Finding Minimum Cost to Time Ratio Cycles with Small Integral Transit Times

Mark Hartmann

Department of Operations Research, University of North Carolina, Chapel Hill, NC 27599

James B. Orlin

Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139

Let $D = (V, E)$ be a digraph with n vertices and m arcs. For each $e \in E$ there is an associated cost c_e and a transit time t_e; c_e can be arbitrary, but we require t_e to be a non-negative integer. The *cost to time ratio* of a cycle C is $\lambda(C) = \Sigma_{e\in C} c_e/\Sigma_{e\in C} t_e$. Let $E' \subseteq E$ denote the set of arcs e with $t_e > 0$, let $T_u = \max\{t_{uv}: (u, v) \in E\}$ for each vertex u, and let $T = \Sigma_{u\in V} T_u$. We give a new algorithm for finding a cycle C with the minimum cost to time ratio $\lambda(C)$. The algorithm's $O(T(m + n \log n))$ running time is dominated by $O(T)$ shortest paths calculations on a digraph with non-negative arc lengths. Further, we consider early termination of the algorithm and a faster $O(Tm)$ algorithm in case $E - E'$ is acyclic, i.e., in case each cycle has a strictly positive transit time, which gives an $O(n^2)$ algorithm for a class of cyclic staffing problems considered by Bartholdi et al. The algorithm can be seen to be an extension of the $O(nm)$ algorithm of Karp for the case in which $t_e = 1$ for all $e \in E$, which is the problem of calculating a minimum mean cycle. Our algorithm can also be modified to solve the related parametric shortest paths problem in $O(T(m + n \log n))$ time.

1. INTRODUCTION

Let $D = (V, E)$ be a digraph with n vertices and m arcs. A *walk* is a forwardly directed arc progression from an initial vertex to a terminal vertex. A *cycle* is a walk in which the initial vertex is equal to the terminal vertex. A *path* is a walk in which no vertex is repeated. For each $e \in E$ there is an associated cost c_e and a transit time t_e; c_e can be arbitrary, but we require t_e to be a non-negative integer. Let $E' \subseteq E$ denote the set of arcs e with $t_e > 0$, let $T_u = \max\{t_{uv}: (u, v) \in E\}$ for each vertex u, and let $T = \Sigma_{u\in V} T_u$. Note that T is an upper bound on the transit time of each path and cycle in D.

The *cost to time ratio* $\lambda(C)$ of a cycle C is $\Sigma_{e\in C} c_e/\Sigma_{e\in C} t_e$. Let $\lambda^* = \min_C \lambda(C)$, where C ranges over all cycles in D; λ^* is called the *minimum cost to time ratio*. If we adopt the convention that the minimum over the empty set is ∞ and that $c/0 = \pm\infty$ depending on whether $c \geq 0$ or $c < 0$, then λ^* can be interpreted as the largest value of λ for which the digraph D has no cycles of negative length with respect to the lengths $c_e(\lambda) = c_e - \lambda t_e$ for $e \in E$. (Throughout the paper, we will refer to $c_e(\lambda)$ as the length of arc e, even if $c_e(\lambda) < 0$.) If λ^* is finite, then we call any cycle C with $\lambda(C) = \lambda^*$ a *minimum cost to time ratio cycle*.

The problem of finding a minimum cost to time ratio cycle has previously been studied by Burns [4], Dantzig et al. [5], Fox [8], Golitschek [12, 13], Hartmann [15], Ishii et al. [16], Karp [17], Karp and Orlin [18], Karzanov [19], Lawler [20, 21] and Megiddo [23]. In the final section of this paper, we discuss an application of the

NETWORKS, Vol. 23 (1993) 567–574

CCC 0028-3045/93/060567-08

minimum cost to time ratio cycle problem with small integral transit times given in Bartholdi et al. [2]. Other applications of the minimum cost to time ratio cycle problem can be found in Boros et al. [3], Burns [4], Dantzig et al. [5], Graves and Orlin [14], Lockyear and Ebeling [22], Orlin and Rothblum [24], Richey [25] and Spaelti and Liebling [26]. In most of these applications, the transit times are small integers and each cycle has a strictly positive transit time.

The main contribution of this paper is an improvement and extension of the $O(n^3)$ algorithm developed by Karp and Orlin for solving a special case of the minimum cost to time ratio cycle problem. It has the following features:

1. It finds the minimum cost to time ratio λ^*, and if λ^* is finite, it also determines a minimum cost to time ratio cycle.
2. The algorithm solves the problem as $O(T)$ shortest paths problems on a related digraph, with a running time of $O(T(m + n \log n))$. In case that each cycle has a strictly positive transit time, the running time can be further improved to $O(Tm)$.
3. In many cases, the algorithm terminates far earlier than indicated by the worst-case running time; this feature improves upon algorithms by Karp [17] and Karp and Orlin [18]
4. The algorithm can be applied to the cyclic staffing problem, and improves the best previous running time.

This also improves on Megiddo's $O(n^2 m \log n)$ algorithm for the general minimum cost to time ratio cycle problem when the transit times are integers which are not much larger than n.

Under the assumptions that $E - E'$ contains no negative-cost cycle, $t_e = 1$ for all $e \in E'$ and there is a path from s to each $v \in V$, Karp and Orlin [18] give $O(n^3)$ and $O(nm \log n)$ algorithms for computing the length of the shortest paths from s to each vertex $v \in V$ parametrically in λ, where the length of an arc e is $c_e(\lambda)$. As a byproduct, these algorithms yield the value λ^*. Young et al. [27] use Fibonacci heaps to improve the running time of the second algorithm to $O(nm + n^2 \log n)$. Our algorithm can be modified to solve this more general problem in $O(T(m + n \log n))$ time.

