Discrete Applied Mathematics 3 (1981) 37–45
North-Holland Publishing Company

PARAMETRIC SHORTEST PATH ALGORITHMS WITH AN APPLICATION TO CYCLIC STAFFING

Richard M. KARP*
Dept. of Electr. Eng. and Comput. Sci., University of California, Berkeley, CA 94720, USA

James B. ORLIN
Sloan School of Management, M.I.T., Cambridge, MA 02139, USA

Received 21 May 1980

Let $G=(V,E)$ be a digraph with n vertices including a special vertex s. Let $E'\subseteq E$ be a designated subset of edges. For each $e\in E$ there is an associated real number $f_1(e)$. Furthermore, let $f_2(e)=1$ if $e\in E'$ and $f_2(e)=0$ if $e\in E-E'$. The *length* of edge e is $f_1(e)-\lambda f_2(e)$, where λ is a parameter that takes on real values. Thus the length varies additively in λ for each edge of E'.

We shall present two algorithms for computing the shortest path from s to each vertex $v\in V$ parametrically in the parameter λ, with respective running times $O(n^3)$ and $O(n\,|E|\log n)$. For dense digraphs the running time of the former algorithm is comparable to the fastest (non-parametric) shortest path algorithm known.

This work generalizes the results of Karp [2] concerning the minimum cycle mean of a digraph, which reduces to the case that $E'=E$. Furthermore, the second parametric algorithm may be used in conjunction with a transformation given by Bartholdi, Orlin, and Ratliff [1] to give an $O(n^2\log n)$ algorithm for the cyclic staffing problem.

Let $G=(V,E)$ be a digraph with n vertices including a special vertex s. Let $E'\subseteq E$ be a designated subset of edges. For each $e\in E$ there is an associated real number $f_1(e)$. Furthermore, let

$$f_2(e)=\begin{cases}1 & \text{if } e\in E',\\ 0 & \text{if } e\in E-E'.\end{cases}$$

The *length* of edge e is $f_1(e)-\lambda f_2(e)$, where λ is a parameter that takes on real values. Thus the length varies additively in λ for each edge of E'.

We shall present two algorithms for computing the shortest path from s to each vertex $v\in V$ parametrically in the parameter λ, with respective running times $O(n^3)$ and $O(n\,|E|\log n)$. For dense digraphs the running time of the former algorithm is comparable to the fastest (non-parametric) shortest path algorithm known.

This work generalizes the results of Karp [2] concerning the minimum cycle mean of a digraph, which reduces to the case that $E'=E$. Furthermore, the second parametric algorithm may be used in conjunction with a transformation given by

* Research supported by National Science Foundation Grant MCS77-09906.

Bartholdi, Orlin, and Ratliff [1] to give an $O(n^2 \log n)$ algorithm for the cyclic staffing problem.

In the following a *walk* is a directed edge 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 walk P and for each $\lambda \in \mathbb{R}$ define

$$l(P, \lambda) = \sum_{e \in P} (f_1(e) - \lambda f_2(e)).$$

Thus $l(P, \lambda)$ is the *length* of walk P. For each integer k and each $\lambda \in \mathbb{R}$ let

$$g_k(v, \lambda) = \left\{\min l(P, \lambda)\text{: } P \text{ is a walk from } s \text{ to } v \text{ and } \sum_{e \in E} f_2(e) = k\right\}.$$

Then $g_k(v, \lambda)$ is a linear function of λ which measures the shortest length path from s to v subject to the additional constraint that the path has exactly k edges of E'.

In the following we assume that there is a path from s to v for each vertex $v \in V$ and that there is no cycle of negative length composed solely of edges in $E - E'$. Thus for all $v \in V$ and for sufficiently small λ there is a minimum length path from s to v. Of course, these assumptions may both be checked in $O(n\,|E|)$ steps.

Let $F(v, \lambda)$ be the minimum length of a walk from s to v for fixed value λ, where $F(v, \lambda) = -\infty$ if there is a sequence of walks with length unbounded from below.

