Chapter 44

Computing a Minimum-Weight k-Link Path in Graphs with the Concave Monge Property[*]

Baruch Schieber [†]

Abstract

Let G be a weighted, complete, directed acyclic graph (DAG) whose edge weights obey the concave Monge condition. We give an efficient algorithm for finding the minimum-weight k-link path between a given pair of vertices for any given k. The algorithm runs in $n2^{O(\sqrt{\log k \log\log n})}$ time. Our algorithm can be applied to get efficient solutions for the following problems, improving on previous results: (1) computing length-limited Huffman codes. (2) computing optimal discrete quantization. (3) computing maximum k-cliques of an interval graph. (4) finding the largest k-gon contained in a given convex polygon. (5) finding the smallest k-gon that is the intersection of k half-planes out of n half-planes defining a convex n-gon.

1 Introduction

Let $G = (V, E)$ be a weighted, complete, directed acyclic graph (DAG) with the vertex set $V = \{v_1, v_2, \ldots, v_n\}$. (For convenience, we sometimes represent v_i by i.) For $1 \le i < j \le n$, let $w(i,j)$ denote the weight associated with the edge (i,j). (See Figure 1.)

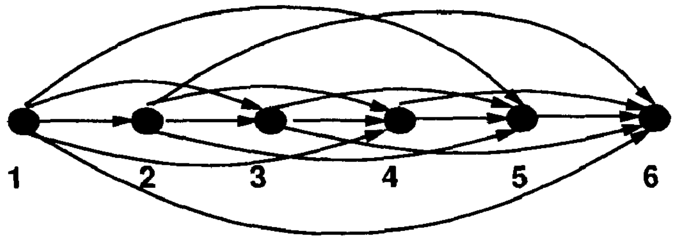

Figure 1: Complete DAG

An edge in a path in G is called a *link* of the path. We call a path in G a *k-link* path if the path contains exactly k links. For any two vertices, i and j, we call a path from i to j a *minimum* k-link path if it contains exactly k links and among all such paths it has the minimum-weight. A weighted DAG, G, satisfies the concave Monge property if the inequality $w(i,j) + w(i+1, j+1) \le w(i, j+1) + w(i+1, j)$ holds for all $1 < i+1 < j < n$.

In this paper, we are interested in computing the minimum-weight k-link path from 1 to n in concave Monge DAGs, i.e., weighted DAGs whose weights satisfy the concave Monge property.

Using the results of Aggarwal *et al.* [1] and Aggarwal and Park [2], it is easy to show that the minimum-weight k-link path can be computed in $O(nk)$ time for a concave Monge DAG. Recently, Bein *et al.* [7] and Aggarwal *et al.* [3] gave a weakly-polynomial algorithm for this problem that runs in $O(n \log U)$ time, assuming that the weights are integral and U is the maximum absolute value of these weights. Aggarwal *et al.* [3] also gave an improved strongly-polynomial algorithm that runs in $O(n\sqrt{k \log n} + n \log n)$ time. The main result of this paper is a $n2^{O(\sqrt{\log k \log\log n})}$ time algorithm for computing the minimum-weight k-link path. Note that this algorithm is superior to the algorithm given in [3]. It is superior to the $O(nk)$ time naive algorithm whenever $k = \Omega(\log n)$. From now on, we assume that this is the case.

In [3], Aggarwal *et al.* posed the question of designing an $O(n \cdot \mathrm{polylog}(n,k))$ time algorithm for computing the minimum-weight k-link path. Although we are still unable to answer this question in the affirmative, we may be a step closer to this goal since our algorithm runs in $o(nk^\varepsilon)$ time, for any fixed ε.

Our algorithm is recursive. It uses some properties of concave Monge DAGs together with a variant of the parametric search technique [13, 9] – a powerful technique for designing algorithms, especially in computational geometry [8]. Interestingly, our algorithm uses the parametric search in the most naive way, in

*Extended summary

†IBM – Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. Email: sbar@watson.ibm.com

contrast to the more sophisticated way it was used in [3]. We leave open the question whether a more clever way of applying the parametric search paradigm would yield a better algorithm.

1.1 Applications

The algorithm for minimum-weight k-link path in concave Monge DAGs has several applications. Given below are such applications to data optimization (App. I), data compression (App. II), interval graphs (App. III), and geometric path finding (App. IV and V).

Application I. Given a weighted alphabet of size n, we want to find an optimal prefix-free binary code for the alphabet with the restriction that no code string be longer than k bits. Using the reduction of this problem to the minimum-weight k-link path problem [12], we solve it in $n2^{O(\sqrt{\log k \log\log n})}$ time, improving on [11, 3].