2. PRELIMINARY REDUCTIONS

Dantzig et al. [5] formulate the minimum cost to time ratio cycle problem as the linear programming problem

$$\begin{aligned} &\text{minimize } \textstyle\sum_{e\in E} c_e x_e \\ &\text{subject to } \textstyle\sum_{e\in E} t_e x_e = 1 \\ &\qquad Ax = 0 \\ &\qquad x \geq 0, \end{aligned} \tag{1}$$

where A is the vertex-arc incidence matrix of D. The dual of this problem is the linear programming problem

$$\begin{aligned} &\text{maximize } \lambda \\ &\text{subject to } \lambda t_{(u,v)} - \pi_u + \pi_v \leq c_{(u,v)} \\ &\qquad \text{for all } (u, v) \in E. \end{aligned} \tag{2}$$

The minimum cost to time ratio λ^* can be computed as the optimal value of (1) or (2), and it is possible to obtain a minimum cost to time ratio cycle from an optimal solution to either problem.

If λ^* and π^* are an optimal solution to (2), we can find a cycle C with $\lambda(C) = \lambda^*$ in $O(n + m)$ time as follows. First note that (2) implies that $\lambda(C) = \lambda^*$ if and only if $C \cap E' \neq \emptyset$ and $C \subseteq E''$, where $E'' \subseteq E$ is the set of arcs (u, v) with $c_{(u,v)} = \lambda^* t_{(u,v)} - \pi_u^* + \pi_v^*$. To find an arc $(u, v) \in E'$ which lies on such a cycle, we first find the strongly connected components of the digraph with arc set $E'' - E'$ in $O(n + m)$ time and look for an arc $(u, v) \in E'' \cap E'$ with both u and v in the same strongly connected component. If there is no such arc, we contract the strongly connected components. Then any directed cycle $C \subseteq E''$ must contain an arc $(u, v) \in E'$. We can find a path P from v to u consisting of arcs in E'' in $O(n + m)$ time; adding (u, v) to P yields a minimum cost to time ratio cycle. (If instead we have an optimal solution x^* to (1), we can let E'' consist of those arcs e with $x_e^* > 0$.)

The dual constraints (2) can also be written as $\pi_v \leq \pi_u + c_{(u,v)}(\lambda)$ for all $(u, v) \in E$, the familiar necessary conditions for the shortest paths problem with lengths $c_e(\lambda)$. Therefore if $\lambda > \lambda^*$ there will be a negative-length cycle, and if $\lambda \leq \lambda^*$ the shortest path distances from a vertex s yield a dual feasible π. Lawler [20, 21] used this as the basis for a polynomial algorithm which computes λ^* by binary search. We will use (2) to terminate early in our algorithm, by attempting to construct a dual feasible π corresponding to the cost to time ratio $\lambda(C)$ of a cycle C. Before describing the algorithm, we will give some general reductions of the problem.

First of all, if the digraph D is not strongly connected, we can find its strongly connected components in $O(n + m)$ time, find a minimum cost to time ratio cycle in each component separately, and then choose the one with the smallest ratio. A better strategy would be to consider the current minimum cost to time ratio

λ, since a minimum cost to time ratio cycle would only need to be found in the current component if it had a cycle with cost to time ratio less than λ. This can be checked using the $O(nm)$ time shortest paths algorithm described in Goldfarb et al. [11], which allows for early termination if a negative-length cycle is detected. If we process p strong components in random order, then on the average we would only need to compute $O(\log p)$ minimum cost to time ratio cycles (see Dynkin and Yushkevich [6, pp. 87–89]). Henceforth we may assume that D is strongly connected.

We may also assume that $c_e \geq 0$ for all $e \in E$. To justify this second assumption, consider the graph with arc set $E - E'$. For all $v \neq s$, add an arc (s, v) with large cost. The $O(nm)$ algorithm of Goldfarb et al. can then be used to find the minimum cost d_v of a path from s to each vertex v in this digraph, unless a negative-cost cycle is detected in which case $\lambda^* = -\infty$. If $E - E'$ is acyclic and s is chosen to be a source, then we can find d_v for $v \in V$ in $O(m)$ time. If the digraph does not contain a negative-cost cycle, then these minimum costs will satisfy $d_v \leq d_u + c_{(u,v)}$ for all $(u, v) \in E - E'$. Then we set $\hat{\lambda} = \min_{(u,v)\in E'}\{(c_{(u,v)} + d_u - d_v)/t_{(u,v)}\}$ so that $\hat{c}_{(u,v)} = c_{(u,v)}(\hat{\lambda}) + d_u - d_v \geq 0$ for all $(u, v) \in E$. It is easy to see that a cycle with cost to time ratio λ with respect to the costs $\hat{c}_e$ has cost to time ratio $\lambda + \hat{\lambda}$ with respect to the costs c_e. Finally, we may assume that $E' \neq \emptyset$, for otherwise $\lambda^* = +\infty$.