Lemma 1. $F(v, \lambda) = \min_{0 \leq k \leq n-1} g_k(v, \lambda)$ *for all* $v \in V$ *and for all* $\lambda \in \mathbb{R}$ *for which the resulting digraph has no cycles of negative length.*

Proof. Since there are no negative length cycles, there is always a minimum length walk which is a path and hence has at most $n-1$ edges in E'. □

The following result is a direct extension of a theorem proved by Karp in [2].

Let λ^* be the maximum value of λ for which there are no cycles of negative length. Let $V(k) = \{v \in V\text{: } |g_k(v)| \neq \infty\}$. Let

$$G(\lambda) = \min_{v \in V} \max_{\substack{0 \leq k \leq n-1 \\ v \in V(k)}} (g_n(v, \lambda) - g_k(v, \lambda)).$$

Theorem 2. *Either* $\lambda^* = \infty$ *or else* λ^* *is the unique value of* λ *for which* $G(\lambda) = 0$.

Proof. Note first that, if $G(\lambda)$ is finite (i.e., there is some v such that $g_n(v, \lambda) \neq +\infty$), then $G(\lambda)$ is a strictly decreasing function, and thus there is a unique value of λ for which $G(\lambda) = 0$.

For $\lambda = \lambda^*$ we have that all cycles have non-negative length and hence by

Lemma 1, for all $v \in V$

$$g_n(v, \lambda^*) \geq F(v, \lambda^*) = \min_{0 \leq k \leq n-1} g_k(v, \lambda^*).$$

Hence, $G(\lambda^*) \geq 0$.

Conversely, for value λ^* there must be a cycle C of length 0 with at least one edge in E'. Else we could increase λ^* without creating a negative length cycle.

Let v be a vertex on cycle C and let P be a minimum length path from s to v for value λ^*. P followed by any number of repetitions of C is also a minimum length walk from s to v. Hence, any initial part of such a walk is a minimum length walk from s to its endpoint. After sufficient repetitions of c there will be an initial part P' with exactly n edges in E' from s to w. Then

$$l(P, \lambda^*) = g_n(w, \lambda^*) = \min_{0 \leq k \leq n-1} g_k(w, \lambda^*).$$

Thus $G(\lambda^*) \leq 0$. Thus $G(\lambda^*) = 0$. $\square$

Let $d(u, v)$ be the minimum length of a path from u to v in G consisting solely of edges of $E - E'$. Let $g_k'(v, \lambda)$ be the minimum length of a walk from s to v with exactly k edges in E' and such that the last edge of the walk is in E'. Then we may calculate g and g' by the following recursive scheme:

$$g_k'(v, \lambda) = \min_{(u,v) \in E'} (g_{k-1}(u, \lambda) + f_1(u, v) - \lambda)$$

and

$$g_k(v, \lambda) = \min_{u \in V} (g_k'(u, \lambda) + d(u, v)) \quad \text{for } k = 1, \ldots, n$$

subject to the initial conditions

$$g_0(v, \lambda) = d(s, v).$$

In order to compute $F(v, \lambda)$ for all $\lambda \leq \lambda^*$, we may calculate λ^* in $O(n^3)$ steps from Theorem 2 and the functions $g_k(v, \lambda)$, and we may calculate $F(v, \lambda)$ from Lemma 1 in another $O(n^3)$ steps. Since the calculations of g' and g take $O(n^3)$ steps, the entire algorithm requires $O(n^3)$ steps.

An $O(n\,|E| \log n)$ algorithm

The previous algorithm is not typical for parametric programming algorithms in that the solutions are calculated simultaneously for all values of λ. The following is a more standard approach in that optimal solutions are calculated for one value of λ at a time as λ increases from $-\infty$.

In the following a *tree* will refer to a connected subgraph of G with exactly $n-1$ edges, with one directed into each vertex of $V-\{s\}$. (Thus a tree is really an arborescence with root s.)

