An efficient algorithm for on-line searching of minima in Monge path-decomposable tridimensional arrays *

Alfredo García *, Pedro Jodrá 1, Javier Tejel 2

Dpto. Métodos Estadísticos, Facultad de Ciencias, Universidad de Zaragoza, 50009 Zaragoza, Spain

Received 5 March 1998; received in revised form 10 August 1998

Abstract

We consider the problem of computing the recurrence \( E[i] = \min_{j=1 \ldots m} \min_{k:1 \leq k \leq i} \{ b(i, j) + c(j, k) + E[k - 1] \} \), for \( i = 1, \ldots, n \), where \( E[0] \) is known and \( B = \{ b(i, j) \} \) and \( C = \{ c(j, k) \} \) are known weight Monge matrices of size \( n \times m \) and \( m \times n \), respectively. We provide an \( \Theta(m + n) \)-algorithm for calculating the \( E[i] \) values. This algorithm allows us to linearly solve the two following problems: Finding the minimum Hamiltonian curve from point \( p_1 \) to point \( p_m \) for \( N \) points on a convex polygon, and solving the traveling salesman problem for \( N \) points on a convex polygon and a segment line inside it, improving the previous \( \Theta(N \log N) \)-algorithms for both these problems. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Monge matrices; Dynamic programming; Computational geometry; Computational complexity; Traveling salesman problem

1. Introduction

Let \( W = \{ w(i, k) \} \) be an \( n \times n' \) weight matrix, where each \( w(i, k) \) can be calculated in constant time. Given an integer constant \( c_1 \geq 1 \) and the values \( F[k] \), for \( k = 1, \ldots, c_1 \), the so-called one-dimensional dynamic programming problem consists in solving the on-line recurrence:

\[
E[i] = \min_{1 \leq k \leq c_1} \{ w(i, k) + F[k] \}, \quad i = 1, \ldots, n,
\]  

where \( c_i \) and \( F[k] \), for \( k = c_{i-1} + 1, \ldots, c_i \), can be computed from \( E[i - 1] \) in constant time, and the integer constants \( c_1, \ldots, c_n \) verify \( 1 \leq c_1 \leq \cdots \leq c_n \leq n' \).

Problems of this type, usually with the weight matrix verifying some additional property, arise in many fields: biology [8], economics [3], operation research [9,7], computational geometry [1], etc.

Solving (1) is equivalent to the problem of on-line searching of the minimum of each row of the partial matrix \( A \) with entries \( \{ a(i, k) = w(i, k) + F[k] \} \), defined when \( k \leq c_i \). A partial matrix of this shape is called a generalized lower triangular matrix, and it is concave totally monotone if \( a(i, k) \geq a(i', k') \) implies \( a(i', k) \geq a(i', k') \), for \( 1 \leq i < i' \leq n, \ 1 \leq k < k' \leq n' \), when these four entries are defined. If we denote by \( k(i) \) the smallest column index where the minimum of row \( i \) is found, the main property of concave totally monotone matrices is that \( k(1) \leq k(2) \leq \cdots \leq k(n) \) (monotonicity), and this property is also verified by the minima of any submatrix (total...
monotonicity). For concave totally monotone matrices there are several \( O(n + n') \)-algorithms which solve the on-line row minima search problem. We will use the one by Larmore and Schieber, described in [8], and we will call it the LARSCH algorithm.

The most frequent case of concave totally monotone matrices is that of Monge matrices. A full \( n \times n' \) matrix

\[
A
\]

is Monge if

\[
a(i, j) + a(i + 1, j + 1) < a(i, j + 1) + a(i + 1, j)
\]

for \( 1 \leq i < n \) and \( 1 \leq j < n' \).

For a review on Monge properties and applications see [4]. The following properties of Monge matrices can be easily checked:

(i) the transpose of a Monge matrix is Monge,
(ii) the sum of two Monge matrices is a Monge matrix,
(iii) if \( A \) with elements \( \{a(i, j)\} \) is Monge, \( u \in \mathbb{R}^n \) and \( v \in \mathbb{R}^{n'} \), also \( B \) with elements \( \{b(i, j) = a(i, j) + u_i + v_j\} \) is Monge, and
(iv) any generalized lower triangular submatrix of a Monge matrix is concave totally monotone.