3. MINIMUM CYCLE MEANS AND 0–1 TRANSIT TIMES

When $t_e = 1$ for all $e \in E$, the cost to time ratio of a cycle C is simply $1/|C| \sum_{e\in C} c_e$, the cycle mean. The problem of computing the minimum cycle mean arises in connection with the minimum cost flow problem (see Ahuja et al. [1], Engel and Schneider [7] and Goldberg and Tarjan [10]). Karp [17] gave an $O(nm)$ algorithm for finding the minimum cycle mean, based on the following characterization for finite λ^*:

$$\lambda^* = \min_{v\in V}\left\{\max_{0\leq k\leq n-1}\left[\frac{G_n(v) - G_k(v)}{n - k}\right]\right\}, \qquad (3)$$

which holds under the assumption that there is a path from s to every other vertex. Here $G_k(v)$ is the minimum cost of a walk from s to v with transit time exactly k. If no such walk exists, then we say that $G_k(v) = \infty$. The quantities $G_k(v)$ may be computed using the following recursive scheme:

$$G_k(v) = \min_{uv\in E}\{G_{k-1}(u) + c_{(u,v)}\} \quad \text{for all } v \in V \qquad (4)$$

for $k = 1, \ldots, n$ subject to the initial conditions $G_0(s) = 0$ and $G_0(v) = \infty$ for $v \neq s$. The main drawback of Karp's algorithm is that the best-case and worst-case running times are the same.

Next we explain how to terminate early in Karp's algorithm after computing $G_0(v), \ldots, G_k(v)$ for all $v \in V$ for some $k < n$. At this point, minimum cost walks have been determined from s to v with transit time exactly j for all $v \in V$ and $j = 0, 1, \ldots, k$. Many of these walks will contain cycles, and if C has the minimum cost to time ratio of all cycles detected in these walks, then we compute the value $\lambda = \lambda(C)$ and

$$\pi_v^k(\lambda) = \min_{j=0,1,\ldots,k}\{G_j(v) - \lambda j\} \quad \text{for all } v \in V. \qquad (5)$$

If λ and $\pi^k(\lambda)$ satisfy the dual constraints (2), then C must be a minimum mean cycle, and the algorithm can terminate early.

To detect cycles, we maintain a *predecessor digraph* $D^< = (V^<, E^<)$, which grows as the algorithm progresses. After computing $G_k(v)$ for $v \in V$, the vertex set $V^<$ contains vertices v^j for all $v \in V$ and $j = 0, 1, \ldots, k$ and the arc set $E^<$ contains arcs (u^{j-1}, v^j) corresponding to the arcs (u, v) which achieved the minimum in (4) for $v \in V$ and $j = 1, 2, \ldots, k$. (To represent the predecessor digraph, we keep track of the unique predecessor u^{j-1} of each such vertex v^j.) After updating $D^<$, we follow predecessors back from each u^k for $u \in V$ until the walk contains a cycle. If we detect a cycle C which begins at v^j and ends at v^i for some $j > i$, then we compute the cost to time ratio $\lambda(C) = (G_j(v) - G_i(v))/(j - i)$. If C has the minimum cycle mean of all cycles detected for $u \in V$, we store the vertex v^j from which we can later recover C. Since we can detect a cycle and compute $\pi^k(\lambda)$ in $O(kn)$ time and check dual feasibility in $O(m)$ time, we can check for early termination at $k = 1, 2, 4, \ldots, 2^{\lfloor \log_2 n \rfloor}$ with at worst a small constant factor increase in running time.

Karp and Orlin [18] show that (3) also holds for 0–1 transit times, and this is the basis of their $O(n^3)$ algorithm. As is the case for Karp's algorithm, the best-case and worst-case running times of Karp and Orlin's algorithm are the same. They compute the quantities $G_k(v)$ using the following recursive scheme:

$$G'_k(v) = \min_{(uv)\in E'}\{G_{k-1}(u) + c_{(uv)}\} \quad \text{for all } v \in V \qquad (6)$$

$$G_k(v) = \min_{u\in V}\{G'_k(u) + d(u, v)\} \quad \text{for all } v \in V \qquad (7)$$

for $k = 1, \ldots, n$ subject to the initial conditions $G_0(v) = d(s, v)$. Here $G'_k(v)$ is the minimum cost of a walk from s to v with transit time exactly k such that the last arc of the walk is in E', and $d(u, v)$ is the minimum

cost of a path from u to v in D consisting solely of arcs in $E - E'$ (which is computed in advance).

Instead of using (7), $G_k(v)$ for $v \in V$ can be computed from $G'_k(v)$ for $v \in V$ by solving a shortest paths problem in a related digraph with non-negative arc lengths. Because we have assumed that $c_e \geq 0$ for all $e \in E$, $G_0(v)$ for $v \in V$ can be determined in $O(m + n \log n)$ time using the shortest paths algorithm described in Fredman and Tarjan [9], and in $O(m)$ time if $E - E'$ is acyclic. Next we show how to compute $G_k(v)$ for $v \in V$ given $G_j(v)$ for $v \in V$ and $j = 0, 1, \ldots, k - 1$. Let $D^* = (V^*, E^*)$, where V^* is obtained from V by adding a source node r. For each arc $(u, v) \in E - E'$, there is a corresponding arc in E^* with the same cost. For each node $v \in V$ with $G'_k(v) < \infty$, there is also an arc (r, v) with cost $G'_k(v)$; these quantities can be computed in $O(m)$ time using (6). It follows that $G_k(v)$ is the minimum cost of a path in D^* from r to v (if no such path exists, then $G_k(v) = \infty$). Since $c_e \geq 0$ for all $e \in E^*$, we can also determine $G_k(v)$ for $v \in V$ in $O(m + n \log n)$ time using Fibonacci heaps, although we could use any shortest paths algorithm for the computation. If $E - E'$ is acyclic, then so is E^* and we can determine $G_k(v)$ for $v \in V$ in $O(m)$ time (this was observed independently by Hartmann [15] and Ishii et al. [16]).