Application II. Let $f : \{x_1, x_2, \ldots, x_n\} \to \mathcal{R}$ be a real valued function, where $\mathcal{R}$ is the set of the real numbers and $x_1 \leq x_2 \leq \cdots \leq x_n$ are real numbers. Fix k and consider a sorted set of real numbers $Z = \{z_1, z_2, \ldots, z_k\}$ and a mapping $\psi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$. The pair (Z, ψ) is called a quantization, and the sum $\sum_{i=1}^{n} f(x_i)(x_i - z_{\psi(i)})^2$ the error of the quantization. Optimal quantization is the one which minimizes the error. It is easy to see that in optimal quantization $\psi^{-1}(j)$ is an interval for each $j = 1, 2, \ldots, k$. Quantization can be regarded as a data compression of n data items into k items, as illustrated in Figure 2. Wu [14] showed that computing optimal quantization can be reduced to finding a minimum-weight k-link path. Hence, it can be solved in $n2^{O(\sqrt{\log k \log\log n})}$ time by applying our algorithm, improving on [14, 3].

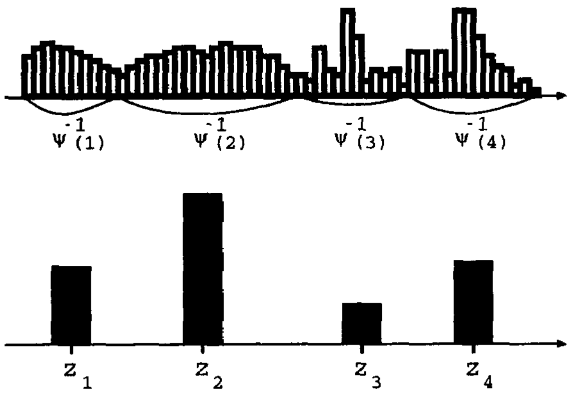

Figure 2: Quantization (k=4)

Application III. Let H be an interval graph generated by m weighted intervals on n terminals. Given k, *find* k *cliques of* H *so that the sum of the weights of intervals in the union of the cliques is maximized.* (See Figure 3.) By applying our algorithm this problem can be solved in $O(m) + n2^{O(\sqrt{\log k \log\log n})}$ time, improving on previous results of [5, 3, 4].

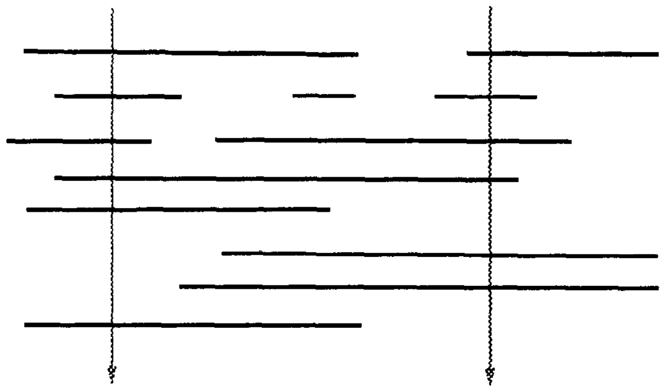

Figure 3: k maximum weight cliques of interval graph (k=2)

Application IV. Computing the maximum area k-gon and the maximum perimeter k-gon that are contained in a given convex n-gon. (See Figure 4.) For this problem Boyce *et al.* [6] provided an $O(nk \log n)$ time algorithm that was later improved by Aggarwal *et al.* [1] to $O(nk + n \log n)$ time, and by Aggarwal *et al.* [3] to $O(n\sqrt{k \log n} + n \log n)$ time. By incorporating the main result of this paper, this problem can now be solved by an algorithm that runs in $n2^{O(\sqrt{\log k \log\log n})}$ time.

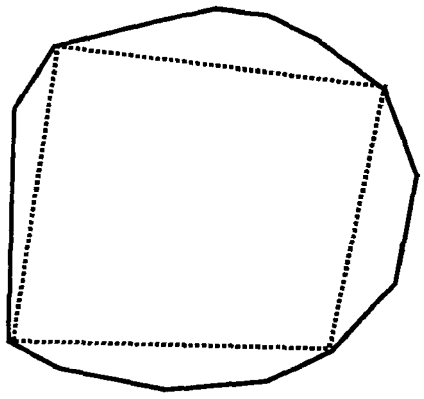

Figure 4: Max-area inscribed polygon

Application V. Computing the minimum area k-gon that is the intersection of k half-planes out of n half-planes defining a given convex n-gon. In other words, computing the minimum area circumscribing polygon touching edge-to-edge. (See Figure 5.) This problem can also be solved in $n2^{O(\sqrt{\log k \log\log n})}$ time, improving on the previous results mentioned above.