It is well known that for $\lambda \leq \lambda^*$ there is a tree T_λ such that the unique path from s to v on T_λ is a path of minimum distance. For each tree T and for each vertex $v \in V$, let $d_1(T, v) - \lambda d_2(T, v)$ denote the distance from s to v on T. Thus $d_2(T, v)$ is the number of edges of E' on the path from s to v in T.

A *neighbor* of tree T is a tree which differs from T in exactly one edge. Let T be any tree and let $e = (u, v) \in E$. then we let $N(T, e)$ denote the subgraph obtained from T by adding edge e and deleting the unique edge of T directed into v. The following is an immediate consequence of this construction.

Lemma 3. *Let T be a tree and let $e = (u, v)$. Then $N(T, e)$ is a tree if and only if v is not on the path from s to u in T.*

If $e = (u, v)$, let

$$\Delta_i(T, e) = d_i(T, u) + f_i(e) - d_i(T, v) \quad \text{for } i = 1, 2$$

and let

$$P(T, v) = \{w \in V \text{ such that } v \text{ is a vertex on the path from } s \text{ to } w \text{ in } T\}.$$

Lemma 4. *Suppose $T' = N(T, e)$ is a tree. Then for $i = 1, 2$*

$$d_i(T', w) = \begin{cases} d_i(T, w) + \Delta_i(T, e) & \text{if } w \in P(T, v), \\ d_i(T, w) & \text{if } w \notin P(T, v) \end{cases}$$

and if $\Delta_2(T, e) \neq 0$, then $\lambda' = \Delta_1(T, e)/\Delta_2(T, e)$ is the unique value of λ for which

$$d_1(T, w) - \lambda d_2(T, w) = d_1(T', w) - \lambda d_2(T', w) \quad \textit{for all } w \in V.$$

Proof. It is easy to verify that for $i = 1, 2$ and $w \in P(T, v)$,

$$d_i(T', w) - d_i(T, w) = d_i(T', v) - d_i(T, v) = \Delta_i(T, e).$$

And for $w \notin P(T, v)$, $d_i(T', w) = d_i(T, w)$. The last statement in the theorem follows from the fact that λ' is the unique value of λ for which $\Delta_1(T, e) - \lambda \Delta_2(T, e) = 0$. □

Lemma 5. *If $T' = N(T, e)$ is not a tree and $\Delta_2(T, e) \neq 0$, then $\Delta_1(T, e)/\Delta_2(T, e)$ is the unique value of λ for which the digraph T' has a circuit of length 0.*

Proof. Suppose that T' is not a tree. Let P_v and P_u denote the subpaths in T from s to v and from s to u. Then a cycle is formed in T' by deleting the subpath P_v from the subpath P_u and then adding edge (u, v). The length of this circuit is thus:

$$d_1(T, u) - d_1(T, v) + f_1(e) - \lambda(d_2(T, u) - d_2(T, v) + f_2(e)) = \Delta_1(T, e) - \lambda \Delta_2(T, e). \quad \square$$

The following algorithm will calculate a sequence of trees $T_1, \dots, T_m$ and a sequence of real numbers $-\infty = \lambda_0, \lambda_1, \dots, \lambda_m = \lambda^*$ in such a way that the tree T_i is a tree of minimum distances for $\lambda \in [\lambda_{i-1}, \lambda_i]$, except for $\lambda = \infty$ or $\lambda = -\infty$ in which case we do not define the trees. The trees will only be stored implicitly via appropriate data structures; otherwise, to store all the trees would take $\Omega(n^3)$ space.

The algorithm requires a data structure to store and retrieve entries of the form $(e, c(e))$, where $e \in E$ and $c(e) \in R$. The operations performed on the structure are the insertion of an entry, the deletion of the unique entry $(e, c(e))$ associated with a given edge e, and the retrieval of an entry $(e, c(e))$ for which $c(e)$ is minimum. If a balanced tree [3] is used for this purpose, then each of the basic operations can be performed in $O(\log |E|)$ steps.

