Computing 60, 109–119 (1998)

Computing
© Springer-Verlag 1998
Printed in Austria

Spanning Trees and Shortest Paths in Monge Graphs*

T. Dudás and **R. Rudolf**, Graz

Received December 3, 1996; revised August 1, 1997

Abstract

We investigate three problems on *Monge graphs*, i.e. complete, undirected weighted graphs whose distance matrix is a Monge matrix: (A) the minimum spanning tree problem, (B) the problem of computing all-pairs shortest paths and (C) the problem of determining a minimum weighted 1-to-all shortest path tree. For all three problems best possible algorithms (in terms of complexity) are presented.

AMS Subject Classifications: 90C27, 05C50, 05C05, 05C12.

Key words: Minimum spanning tree, Monge matrix, shortest paths.

1. Introduction

Let $G=(V,E)$ be an undirected graph with node set $V=\{1,\dots,n\}$ and edge set $E\subseteq V\times V$, $(|E|=m)$. Moreover, to each edge (i,j) in E we associate the weight (distance) c_{ij}. If additionally G is a complete graph, i.e. $m=\binom{n}{2}$, and the associated symmetric distance matrix C fulfills the property that

$$c_{ij}+c_{kl}\le c_{il}+c_{kj} \text{ for all } 1\le i<k\le n,\ 1\le j<l\le n,\ i\ne j,\ k\ne l,\ i\ne l,\ k\ne j \quad (1)$$

then we call G a *Monge graph.*

In this paper we revisit three classical graph problems under the assumption that the underlying graph G is a Monge graph. For all three problems we derive algorithms which are best possible in terms of running time complexity.

The first problem under investigation is the problem of finding a minimum spanning tree in G. It is well known that a minimum spanning tree can be found

*This research has been supported by the Spezialforschungsbereich F 003 'Optimierung und Kontrolle'/Projektbereich Diskrete Optimierung.

in $O(n^2)$ time in case that G is a complete graph (Prim [12]), and in $O(m \log \beta(m,n))$ time, if G has m edges, where $\beta(m,n) = \min\{i | \log^{(i)} n \leq m/n\}$ (Gabow, Galil, Spencer and Tarjan [8]). In this paper we present an algorithm which constructs a minimum spanning tree in a Monge graph G in $O(n)$ time.

As a second problem we revisit the problem of computing shortest paths in an undirected weighted graph G, if all weights are nonnegative. Due to Dijkstra's classical algorithm (Dijkstra [5]) a shortest path from a source to a sink can be computed in $O(n \log n + m)$ time using an implementation based on Fibonacci-Heaps given in Fredman and Tarjan [7]. Moreover, the same algorithm may be used to solve the 1-to-all shortest path problem. All-pairs shortest paths, i.e. the shortest paths from each node to each other node, can be found in $O(n^3)$ time using the Floyd-Warshall algorithm (Floyd [6]). If we assume G to be a Monge graph, then the 1-to-all shortest path tree (and therefore also the shortest path from 1 to n) may be computed in $O(n)$ time due to an algorithm by Wilber [14]. Pferschy et al. [11] have investigated the problem of computing a longest path in a bipartite Monge graph. Our contribution is an algorithm computing all-pairs shortest paths in a Monge graph in $O(n^2)$ time.

The third problem treated here is somehow a combination of a minimum spanning tree problem and the problem of computing a 1-to-all shortest path problem, namely the problem of finding a minimum weighted 1-to-all shortest path tree. That means, we are searching for a 1-to-all shortest path tree, which has minimum weight among all 1-to-all shortest path trees. In Section 5 we will show that Dijkstra's algorithm is applicable and therefore this problem may be solved in $O(n \log n + m)$ time for general undirected graphs. Khuller, Raghavachari and Young [9] have shown that the minimum weighted 1-to-all shortest path tree can be found in $O(n + m)$ time in general directed graphs with nonnegative weights, if an arbitrary 1-to-all shortest path tree is already at hand. Our main result concerning this problem is an $O(n)$ time algorithm for Monge graphs.

