JOURNAL OF ALGORITHMS **29,** 204–222 (1998)
ARTICLE NO. AL980955

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

Baruch Schieber*

IBM—Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598

Received January 1995; revised April 3, 1997

Let G be a weighted, complete, directed acyclic graph 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, for $k = \Omega(\log n)$. 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 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 Fig. 1.)

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 article, we are interested in computing the minimum weight k-link path from 1 to n in concave Monge DAGs, i.e., weighted DAGs

*E-mail address: sbar@watson.ibm.com.

0196-6774/98 $25.00

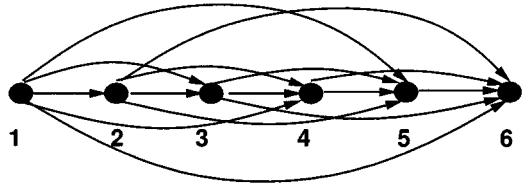

FIG. 1. Complete DAG.

whose weights satisfy the concave Monge property. We assume that the weights are not given explicitly, instead whenever a weight $w(i, j)$ is required it can be computed in constant time. (Otherwise, any algorithm would require $\Omega(n^2)$ running time.)

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. Bein, Larmore, and Park [7] and Aggarwal, Schieber, and Tokuyama [3] gave a weakly polynomial algorithm for this problem when the weights are restricted to be integers. The algorithm runs in $O(n \log U)$ time, where U is the maximum absolute value of the weights. Aggarwal, Schieber, and Tokuyama [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 article is a $n2^{O(\sqrt{\log k \log\log n})}$ time algorithm for computing the minimum weight k-link path, for $k = \Omega(\log n)$. Note that this algorithm is superior to both the algorithm given in [3] and the $O(nk)$ time naive algorithm whenever $k = \Omega(\log n)$. From now on, we assume that this is the case.

In [3], Aggarwal, Schieber, and Tokuyama posed the question of designing an $O(n \cdot \text{polylog}(n, k))$ 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 because for any k, such that $\log k = \Omega((\log\log n)^{(1+\alpha)})$, for some $\alpha > 0$, 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 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.

In [3], Aggarwal, Schieber, and Tokuyama describe five applications of the algorithm for minimum weight k-link path in concave Monge DAGs.

These applications are to the following problems:

I. Computing optimal length limited prefix free binary codes.

II. Data compression by quantization.

III. Computing k maximum weight cliques in interval graphs.

IV. Computing the maximum area k-gon and the maximum perimeter k-gon that are contained in a given convex n-gon.

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 all of these applications we improve upon the results in [3]. Specifically, for $k = \Omega(\log n)$, the improved running time for Applications I, II, IV, and V is $n2^{O(\sqrt{\log k \log\log n})}$, while the improved running time for Application III is $O(m) + n2^{O(\sqrt{\log k \log\log n})}$. To make the article self-contained we describe these applications in more detail in Section 4.

The rest of the article 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 DIRECTED ACYCLIC GRAPHS

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 $G(\tau)$ also 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 that G has the concave Monge property.

LEMMA 1. *If for some τ a minimum weight diameter path in $G(\tau)$ has k links, then this path is a minimum weight k-link diameter path in G.*

Proof. The lemma follows from the fact that the difference between the weight of any k-link diameter path in $G(\tau)$ and the weight of the same path in G is $k\tau$. ∎

LEMMA 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 that 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,

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

Because $\tau - \xi > 0$, we have that $\ell \geq k$. ∎

DEFINITION 1. An edge (i_1, j_1) *covers* 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). (See Fig. 2.)

LEMMA 3. *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.*

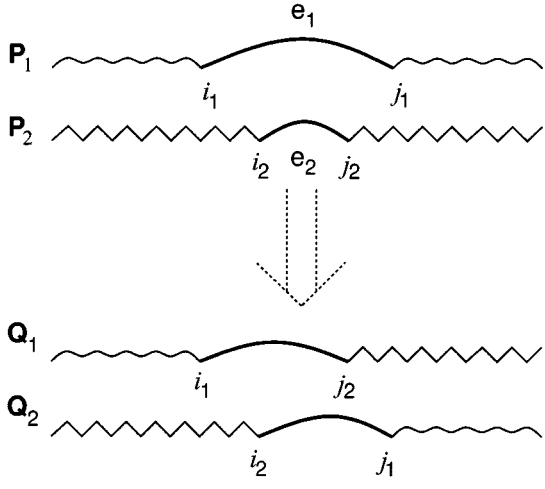

FIG. 2. The swap operation.

