SAND--92-1724C

DE93 009828

The d-Edge Shortest-Path Problem for a Monge Graph

(extended abstract)

Wolfgang W. Bein * Lawrence L. Larmore † James K. Park ‡

July 14, 1992

Abstract

A complete edge-weighted directed graph on vertices $1, 2, \ldots, n$ that assigns cost $c[i, j]$ to the edge (i, j) is called *Monge* if its edge costs form a Monge array, i.e., for all $i < k$ and $j < \ell$, $c[i, j]+c[k, \ell] \leq c[i, \ell]+c[k, j]$. One reason Monge graphs are interesting is that shortest paths can be computed quite quickly in such graphs. In particular, Wilber (1988) showed that the shortest path from vertex 1 to vertex n of a Monge graph can be computed in $O(n)$ time, and Aggarwal, Klawe, Moran, Shor, and Wilber (1987) showed that the shortest d-edge 1-to-n path (i.e., the shortest path among all 1-to-n paths with exactly d edges) can be computed in $O(dn)$ time. This paper's contribution is a new algorithm for the latter problem. Assuming $0 \leq c[i, j] \leq U$ and $c[i, j + 1] + c[i + 1, j] - c[i, j] - c[i + 1, j + 1] \geq L > 0$ for all i and j, our algorithm runs in $O(n(1 + \lg(U/L)))$ time. Thus, when $d \gg 1 + \lg(U/L)$, our algorithm represents a significant improvement over Aggarwal et al.'s $O(dn)$-time algorithm. We also present several applications of our algorithm; they include length-limited Huffman coding, finding the maximum-perimeter d-gon inscribed in a given convex n-gon, and a digital-signal-compression problem.

1 Introduction

Given an edge-weighted directed graph $G = (V, E)$ and two distinguished vertices s and t, the problem of computing the shortest (i.e., minimum-cost) path from s to t is a fundamental problem in combinatorial optimization. Assuming the edge costs are nonnegative, the best general algorithms known for this problem, due to Fredman and Tarjan [7], Johnson [11], and Ahuja, Melhorn, Orlin, and Tarjan [4], run in $O(m + n \lg n)$, $O(m \lg\lg U)$, and $O(m + n\sqrt{\lg U})$ time, respectively, where $m = |E|$, $n = |V|$, and U is the maximum edge cost. If the path between s and t is restricted to have exactly d edges, then the shortest-path problem seems to become somewhat more difficult; the best algorithms (that we know of) for this problem run in $O(dm)$ and $O(n^2 \lg d)$ time.

In this paper, we focus on a special case of the d-edge shortest-path problem defined above. Specifically, we consider the problem of finding the shortest d-edge path from vertex 1 to vertex n of an n-vertex Monge graph. A complete edge-weighted directed graph on vertices $1, 2, \ldots, n$ that assigns cost $c[i, j]$ to the edge (i, j) is called *Monge* if its edge costs form a Monge array. An $m \times n$ array $C = \{c[i, j]\}$ is called *Monge* if for all rows i and columns j satisfying $1 \leq i < m$ and $1 \leq j < n$, we have

$$c[i, j] + c[i + 1, j + 1] \leq c[i, j + 1] + c[i + 1, j] .$$

*Department of Computer Science, University of New Mexico, Albuquerque, NM 87131.

†Department of Computer Science, University of California, Riverside, CA 92521.

‡Algorithms and Discrete Mathematics Department (Org. 1423), Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185. This work was supported by the U.S. Department of Energy under Contract DE-AC04-76DP00789.

Equivalently, C is Monge if for all $i < k$ and $j < \ell$,

$$c[i,j] + c[k,\ell] \;\leq\; c[i,\ell] + c[k,j]\;,$$

as it is not hard to show that this latter property follows from the former. The name of the French mathematician Gaspard Monge (1746–1818) is associated with such arrays because of work done by Hoffman [9] on easily-solved special cases of the transportation problem. (Hoffman showed that if the cost array associated with a transportation problem is an $m \times n$ Monge array, then a simple greedy algorithm solves the transportation problem in $O(m+n)$ time.)