The paper is organized as follows: Section 2 summarizes some preliminary considerations and basic properties of Monge matrices used later on. Section 3 is dedicated to the minimum spanning tree problem, Section 4 contains the algorithm for computing all-pairs shortest paths in Monge graphs and Section 5 deals with the problem of finding a minimum weighted 1-to-all shortest path tree.

2. Basic Properties

In this section we mention the basic properties of Monge structures which we will explicitly use in the paper. For a survey concerning Monge properties and their applications in combinatorial optimization we refer to Burkard et al. [3].

First of all note that in (1) diagonal elements are not involved, meaning that loops may have an arbitrary weight. However, in the problems treated in this paper loops are never part of a feasible solution. Therefore the diagonal elements may be replaced by arbitrary values. It has been shown in Burkard [2] that in the case that C is a symmetric matrix and fulfills conditions (1), the diagonal elements can always be redefined in such a way that the new matrix is a symmetric Monge matrix, i.e. fulfills the *Monge property*, namely

$$c_{ij} + c_{kl} \leq c_{il} + c_{kj} \qquad \text{for all } 1 \leq i < k \leq n,\ 1 \leq j < l \leq n. \tag{2}$$

Moreover, this redefinition of the diagonal elements can be done in $O(n)$ time. Therefore we may assume w.l.o.g. for the remainder of the paper that the associated weight matrix of a Monge graph is a symmetric Monge matrix, i.e. the diagonal entries of C are already defined in an appropriate way.

It should also be mentioned that symmetric matrices C fulfilling (1) also arise in the context of efficiently solvable travelling salesman problems and are known in this area as *Supnick matrices* (e.g. Supnick [13], Burkard et al. [4]).

A basic property of Monge matrices is the fact that each Monge matrix is also a *totally monotone* matrix. An $n \times n$ matrix C is called totally monotone, if

$$c_{ij} > c_{is} \Rightarrow c_{rj} > c_{rs} \qquad \text{for all } 1 \leq i < r \leq n,\ 1 \leq j < s \leq n. \tag{3}$$

If we let $r(i)$ denote the column index of the leftmost minimum entry in row i of a totally monotone matrix C, $1 \leq i \leq n$, then

$$r(1) \leq r(2) \leq \cdots \leq r(n). \tag{4}$$

Since Monge matrices form a proper subclass of totally monotone matrices, property (4) holds for Monge matrices as well.

3. MST for Complete Graphs with Totally Monotone Distance Matrices

In this section an $O(n)$ time algorithm for computing a minimum spanning tree in Monge graphs with n nodes is presented. Additionally we will see that this algorithm is also suited for graphs with weight matrices which are symmetric and totally monotone.

Before describing the algorithm in detail we start with some basic definitions and notations.

A *cut* $(S, V \backslash S)$ of $G = (V, E)$ is a partition of V. An edge (u, v) is said to *cross* the cut $(S, V \backslash S)$ if one of its endpoints lies in S and the other one lies in $V \backslash S$. Finally, given an $n \times n$ matrix C we denote by $C[r_1, \ldots, r_2][s_1, \ldots, s_2]$ the $(r_2 - r_1 + 1) \times (s_2 - s_1 + 1)$ matrix which results from C by deleting all rows except those contained in $\{r_1, \ldots, r_2\}$ and all columns except those contained in $\{s_1, \ldots, s_2\}$.

Algorithm MST:

```
(1) Given an $n \times n$ totally montone matrix $C$ determine the values $r(i)$,
    $i = 1, \ldots, n$.
    Set $T := \{(i, r(i)) | 1 \le i \le n, i \ne r(i)\}$.
(2) For all $1 \le i < n$ do:
    If $r(i) \le i$ and $r(i+1) \ge i+1$ then
      (a) Let $c_{kl}$ be the minimum entry of the matrix $D_i$ where
          $D_i := C[i+1, \ldots, r(i+1)][r(i), \ldots, i]$.
      (b) Set $T := T \cup \{(k, l)\}$.
(3) Output $T$.
```

Theorem 3.1. *The time complexity of Algorithm* MST *is* $O(n)$.