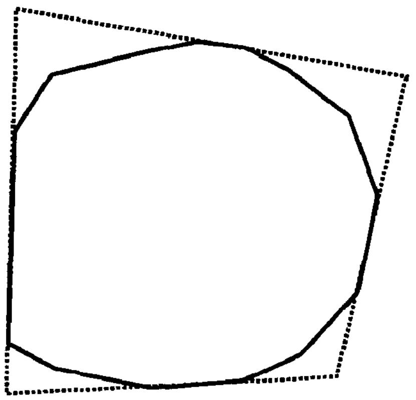

Figure 5: Min-area inscribed polygon with edge-to-edge contact

The rest of the paper is organized as follows. Section 2 proves some properties of concave Monge DAGs, and Section 3 describes the algorithm and analyzes its complexity.

2 Properties of concave Monge DAGs

Let G be a concave Monge DAG. For a real number τ, define $G(\tau)$ to be the weighted DAG with the same sets of vertices and edges as G, in which each edge e in $G(\tau)$ has the weight $w(e)+\tau$ (where $w(e)$ is the weight of e in G). Note that if G has the concave Monge property, then also $G(\tau)$ has this property. Define a *diameter* path in G to be a path from 1 to n.

The first two lemmas hold for any DAG and do not depend on the fact the G has the concave Monge property.

Lemma 2.1. *If for some τ a minimum-weight diameter path in $G(\tau)$ has k links, then this path is the minimum-weight k-link diameter path in G.*

Lemma 2.2. *If a minimum-weight diameter path in $G(\tau)$ has k links, then for every $\xi < \tau$, any minimum-weight diameter path in $G(\xi)$ has at least k links.*

Proof. Let P and Q be minimum-weight diameter paths in $G(\tau)$ and $G(\xi)$, respectively. Suppose that P has k links, and Q has ℓ links. Let $W_\tau(P)$ denote the weight of P in $G(\tau)$. Then, $W_\tau(Q) - W_\tau(P) \geq 0$ and $W_\xi(Q) - W_\xi(P) \leq 0$. Thus,

$$\ell(\tau-\xi) = W_\tau(Q)-W_\xi(Q) \geq W_\tau(P)-W_\xi(P) = k(\tau-\xi).$$

Since $\tau - \xi > 0$, we have that $\ell \geq k$. □

Definition 2.3. *An edge (i_1, j_1) <u>covers</u> another edge (i_2, j_2) if $i_1 \leq i_2 < j_2 \leq j_1$ and $(i_1, j_1) \neq (i_2, j_2)$.*

Let P_1 and P_2 be paths in G. Suppose that there exists a link (i_1, j_1) of P_1 and a link (i_2, j_2) of P_2 such that (i_1, j_1) covers (i_2, j_2). We define a *path swap* operation with respect to this pair of edges. This operation creates two new paths Q_1 and Q_2. Path Q_1 is given by connecting the prefix of P_1 ending at i_1 with the suffix of P_2 starting at j_2 by edge (i_1, j_2). Path Q_2 is given by connecting the prefix of P_2 ending at i_2 with the suffix of P_1 starting at j_1 by edge (i_2, j_1).

Lemma 2.4. *Let Q_1, Q_2 be paths obtained from P_1 and P_2 by a path swap operation with respect to (i_1, j_1) and (i_2, j_2). The sum of the weights of paths Q_1 and Q_2 is at most the sum of the weights of paths P_1 and P_2. In particular, if P_1 and P_2 are minimum-weight paths so are Q_1 and Q_2.*

Proof. In case $i_1 = i_2$ or $j_1 = j_2$, clearly, $W(Q_1) + W(Q_2) = W(P_1) + W(P_2)$. Otherwise, i.e., $i_1 < i_2 < j_2 < j_1$, we have

$$\begin{aligned} W(Q_1) + W(Q_2) &= W(P_1) + W(P_2) - \\ &\quad (w(i_1, j_1) + w(i_2, j_2)) + \\ &\quad (w(i_1, j_2) + w(i_2, j_1)) \\ &\leq W(P_1) + W(P_2). \end{aligned}$$

The inequality follows from the concave Monge property of the edge weights. □

For $a \leq b$, let P_a and P_b be paths from v_1 to v_a and from v_1 to v_b, respectively. Suppose that P_a has k_a links, P_b has k_b links, and $k_a > k_b$.