The main result in this paper is a linear time algorithm solving the extension of problem (1) when the weight matrix is given by

\[
W(i, k) = \sum_{j=1}^{m} b(i, j) + c(j, k) + E[k - 1],
\]

where \( B = \{b(i, j)\} \) and \( C = \{c(j, k)\} \) are known Monge matrices of size \( n \times m \) and \( m \times n' \), respectively. In the applications described later, \( c_i \) will be \( i \), \( n' \) will be \( n \) and \( F[k] \) will be \( E[k - 1] \). For this reason, we explain the algorithm only for this particular case, although it can be modified for solving the more general formulation. Then, the problem is that of calculating

\[
w(i, k) = \min_{j=1}^{m} \{b(i, j) + c(j, k)\},
\]

where \( B = \{b(i, j)\} \) and \( C = \{c(j, k)\} \) are known Monge matrices of size \( n \times m \) and \( m \times n' \), respectively. In the applications described later, \( c_i \) will be \( i \), \( n' \) will be \( n \) and \( F[k] \) will be \( E[k - 1] \). For this reason, we explain the algorithm only for this particular case, although it can be modified for solving the more general formulation. Then, the problem is that of calculating

\[
E[i] = \min_{j=1}^{m} \min_{k=1}^{i} \{b(i, j) + c(j, k) + E[k - 1]\},
\]

where \( E[0] \) is given. In [2], a tridimensional array that can be expressed as \( b(i, j) + c(j, k) \) is called a Monge path-decomposable tridimensional array. So, our problem is the on-line calculating of the minima of a partial (because \( k \leq i \)) Monge path-decomposable tridimensional array.

### 2. On-line algorithm

In order to solve (2), we use mainly the \( m \times n \) Monge matrix

\[
\overline{C} = \{\overline{c}(j, k) = c(j, k) + E[k - 1]\}.
\]

Given \( i \), let \( \overline{C}^i \) be the \( m \times i \) submatrix formed by the first \( i \) columns of \( \overline{C} \) and let \( A^i \) be the matrix obtained by adding \( b(i, j) \) to each row \( j \) of \( \overline{C}^i \). \( E[i] \) is the global minimum of \( A^i \) and we denote by \( J(i) \) and \( K(i) \) the row and column of \( A^i \) where this minimum is, i.e.,

\[
E[i] = a^i(J(i), K(i)) = b(i, J(i)) + \overline{c}(J(i), K(i))
\]

for each \( i \), we want to calculate \( (J(i), K(i)) \) and the key idea of our algorithm is that only \( O(m + n) \) candidate positions \((j, k)\) need to be considered for calculating \( (J(i), K(i)) \), \( \forall i \). In addition, all the matrices involved in the algorithm are concave totally monotone.

Let \( k^i(j), j = 1, \ldots, m \), be the column with the minimum of row \( j \) of \( A^i \), and let \( j^i(k), k = 1, \ldots, i \), be the row with the minimum of column \( k \) of \( A^i \). Notice that \( k^i(j) \) is also the position of the minimum of the row \( j \) of \( \overline{C}^i \) because row \( j \) of \( A^i \) is obtained from the same row of \( \overline{C}^i \) by adding the constant \( b(i, j) \). When a row has more than one minimum, we will always take the one in the leftmost column. Similarly for columns, we will take the one in the topmost row and as a global minimum, we will take the one in the leftmost column and topmost row. The following lemma gives some additional properties about these row and column minima.

#### Lemma 1.

1. \( k^i(j) \leq k^i(j + 1), j = 1, \ldots, m - 1 \).
2. \( j^i(k) \leq j^i(k + 1), k = 1, \ldots, i - 1 \).
3. \( k^i(j) \leq k^{i+1}(j), i = 1, \ldots, n - 1 \).
4. \( j^i(k) \leq j^{i+1}(k), i = k, \ldots, n - 1 \).