Proof: Note that the minimum entry of an $n_1 \times n_2$ totally monotone matrix can be found in $O(n_1 + n_2)$ time and that the leftmost minimum entries in each row of an $n \times n$ totally monotone matrix can be computed in $O(n)$ time (Aggarwal et al. [1]). Thus Step (1) can be performed in $O(n)$ time. Next, it is shown that the time complexity of Step (2) is also $O(n)$. This is due to the fact that the search for the minimum entry of the matrix D_i takes at most $O(r(i+1) - r(i) + 1)$ operations. Summing over all indices i, we have $\sum_{i=1}^{n-1} O(r(i+1) - r(i) + 1) = O(n + r(n) - r(1)) = O(n)$, thus the total time is bounded by $O(n)$. ■

After having proven the time complexity of Algorithm MST the correctness needs to be shown. Therefore consider the following lemma.

Lemma 3.2. *The set T defined in Step* (1) *of Algorithm* MST *is a subset of a minimum spanning tree.*

Proof: Suppose that after Step (1) $T = \{(i_1, r(i_1)), \ldots, (i_k, r(i_k))\}$, where $i_1 < \cdots < i_k$. First of all we show that T contains no cycle, i.e. is a subset of a spanning tree. To this end investigate the directed graph T' corresponding to T where each edge $(i_l, r(i_l))$ of T is directed from i_l to $r(i_l)$. It is clear that the outdegree of any node is exactly one, therefore a cycle in T exists if and only if this cycle is a directed cycle in T'. So assume that such a cycle C exists in T'. Let i be the minimal node in C. Then $r(i) > i$ and there exists a node $k > i$ in C such that $r(k) = i$. But this contradicts property (4), since $r(i) \le r(k)$. Therefore no cycle is contained in T' and T is indeed a subset of a spanning tree.

In order to show that T is a subset of a minimum spanning tree, it is sufficient to investigate the cuts $(\{i_l\}, V \setminus \{i_l\})$ for all $1 \le l \le k$. Due to the definitions of the edges $(i_l, r(i_l))$ it is clear that these edges are minimal edges crossing the above cuts and therefore they are contained in a minimum spanning tree. ■

Now we are prepared to prove the correctness of Algorithm MST.

Theorem 3.3. *Given a totally monotone, symmetric $n \times n$ matrix C. Then Algorithm* MST *constructs a minimum spanning tree.*

Proof: Lemma 3.2 implies that after Step (1) of Algorithm MST T is a subset of a minimum spanning tree. In the sequel it is shown that whenever in Step (2) an edge is added to T, T remains a subset of a minimum spanning tree. So suppose that there exists an index $1 \leq i < n$ such that $r(i) \leq i$ and $r(i+1) \geq i+1$: Consider the cut $(\{1,\dots,i\}, \{i+1,\dots,n\})$. It is clear that no edge of T obtained so far crosses this cut. Due to the properties of a minimum spanning tree, the smallest edge crossing this cut must be included in the set T, i.e. the edge corresponding to the minimum entry in the matrix $C' := C[1,\dots,i]\,[i+1,\dots,n]$. Due to the total monotonicity of C (and also C') we will see that the minimum entry in C' is always contained in the submatrix D_i, defined in Step (2a). To that end distinguish two cases:

Case (i): $1 \leq p < i+1$ and $r(i+1) < q \leq n$: Then $c_{pq} \geq c_{p,r(i+1)}$. Assume that $c_{p,r(i+1)} > c_{pq}$. Since C is totally monotone, $c_{i+1,r(i+1)} > c_{i+1,q}$ which is a contradiction to the fact that $c_{i+1,r(i+1)}$ is the leftmost minimum entry in row $i+1$.