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

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

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

For $a \le b$, let P_1 and P_2 be two distinct paths from 1 to a and from 1 to b, respectively. Suppose that P_1 has k_1 links, P_2 has k_2 links, where $k_1 \ge k_2$.

LEMMA 4. *For any $0 \le x \le k_1 - k_2$ there are links $e_1 = (i_1, j_1)$ of P_1 and $e_2 = (i_2, j_2)$ of P_2 with the following two properties.*

1. *Edge e_2 covers edge e_1.*
2. *The prefix of P_1 ending at i_1 has x more links than the prefix of P_2 ending at i_2.*

Proof. We prove the lemma by (double) induction on k_1 and k_2: the lengths of paths P_1 and P_2.

Induction Basis. Consider a path P_1 from 1 to a with k_1 links. Suppose that $k_2 = 1$, that is, path P_2 consists of a single link $(1, b)$. In this case set $e_2 = (1, b)$, and set e_1 as the $(x + 1)$st link in P_1. It is easy to see that e_2 covers e_1 and that the second property holds because the prefix of P_1 up to the left endpoint of e_1 consists of x links.

Induction Step. Assume that the lemma holds for (i) all pairs of paths P_1 and P_2 (that obey the lemma conditions), such that P_1 has less than k_1 links, and for (ii) all pairs of paths P_1 and P_2, such that P_1 has k_1 links and P_2 has k' links, for $1 \le k' < k_2 \le k_1$. We show that the lemma holds also for all pairs of paths P_1 with k_1 links and P_2 with k_2 links. Consider such a pair P_1 and P_2. We distinguish between two cases depending on the last link of P_2, denoted (c, b).

Case 1. The edge (c, b) does not cover any edges of P_1. This implies that $c > d$, where (d, a) is the last link of P_1. Let P_1' be the prefix of P_1 ending at d, and let P_2' be the prefix of P_2 ending at c. Note that the pair of paths P_1' and P_2' with $k_1 - 1$ and $k_2 - 1$ links, respectively, obey the lemma conditions and thus by the induction hypothesis for any $0 \le x \le (k_1 - 1) - (k_2 - 1) = k_1 - k_2$ there are links $e_1 = (i_1, j_1)$ of P_1' and $e_2 = (i_2, j_2)$ of P_2' that satisfy the lemma properties. Because e_1 and e_2 are also in P_1 and P_2 it follows that the lemma holds also for P_1 and P_2.

Case 2. The edge (c, b) covers the last $y > 0$ edges of P_1. Denote the last y edges of P_1 by $(d_1, d_2), (d_2, d_3), \ldots, (d_y, d_{y+1} = a)$. Suppose that $k_1 - k_2 - y < x \le k_1 - k_2$, in this case set $e_2 = (c, b)$ and $e_1 = (d_{x-k_1+k_2+y}, d_{x-k_1+k_2+y+1})$. Clearly, e_2 covers e_1, also the prefix of P_1 ending at $d_{x-k_1+k_2+y}$ has x more links than the prefix of P_2 ending at c. The remaining case we have to consider is $0 \le x \le k_1 - k_2 - y$, and $y \le k_1 - k_2$. Let d be the vertex preceding d_1 in P_1, let P_1' be the prefix of P_1 ending at d, and let P_2' be the prefix of P_2 ending at c. Note that P_1' has $k_1 - y - 1$ links and note that P_2 has $k_2 - 1$ links. The pair P_1' and P_2' obey the lemma conditions and thus by the induction hypothesis for any $0 \le x \le (k_1 - y - 1) - (k_2 - 1) = k_1 - k_2 - y$ there are links $e_1 = (i_1, j_1)$ of P_1' and $e_2 = (i_2, j_2)$ of P_2' that satisfy the lemma properties. Because e_1 and e_2 are also in P_1 and P_2 it follows that the lemma holds also for P_1 and P_2. ▮

LEMMA 5. *Let a, b, P_1, P_2, k_1, and k_2 be as in the previous text. For any k in the range $[k_2, k_1]$, there are paths Q_1 with k links from* 1 *to a and Q_2 with $k_1 + k_2 - k$ links from* 1 *to b such that 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. Fix some k in the range $[k_2, k_1]$. By Lemma 4 there are links $e_1 = (i_1, j_1)$ in P_1 and $e_2 = (i_2, j_2)$ in P_2 such that edge e_2 covers edge e_1, and the prefix of P_1 ending at i_1 has $k - k_2$ more links than the prefix of P_2 ending at i_2. Perform a path swap with respect to e_1 and e_2 to obtain two paths Q_1 and Q_2 from 1 to a and b, respectively. Because Q_1 is created by connecting the prefix of P_1 ending at i_1 with the suffix of P_2 starting at j_2, the length of Q_1 is $k_2 + (k - k_2) = k$. Similarly, the length of Q_2 is $k_1 - (k - k_2) = k_1 + k_2 - k$. Lemma 3 implies that 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. ▮

DEFINITION 2. For $1 < a \le n$ and $1 \le \ell < a$, let $P(a, \ell)$ denote a 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 propositions follow from Lemma 5.

