Math. Meth. Oper. Res. (2006) 63: 543–551
DOI 10.1007/s00186-005-0053-1

ORIGINAL ARTICLE

Martin Gavalec · Ján Plavka

Computing an eigenvector of a Monge matrix in max-plus algebra

Received: 15 October 2003 / Accepted: 15 January 2005 / Published online: 29 April 2006

Abstract The problem of finding one eigenvector of a given Monge matrix A in a max-plus algebra is considered. For a general matrix, the problem can be solved in $O(n^3)$ time by computing one column of the corresponding metric matrix $\Delta(A_\lambda)$, where λ is the eigenvalue of A. An algorithm is presented, which computes an eigenvector of a Monge matrix in $O(n^2)$ time.

Keywords Eigenproblem · Monge matrix

Mathematics Subject Classification (2000) Primary: 90C27 · Secondary: 05B35

1 Introduction

The problem of computing one eigenvector of a given Monge matrix A in a max-plus algebra is considered in this note. The problem is equivalent to computing the maximal weights of paths from all vertices to a fixed vertex j in the associated digraph D_A, and, in the general case, it can be solved in $O(n^3)$ time by computing the j-th column of the metric matrix $\Delta(A_\lambda)$, Cuninghame-Green (1979). We present a faster algorithm for Monge matrices, using techniques known in the area of efficiently solvable cases of the TSP and other problems.

M. Gavalec (✉)
Department of Information Technologies,
Faculty of Informatics and Management,
University Hradec Králové, Rokitanského 62
50003 Hradec Králové, Czech Republic
E-mail: martin.gavalec@uhk.cz

J. Plavka
Departments of Mathematics,
Faculty of Electrical Engineering and Informatics,
Technical University in Košice, B.N mcovej 32,
04200, Košice, Slovak Republic
E-mail: jan.plavka@tuke.sk

An analogous problem of finding the minimal weights of all paths ending in the last vertex n in a Monge digraph, can be solved in $O(n^2)$ time by a standard dynamic algorithm based on an $O(n)$ algorithm proposed by Wilber (1988). However, this result does not help to compute an eigenvector of a Monge matrix, where maximal weights are needed. In general, the maximal-weight paths problems in Monge graphs are more difficult than the minimal-weight ones (see the survey article of Burkard, Klinz and Rudolf (1996)).

We show in this paper how the computation of one eigenvector of a Monge matrix can be done in $O(n^2)$ time. The main idea of our approach lies in reducing the considered paths between two nodes in D_A to paths of a special form, so called spirals.

2 Notions and notation

By a max-plus algebra we understand the algebraic structure $(G, \oplus, \otimes) = (\mathcal{R}^\star, \max, +)$, where $G = \mathcal{R}^\star$ is the set of all real numbers $\mathcal{R}$ extended by an infinite element $-\infty$, and $\oplus$, $\otimes$ are the binary operations on $\mathcal{R}$: $\oplus = \max$ and $\otimes = +$. The infinite element is neutral with respect to the maximum operation and absorbing with respect to addition.

All the results presented in this paper for the max-plus algebra $(\mathcal{R}^\star, \max, +)$ are valid also for the more general notion of max-plus algebra, in which $(G, \oplus, \otimes)$ is derived in a similar way from an arbitrary divisible commutative linearly ordered group in additive notation. In the case of general G, the neutral element $e \in G$ of the additive group must be used instead of $0 \in \mathcal{R}$.

For any natural $n > 0$, we denote $N = \{1, 2, \ldots, n\}$. Further, we denote by G_n the set of all $n \times n$ matrices over G. The matrix operations over the max-algebra G are defined with respect to $\oplus$, $\otimes$, formally in the same manner as the matrix operations over any field. The operation $\otimes$ for matrices denotes the formal matrix product with operations $\oplus = \max$ and $\otimes = +$ replacing the usual operations $+$, $\cdot$, while the operation $\oplus$ for matrices is performed componentwise. The problem of finding a vector $x \in G^n$ and a value $\lambda \in G$ satisfying

$$A \otimes x = \lambda \otimes x$$

is called an extremal eigenproblem corresponding to the matrix A, the value λ is called (extremal) eigenvalue, and x is called (extremal) eigenvector of A. The word "extremal" is usually omitted. A survey of the results concerning various types of eigenproblems can be found in Zimmermann (1981).