Case (ii): $1 \leq p < r(i)$ and $i+1 \leq q \leq r(i+1)$: Then $c_{pq} \geq c_{r(i),q}$. Assume that $c_{pq} < c_{r(i),q}$. Since $c_{ip} > c_{i,r(i)}$ ($c_{i,r(i)}$ is the leftmost minimum entry in row i), it follows from the total monotonicity of C that $c_{qp} > c_{q,r(i)}$. But this contradicts our assumption, since C is symmetric.

Thus it is sufficient to investigate only the matrix $C[r(i),\dots,i]\,[i+1,\dots,r(i+1)]$ in order to obtain the minimal edge from $\{1,\dots,i\}$ to $\{i+1,\dots,n\}$. Since C is symmetric, this task is equivalent with finding the minimum entry in matrix D_i.

Finally it remains to be proven that Algorithm MST stops with a set T where $|T| = n-1$. Let $r := |\{i \mid r(i) = i, i = 1,\dots,n\}|$. After executing Step (1) T consists of exactly $n - r$ edges. In Step (2) exactly $r - 1$ times an edge is added to T. Thus T is indeed the edge set of a minimum spanning tree. ■

4. Shortest Paths in Monge Graphs

This section deals with the problem of determining all-pairs shortest paths in a Monge graph G having n nodes under the assumption that G contains no negative cycle. (This is e.g. achieved if the associated distance matrix C is nonnegative.) The main result is an algorithm with optimal running time $O(n^2)$, thereby improving Floyd's $O(n^3)$ time algorithm for complete graphs with arbitrary distances.

First of all we recall a property of shortest paths from 1 to n in Monge graphs:

Lemma 4.1. (*e.g.* [3])
In a Monge graph G there always exists a shortest path P from 1 *to n which is an ascending path, i.e. $P = (p_1, p_2, \dots, p_l)$ with $1 = p_1 < p_2 < \dots < p_l = n$.*

As an immediate consequence of Lemma 4.1 the following may be concluded: if we restrict the node set to the set $\{i,\dots,j\}$, $1 \le i < j \le n$, and ask for a shortest (restricted) path from i to j then there always exists an ascending path between i and j which is also a shortest path. However, it is obvious that by dropping the restriction on the node set the shortest ascending path from i to j may not be a shortest path, i.e. there may exist a shorter path from i to j visiting some nodes from $V \setminus \{i,\dots,j\}$. In the sequel we explore the structure of a shortest path from i to j which may visit any node in the graph G.

To make things easier to illustrate we use some definitions and notations. Denote by $P_{r,s}$ the subpath of P starting at node r and ending in node s using the same edges as P and by the *reverse* subpath according to $P_{r,s}$ we understand the subpath of P with opposite orientation, i.e. this is a path from s to r using all edges of $P_{r,s}$ in opposite direction. Furthermore, given a path $P = (p_1, p_2, \dots, p_k)$ we denote by μ_P and ν_P the minimum and the maximum inner node visited by path P, i.e. $\mu_P := \min\{p_i | 1 < i < k\}$ and $\nu_P := \max\{p_i | 1 < i < k\}$. Finally $L(P)$ denotes the total length of path P.

Theorem 4.2. *In a Monge graph G there always exists a shortest path P from node i to j, $i < j$, such that $P = (i = p_1, p_2, \dots, p_k = j)$ with $p_2 < \cdots < p_{k-1}$.*

Proof: We show that there always exists a shortest path P from i to j fulfilling these conditions. Let $Q = (i = q_1, q_2, \dots, q_t = j)$ be an arbitrary shortest path from i to j visiting at least four distinct nodes (if Q contains less nodes, Q itself fulfills all conditions). Now let r and s be those indices such that $q_r = \mu_Q$ and $q_s = \nu_Q$. (Since Q contains at least two distinct inner nodes, $r \ne s$.)

First suppose that $r < s$, i.e. node μ_Q is visited before ν_Q. We show that there always exists a shortest path Q such that

(a) $q_r < q_{r+1} < \dots < q_s$.
(b) $q_2 = \mu_Q$, i.e. the smallest node is the first inner node of path Q.
(c) $q_{t-1} = \nu_Q$, i.e. the last inner node is always the largest one.