Lemma 2.5. *For any $0 \leq x \leq k_a - k_b$ there are links $e_a = (i_a, j_a)$ of P_a and $e_b = (i_b, j_b)$ of P_b with the following two properties.*

1. *Edge e_b covers edge e_a.*
2. *The prefix of P_a ending at i_a has x more links than the prefix of P_b ending at i_b.*

Proof. Let $e = (i_b, j_b)$ be the leftmost link of P_b that covers some link of P_a. Such a link must exist since $b \geq a$ and $k_a > k_b$. Suppose that e covers c links of P_a, and let $f = (i_a, j_a)$ be the leftmost such link. Let d be the difference between the length of the prefix of P_a ending at i_a and the length of the prefix of P_b ending at i_b. It follows from our selection of e that this difference is less than or equal to zero. Observe that for any $d \leq 0 \leq x < d+c$ we can set e_b to e and e_a to one of the links of P_a covered by e and have the two properties of the lemma satisfied. In case $k_a - k_b < d + c$ we are done. Otherwise, there must be another link of P_b (to the right of e) that covers some link of P_a. Let $e' = (i'_b, j'_b)$ be the leftmost such link. Again, suppose that e' covers c' links of P_a, and let $f' = (i'_a, j'_a)$ be the leftmost such link. Note that the difference d' between the length of the prefix of P_a ending at i'_a and the length of the prefix of P_b ending at i'_b is less than or equal to $d + c$. Hence, for any $d' \leq d+c \leq x < d'+c'$ we can set e_b to e' and e_a to one of the links of P_a covered by e' and have the two properties of the lemma satisfied. If $k_a - k_b < d' + c'$ we are done. Otherwise, we continue in the same manner. □

LEMMA 2.6. *Let a, b, P_a, P_b, k_a and k_b be as above. For any k in the range $[k_b, k_a]$, there are paths Q_a with k links from v_1 to v_a and Q_b with $k_a + k_b - k$ links from v_1 to v_b such that the sum of the weights of paths Q_a and Q_b is at most the sum of the weights of paths P_a and P_b. In particular, if P_a and P_b are minimum-weight paths so are Q_a and Q_b.*

Proof. Fix some k in the range $[k_b, k_a]$. By Lemma 2.5 there are links $e_a = (i_a, j_a)$ in P_a and $e_b = (i_b, j_b)$ in P_b such that edge e_b covers edge e_a, and the prefix of P_a ending at i_a has $k_a - k$ more links than the prefix of P_b ending at i_b. Perform a path swap with respect to e_a and e_b to obtain two paths Q_a and Q_b from v_1 to v_a and v_b, respectively. Since Q_a is created by connecting the prefix of P_b ending at i_b with the suffix of P_a starting at j_a, the length of Q_a is $k_a - (k_a - k) = k$. Similarly, the length of Q_b is $k_a + k_b - k$. Lemma 2.4 implies that the sum of the weights of paths Q_a and Q_b is at most the sum of the weights of paths P_a and P_b. □

DEFINITION 2.7. *For $1 < a \leq n$ and $1 \leq \ell < a$, let $P(a, \ell)$ denote the minimum-weight ℓ-link path in G from 1 to a, and let $W(a, \ell)$ denote the weight of this path. Let $P(\ell) = P(n, \ell)$ and $W(\ell) = W(n, \ell)$.*

The next corollaries follow from Lemma 2.6. (Proofs omitted.)

COROLLARY 2.8. *For $1 < a < b \leq n$ and $1 \leq \ell < a - 1$,*

$$W(a, \ell) - W(a, \ell + 1) \leq W(b, \ell) - W(b, \ell + 1).$$

COROLLARY 2.9. *For $1 < a \leq n$ and $1 < \ell < a - 1$,*

$$W(a, \ell) - W(a, \ell + 1) \leq W(a, \ell - 1) - W(a, \ell).$$

Corollary 2.9 implies that the function $W(a, .)$ is unimodal; or in other words, any local minimum of the function is a global minimum.