#### Proof.

Given \( i \), \( A^i \) and its transpose are Monge matrices, hence (1) and (2) hold by monotonicity. On the other hand, as \( \overline{C}^{i+1} \) is obtained from \( \overline{C}^i \) by adding column \( i + 1 \) of \( \overline{C} \), \( k^{i+1}(j) \) is either \( k^i(j) \) or \( i + 1 \), hence (3). Finally, given \( k \leq i \), column \( k \) of \( A^{i+1} \) is obtained from column \( k \) of \( A^i \) by adding \( b(i + 1, j) - b(i, j) \) to each entry. As \( B \) is Monge, \( b(i + 1, j + 1) - b(i, j + 1) \leq b(i + 1, j) - b(i, j) \), for \( j = 1, \ldots, m - 1 \), so, what is added is a decreasing
amount in each row. Therefore, the minimum of the column $k$ of $A^{i+1}$ has to be in or after $j^i(k)$. □

The global minimum of $A^i$ is always the minimum in its row and in its column, so we have $K(i) = k^i(j)$ for an index $j \leq m$, and $J(i) = j^i(k)$ for an index $k \leq i$. Hence, by previous lemma $J(i) \leq j^i(i)$. These values $j^i(i)$, $i = 1, \ldots, n$, can be precalculated as follows: let $D$ be the $n \times m$ Monge matrix defined as $d(i, j) = b(i, j) + c(j, i)$ and let $d(i)$ be the column where row $i$ of $D$ has its minimum. Column $i$ of $A^i$ is obtained by adding the constant $E[i - 1]$ to each element of row $i$ of $D$, so $d(i) = j^i(i)$.

**Lemma 2.** $K(i) \leq i$ and $d(K(i)) \leq J(i) \leq d(i)$ for $i = 1, \ldots, n$.

**Proof.** Obviously, $K(i) \leq i$ because $A^i$ is an $m \times i$ matrix. In addition, $J(i) = j^i(k)$ for a $k \leq i$, hence by Lemma 1, part (2),

\[ J(i) \leq j^i(i) = d(i). \]

Similarly, as $K(i) \leq i$, using (4) of Lemma 1 we have:

\[ d(K(i)) = j^{K(i)}(K(i)) \leq j^i(K(i)) = J(i). \] □

In Fig. 1 the lemma is illustrated. Let $\overline{C}'$ be the partial matrix formed by the elements $\overline{c}(j, k)$ such that $d(k) \leq j \leq d(n)$, for $1 \leq k \leq n$. Given $i$, the previous lemma implies that the position $(J(i), K(i))$ is in $R_i$ (shaded region in Fig. 1) defined as

\[ R_i = \{(j, k) \ni d(k) \leq j \leq d(i), \ 1 \leq k \leq i\}. \]

Now, let $H$ be the partial matrix of $\overline{C}$ defined as $h(j, k) = c(j, k) + E[k - 1]$ if $d(k) < j \leq d(n)$, for $1 \leq k \leq n$ (submatrix delimited by the thick line in Fig. 1). Notice that each row $j$ of $H$, $j = d(1) + 1, \ldots, d(n)$, is defined until the column $\max\{k \ni d(k) < j\}$ and that $H$ is concave totally monotone. Let $h(j)$ be the position where the minimum of row $j$ of $H$ is achieved. Then, the following lemma shows that it suffices to consider $O(m + n)$ candidates for the positions of all the global minima.

**Lemma 3.** Either $J(i) = d(K(i))$ or $K(i) = h(J(i))$ for $i = 1, \ldots, n$.

**Proof.** Given $i$, we know that $K(i)$ is the column where the minimum of row $J(i)$ of $\overline{C}'$ is found and, by Lemma 2, that $(J(i), K(i)) \in R_i$. Then, either the minimum of row $J(i)$ is in $H$, and hence $K(i) = h(J(i))$, or $K(i) \leq i$ is a column such that $J(i) = d(K(i))$. □

In Fig. 1, the positions $(j, h(j))$ of $H$ are marked with a black dot and the positions $(d(i), i)$ with a white one. Lemmas 2 and 3 imply that, given $i$, candidate positions for containing $(J(i), K(i))$ are only those...
belonging to $R_i$ and marked with a black or a white dot.