Early termination in Karp and Orlin's algorithm is complicated by the arcs in $E - E'$. We still compute $\pi^k(\lambda)$ for some $\lambda = \lambda(C)$, but recovering the cycle C requires $O(n + m)$ additional time. The arc set $E^<$ of the predecessor digraph must also be modified as follows. After computing $G_k(v)$ for $v \in V$, we add an arc to $E^<$ corresponding to each arc in the shortest paths tree T for D^*: for every arc $(r, v) \in T$ there is an arc (u^{k-1}, v^k) corresponding to the arc $(u, v) \in E'$ which achieved the minimum in (6) for v, and for every arc $(u, v) \in T$ with $u \neq r$ there is an arc (w^k, v^k), where (r, w) is the first arc on the path from r to v in T. (This modification ensures that each path in the predecessor digraph has $O(k)$ arcs.) The determination of C procedes as before, except that we may no longer be able to recover C by following predecessors back from v^j; however, we can easily obtain an arc $e \in C$ with $t_e = 1$. Then in $O(n + m)$ time we can find a minimum cost to time ratio cycle as described in Section 2.

4. MINIMUM COST TO TIME RATIO CYCLES

When the transit times are arbitrary non-negative integers, there is no analogue of n in (3) and hence no similar characterization. On the other hand, when the transit times are 0–1 it is not necessary to make use of (3); if we check for early termination when $k \geq n$, we will detect a cycle with $\lambda(C) = \lambda^*$ and $\pi^k(\lambda^*)$ will be a corresponding dual feasible solution. A similar result holds when the transit times are arbitrary non-negative integers, but we must modify the method used to detect cycles. The following theorem gives an upper bound on the number of shortest paths problems which need to be solved. The proof is similar to the proof of [17, Theorem 1].

Theorem. *If $D = (V, E)$ is a strongly connected digraph for which the minimum cost to time ratio λ^* is finite, then for some $v \in V$ and k with $T \leq k < 2T$, every minimum cost walk $W_k(v)$ from s to v with transit time exactly k contains a minimum cost to time ratio cycle. Further, every cycle contained in such a walk $W_k(v)$ is either a minimum cost to time ratio cycle or a zero-cost cycle in $E - E'$.*

Proof. First note that $W_k(v)$ is a minimum cost walk with transit time exactly k with respect to the costs c_e if and only if it is a minimum cost walk with transit time exactly k with respect to the costs $c_e(\lambda^*)$. Then since C is a minimum cost to time ratio cycle with respect to the costs c_e if and only if it is a minimum cost to time ratio cycle with respect to the costs $c_e(\lambda^*)$, we may assume that $\lambda^* = 0$.

So let C^* be a minimum cost to time ratio cycle and let v be a vertex in C^*. Since there can be no negative cycles, there must be a minimum cost path $P(v)$ from s to v, which therefore has transit time at most T. Since C^* is a zero-cost cycle, adding any number of repetitions of C^* to $P(v)$ results in a minimum cost walk from s to v. Then the fact that $C^* \cap E' \neq \emptyset$ implies that there must be a minimum cost walk $W(v)$ from s to v which has transit time k for some $T \leq k < 2T$. Now consider any minimum cost walk $W_k(v)$ from s to v with transit time exactly k. Since $k \geq T$, $W_k(v)$ must contain a cycle C with $C \cap E' \neq \emptyset$. Since $W_k(v)$ has the same transit time as $W(v)$, it too must be a minimum cost walk from s to v. This implies that C is a zero-cost cycle, and so C must be a minimum cost to time ratio cycle. For the same reason every cycle contained in $W_k(v)$ must be a zero-cost cycle, which gives the desired result. ■

This theorem guarantees that we will detect a minimum cost to time ratio cycle if we follow pointers back from each u^k for $u \in V$ until the walk contains a cycle, but when the transit times are not 0–1 we cannot determine k in advance. (If the transit times are 0–1 we can take $k = n$, since we could instead follow $W(v)$ until we obtained a minimum cost walk W of transit time exactly n from s to some vertex $u \in V$.) When the transit times are arbitrary non-negative integers, we need to modify (6) to

$$G'_k(v) = \min_{(uv)\in E'} \{G_{k-t_{(u,v)}}(u) + c_{(uv)}\} \quad \text{for all } v \in V. \quad (8)$$

These quantities can also be computed in $O(m)$ time given the last T_u values of $G_j(u)$ for each $u \in V$, and the same shortest paths computation can be used to compute $G_k(v)$ for $v \in V$. Instead of storing $G_j(v)$ for each $v \in V$ and $j = 1, \ldots, k$ to determine λ^* by (3), we can recompute them as needed while exploring the predecessor digraph.

As was the case for 0–1 transit times, the arc set $E^<$ of the predecessor digraph contains an arc corresponding to each arc in the shortest paths tree T for D^*: for every arc $(r, v) \in T$ there is an arc $(u^{k-t_{(u,v)}}, v^k)$ for some (u, v) which achieved the minimum in (8), and for every arc $(u, v) \in T$ with $u \neq r$ there is an arc (u^k, v^k). (To represent the predecessor digraph, we now maintain for each vertex u^j in $V^<$ a list of arcs $(u^j, v^{j+t_{(u,v)}})$ in $E^<$.) It is not hard to see that if $G_k(v) < \infty$ there is a unique path from s^0 to v^k in $D^<$ whose image in D is a walk $W_k(v)$ which has transit time k and cost $G_k(v)$. It is also the case that the initial cycle contained in $W_k(v)$ must contain an arc $e \in E'$, since $D^<$ is acyclic. Although the predecessor digraph is built up as the algorithm progresses, we will show that many of the arcs and vertices of $D^<$ can be pruned after checking for early termination.