PROPOSITION 6. *For $1 < a < b \le n$ and $1 \le \ell < a - 1$,*

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

Proof. Let P_1 be a minimum weight $(\ell + 1)$-link path in G from 1 to a and let P' be a minimum weight ℓ-link path from 1 to b. By Lemma 5 there are paths Q_1 with ℓ links from 1 to a and Q_2 with $(\ell + 1)$ links

from 1 to b such that the sum of the weights of these paths is at most the sum of the weights of paths P_1 and P_2. The proposition follows. ∎

PROPOSITION 7. *For* $1 < a \leq n$ *and* $1 < \ell < a - 1$,

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

Proof. Let P be a minimum weight $(\ell + 1)$-link path in G from 1 to a and let P' be a minimum weight $(\ell - 1)$-link path from 1 to a. By Lemma 5 there are paths Q and Q' with ℓ links from 1 to a such that the sum of the weights of these paths is at most the sum of the weights of paths P and P'. The proposition follows. ∎

Proposition 7 implies that the function $W(a, .)$ is unimodal; or in other words, any local minimum of the function is the global minimum.

In the next proposition we consider minimum weight paths with ℓ links from 1 to some vertex i. Note the distinction between minimum weight ℓ-link paths which are paths of minimum weight among all path with exactly ℓ links from 1 to i, and minimum weight paths with ℓ links which are paths of minimum weight among *all* paths from 1 to i that also have ℓ links, such paths may not necessarily exist.

PROPOSITION 8. *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*.

Proof. To obtain a contradiction assume that there exist three vertices $a < b < c$, such that there exist minimum weight paths P_1 and P_3 with ℓ links from 1 to a and to c, respectively, but there is no minimum weight path from 1 to b with ℓ links. Let P_2 be a minimum weight path from 1 to b with k links. If $k < \ell$, then, by Lemma 5, there are minimum weight paths Q_1 with k links from 1 to a and Q_2 with ℓ links from 1 to b. Otherwise, there are minimum weight paths Q_3 with k links from 1 to c and Q_2 with ℓ from 1 to b; a contradiction. ∎

LEMMA 9. *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 $k < \ell < n$. Note that a minimum weight x-link path in $G(\tau)$ weighs $W(x) + x\tau$, it follows that for $\tau \geq (W(k) - W(\ell))/(\ell - k)$, a minimum weight k-link path in

$G(\tau)$ weighs no more than a minimum weight ℓ-link path in $G(\tau)$. By Proposition 7,

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

We get that for all $\tau \ge W(k) - W(k+1)$, a minimum weight k-link path in $G(\tau)$ weighs no more than a minimum weight ℓ-link path in $G(\tau)$, for any $\ell > k$. Now consider any $1 \le \ell < k$. For $\tau \le (W(\ell) - W(k))/(k - \ell)$, a minimum weight k-link path in $G(\tau)$ weighs no more than a minimum weight ℓ-link path in $G(\tau)$. Applying Proposition 7 again we get that for all $\tau \le W(k-1) - W(k)$, 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 $\tau \ge W(1) - W(2)$, a minimum weight 1-link path in $G(\tau)$ weighs no more than a minimum weight ℓ-link path; and for any $1 \le \ell < n - 1$, and for all $\tau \le W(n-2) - W(n-1)$, 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 $G(\tau_{\text{opt}})$ with k links) we do the following. First, we apply either the linear time algorithm for finding a minimum weight diameter path in DAGs with the concave Monge property given by Wilber [15] or the one given by Klawe [10], to find two minimum weight diameter paths in $G(\tau_{\text{opt}})$: P_1 with the maximum number of links and P_2 with the minimum number of links. The modification required in either of these algorithms in order to find P_1 (resp., P_2) is in the comparisons of path weights: whenever the weights of two paths are compared and found to be equal the path with more (resp., less) links is considered lighter. Second, we apply Lemma 5 to find a minimum weight diameter path with k links. It is easy to see that finding the required links e_1 and e_2 in the proof of Lemma 5 and that performing the path swap can be done in time proportional to the length of P_1; that is, $O(n)$ time.