Now, let $m'$ be the number of these positions that can contain a global minimum in any step $i$. For the moment, we assume that all of them have been calculated, i.e., we know where a black or white dot appears in $C'$. We can enumerate these positions from 1 to $m'$ beginning with the first row and then in each row, from left to right. If $(j, k)$ is the candidate position with number $p$, $1 \leq p \leq m'$, we define $\text{row}(p) = j$ and $\text{col}(p) = k$. Given $i$, $E[i]$ is achieved in one candidate position of $R_i$, so, if $l(i)$ is the number of candidate positions in $R_i$, then

$$E[i] = \min_{1 \leq p \leq l(i)} b(i, \text{row}(p)) + \bar{c}(\text{row}(p), \text{col}(p)).$$

Therefore, if we define $A'$ as the $n \times m'$ partial matrix with elements

$$a'(i, p) = b(i, \text{row}(p)) + c(\text{row}(p), \text{col}(p)) + E[\text{col}(p) - 1]$$

for $i = 1, \ldots, n$, $p = 1, \ldots, l(i)$,

then $E[i]$ is the minimum of row $i$ of $A'$.

**Lemma 4.** The $n \times m'$ matrix $A'$ is concave totally monotone.

**Proof.** The values $l(i)$ are non-decreasing, so $A'$ is a generalized lower triangular matrix. We only need to prove that $a'(i, p) + a'(i + 1, p + 1) \leq a'(i + 1, p) + a'(i, p + 1)$ when these four entries are defined, which is equivalent to proving that

$$b(i, \text{row}(p)) + b(i + 1, \text{row}(p + 1)) \leq b(i + 1, \text{row}(p)) + b(i, \text{row}(p + 1)).$$

This last inequality is true because $\text{row}(p + 1)$ is either $\text{row}(p)$ or $\text{row}(p) + 1$, and $B$ is Monge. \(\square\)

Now, we are ready to solve the initial problem (2). In step $i$, let us assume that $E[1], \ldots, E[i - 1]$ and $h(d(1) + 1), \ldots, h(d(i - 1))$ have been calculated. Then, the first $i$ columns of $\bar{C}$ and $H$ are defined. Hence, the rows $d(i - 1) + 1, \ldots, d(i)$ of $H$ are also known and, if $d(i - 1) < d(i)$, then we can calculate $h(d(i - 1) + 1), \ldots, h(d(i))$. In order to calculate $E[i]$, we need to know all the positions in $R_i$ with black and white dots and the value of $a'(i, p)$ in these positions. This obviously can be done because the first $i$ columns of $\bar{C}$ are defined and because $h(d(1) + 1)$,

**Algorithm MINIMA**

begin
activate LARSCH over each row of $D$, obtaining $d(i), i = 1, \ldots, n$. 
initialize $p = 0; d(0) = d(1)$; 
for $i$ from 1 to $n$ do 
comment The $i$ first columns of $\bar{C}$ are defined. 
if $(d(i) > d(i - 1))$ then 
comment The rows from $d(i - 1) + 1$ to $d(i)$ of $H$ are defined. 
for $j$ from $d(i - 1) + 1$ to $d(i)$ do 
activate LARSCH over $H$ for calculating $h(j)$ 
p = p + 1; row(p) = d(i); col(p) = h(j); 
end for 
end if 
$E[i] =$ \ynici,b(i, row(p)) + \yni(row(p), col(p)). 
Therefore, if we define $A'$ as the $n \times m'$ partial matrix with elements

$$a'(i, p) = b(i, \text{row}(p)) + c(\text{row}(p), \text{col}(p)) + E[\text{col}(p) - 1]$$

for $i = 1, \ldots, n$, $p = 1, \ldots, l(i)$,

then $E[i]$ is the minimum of row $i$ of $A'$.