Now we address the question of finding the minimum cost to time ratio of a cycle which is contained in $W_j(v)$ for some $v \in V$ and $j = 0, 1, \ldots, k$ after determining $G_k(v)$ for $v \in V$. Since the initial parts of many of these walks will be the same, there should be some way to combine the search. Consider what happens when we carry out a depth first search from s^0 to all other vertices in the predecessor digraph $D^<$. The depth first search will grow a path rooted at s^0. Suppose u^j is the terminal vertex of P: the depth first search algorithm will try to add an unscanned vertex $v^{j+t_{(u,v)}}$ emanating from u^j to P; if there is no such unscanned vertex, then the algorithm will eliminate u^j from P and try to extend P from the predecessor of u^j in the same way.

Next we will show that this depth first search can be truncated as soon as the image of P in D contains a cycle. Suppose that C is the initial cycle encountered in one of these walks, and that C begins at v^i and ends at v^j for some $j > i$. Then v^i and v^j are the only copies of v on the path from s^0 to v^j in $D^<$. The theorem allows us to truncate the path P in the depth first search at v^j, because we know that for *some* $v \in V$ and $k < 2T$, the initial cycle encountered in $W_k(v)$ must be a minimum cost to time ratio cycle. Moreover, the cost to time ratio $\lambda(C) = (G_j(v) - G_i(v))/(j - i)$, which can be determined in $O(1)$ time provided that we store the transit time i and cost $G_i(v)$ of a copy v^i of each vertex v on the current path P.

We can therefore perform this truncated depth first search in $O(kn)$ time using $O(n)$ storage space in addition to that required to represent the predecessor digraph. After we have computed the minimum cost to time ratio $\lambda = \lambda(C)$ of an initial cycle C which is contained in $W_j(v)$ for some $v \in V$ and $j = 0, 1, \ldots, k$, we can determine $\pi^k(\lambda)$ by (5) in $O(kn)$ time using $O(n)$ additional space by performing another truncated depth first search in the predecessor digraph. It is unnecessary to perform a full depth first search, since if $\lambda > \lambda^*$ then the algorithm cannot terminate and if $\lambda = \lambda^*$ there must be a shortest *path* from s to v with lengths $c_e(\lambda^*)$. (More precisely, if t is the smallest transit time of such a path then the walk $W_t(v)$ found by the algorithm must also be a path.)

Since we can detect a cycle and compute $\pi^k(\lambda)$ in $O(kn)$ time and check dual feasibility in $O(m)$ time, if we check for early termination at $k = 1, 2, 4, \ldots, 2^{\lfloor \log_2 T \rfloor}$ then the running time will increase by at most $O(Tn + \log_2 T\, m)$. In case we are not able to terminate when $k = 2^{\lfloor \log_2 T \rfloor}$, then the theorem guarantees we will find the minimum cost to time ratio when $k = 2T - 1$. Since a path can have transit time at most T, $\pi_v^{2T-1}(\lambda^*)$ must then be the length of a shortest path from s to v with lengths $c_e(\lambda^*)$ and thus λ^* and $\pi^{2T-1}(\lambda^*)$ must satisfy (2). This yields an $O(T(m + n \log n))$ algorithm for finding the minimum cost to time ratio λ^*. Once we have determined λ^*, we can obtain a minimum cost to time ratio cycle in $O(n + m)$ time as described in Section 2.

The process of finding a dual feasible π corresponding to $\lambda = \lambda(C)$ can often be accelerated by mimicking a shortest paths algorithm as we perform the truncated depth first search. Initially we set $\hat{\pi}_s^k(\lambda) = 0$ and $\hat{\pi}_v^k(\lambda) = \infty$ for $v \neq s$. During the depth first search, as the arc $(u^j, v^{j+t_{(u,v)}})$ is examined, we set

$$\hat{\pi}_v^k(\lambda) = \min\{\hat{\pi}_v^k(\lambda), \hat{\pi}_u^k(\lambda) + c_{(u,v)}(\lambda)\}. \quad (9)$$

After the depth first search is completed, we will have $\hat{\pi}^k(\lambda) \leq \pi^k(\lambda)$, and some of the $\hat{\pi}_v^k(\lambda)$ may correspond to paths whose transit time exceeds k. Boros et al. [3] describe an application in which it suffices to compute $\lfloor \lambda^* \rfloor$ and a corresponding dual feasible π. In this case, we may be able to terminate even earlier by instead computing $\pi^k(\lfloor \lambda \rfloor)$ or $\hat{\pi}^k(\lfloor \lambda \rfloor)$.

If λ and $\pi^k(\lambda)$ or $\hat{\pi}^k(\lambda)$ do not satisfy (2), we can still profit from the depth first searches. First of all, if we store the initial cycle with the minimum cost to time ratio then it is only necessary to maintain those arcs of the predecessor digraph which lie on *paths* encountered in the truncated depth first search, since subsequent searches will not examine the other arcs. Further, it is possible to set $G_j(u) = \infty$ for all vertices u^j pruned from the predecessor digraph by deleting the

other arcs. This may lead to $G_k'(v) = \infty$ as determined by (4), and hence the removal of arcs (r, v) from $E^<$, which in turn may allow the shortest paths calculations to be simplified. Finally, if we keep track of the arcs achieving the minimum in (9) as in a shortest paths algorithm, these arcs may form a cycle $\hat{C}$ which would necessarily have $\lambda(\hat{C}) < \lambda(C)$.