COROLLARY 2.10. *For $1 \leq \ell \leq n - 1$, all the vertices to which there exists a minimum-weight path (from 1) with ℓ links are consecutive.*

LEMMA 2.11. *For any $1 \leq k \leq n - 1$, there exists a real number τ such that a minimum-weight diameter path of $G(\tau)$ has k links.*

Proof. First, consider some $1 < k < n-1$. We claim that for any τ in the interval $[W(k) - W(k+1), W(k-1) - W(k)]$ there is a minimum-weight diameter path in $G(\tau)$ with k links. Consider any $k < \ell < n$. It is easy to verify that for all $\tau \geq (W(k) - W(\ell))/(\ell - k)$, a minimum-weight k-link path in $G(\tau)$ weighs no more than a minimum-weight ℓ-link path. By Corollary 2.9

$$\begin{aligned} W(k) - W(\ell) &= W(k) - W(k+1) + \\ &\quad W(k+1) - W(k+2) + \cdots + \\ &\quad W(\ell - 1) - W(\ell) \\ &\leq (\ell - k)(W(k) - W(k+1)). \end{aligned}$$

We get that for all $\tau \geq W(k) - W(k+1)$, a minimum-weight k-link path in $G(\tau)$ weighs no more than a minimum-weight ℓ-link path, for any $\ell > k$. Similarly, for any $1 \leq \ell < k$, and for all $\tau \leq W(k-1) - W(k)$, a minimum-weight k-link path in $G(\tau)$ weighs no more than a minimum-weight ℓ-link path. This completes the proof for $1 < k < n - 1$. In a similar way it can be shown that for any $1 < \ell < n$, and for all $\tau \geq W(1) - W(2)$, a minimum-weight 1-link path in $G(\tau)$ weighs no more than a minimum-weight ℓ-link path; and for any $1 \leq \ell < n - 1$, and for all $\tau \leq W(n-2) - W(n-1)$, a minimum-weight $(n-1)$-link path in $G(\tau)$ weighs no more than a minimum-weight ℓ-link path. □

Let I_{opt} be the interval $[W(k) - W(k+1), W(k-1) - W(k)]$. To find a minimum-weight k-link diameter path it is sufficient to find some value τ_{opt} in the interval I_{opt}. Suppose that such a τ_{opt} is found. To compute a minimum-weight k-link diameter path in G (or equivalently, a minimum-weight diameter path in $G(\tau_{opt})$ with k links) we do the following. Apply the linear time algorithm for finding a minimum-weight diameter path in DAGs with the concave Monge property, to find two minimum-weight diameter paths in $G(\tau_{opt})$: P_a with the maximum number of links and P_b with the minimum number of links. It is easy to see that both known linear time algorithms for this problem: the one given by Wilber [15] and the one by Klawe [10] can be used to find these paths. Then, to find a minimum-weight diameter path with k links we apply Lemma 2.6. It is easy to see that finding the required links e_a and e_b in the proof of Lemma 2.6 and performing the path swap can be done in time proportional to the length of P_a; that is $O(n)$ time.

3 The algorithm

Fix an integer ℓ, which will be set appropriately in the analysis. For convenience, assume that both ℓ and k/ℓ are integers. The algorithm consists of k/ℓ stages. In the t-th stage, for $t = 1, \ldots, k/\ell$,

> either find some τ_{opt} or (i) compute the maximal range $[L_t, R_t]$ such that for all $\tau \in I_{opt}$ and all $L_t \leq i \leq R_t$, there exists a minimum-weight path from v_1 to v_i in $G(\tau)$ of length $t\ell$; and (ii) compute a minimum-weight $t\ell$-link path to each of these vertices.

Note that the range $[L_t, R_t]$ cannot be empty. Later we show that in case τ_{opt} is not found in Stage t, then for all $\tau \in I_{opt}$ and all $L_t \leq i \leq R_t$, *all* minimum-weight paths from v_1 to v_i in $G(\tau)$ have $t\ell$ links.

DEFINITION 3.1. *Let P be a path that starts at v_1. The left endpoint of the last link of P is called the* <u>anchor</u> *of P. If the anchor of a path P is in an interval I, we say that the path P is anchored in I.*

We now describe stage t of the algorithm. The input to this stage is minimum-weight $((t-1)\ell)$-link paths to v_i in G, for all $L_{t-1} \leq i \leq R_{t-1}$. All these paths are minimum-weight paths in $G(\tau)$, for all $\tau \in I_{opt}$. For $t > 1$, these paths were computed in the previous stage. For $t = 1$, $L_0 = R_0 = 1$.

Stage t consists of three steps:

Step 1: For all $R_{t-1} < j \leq n$ and all $\tau \in I_{opt}$, compute a minimum-weight path in $G(\tau)$ anchored in $[L_{t-1}, R_{t-1}]$ from v_1 to v_j. All these paths have $(t-1)\ell + 1$ links.