**Lemma 5.** The complexity of algorithm MINIMA is $\Theta(m + n)$. 

**Proof.** The algorithm LARSCH is activated for calculating the minima of $D$, $H$, and $A'$. $D$ is a $n \times m$ full matrix, $H$ is a partial matrix of, at most, size $(m - 1) \times (n - 1)$ and $A'$ is an $n \times m'$ partial matrix, but $m' = n + d(n) - d(1) \leq n + m - 1$. As each entry of these matrices can be calculated in constant time, all these minima are obtained in $\Theta(m + n)$ steps. \(\square\)

3. Applications

3.1. Finding the minimum Hamiltonian curve in a convex polygon

Given $N = m + n$ points on the plane forming a convex polygon $P$, we want to find the minimum Hamiltonian curve $S$, starting at point $p_1$ and finishing at point $p_m$, where $p_1$ and $p_m$ are arbitrarily chosen. In [7], this problem is reformulated as one of dynamic
programming and an $O(N \log N)$ time and $O(N)$ space algorithm is given for solving it.

This reformulation is the following. Let $U_P = \{p_1, \ldots, p_m\}$ be the set of points on $P$ between $p_1$ and $p_m$ (both included), clockwise numbered, and $L_P = \{q_1, \ldots, q_n\}$ the set of points on $P$ between $p_1$ and $p_m$ (both excluded), counterclockwise numbered (see Fig. 3). Let $d(.,.)$ be the Euclidean distance between two points. Let $S(i)$ be the shortest Hamiltonian curve from $p_1$ to $p_m$ visiting all the points of $U_P$ and only the first $i$ points of $L_P$ and let $E[i]$ be the length of $S(i)$.

If we define

$$E[0] = \sum_{j=1}^{m-1} d(p_j, p_{j+1}),$$

then the following scheme of dynamic programming holds (see [7]):

$$E[i] = \min_{j=1, \ldots, m-1} \min_{k \leq i} \left\{ E[k-1] + d(p_j, q_k) + \sum_{l=k}^{i-1} d(q_l, q_{l+1}) + d(q_i, p_{j+1}) - d(p_j, p_{j+1}) \right\}, \quad i = 1, \ldots, n.$$  

By defining:

- $s(q_i) = \sum_{k=1}^{i} d(q_k, q_{k+1})$, for $1 \leq i \leq n$,
- $b(i, j) = s(q_i) + d(q_i, p_{j+1}) - d(p_j, p_{j+1})$, for $1 \leq i \leq n$ and $1 \leq j \leq m - 1$,
- $c(j, k) = d(p_j, q_k) - s(q_k)$, for $1 \leq j \leq m - 1$ and $1 \leq k \leq n$, then the previous scheme is equivalent to:

$$E[i] = \min_{j=1, \ldots, m-1} \min_{k \leq i} \left\{ b(i, j) + c(j, k) + E[k-1] \right\}, \quad i = 1, \ldots, n.$$  

Quantities $s(i)$, $i = 1, \ldots, n$, can be easily computed in $O(n)$ time and stored in $O(n)$ space and hence, given $i$ and $j$, we can calculate $b(i, j)$ and $c(j, i)$ in constant time. In addition, matrices $B$ and $C$ are Monge and then, the on-line algorithm of the previous section can be applied directly.

### 3.2. The convex-polygon-and-line TSP

Now, we are interested in calculating the minimum tour that visits $N = m + n$ points, when $m$ of them are on a convex polygon $P$ and the other $n$ are on a segment line $SL$ inside $P$. In [7], this problem is solved in $O((m + n) \log n)$ time and $O(n)$ space, improving the previous $O(N^2)$-algorithm for this problem described in [6]. A simpler version of this problem, for points on three parallel lines, was studied by Cutler (see [5]) in relation to the problem of connecting nets in printed circuits and he solved it in $O(N^3)$ time.