3. THE ALGORITHM

The algorithm either explicitly computes a minimum k-link diameter path or finds some τ_{opt} which can be used to find such a path as explained in the previous section.

Fix an integer ℓ, which is 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 tth 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 $\tau \in I_{\text{opt}}$ and for all $L_t \le i \le R_t$, there exists a minimum weight path from 1 to i in $G(\tau)$ with $t\ell$ links; 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 $t \in I_{\text{opt}}$ and for all $L_t \le i \le R_t$, *all* minimum weight paths from 1 to i in $G(\tau)$ have $t\ell$ links. This implies that $L_t > R_{t-1}$.

The input to stage t consists of minimum weight $((t-1)\ell)$-link paths from 1 to i in G, for all $L_{t-1} \le i \le R_{t-1}$. All these paths are minimum weight paths in $G(\tau)$, for all $\tau \in I_{\text{opt}}$. For $t > 1$, these paths were computed in the previous stage. For $t = 1$, $L_0 = R_0 = 1$.

The main tool used in stage t is a decision procedure that for a given vertex $R_{t-1} + m$ and a parameter $x > (t-1)\ell$, decides whether for all $\tau \in I_{\text{opt}}$ all minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have less than x links. The decision procedure either terminates with a "yes/no" answer or finds some τ_{opt}. We use the decision procedure to binary search L_t and R_t.

Before turning to stage t we describe the decision procedure. In this procedure we recursively compute minimum k'-link paths for some $k' < k$ in an auxiliary DAG H_k. We start with the following definition.

DEFINITION 3. Let P be a path that starts at 1. The left endpoint of the last link of P is called the *anchor* of P. Similarly, the left endpoint of the ith link of P is called the *i-anchor* of P. If the anchor (i-anchor) of a path P is in an interval I, we say that the path P is anchored (i-anchored) in I.

Fix $x > (t-1)\ell$ and m and consider any minimum weight path P from 1 to $R_{t-1} + m$ in $G(\tau)$, for any $\tau \in I_{\text{opt}}$. We claim that P is $((t-1)\ell + 1)$-anchored at $[L_{t-1}, R_{t-1}]$. This follows from our assumption that $[L_{t-1}, R_{t-1}]$ is the maximal range such that for all $\tau \in I_{\text{opt}}$ and all $L_{t-1} \le i \le R_{t-1}$, there exists a minimum weight path from 1 to i in $G(\tau)$ with $(t-1)\ell$ links.

We now define the auxiliary DAG H_t. In this DAG the prefix of any such path P with $(t-1)\ell + 1$ edges is represented by a single edge. 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 $R_{t-1} + 1 \le i < j \le n$, the weight of edge (i, j) in H_t is the same as its weight in G. The weight of edge (s, i), for $R_{t-1} + 1 \le i \le n$, is the weight of a minimum weight $((t-1)\ell + 1)$-link path in G from 1 to i that is anchored in $[L_{t-1}, R_{t-1}]$. (We describe how

to compute these paths later.) 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 $R_{t-1} + 1 \le i < j \le n$, is incremented by τ, and the weight of each edge (s, i), for $R_{t-1} + 1 \le i \le n$, is incremented by $((t-1)\ell + 1)\tau$. Because for all $\tau \in I_{\text{opt}}$, any minimum weight path in $G(\tau)$ from 1 to $R_{t-1} + m$ is $((t-1)l + 1)$-anchored in $[l_{t-1}, R_{t-1}]$, 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. In order to be able to compute minimum k'-link paths in $H_t(\tau)$ recursively, we need to show that H_t has the concave Monge property.

PROPOSITION 10. *The DAG H_t has the concave Monge property.*

Proof. Because for all $R_{t-1} + 1 \le i < j \le n$, the weight of edge (i, j) in H_t is the same as its weight in G, the inequality corresponding to the concave Monge property holds for all these indices. Thus, all that is left to show is that for every $R_{t-1} + 1 < j < n$, the difference between weight of edge (s, j) and the weight of edge $(s, j+1)$ is at most $w(R_{t-1} + 1, j) - w(R_{t-1} + 1, j + 1)$. Let P_j and P_{j+1} be minimum weight $((t-1)\ell + 1)$-link paths anchored in $[L_{t-1}, R_{t-1}]$ from 1 to j and $j+1$, respectively. Recall that the weights of edges (s, j) and $(s, j+1)$ are the weights of P_j and P_{j+1}, respectively. Suppose that P_{j+1} is anchored at $i \in [L_{t-1}, R_{t-1}]$. Let Q_j be the $((t-1)\ell + 1)$-link path given by the prefix of P_{j+1} ending at i with the edge (i, j). Clearly, the weight of Q_j is at least the weight of P_j. We get that the difference between the weights of P_j and P_{j+1} is less than or equal to the difference between the weights of Q_j and P_{j+1}. Because the prefixes of Q_j and P_{j+1} ending at i are the same, this difference is equal to $w(i, j) - w(i, j+1)$. By the concave Monge property of G we get