5. PARAMETRIC SHORTEST PATHS

In this section, we show that if the algorithm for finding a minimum cost to time ratio cycle terminates early with $k < 2T$ after computing $\pi^k(\lambda)$, then by doing $O(kn \log n)$ postprocessing we can also compute the length of shortest paths from s to each vertex $v \in V$ with respect to the lengths $c_e(\lambda) = c_e - \lambda t_e$ parametrically in λ. More precisely, we create data structures requiring $O(kn)$ storage space such that given any $\lambda \leq \lambda^*$ the length of the shortest path from s to v with respect to the lengths $c_e(\lambda)$ can be recovered in $O(\log n)$ time, and the shortest path itself can be determined in $O(p + \log n)$ time if it has p arcs. A shortest paths tree with respect to $c_e(\lambda)$ can be recovered in $O(n \log n)$ time, as in Young et al. [27].

As a first step, we perform a final truncated depth first search in the pruned predecessor digraph in order to compute $G_j(v)$ for those $v \in V$ and $j = 0, 1, \ldots, k$ for which there is a path from s^0 to v^j. (If the predecessor digraph is represented by predecessors and the $G_j(v)$ are stored, this depth first search is unnecessary.) We will show that the length of a shortest path from s to v with lengths $c_e(\lambda)$ for any $\lambda < \lambda^*$ can be computed as the minimum of $G_j(v) - \lambda j$ over j for which there is a path from s^0 to v^j in the predecessor digraph.

First we will argue that $\pi_v^k(\lambda)$ is the length of a shortest path from s to v with lengths $c_e(\lambda)$ for any $\lambda < \lambda^*$. Suppose that $\pi_v^k(\lambda^*) = G_j(v) - \lambda^* j$. Then since $\pi_v^k(\lambda^*)$ is the length of a shortest path from s to v with lengths $c_e(\lambda^*)$, we must have $G_j(v) - \lambda^* j \leq G_i(v) - \lambda^* i$ for all $i > k$. Therefore,

$$G_j(v) - \lambda j < G_j(v) + \lambda^*(i - j) - \lambda i \leq G_i(v) - \lambda i$$

for all $i > k$, which shows that $\pi_v^k(\lambda) = \pi_v^{n-1}(\lambda)$, the length of a shortest path from s to v with lengths $c_e(\lambda)$. Finally, we must show that if $G_i(v)$ corresponds to a walk $W_i(v)$ which contains a cycle then i cannot achieve the minimum in (5). Since $\lambda < \lambda^*$, an initial cycle C contained in $W_i(v)$ must have $\Sigma_{e \in C} c_e(\lambda) > 0$, so removing C from $W_i(v)$ yields a walk $W(v)$ with transit time $t = i - \Sigma_{e \in C} t_e < i$ and cost $G_i(v) - \Sigma_{e \in C} c_e$. Therefore

$$G_t(v) - \lambda t \leq (G_i(v) - \Sigma_{e \in C} c_e) - \lambda(i - \Sigma_{e \in C} t_e) < G_i(v) - \lambda i,$$

which gives the desired conclusion.

Since we will be interested in determining the value $\pi_v^k(\lambda)$ as well as an index which achieves the minimum in (5), we will use a data structure from the algorithm of Megiddo [23] for determining the minimum of linear functions. For each vertex $v \in V$, we incorporate the linear functions $G_j(v) - \lambda j$ for which there is a path from s^0 to v^j one-by-one into a list of break points and indices associated with v. Before incorporating $G_j(v) - \lambda j$, we will assume that we have a list of breakpoints $-\infty = \lambda_0 < \lambda_1 < \cdots < \lambda_i = \infty$, together with indices $0 \leq j_1 < \cdots < j_i \leq j - 1$ such that j_l achieves the minimum in (5) for all λ with $\lambda_{l-1} \leq \lambda \leq \lambda_l$. Megiddo shows that $G_j(v) - \lambda j$ can be incorporated in this list in $O(1)$ time after performing a binary search over $j_1, \ldots, j_i$. He also shows that the number of breakpoints can be at most $O(nm)$, so the total time required will be $O(kn \log n)$.

Given these data structures, we can determine the value $\pi_v^k(\lambda)$ and an index j_l achieving the minimum in (5) in $O(\log n)$ time. In order to recover a shortest path from s to v, we can just follow predecessors back from v^{j_l} to s^0 in the predecessor digraph. (If the predecessor digraph is represented by lists of arcs $(u^j, v^{j+t(u,v)})$ in $E^<$ for each vertex $u^j \in V^<$, we must first represent the predecessor digraph by predecessors in $O(kn)$ time.) While the worst-case time to recover a shortest path tree T with respect to $c_e(\lambda)$ will be $O(n \log n)$, on the average we can do better. If D_u is the number of descendants of u in T, i.e., the number of vertices v for which the path from s to v in T passes through u, then the expected running time will be only $O(n + \log n \, \Sigma_{u \neq s} 1/D_u)$ if the vertices $v \neq s$ are processed in random order.

6. APPLICATION TO CYCLIC STAFFING

In Bartholdi et al. [2], a certain class of cyclic staffing problems were represented as integer programming problems of the form

$$\begin{aligned} &\text{minimize } x_1 + \cdots + x_n \\ &\text{subject to } Ax \geq b \\ &\qquad x \geq 0 \text{ and integer} \end{aligned} \tag{10}$$

where A is an m by n 0–1 column circular matrix, i.e., in each column of A the 1's occur consecutively, where the first and last entries are considered to be consecutive. This class includes the work scheduling problems described in the introductory textbook of Winston [28, §3–5].