The structure of Monge arrays greatly facilitates the computation of shortest paths in Monge graphs. One immediate consequence of this structure is that only monotonic paths need be considered in computing a shortest unrestricted 1-to-n path or a shortest d-edge 1-to-n path. (A path $((i_1,i_2),(i_2,i_3),\ldots,(i_{k-1},i_k))$ is called *monotonic* if $i_1 < i_2 < \cdots < i_k$.) In other words, setting $c[i,j] = +\infty$ for all $i \geq j$ does not change the solution of either shortest-path problem. (This property is also useful when solving the traveling-salesman problem for a Monge graph; see J. Park [16].)

A second, more important consequence of Mongité is that only a fraction of a Monge graph's n^2 edge costs need be examined in computing shortest paths. This property allows Monge-graph shortest-path algorithms to run significantly faster than the corresponding algorithms for general graphs. In particular, Wilber [17] has shown that a shortest path from vertex 1 to vertex n of a Monge graph can be computed in $O(n)$ time, and Eppstein [6], Klawe [12], Galil and K. Park [8], and Larmore and Schieber [15] have extended Wilber's result to more general settings. (This unrestricted version of the Monge-array 1-to-n shortest-path problem is also known as the *concave least-weight-subsequence problem.*) Furthermore, Aggarwal, Klawe, Moran, Shor, and Wilber [2] have shown that a shortest d-edge 1-to-n path can be computed in $O(dn)$ time.

The main result of this paper is a new algorithm for the Monge-graph d-edge shortest-path problem. If $0 \leq c[i,j] \leq U$ and $c[i,j+1] + c[i+1,j] - c[i,j] - c[i+1,j+1] \geq L > 0$ for all i and j, then our algorithm runs in $O(n(1+\lg(U/L)))$ time. (Our algorithm can also be used when $c[i,j+1]+c[i+1,j]-c[i,j]-c[i+1,j+1] = 0$ for some i and j, but its running time in this case is harder to state; see Section 2 for more information.) Thus, when $d \gg 1+\lg(U/L)$, our algorithm represents a significant improvement over Aggarwal et al.'s $O(dn)$-time algorithm. For example, if the edge costs are nonnegative integers bounded by some polynomial in n and $d = \omega(\lg n)$, then our algorithm is asymptotically faster.

We also present several applications of our algorithm. These applications include length-limited Huffman coding, finding the maximum-perimeter d-gon inscribed in a given convex n-gon, and a digital-signal-compression problem. Though none of these applications is new — they were developed by Larmore and Przytycka [14], Aggarwal et al. [2], and Wu [18], respectively — we must examine each rather closely in order to show that our algorithm's running time is (weakly) polynomial in the size of the application's input.

The remainder of this paper is organized as follows. In Section 2, we describe our new algorithm for computing a shortest d-edge 1-to-n path in a Monge graph. Then, in Section 3, we present the three applications of our algorithm. Finally, in Section 4, we conclude with a few open problems.

2 The Algorithm

In this section, we present our new algorithm for computing the shortest k-edge path from vertex 1 to vertex n of a Monge graph. As we mentioned in the introduction, the key observation behind this algorithm is the following: for every k in the range $1 \leq k < n$, there exists a constant λ_k such

that some unrestricted 1-to-n shortest path in the complete directed graph assigning cost $c[i,j]+\lambda_k$ to the edge (i,j) is a k-edge 1-to-n shortest path in the original graph.

To see why this claim is true, we first need a few definitions. Given any path P, let $c(P)$ denote its cost, and for $1 \leq k < n$, let P_k^* denote a shortest k-edge 1-to-n path. Furthermore, for $1 < k < n$, let $\alpha_k = c(P_{k-1}^*) - c(P_k^*)$. (For notational convenience, we also define $\alpha_1 = +\infty$ and $\alpha_n = -\infty$.) Finally, let $G(\lambda)$ denote the complete directed graph assigning cost $c[i,j]+\lambda$ to the edge (i,j).

Now consider the following series of lemmas.