Property (a) follows directly from the remarks following Lemma 4.1. If the subpath $Q_{r,s}$ is not ascending we may always replace it by an equivalent shortest ascending path from q_r to q_s. Thus we may assume w.l.o.g. that Q has already property (a).

To show property (b) suppose the contrary, i.e. that $q_2 \ne \mu_Q$. Due to the definition of μ_Q, it follows that $q_2 > q_r = \mu_Q$. Distinguish two cases:

Case (i): In the case that $i = q_1 > q_2$ we proceed as follows: Since Q_{rs} is an ascending path and $q_r < q_2 < q_s$, there exists an edge (q_k, q_{k+1}) on Q_{rs} such that $q_k < q_2 < q_{k+1}$. Next we construct another path $Q' = (q_1, q_{k+1}, \dots, q_t)$. From the Monge property and the assumption that G does not contain negative cycles it follows that $L(Q) - L(Q') = c_{i,q_2} + L(Q_{q_2,q_k}) + c_{q_k,q_{k+1}} - c_{i,q_{k+1}} \ge c_{q_k,q_2} + c_{i,q_{k+1}}$

$+L(Q_{q_2,q_k}) - c_{i,q_{k+1}} = c_{q_k,q_2} + L(Q_{q_2,q_k}) \geq 0$. Note that Q' is therefore a shortest path fulfilling properties (a) and (b).

Case (ii): If $i < q_2$, there exists an edge (q_k, q_{k+1}) on Q_{rs} such that $q_k < i < q_{k+1}$. The new path Q' is obtained by replacing the subpath $Q_{i,q_{k+1}}$ by the edge (i, q_k) followed by the reversed path Q_{q_k,q_2} and the edge (q_2, q_{k+1}). Again Q' is a shortest path, since $L(Q) - L(Q') = c_{i,q_2} + c_{q_k,q_{k+1}} - c_{i,q_k} - c_{q_2,q_{k+1}} = c_{i,q_2} + c_{q_{k+1},q_k} - c_{i,q_k} - c_{q_{k+1},q_2} \geq 0$ (due to the symmetry of C and the Monge property). Next we define Q'' by replacing $Q'_{rs} = (q_r, \dots, q_2, q_{k+1}, \dots, q_s)$ with a shortest ascending path from node r to node s. Now we are done. The path Q'' fulfills property (a) and fits into Case (i), since $i > q_k = q''_2$.

In a similar way property (b) may be verified. This follows directly from the fact that the reversed matrix of C, i.e. the matrix obtained by reversing the order of the rows and columns of C, is again a Monge matrix. Applying the proof above to the reversed matrix we get the desired result.

Summarizing all properties of Q we see that there always exists a shortest path P fulfilling the requirements of this theorem.

It remains to be shown what happens in the case when node μ_Q is visited after node ν_Q. i.e. $r > s$. Using again the same considerations as above we can always find an equivalent shortest path P now having the property that $p_2 > p_3 > \cdots > p_{k-1}$. Again using the Monge property, we obtain that the path $P' = (i, p_{k-1}, \dots, p_1, j)$ has at most the same length. Thus the theorem is proven. ■

To illustrate that a shortest path in a Monge graph need not to be an ascending path consider the following illustrative example. Let

$$C = \begin{pmatrix} - & 1 & 5 & 3 \\ 1 & - & 8 & 5 \\ 5 & 8 & - & 1 \\ 3 & 5 & 1 & - \end{pmatrix}$$

be the distance matrix of a Monge graph. Then e.g. the shortest path from 2 to 3 is just $(2, 1, 4, 3)$ having a total length of 5.

After having shown the structural properties of a shortest path in a Monge graph we will present an algorithm exploiting Theorem 4.2 explicitly in order to compute all-pairs shortest paths efficiently. Let Γ be the shortest-distance matrix with respect to G, i.e. γ_{ij} is the value of the shortest path from i to j in G. It is known that for arbitrary complete graphs with n nodes Γ can be computed in $O(n^3)$ time applying the Floyd-Warshall algorithm (Floyd [6]). The following algorithm computes Γ in $O(n^2)$ time whenever the weights fulfill the Monge property. (Note that this is best possible, since Γ has exactly n^2 entries.)