$$\begin{aligned} w(i,j) - w(i,j+1) &\le w(i+1,j) - w(i+1,j+1) \\ &\le w(i+2,j) - w(i+2,j+1) \\ &\le \cdots \le w(R_{t-1}+1,j) - w(R_{t-1}+1,j+1). \end{aligned}$$

∎

Recall that given m and x, the decision procedure decides whether for all $\tau \in I_{\text{opt}}$ all minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have less than x links. The decision procedure consists of two steps.

Step I. Let $y = x - (t-1)\ell$. Find a minimum weight $(y-1)$-link path, and find a minimum weight y-link path in H_t from s to $R_{t-1} + m$, by invoking the algorithm recursively. (However, as a matter of fact, the recursion is done only on the subgraph of H_t induced by the first $m + 1$

vertices.) Let $W_H(m, y-1)$ and $W_H(m, y)$ denote the weights of these paths.

Step II. Set $\xi = W_H(m, y-1) - W_H(m, y)$. Find two minimum weight diameter paths in $G(\xi)$: P_1 with the maximum number of links and P_2 with the minimum number of links. If the number of links of P_1 is greater than or equal to k and the number of links of P_2 is less than or equal to k, then there exists a minimum weight diameter path in $G(\xi)$ with k links. This implies that $\xi \in I_{\text{opt}}$ and we are done. In the next two lemmas we prove: (i) If the number of links of P_1 is strictly less than k, then for all $\tau \in I_{\text{opt}}$, *all* minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have at least x links. (ii) If the number of links of P_2 is strictly greater than k, then for all $\tau \in I_{\text{opt}}$, *all* minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have less than x links. This concludes the decision procedure.

LEMMA 11. *If all minimum weight diameter paths in $G(\xi)$ have less than k links, then for all $\tau \in I_{\text{opt}}$, all* minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have at least x links.

Proof. Because all minimum weight diameter paths in $G(\xi)$ have less than k links, $\xi > W(k-1) - W(k)$. (Recall that $W(k-1) - W(k)$ is the rightmost point of I_{opt}.) By the definition of ξ, for all $\tau < \xi$ the weight of a minimum weight y-link path to $R_{t-1} + m$ in $H_t(\tau)$ is less than the weight of a minimum weight $(y-1)$-link path to $R_{t-1} + m$. It follows from the unimodality of the function $W_H(m, .)$ (Proposition 7) that all minimum weight paths from s to $R_{t-1} + m$ in $H_t(\tau)$ have at least y links. Recall that for all $\tau \in I_{\text{opt}}$, 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. Because $x = y + (t-1)\ell$, for all $\tau \in I_{\text{opt}}$, all minimum weight paths to $R_{t-1} + m$ in $G(\tau)$ have at least x links. ∎

In a similar way we can prove:

LEMMA 12. *If all minimum weight diameter paths in $G(\xi)$ have more than k links, then for all $\tau \in I_{\text{opt}}$, all minimum weight paths from* 1 *to $R_{t-1} + m$ in $G(\tau)$ have less than x links.*

We turn back to the description of stage t. Stage t consists of three steps:

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

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

Step 3. For all $L_t \le j \le R_t$ and for all $\tau \in I_{\text{opt}}$, compute a minimum weight path in $G(\tau)$ from 1 to 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 $t > 1$. Because 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.

Let M_t be 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 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 $[L_{t-1}, R_{t-1}]$.

PROPOSITION 13. *The matrix M_t has the concave Monge property.*

Proof. To prove the property we have to show that for every $1 \le i < n - R_{t-1}$ and for every $1 \le j < R_{t-1} - L_{t-1} + 1$, $M_t[i, j] - M_t[i+1, j] \le M_t[i, j+1] - M_t[i+1, j+1]$. By our definition $M_t[i, j] = W(j + L_{t-1} - 1, (t-1)\ell) + w(j + L_{t-1} - 1, i + R_{t-1})$. Because the weights of G obey the concave Monge property we get

$$\begin{aligned} &M_t[i, j] - M_t[i+1, j] \\ &\quad = w(j + L_{t-1} - 1, i + R_{t-1}) - w(j + L_{t-1} - 1, i + R_{t-1} + 1) \\ &\quad \le w(j + L_{t-1}, i + R_{t-1}) - w(j + L_{t-1}, i + R_{t-1} + 1) \\ &\quad = M_t[i, j+1] - M_t[i+1, j+1]. \end{aligned}$$

∎

It follows that all the minimum weight paths anchored in $[L_{t-1}, R_{t-1}]$ 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. First, we show how to find L_t. This is done in two phases of binary search. In the first phase we find the minimum integer $a \ge 0$ such that for all $\tau \in I_{\text{opt}}$, 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.