Lemma 2.1 For any r, s, and t satisfying $1 \leq s < r < t \leq n$, let P_s and P_t denote paths from vertex 1 to vertex n of a Monge graph containing s and t edges, respectively. Then there exist 1-to-n paths P_r and P_{s+t-r} containing r and $s+t-r$ edges, respectively, such that

$$c(P_r) + c(P_{s+t-r}) \leq c(P_s) + c(P_t) .$$

Proof Let $(i_0,i_1),(i_1,i_2),\ldots,(i_{s-1},i_s)$ and $(j_0,j_1),(j_1,j_2),\ldots,(j_{t-1},j_t)$ denote the edges of P_s and P_t, respectively, where by definition, $i_0 = j_0 = 1$ and $i_s = j_t = n$. By a simple pigeon-hole argument, there must exist an x in the range $0 \leq x < s$ such that $i_x < j_{x+r-s} < j_{x+r-s+1} \leq i_{x+1}$, since $1 = i_0 < j_{r-s}$ and $n = i_s > j_r$. Let P_r denote the r-edge 1-to-n path with edges $(j_0,j_1),\ldots,(j_{x+r-s},i_{x+1}),\ldots,(i_{s-1},i_s)$, and let P_{s+t-r} denote the $(s+t-r)$-edge 1-to-n path with edges $(i_0,i_1),\ldots,(i_x,j_{x+r-s+1}),\ldots,(j_{t-1},j_t)$. Since the Mongité of C implies

$$c[i_x,j_{x+r-s+1}] + c[j_{x+r-s},i_{x+1}] \leq c[i_x,i_{x+1}] + c[j_{x+r-s},j_{x+r-s+1}] ,$$

we must have

$$c(P_r) + c(P_{s+t-r}) \leq c(P_s) + c(P_t) .$$

■

Lemma 2.2 The α_k are nonincreasing in k, i.e., $\alpha_2 \geq \alpha_3 \geq \cdots \geq \alpha_{n-1}$.

Proof Suppose $\alpha_k < \alpha_{k+1}$ for some k in the range $2 \leq k < n-1$. This assumption implies $c(P_{k-1}^*) - c(P_k^*) < c(P_k^*) - c(P_{k+1}^*)$ or $c(P_{k-1}^*) + c(P_{k+1}^*) < 2c(P_k^*)$. However, by Lemma 2.1, there exist two k-edge 1-to-n paths P_k' and P_k'' such that $c(P_k') + c(P_k'') \leq c(P_{k-1}^*) + c(P_{k+1}^*)$. Thus, at least one of P_k' and P_k'' is strictly shorter than P_k^*, which contradicts our assumption that P_k^* is a shortest k-edge 1-to-n path. ■

Lemma 2.3 For any k in the range $2 \leq k \leq n-1$,

1. $\lambda > \alpha_k$ implies every shortest unrestricted 1-to-n path in $G(\lambda)$ has at most $k-1$ edges,
2. $\lambda < \alpha_k$ implies every shortest unrestricted 1-to-n path in $G(\lambda)$ has at least k edges, and
3. $\lambda = \alpha_k$ implies at least one shortest unrestricted 1-to-n path in $G(\lambda)$ has k edges and at least one such path has $k-1$ edges.

Proof To prove the first part of the lemma, suppose some unrestricted shortest path P in $G(\lambda)$ uses $\ell \geq k$ edges. Since the cost of an ℓ-edge path in $G(\lambda)$ is always exactly $\ell\lambda$ more than its cost in the original graph G, P must be a shortest ℓ-edge path in G, i.e., we must have $c(P) = c(P_\ell^*)$. Furthermore, since P is an unrestricted shortest path in $G(\lambda)$, its cost in $G(\lambda)$ must be at most

the cost of P^*_{k-1} in $G(\lambda)$, i.e., we must have $c(P^*_\ell) + \ell\lambda \le c(P^*_{k-1}) + (k-1)\lambda$ or $(\ell - k + 1)\lambda \le c(P^*_{k-1}) - c(P^*_\ell)$. Finally, by Lemma 2.2, we must have

$$\begin{aligned} c(P^*_{k-1}) - c(P^*_\ell) &= \sum_{r=k}^{\ell} \alpha_r \\ &\le (\ell - k + 1)\alpha_k \ . \end{aligned}$$

Combining these last two inequalities, we find $\lambda \le \alpha_k$.