Bartholdi et al. show that a solution to (10) can be obtained by rounding the solution to its continuous relaxation

$$\begin{aligned} &\text{minimize } x_1 + \cdots + x_n \\ &\text{subject to } Ax \geq b \\ &\qquad x \geq 0 \end{aligned} \tag{11}$$

in a particular way, and that the solution to (11) can be obtained by transforming it to a parametric shortest path problem of the type considered in Karp and Orlin [18]. For the sake of completeness, we briefly describe the transformation of (11) to a parametric shortest path problem.

First of all we note that because of the column circularity of A, rows i and $i + 1$ can differ for at most $2n$ row indices i, so we may assume that $m \leq 2n$. Also, if there is a pair j, k of distinct columns such that $a_{ij} \geq a_{ik}$ for all i then column k can be eliminated. Checking this condition for each pair of columns would require $O(n^3)$ time, but there is a more efficient way to eliminate all such columns. We first scan each column in order to obtain a compact representation of the form (i, j), where (i, j) represents a column for which the first 1 in the circular sequence of 1's occurs in row i and the first 0 in the circular sequence of 0's occurs in row j. Thus if $i < j$ the column has consecutive 1's occurring in rows $i, \ldots, j - 1$ and if $i > j$ the column has consecutive 1's in columns $1, \ldots, j - 1$ and $i, \ldots, m$.

We will eliminate all dominated columns in three stages. First we eliminate those (i, j) with $i < j$ for which there is an (i', j') with $i' < j'$ such that $i' \leq i$ and $j' \geq j$. Assuming the columns of this form are sorted primarily by the first index and secondarily by the second index, this can be done in $O(n)$ time using dynamic programming. (The sorting itself takes $O(n)$ time using a bucket sorting technique.) In the same way, we can eliminate those (i, j) with $i > j$ for which there is an (i', j') with $i' > j'$ such that $i' \leq i$ and $j' \geq j$. We note that no column (i', j') with $i' < j'$ can dominate a column (i, j) with $i > j$.

Finally, to see what columns (i', j') with $i' > j'$ dominate columns of the form (i, j) with $i < j$, let i^* be the smallest i and j^* be the largest j for which there is a column (i, j) with $i > j$. Then a column (i, j) with $i < j$ is eliminated if $i \geq i^*$ or if $j \leq j^*$. The entire procedure takes $O(n)$ time once the columns are represented compactly, which requires $O(n^2)$ time since we must examine each a_{ij}. It is easy to see that the compact representation can be used to reconstruct the matrix A in such a way that its remaining columns are in lexicographic order. The matrix A will then be row circular, i.e., in each row of A the 1's occur consecutively where the first and last columns are considered to be consecutive.

A change of variables then transforms (11) into the dual of a parametric shortest path problem: For $j = 0, 1, \ldots, n - 1$, let $y_j = x_1 + \cdots + x_j$; in particular $y_0 = 0$. Also, let $\lambda = -(x_1 + \cdots + x_n)$. The non-negativity constraints on the x_i's can be stated as $y_{i-1} \leq y_i$ for $i = 1, \ldots, n - 1$ and $y_{n-1} \leq y_0 - \lambda$. If row i has consecutive 1's occurring in columns $j + 1, \ldots, k$ for some $0 \leq j < k < n$, the corresponding inequality can be written $y_j \leq y_k - b_i$. If row i has consecutive 1's in columns $1, \ldots, j$ and $k + 1, \ldots, n$ for some $0 \leq j \leq k < n$, the corresponding inequality can be written $y_k \leq y_j - b_i - \lambda$. The relaxation (11) is thus transformed to the maximization of λ subject to linear inequalities of the form $\lambda t_{(u,v)} - y_u + y_v \leq c_{(uv)}$, where $t_{(uv)} \in \{0, 1\}$.

These are simply the dual constraints (2), so the relaxation of the cyclic staffing problem is reduced to the computation of λ^* for a digraph D with vertex set $\{0, 1, \ldots, n - 1\}$. For a solution x to (11) we can take $x_j = \pi_j^k(\lambda^*) - \pi_{j-1}^k(\lambda^*)$ for $j = 1, 2, \ldots, n - 1$ and $x_n = -\lambda^* - \pi_{n-1}^k(\lambda^*)$ with $\pi_v^k(\lambda^*)$ for $v \in V$ determined by (5). Since the rounding used by Bartholdi et al. to obtain a solution to (10) is a special case of the rounding used by Boros et al. [3], it suffices to compute $\lceil \lambda^* \rceil$ and a corresponding dual feasible π as we have remarked in Section 4.

Since this digraph has only $O(n)$ arcs, the $O(nm \log n)$ parametric shortest paths algorithm of Karp and Orlin or the $O(nm + n^2 \log n)$ parametric shortest paths algorithm of Young et al. yield an $O(n^2 \log n)$ algorithm for this class of cyclic staffing problems. Here we note that $v < u$ for every arc (u, v) such that $t_{(uv)} = 0$, so that the corresponding arc set $E - E'$ is acyclic. Thus since $T = O(n)$, the faster $O(Tm)$ algorithm for computing a minimum cost to time ratio cycle yields an $O(n^2)$ algorithm for this class of cyclic staffing problems. Further, if K is the maximum number of 1's in any row, then $T = O(K)$ and thus the algorithm finds a minimum cost to time ratio cycle in $O(Kn)$ time. In terms of cyclic staffing, it means that no shift is working for more than K periods.