The First Phase. Initialize m to ℓ, and test using the decision procedure whether for all $\tau \in I_{\text{opt}}$, *all* minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have strictly less than $t\ell$ links. If some τ_{opt} is found

during the execution then we are done. Else, if the answer is "yes," then double m and iterate. Otherwise, $a = \log_2(m/\ell)$ is the minimum integer such that for all $\tau \in I_{\text{opt}}$, there exists a minimum weight path from 1 to $R_{t-1} + 2^a\ell$ in $G(\tau)$ with at least $t\ell$ links.

The Second Phase. If $a = 0$, then there exists a minimum weight path from 1 to $R_{t-1} + \ell$ in $G(\tau)$ with at least $t\ell$ links. Because *all* minimum weight paths in $G(\tau)$ from 1 to any $i \in [L_{t-1}, R_{t-1}]$ have $(t-1)\ell$ links, any minimum weight path from 1 to $R_{t-1} + \ell$ in $G(\tau)$ has at most $t\ell$ links. It follows that any minimum weight path from 1 to $R_{t-1} + \ell$ in $G(\tau)$ has exactly $t\ell$ links. Because all minimum weight paths from 1 to $R_{t-1} + j$, for $j < \ell$, have strictly less than $t\ell$ links, we get that if $a = 0$ then $L_t = R_{t-1} + \ell$.

Suppose that $a > 0$. In this phase 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 $\tau \in I_{\text{opt}}$, 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 $m = \lfloor(\beta + \alpha)/2\rfloor$, and test whether for all $\tau \in I_{\text{opt}}$, *all* minimum weight paths from 1 to $R_{t-1} + m$ in $G(\tau)$ have less than $t\ell$ links. If some τ_{opt} is found during the execution then we are done. Else, if the answer is "yes," then set $\alpha = m + 1$, and iterate. Otherwise, set $\beta = m$, and iterate.

Note that if no value τ_{opt} is found in the search then by Lemma 11 for all $\tau \in I_{\text{opt}}$, *all* minimum weight paths from 1 to L_t in $G(\tau)$ have at least $t\ell$ links. Also, because 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.

To find the vertex R_t, we search for the minimum b such that for all $\tau \in I_{\text{opt}}$, all the minimum weight paths from 1 to b in $G(\tau)$ have at least $t\ell + 1$ links. This is done by a binary search using the decision procedure similar to the way L_t was found in the previous text. However, as before, if no value τ_{opt} is found in the search then for all $\tau \in I_{\text{opt}}$, *all* minimum weight paths from 1 to b in $G(\tau)$ have at least $t\ell + 1$ links. Clearly, in this case $R_t = b - 1$. It follows that for all $\tau \in I_{\text{opt}}$ and for all $L_t \le i \le R_t$, *all* minimum weight paths from 1 to i in $G(\tau)$ have exactly $t\ell$ links.

Step 3. To compute a minimum weight path in $G(\tau)$ from 1 to j, for all $L_t \le j \le R_t$ and for all $\tau \in I_{\text{opt}}$, we first compute the weights $W_H(L_t, \ell - 1)$, $W_H(L_t, \ell)$, $W_H(R_t, \ell)$, and $W_H(R_t, \ell + 1)$ (in case they were not computed in the previous step), by a recursive call to our algorithm.

Recall that $I_{\text{opt}} = [W(k) - W(k+1), W(k-1) - W(k)]$. Because for all $\tau \in I_{\text{opt}}$ and for all $L_t \le i \le R_t$, *all* minimum weight paths from 1 to i in $G(\tau)$ have exactly $t\ell$ links, *all* minimum weight paths from s to i in

$H_t(\tau)$ have exactly ℓ links. It follows that $W(k-1) - W(k) < W_h(L_t, \ell - 1) - W_H(L_t, \ell)$, and $W(k) - W(k+1) > W_H(R_t, \ell) - W_H(R_t, \ell + 1)$. Alternatively, in other words, the (closed) interval I_{opt} is contained in the (open) interval $I_t = (W_H(R_t, \ell) - W_H(R_t, \ell + 1), W_H(L_t, \ell - 1) - W_H(L_t, \ell))$. By Proposition 6 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 $L_t \le i \le R_t$. 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 we 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 $L_t \le i \le R_t$ in $G(\tau)$. This can be done in one application of the linear time algorithm.