The proof for the second part of the lemma is similar: if we assume some shortest path uses $\ell < k$ edges, we find $c(P^*_\ell) - c(P^*_k) \le (k - \ell)\lambda$, which implies $\lambda \ge \alpha_k$.

For the final part of the lemma, first observe that P^*_k and P^*_{k-1} must have the same cost in $G(\lambda)$, since $\lambda = c(P^*_{k-1}) - c(P^*_k)$. Furthermore, if we assume $c(P^*_\ell) + \ell\lambda < c(P^*_{k-1}) + (k-1)\lambda$ for some $\ell > k$, we obtain the contradiction $\lambda < \alpha_k$, and if we assume $c(P^*_\ell) + \ell\lambda < c(P^*_k) + k\lambda$ for some $\ell < k - 1$, we obtain the contradiction $\lambda > \alpha_k$. ■

If $\alpha_d > \alpha_{d+1}$, then Lemmas 2.2 and 2.3 suggest a natural algorithm for the d-edge shortest-path problem. The algorithm locates a λ in the range $\alpha_{d+1} < \lambda < \alpha_d$ by performing a binary search on an interval containing $[\alpha_{n-1}, \ldots, \alpha_2]$. To test a particular λ, we use Wilber's $O(n)$-time algorithm to obtain a shortest unrestricted 1-to-n path P in $G(\lambda)$. If P uses exactly d edges, then P must be a shortest d-edge path in the original graph G, i.e., we are done. If P uses fewer than d edges, then by Lemma 2.3, we know that $\lambda \ge \alpha_d$. Finally, if P uses more than d edges, then $\lambda \le \alpha_{d+1}$.

To bound the running time of this algorithm, we must first specify the interval to be searched. If we assume $0 \le c[i,j] \le U$ for all i and j, then clearly $c(P^*_1) \le U$, $c(P^*_2) \ge 0$, and $c(P^*_{n-1}) - c(P^*_{n-2}) \le U$, which implies $[\alpha_{n-1}, \ldots, \alpha_2] \subseteq [-U, \ldots, U]$. Thus, if we perform the binary search on this last interval, our algorithm runs in $O(n(1 + \lg(U/(\alpha_d - \alpha_{d+1}))))$ time.

Note that if we assume $c[i, j+1] + c[i+1, j] - c[i,j] - c[i+1, j+1] \ge L > 0$ for all i and j satisfying $2 \le i + 1 < j \le n - 1$, then $\alpha_k - \alpha_{k+1} \ge L$ for all k in the range $2 \le k \le n - 2$, since the proof of Lemma 2.1 actually shows $c(P_s) + c(P_t) - c(P_r) - c(P_{s+t-r}) \ge L$. Thus, if such an L exists, the running time of our algorithm is $O(n(1 + \lg(U/L)))$.

Now suppose $\alpha_d = \alpha_{d+1}$. To handle this case, we need a lower bound L' on the minimum *nonzero* value of $\alpha_k - \alpha_{k+1}$ over all k in range $2 \le k \le n - 1$. Given such a bound, we can solve the d-edge shortest-path problem in $O(n(1 + \lg(U/L')))$ time. Our approach is quite similar to that used for the $\alpha_d > \alpha_{d-1}$ case. We again perform a binary search on the interval $[-U, \ldots, U]$, but here we stop the binary search once we find a λ such that

1. the shortest 1-to-n path P^*_s that we find in the graph $G(\lambda)$ contains $s \ge d$ edges, and
2. the shortest 1-to-n path P^*_t that we find in the graph $G(\lambda + L'/2)$ contains $t \le d$ edges.

If $s = d$ or $t = d$, we are done. Otherwise, note that Lemma 2.3 implies $\lambda \le \alpha_s$ and $\lambda + L'/2 \ge \alpha_{t+1}$. By the definition of L', these inequalities imply $\alpha_{t+1} = \alpha_{t+2} = \cdots = \alpha_s = \lambda'$ for some λ' in the range $\lambda \le \lambda' \le \lambda + L'/2$. Furthermore, $c(P^*_k) + k\lambda'$ must be the same for all k in the range $t + 1 \le k \le s$. In particular, we must have