Algorithm 2

```
Step 0 (Initialize)
    0.1 Let $M = 1 + \sum_{e \in E} |f_1(e)|$.
        Let $\lambda_0 = -\infty$.
    0.2 Let $T_1$ be the tree of minimum distances for $\lambda = -M$. (Use a standard
        $O(n\,|E|)$ shortest path algorithm to determine this tree.)
    0.3 Compute $d_i(T_1, v)$ for $i = 1, 2$ and $v \in V$.
    0.4 For each edge $e \in E$ let
```

$$c(e) = \begin{cases} \infty & \text{if } \Delta_2(T_1, e) \le 0, \\ \dfrac{\Delta_1(T_1, e)}{\Delta_2(T_1, e)} & \text{if } \Delta_2(T_1, e) > 0. \end{cases}$$

```
    0.5 Store the values $c(e)$ for $e \in E$ in a balanced tree $B$.
    0.6 Let $i = 1$.

Step 1 (Compute the new tree)
    1.1 Let $\lambda_i = c(e') = \min_{e \in E} c(e)$ (by finding the minimum element of the bal-
        anced tree $B$). If $\lambda_i = \infty$ then quit. Let $(u, v) = e'$.
    1.2 $T_{i+1} = n(T_i, e')$.
    1.3 Let $P_i = P(T_i, v)$.
        If $u \in P_i$ then quit with $\lambda^* = \lambda_i$; else, let $\Delta_j = \Delta_j(T_i, e')$ for $j = 1, 2$.
    1.4 For each $w \in P_i$ and for $j = 1, 2$
```

$$d_j(T_{i+1}, w) = \begin{cases} d_j(T_i, w) + \Delta_j & \text{if } w \in P, \\ d_j(T_i, w) & \text{if } w \notin P. \end{cases}$$

```
Step 2 (Calculate the next change of trees)
    2.1 Let $\bar{E}$ be the subset of edges of $E$ with at least one endpoint in $P_i$.
```

2.2 For $e \in \bar{E}$ delete the value $c(e)$ from B and instead let

$$c(e) = \begin{cases} \infty & \text{if } \Delta_2(T_{i+1}, e) \leq 0, \\ \dfrac{\Delta_1(T_{i+1}, e)}{\Delta_2(T_{i+1}, e)} & \text{if } \Delta_2(T_{i+1}, e) > 0 \end{cases}$$

and store this new value of $c(e)$ in the balanced tree B.

2.3 Let $i = i + 1$ and return to Step 1.

Theorem 6. *For each $i = 1, \ldots, m$, the tree T_i is optimal for all $\lambda \in [\lambda_{i-1}, \lambda_i]$ with λ finite.*

Proof. Suppose the theorem is false. Choose the minimum i so that there is a value $\lambda' \in [\lambda_{i-1}, \lambda_i]$ such that the result fails at λ'. This can happen for one of two reasons: either there is a tree T that is optimal for $\lambda = \lambda'$ and a vertex v so that the distance from s to v in T is less than the corresponding distance in T_i for $\lambda = \lambda'$; else the theorem fails because there is a negative cycle for $\lambda = \lambda'$. We first investigate the former case.

By our choice of M, for $\lambda \leq -M$ it follows that $F(v, \lambda) = g_k(v, \lambda)$ where k is the minimum index for which $g_k(v, \lambda) \neq \infty$. Thus T_1 is optimal for $\lambda \in (-\infty, -M]$ and thus $\lambda' > -M$.

Furthermore $\lambda' \neq \lambda_{i-1}$. To see this note that by our choice of i we have that T_{i-1} is optimal for value λ_{i-1}. By our choice of λ_{i-1} and by Lemma 4, the distances in T_i and T_{i-1} are the same for $\lambda = \lambda_{i-1}$. Thus T_i is optimal at λ_{i-1}.