3.1. *Time and Space 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. In Step 2 we perform a binary search where in each iteration of this search we call the algorithm recursively to find minimum weight ℓ-links paths in an auxiliary DAG, and we invoke a linear time algorithm for finding a minimum weight diameter path in $G(\tau)$. The size of the auxiliary DAG in the ith stage is bounded by $2(R_i - R_{i-1})$. It follows that $T(n, k)$ satisfies the following recursion,

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

for some constant c, and for some sequence $n_1, n_2, \ldots, n_{k/\ell}$, where $n \le \sum_{i=1}^{k/\ell} n_i \le 2n$. (All logarithms are base 2.)

We claim that if $k = \Omega(\log n)$, then the solution of this recursion is $n2^{O(\sqrt{\log k \log\log n})}$. To see this, it is sufficient to prove that for some ℓ there exists some constant α such that

$$n2^{\alpha\sqrt{\log k \log\log n}} \ge c \cdot \log n \cdot \sum_{i=1}^{k/\ell} \left(n_i 2^{\alpha\sqrt{\log \ell \log\log n_i}} + n\right).$$

For this we prove

$$n2^{\alpha\sqrt{\log k \log\log n}} \ge c \cdot n \cdot \log n \cdot \left(2 \cdot 2^{\alpha\sqrt{\log \ell \log\log n}} + \frac{k}{\ell}\right).$$

This is done by showing that for some α each of the terms in the right-hand side is less than or equal to half of the left-hand side. We set ℓ to be the minimum value for which $n2^{\alpha\sqrt{\log k \log\log n}} \geq 2c \cdot n \cdot \log n \cdot k/\ell$; that is, $\ell = 2c \cdot \log n \cdot k \cdot 2^{-\alpha\sqrt{\log k \log\log n}}$. Taking the base 2 logarithm from both hands, the other inequality translates to

$$\alpha\sqrt{\log k \log\log n} \geq 2 + \log c + \log\log n + \alpha\sqrt{\log\log n} \cdot\sqrt{1 + \log c + \log\log n + \log k - \alpha\sqrt{\log k \log\log n}}\,.$$

By our assumption $k = \Omega(\log n)$. Hence, for large enough n and for some $x \geq \frac{3}{4}$, $x^2 \log\log n \leq \log k \leq (x^2 + 1)\log\log n$. For large enough n: $2 + \log c \leq \frac{1}{2}\log\log n$. We get that it is sufficient to show

$$\alpha x \log\log n \geq 2\log\log n + \alpha\sqrt{\log\log n} \cdot\sqrt{(x^2 + 1)\log\log n - (\alpha - 2)x \log\log n}\,.$$

This inequality holds if

$$\alpha\left(x - \sqrt{x^2 + 1 - (\alpha - 2)x}\right) \geq 2.$$

It is easy to verify that this holds for any $x \geq \frac{3}{4}$ and $\alpha = 4$.

We note that the space complexity of the algorithm is $O(n)$.

4. APPLICATIONS

The algorithm for minimum weight k-link path in concave Monge DAGs has several applications. In the following text, we describe five of these applications to: data compression (Applications I and II), interval graphs (Application III), and geometric path finding (Applications IV and V). All these applications were originally described in [3], and we include them here to make the article self-contained.

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: $\{x_1, x_2, \ldots, x_n\} \to \mathscr{R}$ be a real valued function, where $\mathscr{R}$ is the set of the real numbers and $x_1 \le x_2 \le \cdots \le x_n$ are real numbers. Fix k and consider a sorted set of real numbers $Z = \{z_1, z_2, \ldots, z_k\}$ and a mapping ψ: $\{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$. 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. An optimal quantization is the one that minimizes the error. It is easy to see that in an 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 Fig. 3. Wu [14] showed that computing an 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].

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 Fig. 4.) Aggarwal, Schieber, and Tokuyama [3] showed how this problem can be reduced to finding a minimum weight k-link path in a DAG with n vertices. Aggarwal and Tokuyama [4] designed a data structure in which the overhead involved in this reduction is $O(m + n \log n)$ time. Hence, the application of our algorithm yields an $O(m) + n2^{O(\sqrt{\log k \log\log n})}$ time solution for this problem, improving on previous results of [5, 3, 4].