$$c(P^*_s) + c(P^*_t) = c(P^*_d) + c(P^*_{s+t-d}) \ . \tag{1}$$

With this equality in mind, recall the construction used in the proof of Lemma 2.1. It produces paths P_d and P_{s+t-d} containing d and $s + t - d$ edges, respectively, such that $c(P_d) + c(P_{s+t-d}) \le c(P^*_s) + c(P^*_t)$. By definition, we must have $c(P^*_d) \le c(P_d)$ and $c(P^*_{s+t-d}) \le c(P_{s+t-d})$. Thus, (1)

implies $c(P_d) = c(P_d^*)$ and $c(P_{s+t-d}) = c(P_{s+t-d}^*)$. In other words, P_d is a shortest d-edge 1-to-n path. Since the construction of Lemma 2.1 is easily converted into an $O(n)$-time algorithm for computing P_d, we have our $O(n(1 + \lg(U/L')))$-time algorithm for the $\alpha_d = \alpha_{d+1}$ case.

3 Applications

In this section, we present three applications of our new algorithm for computing the shortest d-edge 1-to-n shortest path in a Monge graph.

3.1 Length-Limited Huffman Codes

Our first application is a restricted version of the Huffman-coding problem. Given an alphabet Σ of size n and a frequency ϕ_i for each symbol $a_i \in \Sigma$, the usual Huffman-coding problem is that of constructing a prefix-free code for Σ minimizing the expected length of an encoded symbol. In his classic paper [10], Huffman showed that this problem can be solved in $O(n \lg n)$ time (or $O(n)$ time, if the ϕ_i are sorted) using a simple greedy algorithm.

This subsection focuses on the length-limited version of Huffman's problem. For this version, we add the restriction that no symbol's code may have length greater than L, where L is an input to the problem. The best algorithm previously known for this problem, due to Larmore and Hirschberg [13], run in $O(nL)$ time.

In order to apply our shortest-path algorithm to the length-limited Huffman-coding problem, we require a very nice reduction developed by Larmore and Przytycka [14]. They showed that the unrestricted Huffman-coding problem is equivalent to the unrestricted 1-to-n shortest-path problem for the Monge graph whose edge-cost array $C = \{c[i,j]\}$ is given by

$$c[i,j] = \begin{cases} \sum_{r=1}^{2j-i} \phi_r & \text{if } i < j \leq (n+i)/2, \\ +\infty & \text{otherwise,} \end{cases}$$

where we assume the ϕ_i have been sorted so that $\phi_1 \leq \phi_2 \leq \cdots \leq \phi_n$. (This assumption insures C is Monge.) Furthermore, a closer examination of their reduction reveals that the depth of the coding tree corresponding to a particular path is precisely the number of edges in this path. Thus, the length-limited Huffman-coding problem is equivalent to the L-edge 1-to-n shortest-path problem for the Monge graph defined above.

Combining Larmore and Przytycka's reduction with Wilber's $O(n)$-time algorithm for the unrestricted Monge-graph shortest-path problem yields an alternate $O(n)$-time algorithm for the unrestricted Huffman-coding problem (provided, of course, that we assume the frequencies ϕ_i are given in sorted order). Similarly, we can use Aggarwal et al.'s algorithm for the d-edge Monge-graph shortest-path problem to obtain an alternate $O(nL)$-time algorithm for the length-limited Huffman-coding problem. Furthermore, if we instead use our algorithm for the d-edge Monge-graph shortest-path problem, we obtain an algorithm for length-limited Huffman coding that is potentially even faster; assuming for simplicity that the ϕ_i are integers from the interval $[1, \ldots, \Phi]$, our algorithm solves the problem in $O(n(1 + \lg \Phi))$ time, since the minimum non-zero difference between the lengths of any two paths in the Monge graph defined above is 1.

3.2 Inscribing and Circumscribing Convex Polygons

One of the original applications of Aggarwal et al.'s $O(dn)$-time algorithm for the Monge-graph d-edge shortest-path problem was the *maximum-perimeter-inscribed-d-gon problem* from computa-

tional geometry: given a convex polygon P in the plane with vertices $p_1, \ldots, p_n$ in clockwise order, find the maximum-perimeter d-vertex convex polygon Q.