The associated digraph D_A of a matrix $A \in G_n$ is defined as a complete arc-weighted digraph with the vertex set $V = N$, and with the arc weights $w(i, j) = a_{ij}$ for every $(i, j) \in N \times N$. If p is a path or a cycle in D_A, of length $r = |p|$, then the weight $w(p)$ is defined as the sum of all weights of the arcs in p. If $r > 0$, then the mean weight of p is defined as $w(p)/r$. Of all the mean weights of cycles in D_A, the maximal one is denoted by $\lambda(A)$. By Cuninghame-Green in Cuninghame-Green (1979), the maximal cycle mean $\lambda(A)$ is the unique eigenvalue of A. The problem of finding the eigenvalue $\lambda(A)$ has been studied by a number of authors and several algorithms are known for solving this problem. The algorithm described by Karp (1978) has the worst-case performance $O(n^3)$.

For $B \in G_n$ we denote by $\Delta(B)$ the matrix $B \oplus B^2 \oplus \cdots \oplus B^n$ where B^s stands for the s-fold iterated product $B \otimes B \otimes \cdots \otimes B$. Further, we denote $A_\lambda = -\lambda(A) \otimes A$ (here we have a formal product of a scalar value $-\lambda(A)$ and a matrix A, i.e. $[A_\lambda]_{ij} = -\lambda(A) + a_{ij}$ for any $(i, j) \in N \times N$). It is shown in Cuninghame-Green (1979) that the matrix $\Delta(A_\lambda)$ contains at least one column, the diagonal element of which is 0 and every such a column is an eigenvector (so called: fundamental eigenvector) of the matrix A. Moreover, every eigenvector of A can be expressed as a linear combination of fundamental eigenvectors.

Let be $\Delta(A_\lambda) = (\delta_{ij})$. It follows from the definition of $\Delta(A_\lambda)$ that δ_{ij} is the maximal weight of a path from i to j in D_{A_λ}. Hence, $\Delta(A_\lambda)$ can be computed in $O(n^3)$ time, using the Floyd-Warshall algorithm Lawler (1976). In this way, a complete set of fundamental eigenvectors can be found by at most $O(n^3)$ operations. However, if we wish to compute only one single eigenvector of A, no better algorithm than $O(n^3)$ is known for matrices of a general type. In the special case, when the matrix A is Monge, the above computations can be performed in a more efficient way.

Definition 2.1 We say that a matrix $A = (a_{ij}) \in G_n$ is Monge if and only if

$$a_{ij} + a_{kl} \leq a_{il} + a_{kj} \quad \text{for all} \quad i < k,\ j < l$$

Similarly, we say that a matrix $A = (a_{ij}) \in G_n$ is inverse Monge if and only if

$$a_{ij} + a_{kl} \geq a_{il} + a_{kj} \quad \text{for all} \quad i < k,\ j < l$$

It has been shown in Gavalec and Plavka (2003) that the eigenvalue $\lambda(A)$ of a Monge matrix can be found in $O(n^2)$ time (in $O(n)$ time for an inverse Monge matrix). In the rest of this paper, we shall show that the computation of a single eigenvector of a Monge matrix can also be performed in $O(n^2)$ time.