Application IV. The computation of the maximum area k-gon and the maximum perimeter k-gon that are contained in a given convex n-gon. (See Fig. 5.) For this problem Boyce *et al.* [6] provided an $O(nk \log n)$ time

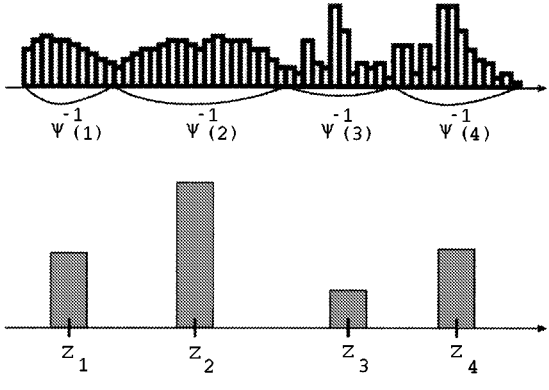

FIG. 3. Quantization ($k = 4$).

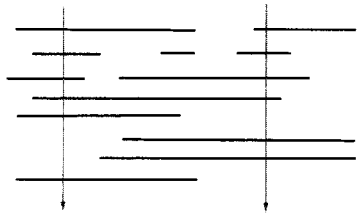

FIG. 4. k maximum weight cliques of interval graph ($k = 2$).

algorithm that was later improved by Aggarwal *et al.* [1] to $O(nk + n \log n)$ time, and by Aggarwal, Schieber, and Tokuyama [3] to $O(n\sqrt{k \log n} + n \log n)$ time. Aggarwal *et al.* [1] showed that the distance matrix involved in computing the maximum area and the maximum perimeter polygon contained in a given convex polygon has the convex Monge property. Because finding the maximum path in convex DAGs is equivalent to finding the minimum path in concave DAGs, we can apply our algorithm to achieve an $n2^{O(\sqrt{\log k \log\log n})}$ time algorithm for the problem.

Application V. The computation of 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 the polygon touching edge-to-edge. (See Fig. 6.) This problem is the dual of the previous problem, and thus can also be solved in $n2^{O(\sqrt{\log k \log\log n})}$ time, improving on the previous results mentioned earlier.

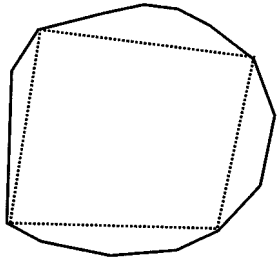

FIG. 5. Max-area inscribed polygon.

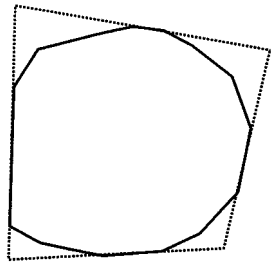

FIG. 6. Min-area inscribed polygon with edge-to-edge contact.

ACKNOWLEDGMENT

We thank Takeshi Tokuyama for helpful discussions, and the anonymous referees for their helpful suggestions.

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, *in* "Proceedings of the Twenty-Ninth IEEE Symposium on Foundations on Computer Science, 1988," pp. 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. Discrete Comput. Geom.* **12** (1994), 263–280.
4. A. Aggarwal and T. Tokuyama, Consecutive interval query and dynamic programming on intervals, *in* "Proceedings of the Fourth International Symposium on Algorithms and Computation," Lecture Notes in Computer Science, Vol. 762, pp. 466–475, Springer-Verlag, Berlin/New York, 1993.
5. T. Asano, Dynamic programming on intervals, *in* "Proceedings of the Second International Symposium on Algorithms and Computation," Lecture Notes in Computer Science, Vol. 557, pp. 199–207, Springer-Verlag, Berlin/New York, 1991.
6. J. Boyce, D. Dobkin, R. Drysdale, and L. Guibas, Finding extremal polygons, *SIAM J. Comput.* **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, *in* "Proceedings of the Eighth ACM Symposium on Computational Geometry, 1992," pp. 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, *in* "Proceedings of the First ACM-SIAM Symposium on Discrete Algorithms, 1990," pp. 310–318.

12. L. Larmore and T. Przytycka, Parallel construction of trees with optimal weighted path length, *in* "Proceedings of the Third ACM Symposium on Parallel Algorithms and Architectures, 1991," pp. 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. Algorithms* **12** (1991), 663–673.
15. R. Wilber, The concave least weight subsequence problem revisited, *J. Algorithms* **9** (1988), 418–425.