Step 2: Find the range $[L_t, R_t]$. If in the course of this computation some value $\tau \in I_{opt}$ is found then we are done; otherwise, continue to the next step.

Step 3: For all $L_t \leq j \leq R_t$ and all $\tau \in I_{opt}$, compute a minimum-weight path in $G(\tau)$ from v_1 to v_j. All these paths have $t\ell$ links.

We now describe each of the steps in detail.

Step 1: For $t = 1$ this step is trivial. Consider some $t > 1$. Since all the minimum-weight paths anchored in $[L_{t-1}, R_{t-1}]$ have $(t-1)\ell + 1$ links, the computation of the minimum over these paths can be done independently of τ as all comparisons involve paths with the same number of links.

Consider the $(n - R_{t-1}) \times (R_{t-1} - L_{t-1} + 1)$ matrix in which the (i, j)-th entry is the weight of a minimum-weight $(t-1)\ell$-link path from 1 to $j + L_{t-1} - 1$ in G, plus the weight of the edge $(j + L_{t-1} - 1, i + R_{t-1})$ in G. It is not difficult to see that: (i) this matrix has the concave Monge Property; and (ii) the minimum entry in row i corresponds to the weight of a minimum-weight $((t-1)\ell + 1)$-link path from 1 to $i + R_{t-1}$ anchored in $[L_{t-1}, R_{t-1}]$. Hence, all these paths can be found in $O(n)$ time by applying the matrix search algorithm of [1]. Note that the matrix need not be stored explicitly. Instead, each entry can be computed upon demand.

Step 2: Define an auxiliary DAG H_t. The DAG H_t has $n - R_{t-1} + 1$ vertices: a new source vertex s, and vertices $R_{t-1}+1, \ldots, n$ of G. For $R_{t-1}+1 \leq i < j \leq n$, the weight of edge (i, j) in H_t is the same as its weight in G. The weight of edge (s, i), for $R_{t-1}+1 \leq i \leq n$, is the weight of a minimum-weight $((t-1)\ell+1)$-link path in G from 1 to i anchored in $[L_{t-1}, R_{t-1}]$, computed in Step 1. Note that H_t has the concave Monge property. Define $H_t(\tau)$ to be the weighted DAG with the same sets of edges and vertices as H_t, in which the weight of each edge (i, j), for $R_{t-1} + 1 \leq i < j \leq n$, is incremented by τ, and the weight of each edge (s, i), for $R_{t-1} + 1 \leq i \leq n$, is incremented by $((t-1)\ell + 1)\tau$. Note that $H_t(\tau)$ has the concave Monge property as well.

We first show how to find L_t. This is done in two phases of binary search. In the first phase we find the minimum integer a such that for all $\tau \in I_{opt}$, there exists a minimum-weight path from 1 to $R_{t-1} + 2^a\ell$ in $G(\tau)$ with at least $t\ell$ links. In the second phase, if $a > 0$, we perform a binary search on all the vertices in the range $[R_{t-1} + 2^{a-1}\ell + 1, R_{t-1} + 2^a\ell]$ to find L_t.

We now describe the first phase. Initialize m to ℓ. The following procedure is done iteratively. Find a minimum-weight $(\ell-1)$-link path and a minimum-weight ℓ-link path in H_t from s to $R_{t-1} + m$. This is done by invoking the algorithm recursively. (As a matter of fact, we do the recursion only on the subgraph of H_t induced by the first $m+1$ vertices.) Let $W_H(m,\ell-1)$ and $W_H(m,\ell)$ denote the weight of these paths. Set $\xi = W_H(m,\ell-1) - W_H(m,\ell)$. Find two minimum-weight diameter paths in $G(\xi)$: P_x with the minimum number of links and P_y with the maximum number of links. If the number of links of P_x is less than or equal to k and the number of links of P_y is greater than or equal to k, then there exists a minimum-weight diameter path in $G(\xi)$ with k links. This implies that $\xi \in I_{opt}$ and we are done. For the rest of the cases we need the following two lemmas.