7. SUMMARY

In this paper, we have considered the problem of finding a minimum cost to time ratio cycle in a digraph when the transit times are small integers. We presented an improvement and extension of the $O(n^3)$ algorithm developed by Karp and Orlin for the case of 0–1 transit times. The algorithm solves the problem as $O(T)$ shortest paths problems on a related digraph and therefore has an $O(T(m + n \log n))$ running time, which can be improved to $O(Tm)$ in case each cycle has a strictly positive transit time. (Here T is an upper bound on the transit time of any path or cycle, so that $T = O(n)$ when the transit times are 0–1.) The algorithm can be applied to the cyclic staffing problem, and improves the running time from $O(n^2 \log n)$ to $O(n^2)$.

We have also described a heuristic method for constructing a dual feasible solution to the linear programming formulation of the minimum cost to time ratio cycle problem, which uses quantities computed by the algorithm. When the dual solution corresponds to the cost to time ratio of a candidate cycle, the algorithm can terminate early. We give several strategies for detecting a candidate cycle, which result in at worst a small constant factor increase in running time but allow the algorithm to terminate far earlier than indicated by the worst-case running time. This feature improves upon Karp's algorithm for the minimum cycle mean problem and Karp and Orlin's algorithm for the parametric shortest paths problems.

REFERENCES

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

[2] J. J. Bartholdi, J. B. Orlin, and H. D. Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research* **28** (1980) 1073–1085.

[3] E. Boros, P. L. Hammer, M. Hartmann, and R. Shamir. Balancing problems in acyclic networks. To appear in *Discrete Applied Mathematics*.

[4] S. M. Burns. Performance Analysis and Optimization of Asyncronous Circuits. Ph.D. Thesis, Department of Computer Science, California Institute of Technology, Pasadena (1991).

[5] G. B. Dantzig, W. Blattner, and M. R. Rao. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. In: P. Rosenstiehl, ed. *Theory of Graphs*. Dunod, Paris, and Gordon and Breach, New York (1967) 77–84.

[6] E. B. Dynkin and A. A. Yushkevich. *Markov Processes: Theorems and Problems*. Plenum Press, New York (1969).

[7] G. M. Engel and H. Schneider. Diagonal similarity and equivalence for matrices over groups with 0. *Czechoslavak Mathematical Journal* **25** (1975) 389–403.

[8] B. Fox. Finding minimal cost-time ratio circuits. *Operations Research* **17** (1969) 546–551.

[9] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network algorithms. *Journal of the ACM* **34** (1987) 596–615.

[10] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by cancelling negative cycles. *Journal of the ACM* **36** (1989) 873–886.

[11] D. Goldfarb, J. Hao, and S.-R. Kai. Shortest path algorithms using dynamic breadth-first search. *Networks* **21** (1991) 29–50.

[12] M. v. Golitschek. An algorithm for scaling matrices and computing the minimum cycle mean in a digraph. *Numerische Mathematik* **35** (1980) 45–55.

[13] M. v. Golitschek. Optimal cycles in doubly weighted graphs and approximation of bivariate functions by univariate ones. *Numerische Mathematik* **39** (1982) 65–84.

[14] S. C. Graves and J. B. Orlin. A minimum concave-cost dynamic network flow problem with an application to lot-sizing. *Networks* **15** (1985) 59–71.

[15] M. Hartmann. On cycle means and cyclic staffing. Technical Report No. UNC/OR/TR-90/14, University of North Carolina, Chapel Hill (1990).

[16] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. An algorithm for the tramp steamer problem based on mean-weight cycles. Preprint (1991).

[17] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics* **23** (1978) 309–311.

[18] R. M. Karp and J. B. Orlin. Parametric shortest path algorithms with an application to cyclic staffing. *Discrete Applied Mathematics* **3** (1981) 37–35.

[19] A. V. Karzanov. On minimal mean cuts and circuits in a digraph. In: *Methods for Solving Operator Equations*. Yaroslavl State University, Yaroslavl (1985) 72–83.

[20] E. L. Lawler. Optimal cycles in doubly weighted linear graphs. In: P. Rosenstiehl, ed. *Theory of Graphs*. Dunod, Paris, and Gordon and Breach, New York (1967) 209–214.

[21] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Reinhart and Winston, New York (1976).

[22] B. Lockyear and C. Ebeling. Optimal retiming of multiphase, level-clocked circuits. Technical Report 91-10-01, Department of Computer Science and Engineering, University of Washington, Seattle (1991).

[23] N. Meggido. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research* **3** (1979) 313–323.

[24] J. B. Orlin and U. G. Rothblum. Computing optimal scalings by parametric network algorithms. *Mathematical Programming* **32** (1985) 1–10.

[25] M. B. Richey. A scheduling problem in the automated parallelization of computer programs. Submitted to *SIAM Journal on Computing* (1990).

[26] S. B. Spaelti and T. M. Liebling. Modelling the satellite placement problem as a network flow problem with one side constraint. *OR Spektrum* **13** (1991) 1–14.

[27] N. E. Young, R. E. Tarjan and J. B. Orlin. Faster parametric shortest path and minimum-balance algorithms. *Networks* **21** (1991) 205–221.

[28] W. Winston. *Operations Research: Applications and Algorithms*. PWS-Kent, Boston (1991).

Received December 1991
Accepted December 1992