For each $w \in V$ let

$$h(w, \lambda) = d_1(T, w) - d_1(T_i, w) - \lambda(d_2(T, w) - d_2(T_i, w)).$$

Then, by the optimality of T at $\lambda = \lambda'$, $h(w, \lambda') \leq 0$ for all w. Also, $h(s, \lambda') = 0$. Since T_i is not optimal at $\lambda = \lambda'$, there is a v such that $h(v, \lambda') < 0$ and $h(u, \lambda') = 0$ for the vertex u which immediately precedes v on the path from s to v in T. Let e be the edge (u, v) of T directed into v. Then

$$h(v, \lambda) = \Delta_1(T_i, e) - \lambda\Delta_2(T_i, e).$$

Furthermore,

$$h(v, \lambda') < 0 < h(v, \lambda_{i-1}).$$

This can only happen if

$$\Delta_2(T_i, e) > 0$$

and

$$\lambda_{i-1} < \frac{\Delta_1(T_i, e)}{\Delta_2(T_i, e)} < \lambda'.$$

But this would contradict our choice of λ_i in Step 1, thus proving that tree T cannot exist.

Suppose instead that there is a negative cycle for $\lambda = \lambda'$. As before $\lambda' > \lambda_{i-1}$, as T_{i-1} is optimal in $[\lambda_{i-1}, \lambda_i]$. Let there be a cycle C of length 0 for λ^* with $\lambda_{i-1} \leq \lambda^* < \lambda_i$ and such that at least one edge of E' is in C. Let $C = v_1, \ldots, v_k, v_1$ and let v_j be a vertex such that v_{j-1} does not precede v_j on the path from s to v_j in T_i. Let $e = (v_{j-1}, v_j)$.

We now claim that $\Delta_1(T_i, e)/\Delta_2(T_i, e) = \lambda^*$ and $\Delta_2(T_i, e) > 0$ thus contradicting our choice of λ_i. To see this we note first that for $\lambda = \lambda^*$ a minimum length walk from s to v followed by C is also a minimum length walk. Thus v_{j-1} precedes v_j on some minimum length walk from s to v_j. Let $e = (v_{j-1}, v_j)$. Since T_i is a minimum distance tree, it follows from the above that

$$d_1(T_i, v_j) - \lambda^* d_2(T_i, v_j) = d_1(T_i, v_{j-1}) + f_1(e) - \lambda^*(d_2(T_i, v_{j-1}) + f_2(e)).$$

Thus

$$\Delta_1(T_i, e) - \lambda^* \Delta_2(T_i, e) = 0. \quad \square$$

Let $\alpha(w)$ be the number of sets P_i as defined in Step 1.3 such that $w \in P_i$. In order to determine the number of steps taken by the algorithm, we will use the following lemma.

Lemma 7. *For each* $w \in V$, $\alpha(w) \leq n-1$.

Proof. If $w \in P_i$ then $d_2(T_i, w) \geq (d_2(T_{i-1}, w) - 1) \geq 0$. Since $d_2(T_i, w) \leq n-1$ for all i it follows that $\alpha(w) \leq n-1$. $\square$

We now can see that Algorithm 2 takes $O(n\,|E| \log n)$ steps as follows:

(1) By Lemma 7, $d_j(T_i, w)$ is updated at most $n-1$ times for each $w \in V$. Thus this updating may be carried out in $O(n^2)$ steps overall.

(2) By Lemma 7, $c(e)$ is updated at most $2n-2$ times, $n-1$ times for each endpoint. Thus the total number of updates of $c(e)$ is $O(n\,|E|)$. At each update we must both delete and insert the value into a balanced tree. These operations may be carried out in $O(\log n)$ steps. Furthermore, we may have to choose the minimum element in the balanced tree as many as n^2 times. Thus the total computation count for these steps is $O(n\,|E| \log n)$.

An application to cyclic staffing

Vector y is said to be *circular* if each component has value 0 or 1 and if the components with value 1 occur consecutively, where the first and last components are considered consecutive. Matrix a is said to be *column* (resp. *row*) *circular* if each column (resp. row) of A is circular.