In [7], a similar reformulation to the previous one is given for solving the convex-polygon-and-line TSP. Let $CP_U$ (the upper convex polygon) and $CP_L$ (the lower convex polygon) be the set of points $\{p_0, p_1, \ldots, p_{m+1}\}$ and the set of points $\{q_0, \ldots, q_{n+1}\}$.
clockwise and counterclockwise numbered, respectively (see Fig. 4). Notice that \( p_1 \in CP_U \) is redefined as \( r_0 \in CP_L \) because sometimes it must be considered as belonging to \( CP_L \). The same happens with points \( r_1, p_m, \) and \( r_{m+1} \). \( V_{SL} = \{q_1, \ldots, q_n\} \) is the set of points on \( SL \) numbered from left to right. Let \( S(i) \) be the shortest tour that visits all the points of \( P \) and only the first \( i \) points of \( SL \) and let \( G[i] \) be its length.

Now, let \( E[i] \) and \( F[i] \), respectively, be the length of the shortest tour that visits all the points of \( P \) and only the first \( i \) points of \( SL \), with the constraint that the last zone of \( SL \), a zone \([q_k, q_{i+1}]\), is linked with two points of \( CP_U \) and with two points of \( CP_L \). Let us define

\[
G[0] = \sum_{j=0}^{m_1} d(p_j, p_{j+1}) + \sum_{j=1}^{m_2-1} d(r_j, r_{j+1})
\]

(the perimeter of \( P \)). Then, the following scheme of dynamic programming holds (see [7]):

\[
E[i] = \min_{j=0, \ldots, m_1, 1 \leq k \leq i} \left\{ G[k-1] + d(q_k, p_j) \right\} + d(q_k, q_i) + d(q_i, p_{j+1}) - d(p_j, p_{j+1})
\]

\[
i = 1, \ldots, n,
\]

\[
F[i] = \min_{j=0, \ldots, m_2, 1 \leq k \leq i} \left\{ G[k-1] + d(q_k, r_j) \right\} + d(q_k, q_i) + d(q_i, r_{j+1}) - d(r_j, r_{j+1})
\]

\[
i = 1, \ldots, n,
\]

\[
G[i] = \min \{ E[i], F[i] \}, \quad i = 1, \ldots, n,
\]

where \( d(\cdot, \cdot) \) is the Euclidean distance between two points except that we define

\[
d(q_k, p_0) = d(q_k, r_0) = \infty \quad \text{if} \ k \neq 1, \quad \text{and}
\]

\[
d(q_k, p_{m+1}) = d(q_k, r_{m+1}) = \infty \quad \text{if} \ k \neq n.
\]

If we define:

\[
\begin{align*}
& s(q_i) = d(q_1, q_i), \quad \text{for} \ 1 \leq i \leq n, \\
& b(i, j) = s(q_i) + d(q_i, p_{j+1}) - d(p_j, p_{j+1}), \quad \text{for} \ i = 1, \ldots, n \ \text{and} \ j = 0, \ldots, m_1, \\
& c(j, k) = d(q_k, p_j) - s(q_k), \quad \text{for} \ j = 0, \ldots, m_1 \ \text{and} \ k = 1, \ldots, n, \\
& b'(i, j) = s(q_i) + d(q_i, r_{j+1}) - d(r_j, r_{j+1}), \quad \text{for} \ i = 1, \ldots, n \ \text{and} \ j = 0, \ldots, m_2, \\
& c'(j, k) = d(q_k, r_j) - s(q_k), \quad \text{for} \ j = 0, \ldots, m_2 \ \text{and} \ k = 1, \ldots, n
\end{align*}
\]

then we have:

\[
E[i] = \min_{j=0, \ldots, m_1, 1 \leq k \leq i} \left\{ G[k-1] + b(i, j) + c(j, k) \right\}, \quad i = 1, \ldots, n,
\]

\[
F[i] = \min_{j=0, \ldots, m_2, 1 \leq k \leq i} \left\{ G[k-1] + b'(i, j) + c'(j, k) \right\}, \quad i = 1, \ldots, n,
\]

\[
G[i] = \min \{ E[i], F[i] \}, \quad i = 1, \ldots, n.
\]

Each element of the matrices \( B, B', C \) and \( C' \) can be calculated in constant time and these matrices are all Monge. Then, by interleaving the computation of \( E[i], F[i] \) and \( G[i] \), the algorithm of the previous section can be used to solve the convex-polygon-and-line TSP in \( \Theta(N) \) time and \( \Theta(N) \) space.
References