It is not hard to see that the vertices of a maximum-perimeter polygon Q must be vertices of P. To select d vertices of P defining a maximum-perimeter Q, Aggarwal et al. used a two-phase approach (first suggested by Boyce, Dobkin, Drysdale, and Guibas [5]). In the first phase, they find a d-gon Q_1 that contains the vertex p_1 and whose perimeter is maximal among all d-gons containing p_1. Then, in the second phase, they use Q_1 to locate a d-gon Q whose perimeter is maximal among all inscribed d-gons.

The problem of computing Q_1 is an instance of the Monge-graph d-edge shortest path problem. To see why, consider the $(n+1) \times (n+1)$ edge-cost array $C = \{c[i,j]\}$ given by

$$c[i,j] = \begin{cases} \mathrm{d}(p_i, p_{((j-1) \bmod n)+1} & \text{if } i < j, \\ +\infty & \text{otherwise.} \end{cases}$$

A d-edge path from vertex 1 to vertex $n+1$ of the graph defined by C clearly corresponds to a convex d-gon contain p_1 along with $d-1$ other vertices of P, and the path's cost is just the d-gon's perimeter. Moreover, as Aggarwal et al. observed, C is Monge. (Roughly speaking, this claim follows from the quadrangle inequality, which says that for any convex quadrilateral in the plane, the sum of the lengths of the quadrilateral's diagonals is at least the sum of the lengths of any pair of opposite sides.) Since any $c[i,j]$ can be computed in constant time, these observations imply that Q_1 can be computed in $O(dn)$ time using Aggarwal et al.'s d-edge shortest-path algorithm.

Once a suitable Q_1 has been located, Aggarwal et al. showed that a globally optimal inscribed d-gon Q can be computed in $O(n \lg n)$ additional time. Thus, Aggarwal et al.'s $O(dn)$-time algorithm for the Monge-graph d-edge shortest-path problem yields an $O(dn + n \lg n)$-time algorithm for the maximum-perimeter-inscribed-d-gon problem.

Given the preceding discussion, it should be clear that we can also use our algorithm to solve the maximum-perimeter-inscribed-d-gon problem. However, in order to bound our algorithm's running time in terms of the problem's input size, we need the following lemma. (We will assume for simplicity that each vertex p_i in P is given by its Cartesian coordinates x_i and y_i and that both x_i and y_i are integers from the interval $[1, \ldots, D]$. We will also assume no three vertices of P are collinear.)

Lemma 3.1 For $1 \leq i+1 < j \leq n-1$, $c[i,j+1] + c[i+1,j] - c[i,j] - c[i+1,j+1]$ is at least $1/D^\gamma$, for some constant γ.

Proof Omitted. ■

This lemma implies that our algorithm for the Monge-graph d-edge shortest-path problem can be used to obtain an $O(n(1 + \lg nD))$-time algorithm for the maximum-perimeter-inscribed-d-gon problem.

As a final note, we remark that the maximum-perimeter-inscribed-d-gon problem is closely related to another problem from computational geometry, the minimum-area-circumscribing-d-gon problem. This problem may be defined as follows: given a convex polygon P in the plane with vertices $p_1, \ldots, p_n$ in clockwise order, find the minimum-area d-vertex convex polygon Q containing P. Aggarwal and J. Park [3] showed that this problem can be solved in $O(dn + n \lg n)$ time using Aggarwal et al.'s algorithm for the d-edge shortest-path problem. In fact, it is not hard to show that our d-edge shortest-path algorithm may be substituted for Aggarwal et al.'s algorithm to obtain an $O(n(1 + \lg nD))$-time algorithm for the minimum-area-circumscribing-d-gon problem, where the polygon's vertices are assumed to have integer coordinates and D is the maximum coordinate.

3.3 d:n Quantization

Consider the problem of converting a digital signal of n discrete levels into another digital signal of $d < n$ levels. Of course such a signal compression necessarily results in the loss of information, and a natural question to ask is how to minimize the error introduced by the compression. Wu [18] formalized this problem (which he called the *optimal d:n quantization problem*) as follows: given an increasing sequence of n levels $x_1, \ldots, x_n$, together with n weights $w_1, \ldots, w_n$, find an increasing sequence of d levels $y_1, \ldots, y_d$ and a mapping $f : \{1, \ldots, n\} \to \{1, \ldots, d\}$ such that the weighted sum

$$\sum_{i=1}^{n} w_i (x_i - y_{f(i)})^2$$

is minimized.