LEMMA 3.2. *If all minimum-weight diameter paths in $G(\xi)$ have more than k links, then for all $\tau \in I_{opt}$, all minimum-weight paths from 1 to $R_{t-1}+m$ in $G(\tau)$ have less than $t\ell$ links.*

Proof. Since all minimum-weight diameter paths in $G(\xi)$ have more than k links, $\xi < W(k) - W(k+1)$. (Recall that $W(k)-W(k+1)$ is the leftmost point of I_{opt}.) By the definition of ξ, for all $\tau > \xi$ the weight of a minimum-weight $(\ell-1)$-link path to $R_{t-1}+m$ in $H_t(\tau)$ is less than the weight of a minimum-weight ℓ-link path to $R_{t-1}+m$. It follows from Corollary 2.9 that all minimum-weight paths from s to $R_{t-1}+m$ in $H_t(\tau)$ have less than ℓ links. By the definition of H_t, for all $\tau \in I_{opt}$, a minimum-weight path in $H_t(\tau)$ from s to $R_{t-1}+m$ with d links corresponds to a minimum-weight path in $G(\tau)$ from 1 to $R_{t-1}+m$ with $d+(t-1)\ell$ links. It follows that for all $\tau \in I_{opt}$, all minimum-weight paths to $R_{t-1}+m$ in $G(\tau)$ have less than $t\ell$ links. □

In the same way we can prove:

LEMMA 3.3. *If all minimum-weight diameter paths in $G(\xi)$ have less than k links, then for all $\tau \in I_{opt}$, all minimum-weight paths from 1 to $R_{t-1}+m$ in $G(\tau)$ have at least $t\ell$ links.*

Given these two lemmas the search proceeds as follows. If P_x has more than k links then double m and iterate. Otherwise, $R(t-1)+m = R(t-1)+2^a\ell$ is the desired vertex.

We turn to the second phase of the search. Suppose that $a > 0$. (Otherwise, this phase is trivial.) We search for L_t: the first vertex in the range $[R(t-1)+2^{a-1}\ell+1, R(t-1)+2^a\ell]$ such that for all $\tau \in I_{opt}$, there exists a minimum-weight path from 1 to L_t in $G(\tau)$ with at least $t\ell$ links. This is done using binary search similar to the first phase. Initialize $\alpha = 2^{a-1}\ell+1$, $\beta = 2^a\ell$. The following procedure is done iteratively. If $\alpha = \beta$ then we are done and $L_t = R_{t-1}+\alpha$. Else, set $m = \lfloor(\beta-\alpha)/2\rfloor$. Recursively, find a minimum-weight $(\ell-1)$-link path and a minimum-weight ℓ-link path in H_t from s to $R_{t-1}+m$. Set $\xi = W_H(m,\ell-1) - W_H(m,\ell)$. Find two minimum-weight diameter paths in $G(\xi)$: P_x with the minimum number of links and P_y with the maximum number of links. If there exists a minimum-weight diameter path in $G(\xi)$ with k links, then $\xi \in I_{opt}$ and we are done. If P_x has more than k links then set $\alpha = m+1$, and iterate. Otherwise, set $\beta = m$, and iterate.

Note that if no value $\xi \in I_{opt}$ is found in this phase, then $W_H(L_t,\ell-1) - W_H(L_t,\ell) > W(k-1) - W(k)$, and hence, for all $\tau \in I_{opt}$, *all* minimum-weight paths from 1 to L_t in $G(\tau)$ have at least $t\ell$ links. Also, since L_t is the first such vertex, there must be a minimum-weight path from 1 to L_t in $G(\tau)$ with exactly $t\ell$ links.

The vertex R_t is found as follows. First, we find the first vertex b such that for all $\tau \in I_{opt}$, there exists a minimum-weight path from 1 to b in $G(\tau)$ with at least $t\ell+1$ links. This is done in the same way L_t was found above. If no value $\xi \in I_{opt}$ is found in this search, then for all $\tau \in I_{opt}$, *all* minimum-weight paths from 1 to b in $G(\tau)$ have at least $t\ell+1$ links. Next, find a minimum-weight ℓ-link path and a minimum-weight $(\ell+1)$-link path in H_t from s to $b-1$. Set $\xi = W_H(b-1,\ell) - W_H(b-1,\ell+1)$. As before, Find P_x and P_y in $G(\xi)$. If there exists a minimum-weight diameter path in $G(\xi)$ with k links, then $\xi \in I_{opt}$ and we are done. Otherwise, it must be that P_x has more than k links and $W_H(b-1,\ell) - W_H(b-1,\ell+1) < W(k) - W(k+1)$. In this case set $R_t = b-1$.

Step 3: The input to this step consists of vertices L_t and R_t; and weights $W_H(L_t,\ell-1)$, $W_H(L_t,\ell)$, $W_H(R_t,\ell)$, and $W_H(R_t,\ell+1)$. All this input is computed in the previous step.