3 Spiral paths

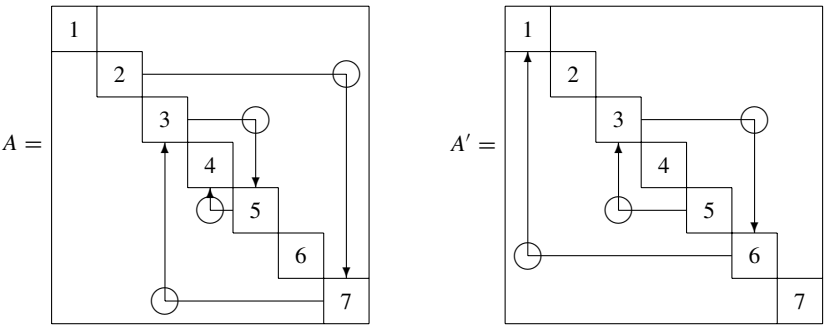
In the following we will make use of the notions of peaks and valleys known from the theory of special cases of the TSP to describe specially stuctured paths in the digraph D_A. We say that a node i_k in a path $p = (i_0, i_1, \ldots, i_r)$ is a *peak* in p, if the incoming arc (i_{k-1}, i_k) is increasing and the outgoing arc (i_k, i_{k+1}) is decreasing, i.e. if $i_{k-1} < i_k$ and $i_k > i_{k+1}$. If p is not a cycle, i.e. if $i_0 \neq i_r$, then for $k = 0$ ($k = r$), the first (second) condition is left out. The notion of a *valley* in p is defined dually. A *pyramidal tour* can be characterized by the requirement that node 1 is the only valley and node n is the only peak. Pyramidal tours are useful in well-solvable cases of hard combinatorial problems, such as the TSP (see the survey article Burkard et al. (1998)). Other interesting results were found by Russian mathematicians Aizenshtat and Kravchuk (1968); Aizenshtat and Maksimovich (1978, 1979) using the key property that every node is a valley or a peak. For our purpose, we use a similar notion of a spiral path (shortly: a spiral).

Definition 3.1 We say that an elementary path $p = (i_0, i_1, \ldots, i_r)$ *in* D_A is a spiral, if and only if

(i) Every node in p is either a peak or a valley, and the peaks and the valleys alternate.
(ii) The subsequence of peaks and the subsequence of valleys are dually monotonous, i.e. one of them is increasing and the other one is decreasing.

We say that a spiral is convergent*, if the peak subsequence in the spiral is decreasing, and the valley subsequence is increasing. In the dual case, the spiral is called* divergent*. The set of all convergent spirals from node i to node j is denoted by* $\mathrm{Sc}(i, j)$*, and the set of all divergent spirals from i to j is denoted by* $\mathrm{Sd}(i, j)$.

By the above definition, every spiral path p in D_A of length > 1 is either convergent, or divergent. Paths of length 0 (one node) or 1 (one arc) in D_A are the only spirals, which are both convergent and divergent. The following picture shows an example of a convergent and of a divergent spiral. The vertices $1, 2, \ldots, n$ of the digraph D_A are identified with the diagonal elements of the matrix A and the circles on the arcs indicate the corresponding matrix elements.



$p = (2, 7, 3, 5, 4) \in \mathrm{Sc}(2, 4)$ $\qquad$ $p' = (5, 3, 6, 1) \in \mathrm{Sd}(5, 1)$

Further, we say that a pair of arcs (i_k, i_{k+1}), (i_l, i_{l+1}) in a path $p = (i_0, i_1, \ldots, i_r)$ is reducible, if and only if the inequalities

$$\begin{aligned} i_k < i_l \quad &\text{and} \quad i_{k+1} < i_{l+1} \qquad \text{or,} \\ i_k > i_l \quad &\text{and} \quad i_{k+1} > i_{l+1} \end{aligned}$$

hold true. Lemma 3.1 shows that the reducible pairs of arcs are closely connected with spiral paths.

Lemma 3.1 *An elementary path p in D_A is a spiral if and only if p contains no reducible pair of arcs.*

Proof First, let p be a spiral and let us assume that p contains a reducible pair of arcs (i_k, i_{k+1}), (i_l, i_{l+1}) fulfilling the inequalities $i_k < i_l$ and $i_{k+1} < i_{l+1}$ (the reversed inequalities are handled analogously). Without any loss of generality, we may assume that $k < l$, i.e. $k+1 \leq l$. If $k+1 = l$, then $i_k < i_{k+1} = i_l < i_{l+1}$ holds, and the node $i_{k+1} = i_l$ is neither peak nor valley, which contradicts to condition (i) in Definition 3.1. For $k + 1 < l$, we shall consider two cases.

case 1. Nodes i_k, i_l are of the same type. Then both nodes i_{k+1}, i_{l+1} also have the same type, which is dual to the type of nodes i_k, i_l. In other words, if i_k, i_l are

valleys, then nodes i_{k+1}, i_{l+1} are peaks, and conversely. In both alternatives, the inequalities $i_k < i_l$ and $i_{k+1} < i_{l+1}$ are in contradiction to condition (ii).

case 2. Nodes i_k, i_l are of different types. Then nodes i_{k+1}, i_{l+1} are of different (dual) types. In particular, if i_k is a valley and i_l is a peak, then i_{k+1} is a peak and i_{l+1} is a valley, and then the inequalities $i_k < i_{k+1} < i_{l+1} < i_l$ hold true. In the dual situation, when i_k is a peak and i_l is a valley, we get the inequalities $i_{k+1} < i_k < i_l < i_{l+1}$. In both alternatives, the inequalities imply $i_k < i_{l+1}$ and $i_{k+1} < i_l$, a contradiction to (ii).