Algorithm ALL-PAIRS SHORTEST PATHS:

(1) Compute the symmetric $n \times n$ matrix F, where f_{ij} represents the value of the shortest ascending path from i to j, where $i<j$.

(2) Compute $\tilde{F}$ by

$$\tilde{f}_{ij} := \min\left\{f_{ij}, \min_k d^i_{kj}\right\},$$

where D^i, $i = 1,\dots,n-1$, is a matrix defined by

$$d^i_{kj} := \begin{cases} f_{ik} + c_{kj}, & i<j<k \\ \infty & \text{otherwise.} \end{cases}$$

(3) Compute Γ by

$$\gamma_{ij} := \min\left\{\tilde{f}_{ij}, \min_k \tilde{d}^j_{ik}\right\},$$

where for all $j = 2,\dots,n$ the matrix $\tilde{D}^j$ is defined by

$$\tilde{d}^j_{ik} := \begin{cases} c_{ik} + \tilde{f}_{kj}, & k<i<j \\ \infty & \text{otherwise.} \end{cases}$$

Theorem 4.3. *Given a complete Monge graph $G=(V,E)$, Algorithm* ALL-PAIRS SHORTEST PATHS *computes all-pairs shortest paths in $O(|V|^2)$ time.*

Proof: To show the correctness recall that the value d^i_{kj} denotes the length of a shortest ascending path from i to k plus the length of the edge (k,j) and $\tilde{f}_{ij}$ is the shortest path value from i to j among all paths which use the node set $\{i,\dots,n\}$. In a similar way, $\tilde{d}^j_{ik}$ represents the value of a path starting with edge (i,k) plus the length of a shortest path from k to j using only nodes from $\{k,\dots,n\}$. Due to Theorem 4.2 it is clear that finally γ_{ij} is the value of a shortest path from i to j.

To show that the total running time complexity is $O(n^2)$ we make use of Wilber's algorithm [14] and the SMAWK-algorithm (Aggarwal et al. [1]). In Step (1) the matrix F is computed in $O(n^2)$. For each $1 \leq i < n$ we use once Wilber's algorithm to compute the shortest (ascending) path from i to n. (Looking at this algorithm in detail it turns out that beside the shortest path from i to n also all shortest ascending paths from i to any arbitrary node j, $i<j$, are computed at the same time.) Thus F can be computed in $O(n^2)$ time. Next it is shown that Step (2) and Step (3) can also be performed in $O(n^2)$ time. First note that all matrices D^i and $\tilde{D}^j$ are totally monotone. This follows directly from the fact that adding a constant column vector to each column of a totally monotone matrix results again in a totally monotone matrix. Thus thanks to the SMAWK-Algorithm all row minima and therefore the matrices $\tilde{F}$ and Γ may be computed in $O(n^2)$ time without explicitly calculating each entry of the matrices D^i and $\tilde{D}^j$. ■

5. Minimum Weighted Shortest Path Trees

In the previous section it has been shown how to compute all-pairs shortest paths in a Monge graph in linear time. As already mentioned a 1-to-all shortest path tree containing all shortest paths emanating from 1 can be computed in $O(n)$ time as a byproduct of Wilber's algorithm. In the sequel we investigate the problem of finding a minimum 1-to-all shortest path tree, i.e. a 1-to-all shortest path tree which has minimum total weight among all 1-to-all shortest path trees emanating from node 1.