In [1], a certain class of cyclic staffing problems was represented as an integer programming problem

$$\begin{aligned} &\text{minimize} && \mu, \\ &\text{subject to} && x_1 + \cdots + x_n = \mu, \\ &&& Ax \geqslant b, \\ &&& x \geqslant 0 \text{ and integer} \end{aligned} \tag{3}$$

where A is column circular. An example of the problem is to find the minimum number of persons needed to staff a weekly schedule so as to satisfy demands that vary from day to day within the week but repeat weekly and so that each person receives two consecutive days off per week. This problem may be represented as the integer program (3) where

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

The following is proved in [1]: If μ is considered as a parameter then every basic solution to (3) is integral whenever μ is integral. To solve (3) it suffices to determine the minimum integral value of μ for which (3) is feasible. Furthermore, this may be accomplished by transforming this parametric problem into an equivalent parametric shortest path problem of the type solved in this paper. The number of edges in this problem is linear in n and thus the entire problem takes $O(n^2 \log n)$ steps.

We briefly describe the problem transformation given in [1]. Without loss of generality, we may assume that there is no pair j, k of distinct column indices such that $a_{ij} \geqslant a_{ik}$ for all i; for if such a pair did exist, then column k could be eliminated. We may also assume that the columns of A are in lexicographic order; i.e., if $j > k$ then there exists some q such that $a_{ij} = a_{ik}$, $i < q$ and $a_{iq} > a_{jq}$. It follows that a is row circular as well as column circular; i.e., in each row of A, either all the 1's are consecutive or all the 0's are consecutive.

We next make a change of variables that will transform our problem to a parametric shortest-path problem. For $j = 0, 1, \ldots, n-1$, let $y_j = x_1 + x_2 + \cdots + x_j$; in particular, y_0 is identically zero. Also, let $\lambda = -\mu = -(x_1 + x_2 + \cdots + x_n)$. The nonnegativity constraints on the x_i become

$$y_0 \leqslant y_1,\ y_1 \leqslant y_2, \ldots, y_{n-2} \leqslant y_{n-1},\ y_{n-1} \leqslant -\lambda.$$

Suppose row i has consecutive 1's occurring in columns $j+1, \ldots, j+k$. Then this row represents the inequality

$$x_{j+1} + x_{j+2} + \cdots + x_k \geqslant b_i,$$

which can be rewritten $y_j \leq y_k - b_i$. Suppose row i does not have consecutive 1's but has consecutive 0's occurring in columns $j+1, \ldots, k$, where $k < n$. Then this row represents the inequality

$$(x_1 + x_2 + \cdots + x_n) - (x_{j+1} + \cdots + x_k) \geq b_i,$$

which can be rewritten

$$y_k \leq y_j - b_i - \lambda.$$

The cyclic staffing problem is thus transformed to the maximization of λ subject to linear inequalities, each of which is of the form

$$y_v \leq y_u + b_e - \lambda B_e, \quad \text{where } B_e \in \{0, 1\}. \tag{4}$$

Let $\mathcal{N}$ be a network with vertex set $\{0, 1, \ldots, n-1\}$ and, for each inequality of the form (4), an edge e from u to v of parametric cost $b_e - \lambda B_e$. Then the cyclic staffing problem is reduced to the computation of λ^* for the network $\mathcal{N}$. Once λ^* is determined, y_v is simply the cost of a shortest path in $\mathcal{N}$ from 0 to v, at $\lambda = \lambda^*$.

The transformation just given, coupled with algorithm 3, yields an algorithm to solve the cyclic staffing problem in time $O(n^2 \log n)$. An experimental computer program that executes this algorithm was run on several examples in which the columns of the constraint matrix consisted of all rotations of an n-vector of k consecutive 1's, and the components of the right hand side were drawn from a uniform distribution. The number of balanced tree operations never exceeded $2n$, suggesting that the expected time of the algorithm is $O(n \cdot \log n)$.

References

[1] J.J. Bartholdi, J.B. Orlin and H.D. Ratliff, Cyclic scheduling via integer programs with circular ones, to appear in Operations Research.

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

[3] D.E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching (Addison-Wesley, Reading, Ma, 1973).