shortest path from $s$ to $y$ of time at most $t$ in the original graph $G$. Suppose that this is not true, namely, there exists another path $\hat{P}$ from $s$ to $y$, which has a length $\mathcal{L}_{\hat{P}} < d_A(y, t)$. Clearly, this cannot be a path of type 1, 2, or 3; otherwise, such a path would have implied that the procedure SP, namely, Dijkstra's algorithm, is not optimal. The only possibility is that $\hat{P}$ is a path with time greater than $t$, which is, however, infeasible for the given $t$. This proves the claim, and therefore the lemma. ■

For each $t = 0, 1, 2, \ldots, T$, the subgraphs $G'$ and $G''$ can be constructed in $O(m + n)$ time, and the procedure SP can be implemented such that it runs in $O(m + n \log n)$ time (see [1]). This is the additional running time needed to update the solutions obtained by the algorithm TCSP-AWT, TCSP-ZWT, or TCSP-CWT. In summary, we have

**Theorem 4.** *The problems TCSP-AWT, TCSP-ZWT, and TCSP-CWT with non-negative transit times can be optimally solved in times* $O(T(m + n \log n))$, $O(T(m + n \log n))$, *and* $O(T(m + n \log T + n \log n))$, *respectively.*

## 7. CONCLUSIONS

In this article, we considered a new extension of the ordinary shortest path problem. Three variants were examined in detail. Algorithms with pseudopolynomial time complexity were proposed, which can solve the problems optimally.

Of course, there are many variations of the problems discussed in this paper. For example, one could study the problem where a *speedup* is also allowed. The transit time of an arc may be shortened, which, however, may incur an additional cost. In such a case, both the transit times on arcs and the waiting times at vertices become decision variables. One should determine not only an optimal path, but also the optimal transit times on the arcs and the waiting times at the vertices along the path.

The authors wish to express their gratitude to the referees for their valuable comments, which improve the presentation of the paper significantly.

## REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ (1993).

[2] Y. P. Aneja and K. P. K. Nair, The constrained shortest path problem. *Naval Res. Logist. Qt.* **25** (1978) 549–555.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA (1990).

[4] E. W. Dijkstra, A note on two problems in connection with graphs. *Numer. Math.* **1** (1959) 269–271.

[5] G. Y. Handler and I. Zang, A dual algorithm for the constrained shortest path problem. *Networks* **10** (1980) 293–310.

[6] M. M. D. Hassan, Network reduction for the acyclic constrained shortest path problem. *Eur. J. Oper. Res.* **63** (1992) 121–132.

[7] H. C. Joksch, The shortest route problem with constraints. *J. Math. Anal. Appl.* **14** (1966) 191–197.

[8] C. C. Skiscim and B. L. Golden, Solving *k*-shortest and constrained shortest path problems efficiently. *Ann. Oper. Res.* **20** (1989) 249–282.

[9] C. Witzgall and A. J. Goldman, Most profitable routing before maintenance. *Bull. Oper. Res. Soc. Am.* (1965) B82.