Wu argued that the minimum-error mapping f must be a partition of the x_i into d nonoverlapping intervals and that the minimum-error y_k for a particular interval $(i, j]$ of the original signal's levels must be the weighted average of x_{i+1} through x_j. Thus, Wu's $d : n$ quantization problem is just an instance of the d-edge shortest-path problem with the $(n+1) \times (n+1)$ edge-cost array $C = \{c[i, j]\}$ given by

$$c[i,j] = \begin{cases} \displaystyle\sum_{r=i+1}^{j} w_r (x_r - \mu(i,j))^2 & \text{if } 0 \le i < j \le n, \\ +\infty & \text{otherwise,} \end{cases}$$

where

$$\mu(i,j) = \frac{\displaystyle\sum_{r=i+1}^{j} w_r x_r}{\displaystyle\sum_{r=i+1}^{j} w_r}$$

Wu also showed that (1) C is Monge, and (2) given $O(n)$-time preprocessing, any entry of C can be computed in constant time. Thus, Aggarwal et al.'s $O(dn)$-time algorithm for the d-edge shortest-path problem yields an $O(dn)$-time algorithm for Wu's d:n quantization problem.

Of course, our algorithm can also be used to solve the d:n quantization problem. To bound its running time, we recall that Wu showed

$$\begin{aligned} & c[i, j+1] + c[i+1, j] - c[i, j] - c[i+1, j+1] \\ &= (\mu(i+1, j+1) - \mu(i, j+1))(2x_{j+1} - \mu(i+1, j+1) - \mu(i, j+1))w_{j+1} \end{aligned}$$

for $1 \le i+1 < j \le n-1$. Thus, if we assume for simplicity that the x_i are distinct integers and the w_i are positive integers, the minimum value attained by $c[i, j+1] + c[i+1, j] - c[i, j] - c[i+1, j+1]$ must be at least

$$\left(\frac{\displaystyle\sum_{r=i+2}^{j+1} w_r x_r}{\displaystyle\sum_{r=i+2}^{j+1} w_r} - \frac{w_{i+1}x_{i+1} + \displaystyle\sum_{r=i+2}^{j+1} w_r x_r}{w_{i+1} + \displaystyle\sum_{r=i+2}^{j+1} w_r} \right) \left(2x_{n+1} - \frac{w_{i+1}x_{i+1} + \displaystyle\sum_{r=i+2}^{j+1} w_r x_r}{w_{i+1} + \displaystyle\sum_{r=i+2}^{j+1} w_r} - \frac{\displaystyle\sum_{r=i+2}^{j+1} w_r x_r}{\displaystyle\sum_{r=i+2}^{j+1} w_r} \right) w_{j+1}$$

$$\geq \left(\frac{w_{i+1}}{w_{i+1} + \sum_{r=i+2}^{j+1} w_r} \right) \left(\frac{2 \sum_{r=i+2}^{j} w_r}{\sum_{r=i+2}^{j+1} w_r} \right) w_{j+1}$$

$$= \Omega\left(\frac{1}{nW}\right) ,$$

where $W = \max_r w_r$. Furthermore, no $c[i,j]$ can be larger than WX^2, where $X = \max_r x_r$. These bounds together imply that our algorithm solves Wu's $d : n$ quantization problem in $O(n(1 + \lg(nWX)))$ time.

4 Concluding Remarks

We conclude with the following open questions:

1. What is the true time complexity of the Monge-graph d-edge shortest-path problem? In particular, is it possible to obtain an algorithm whose running time is both $o(dn)$ and strongly polynomial (i.e., independent of the sizes of the edge costs)?