To that end we attach to each edge $(i, j) \in E$ the pair $(d_i + c_{ij}, c_{ij})$ and call the associated $n \times n$ matrix A. (Note that d_i represents the value of the shortest path from node 1 to node i.) Additionally let $\prec$ ($\preccurlyeq$) denote the usual lexicographic order, i.e. $(a, b) \prec (c, d)$, iff either $a < c$ or $a = c$ and $b < d$ holds. A closer look to the values $d_i + c_{ij}$ reveals that only the edges corresponding to minimum entries in column j of A can occur in a shortest path tree. That means that each edge (k, j) for which

$$d_k + c_{kj} > \min_{1 \le i \le n} \{d_i + c_{ij}\}$$

is never part of a shortest path tree and therefore not contained in any minimum weighted 1-to-all shortest path tree.

Let us start with the following observation.

Lemma 5.1. *If C is a Monge matrix, then A^T, the transposed matrix of A, is totally monotone with respect to $\prec$.*

Proof: It is sufficient to show that for all $i < r$ and $j < s$: $a_{ij} \succ a_{rj} \Rightarrow a_{is} \succ a_{rs}$. Assume the contrary, i.e. that $a_{ij} \succ a_{rj}$ and $a_{is} \preccurlyeq a_{rs}$. Now $a_{ij} \succ a_{rj}$ implies that either $d_i + c_{ij} > d_r + c_{rj}$ or that $d_i + c_{ij} = d_r + c_{rj}$ and $c_{ij} > c_{rj}$. On the other hand $a_{is} \preccurlyeq a_{rs}$ is equivalent to $d_i + c_{is} > d_r + c_{rs}$ or $d_i + c_{is} = d_r + c_{rs}$ and $c_{is} \le c_{rs}$. By combining these equations and inequalities in an appropriate way we get in any case a contradiction to the Monge property of C. ■

In the following we briefly show that a small adaption of Algorithm MST — further referred to as Algorithm MWSPT — is sufficient to obtain an algorithm for constructing a minimum weighted 1-to-all shortest path tree (each comparison must be evaluated according to $\prec$ instead of $<$).

Theorem 5.2. *Algorithm* MWSPT *applied to A^T computes a minimum weighted 1-to-all shortest path tree in $O(n)$ time in a Monge graph having nonnegative weights.*

Proof: This theorem can be proven in the same way as the results for computing the MST shown in Section 3. The main difference here is that the underlying matrix $B := A^T$ is not necessarily symmetric. (To simplify and to shorten the proof we highlight only those parts which must be changed due to the unsymmetry of the matrix.) It is clear that the running time is $O(n)$ and that the algorithm computes an undirected tree at all. Thus it remains to be shown that this tree is a minimum weighted 1-to-all shortest path tree.

First consider an edge (i, j), $j = r(i) \neq i$, which is inserted during the execution of Step (1). Since this entry, say $(d_j + c_{ji}, c_{ji})$, corresponds to the leftmost minimum entry in row i of A^T, it follows $(d_j + c_{ji}, c_{ji}) \preccurlyeq (d_k + c_{ki}, c_{ki})$ for all $1 \leq k \leq n$. This means that this edge must be part in our desired tree, since it is contained in a shortest path tree and since it has the smallest weight among all such edges.

The proof for edges which are inserted Step (2) is in principle the same as for Theorem 3.3 except that we distinguish here three cases:

Case (i): $i < q \leq n$ and $1 \leq p < r(i)$: Since $b_{ip} \succ b_{i,r(i)}$ (by definition of $r(i)$), it follows that $b_{qp} \succ b_{q,r(i)}$ (since B is totally monotone).

Case (ii): $r(i+1) < q \leq n$ and $r(i) \leq p \leq i$: Assume that b_{qp} is the minimal edge crossing the cut $(\{1, \ldots, i\}, \{i+1, \ldots, n\})$ and especially that $b_{qp} \prec b_{r(i+1),p)}$. (If this is not the case, choose entry $b_{r(i+1),p}$ instead of entry b_{qp}.) The minimality of b_{qp} and the construction of matrix B imply the existence of a shortest path from 1 to q with the final edge (p, q). Thus $d_q = d_p + c_{qp}$ and $d_q \leq d_j$ for all $i+1 \leq j \leq n$. From $b_{qp} \prec b_{r(i+1),p}$ it follows that $c_{qp} < c_{r(i+1),p}$ or, equivalently, that $c_{pq} < c_{p,r(i+1)}$ (C is symmetric.) The total monotonicity of C implies the inequality $c_{i+1,r(i+1)} > c_{i+1,q}$. On the other hand $b_{i+1,r(i+1)} \preccurlyeq b_{i+1,q}$ since $(i+1, r(i+1))$ is the leftmost minimum entry in row $i+1$ of matrix B. Combining these two inequalities we get that $d_{r(i+1)} + c_{i+1,r(i+1)} < d_q + c_{i+1,q}$ and that $d_{r(i+1)} < d_q$. But this contradicts the fact that $d_q \leq d_{r(i+1)}$.