Let I_t be the *open* interval $(W_H(R_t,\ell) - W_H(R_t,\ell+1), W_H(L_t,\ell-1) - W_H(L_t,\ell))$. Observe that by our construction I_t contains the (closed) interval I_{opt}. By Corollary 2.8 it follows that I_t is contained in all the open intervals $(W_H(i,\ell) - W_H(i,\ell+1), W_H(i,\ell-1) - W_H(i,\ell))$, for $L_t \leq i \leq R_t$. Hence, for all $\tau \in I_t$ all minimum-weight paths from s to i in $H_t(\tau)$ have ℓ links, and correspondingly, all minimum-weight paths from 1 to i in $G(\tau)$ have $t\ell$ links. We pick one such τ and apply the linear time algorithm for finding a minimum-weight diameter path in DAGs with the concave Monge property to find the minimum-weight path from 1 to all $L_t \leq i \leq R_t$ in $G(\tau)$. This can be done in one application of the linear time algorithm.

3.1 Time complexity

Let $T(n, k)$ denote the time complexity of our algorithm when the input DAG has n vertices and we are required to find a minimum-weight k-link path. It is not difficult to verify that the time complexity of the algorithm is dominated by Step 2 of each stage. It follows that $T(n, k)$ satisfies the following recursion:

$$T(n,k) \quad = \quad c \cdot \log n \cdot \sum_{i=1}^{k/\ell} (T(n_i, \ell) + n),$$

for some constant c, and some sequence $n_1, n_2, \ldots, n_{k/\ell}$, where $n \leq \sum_{i=1}^{k/\ell} n_i \leq 2n$.

Setting ℓ appropriately, it can be shown that the solution of this recursion is $T(n, k) = n2^{O(\sqrt{\log k \log\log n})}$.

We note that the space complexity of the algorithm is linear in n.

Acknowledgement. We thank Takeshi Tokuyama for helpful discussions.

References

[1] A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber, Geometric Applications of a Matrix-Searching Algorithm, *Algorithmica* 2 (1987), 195-208.

[2] A. Aggarwal and J. Park, Notes on Searching in Multidimensional Monotone Arrays, *Proc. 29th IEEE Symp. on Foundations on Computer Science* (1988), 497-512.

[3] A. Aggarwal, B. Schieber and T. Tokuyama, Finding a minimum weight K-link path in graphs with Monge property and applications, *J. of Discrete and Computational Geometry* **12** (1994), 263-280.

[4] A. Aggarwal and T. Tokuyama, Consecutive Interval Query and Dynamic Programming on Intervals, *Proc. 4th Int'l Symp. on Algorithms and Computation, Lecture Notes in Computer Science* **762**, Springer–Verlag (1993), 466-475.

[5] T. Asano, Dynamic Programming on Intervals, *Proc. 2nd Int'l Symp. on Algorithms, Lecture Notes in Computer Science* **557**, Springer–Verlag (1991), 199-207.

[6] J. Boyce, D. Dobkin, R. Drysdale, and L. Guibas, Finding Extremal Polygons, *SIAM J. on Computing* **14** (1985), 134-147.

[7] W. Bein, L. Larmore, and J. Park, The *d*-Edge Shortest-Path Problem for a Monge Graph, Preprint, 1992.

[8] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, Diameter, Width, Closest Line Pair, and Parametric Searching, *Proc. 8th ACM Symp. on Computational Geometry* (1992), 120-129.

[9] R. Cole, Slowing Down Sorting Networks to Obtain Faster Sorting Algorithms, *J. ACM* **34** (1987), 200-208.

[10] M. Klawe, A Simple Linear Time Algorithm for Concave One-Dimensional Dynamic Programming, Technical Report 89-16, University of British Columbia, Vancouver, 1989.

[11] L. Larmore and D. Hirschberg, Length-Limited Coding, *Proc. 1st ACM-SIAM Symp. on Discrete Algorithms* (1990), 310-318.

[12] L. Larmore and T. Przytycka, Parallel Construction of Trees with Optimal Weighted Path Length, *Proc. 3rd ACM Symp. on Parallel Algorithms and Architectures* (1991), 71-80.

[13] N. Megiddo, Applying Parallel Computation Algorithms in the Design of Serial Algorithms, *J. ACM* **30** (1983), 852-865.

[14] X. Wu, Optimal Quantization by Matrix Searching, *J. of Algorithms* **12** (1991), 663-673.

[15] R. Wilber, The concave least weight subsequence problem revisited, *J. of Algorithms* **9** (1988), 418-425.