2. Is it possible to extend the techniques developed in this paper and obtain improved algorithms for other related problems, such as the minimum-cost matching problem for a bipartite Monge graph and the Hamiltonian-path problem for a convex polygon? For the former problem, we are given a complete bipartite graph $G = (A \cup B, A \times B)$, where $A = \{u_1, \ldots, u_m\}$, $B = \{v_1, \ldots, v_n\}$, and $m \leq n$, together with an $m \times n$ Monge edge-cost array $C = \{c[i,j]\}$ assigning cost $c[i,j]$ to edge (u_i, v_j), and we want to find a minimum-cost maximum (i.e., cardinality m) matching. (If $m = n$, this problem is trivial, and for $m < n$, it can be solved in $O(m(n-m))$ time using a straightforward dynamic-programming algorithm. Furthermore, Aggarwal, Bar-Noy, Khuller, Kravets, and Schieber [1] have very recently obtained faster algorithms for several interesting special cases.) For the latter problem, we are given a convex polygon P in the plane with vertices $p_1, \ldots, p_n$ in clockwise order, together with two distinguished vertices p_i and p_j, and we want to find a minimum-cost path from p_i to p_j that visits every vertex of P exactly once. (The best algorithm currently known for this problem takes $O(m(n-m))$ time, where m is the number of vertices between p_i and p_j in the clockwise ordering of P's vertices.) Each of these problems can be formulated as a generalized Monge-graph d-edge shortest-path problem where the cost of traversing an edge may vary with its location in the path. Specifically, the problem has a three-dimensional Monge edge-cost array $C = \{c[i,j,k]\}$, where $c[i,j,k]$ is the cost of traversing the edge (i,j) as the kth edge in a path.

Acknowledgements

The third author would like to thank David Greenberg and Bruce Hendrickson (Sandia National Laboratories) for several helpful technical discussions.

References

[1] A. Aggarwal, A. Bar-Noy, S. Khuller, D. Kravets, and B. Schieber. Efficient minimum cost matching using quadrangle inequality. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, 1992. To appear.

[2] A. Aggarwal, M. M. Klawe, S. Moran, P. W. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(2):195–208, 1987.

[3] A. Aggarwal and J. K. Park. Sequential searching in multidimensional monotone arrays. Research Report RC 15128, IBM T. J. Watson Research Center, Yorktown Heights, NY, November 1989. Submitted to *Journal of Algorithms*. Portions of this paper appear in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 497–512, 1988.

[4] R. K. Ahuja, K. Melhorn, J. B. Orlin, and R. E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2):213–223, 1990.

[5] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, and L. J. Guibas. Finding extremal polygons. *SIAM Journal on Computing*, 14(1):134–147, 1985.

[6] D. Eppstein. Sequence comparison with mixed convex and concave costs. *Journal of Algorithms*, 11(1):85–101, 1990.

[7] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987.

[8] Z. Galil and K. Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Information Processing Letters*, 33(6):309–311, 1990.

[9] A. J. Hoffman. On simple linear programming problems. In V. Klee, editor, *Convexity: Proceedings of the Seventh Symposium in Pure Mathematics of the AMS*, volume 7 of *Proceedings of Symposia in Pure Mathematics*, pages 317–327. American Mathematical Society, Providence, RI, 1963.

[10] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[11] D. B. Johnson. A priority queue in which initialization and queue operations take $O(\log \log D)$ time. *Mathematical Systems Theory*, 15(4):295–309, 1982.

[12] M. M. Klawe. A simple linear time algorithm for concave one-dimensional dynamic programming. Technical Report 89-16, Department of Computer Science, University of British Columbia, Vancouver, Canada, 1989.

[13] L. L. Larmore and D. S. Hirschberg. A fast algorithm for optimal length-limited Huffman codes. *Journal of the ACM*, 37(3):464–473, 1990.

[14] L. L. Larmore and T. M. Przytycka. Parallel construction of trees with optimal weighted path length. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 71–80, 1991.

[15] L. L. Larmore and B. Schieber. On-line dynamic programming with applications to the prediction of RNA secondary structure. *Journal of Algorithms*, 12(3):490–515, 1991.

[16] J. K. Park. A special case of the n-vertex traveling-salesman problem that can be solved in $O(n)$ time. *Information Processing Letters*, 40(5):247–254, 1991.

[17] R. Wilber. The concave least-weight subsequence problem revisited. *Journal of Algorithms*, 9(3):418–425, 1988.

[18] X. Wu. Optimal quantization by matrix searching. *Journal of Algorithms*, 12(4):663–673, 1991.

END

DATE
FILMED
6/21/93