Case (iii): $1 \leq p \leq i$ and $i+1 \leq q \leq n$: Assume that b_{pq} corresponds to the minimal edge crossing the cut and that $b_{bp} \prec b_{qp}$ (otherwise we exchange and are done). This implies that $d_p = d_q + c_{pq}$ and since $b_{qp} \prec b_{pq}$, $c_{pq} > 0$ and $d_q < d_p$. On the other hand, since the node p is already connected to node 1 (and this can only be done by a shortest path), we have that the path from 1 to q would have a length of $d_p + c_{pq} > d_p > d_q$, a contradiction.

Thus, in the final step we obtain an undirected tree which corresponds to a minimum weighted 1-to-all shortest path tree. After directing it we are done. ∎

Acknowledgement

We would like to thank Katja Wolf for careful reading of a first version of this paper.

References

[1] Aggarwal, A., Klawe, M. M., Moran, S., Shor, P., Wilber, R.: Geometric applications of a matrix-searching algorithm. Algorithmica *2*, 195–208 (1987).
[2] Burkard, R. E.: Special cases of the travelling salesman problem and heuristics. Acta Math. Appl. Sin. *6*, 273–288 (1990).
[3] Burkard, R. E., Klinz, B., Rudolf, R.: Perspectives of Monge properties in optimization. Discr. Appl. Math. *70*, 95–161 (1996).
[4] Burkard, R. E., Deĭneko, V., van Dal, R., van der Veen, J. A. A., Woeginger, G. J.: Well-solvable cases of the TSP: A survey. SFB-Report No. 52, SFB 'Optimierung und Kontrolle', Institute of Mathematics, University of Technology, Graz, Austria, December 1995, submitted for publication.
[5] Dijkstra, E. W.: A note on two problems in connection with graphs. Numer. Math. *1*, 269–271 (1959).
[6] Floyd, R. W.: Algorithm 97 (Shortest Path). Comm. ACM *5*, 345 (1962).
[7] Fredman, M. L., Tarjan, R. E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. Assoc. Comput. Mach. *34*, 596–615 (1987).
[8] Gabow, H. N., Galil, Z., Spencer, T., Tarjan, R. E.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica *6*, 109–122 (1986).
[9] Khuller, S., Raghavachari, B., Young, N.: Balancing minimum spanning trees and shortest-path trees. Algorithmica *44*, 305–321 (1995).
[10] Klein, P., Tarjan, R. E.: A randomized linear-time algorithm for finding minimum spanning trees. Proc. 26th Annual Symposium on Theory of Computing, pp. 9–15 (1994).
[11] Pferschy, U., Rudolf, R., Woeginger, G. J.: Monge matrices make maximization manageable. Oper. Res. Lett. *16*, 245–254 (1994).
[12] Prim, R. C.: Shortest connection networks and some generalizations. Bell System Tech. J. *36*, 1389–1401 (1957).
[13] Supnick, F.: Extreme Hamiltonian lines. Ann. Math. *66*, 179–201 (1957).
[14] Wilber, R.: The concave least-weight subsequence problem revisited. J. Algorithms *9*, 418–425 (1988).

T. Dudás
R. Rudolf
Institut für Mathematik B
Graz
Steyrergasse 30
A-8010 Graz
Austria
e-mails:{dudas, rudolf}@opt.math.tu-graz.ac.at