Second, let the path p contain no reducible pair of arcs. We shall show that conditions (i), (ii) in Definition 3.1 are satisfied. If i_k is a node in p which is neither peak nor valley, then the inequalities $i_{k-1} < i_k < i_{k+1}$, or $i_{k-1} > i_k > i_{k+1}$, hold true. Therefore, the arcs (i_{k-1}, i_k), (i_k, i_{k+1}) form a reducible pair in p, a contradiction. Hence, every node in p must be either a peak, or a valley. As two consecutive peaks (or valleys) in a path are not possible, we get condition (i).

Condition (ii) is trivially fulfilled, if $|p| < 3$, i.e. if p contains at most three nodes. Let $|p| \geq 3$ and let $i_{k-1}, i_k, i_{k+1}, i_{k+2}$ be four consecutive nodes in p. Let us suppose that the inequalities $i_{k-1} < i_{k+1}$ and $i_k < i_{k+2}$ hold true. Then the arcs (i_{k-1}, i_k), (i_{k+1}, i_{k+2}) form a reducible pair, in contradiction to our assumption. By an analogous argument, the inequalities $i_{k-1} > i_{k+1}$ and $i_k > i_{k+2}$ cannot hold simultaneously.

Thus, of any four consecutive nodes in p, the first and third node are ordered in the opposite direction than the second and fourth one. As a consequence, either the subsequence of the nodes with the odd indices is increasing and the subsequence of the nodes with the even indices is decreasing, or conversely. Therefore, condition (ii) holds true. □

Theorem 3.2 *If $A = (a_{ij}) \in G_n$ is a Monge matrix over a max-plus algebra G, and if $\lambda(A) = 0$, then any non-diagonal element of the matrix $\Delta(A) = (\delta_{ij})$ can be expressed in the form*

$$\delta_{ij} = \max\{\, w(p);\ p \in \mathrm{Sc}(i,j) \cup \mathrm{Sd}(i,j)\,\} \quad \textit{for } i \neq j$$

Proof It is shown in Cuninghame-Green (1979) that under the assumption $\lambda(A) = 0$, the matrix $\Delta(A)$ is the metric matrix of the digraph D_A, i.e. any element δ_{ij} with $i \neq j$ is equal to the maximal weight of a path from i to j. By the assumption $\lambda(A) = 0$, the weight of any cycle is non-positive. Therefore, the computation of δ_{ij} can be restricted to elementary paths, simply reducing any non-elementary path by clipping out all of its subcycles.

If an elementary path $p = (i_0, i_1, \ldots, i_r)$ from $i = i_0$ to $j = i_r$ is not a spiral, then, by Lemma 3.1, p contains a reducible pair of arcs (i_k, i_{k+1}), (i_l, i_{l+1}) with $i_k < i_l$ and $i_{k+1} < i_{l+1}$. If we substitute these arcs in p by (i_k, i_{l+1}), (i_l, i_{k+1}), then, in view of the Monge property, the total weight of the used arcs will not decrease. It is easy to see, that then the path p splits into a shorter subpath p' from i to j and a cycle c. Namely, we may assume without any loss of generality, that $k < l$, i.e. $k + 1 \leq l$. Then the path p can be expressed in the form of a concatenation of subpaths and arcs as follows $p = s^{(1)}(i_k, i_{k+1})s^{(2)}(i_l, i_{l+1})s^{(3)}$. Some of the subpaths $s^{(1)}, s^{(2)}, s^{(3)}$ may be of length zero, e.g. $|s^{(2)}| = 0$, if $k + 1 = l$. By the above mentioned substitution of arcs, we get a subpath $p' = s^{(1)}(i_k, i_{l+1})s^{(3)}$ and a cycle $c = (i_l, i_{k+1})s^{(2)}$. As the cycle-weight $w(c)$ is non-positive, we have $w(p') \geq w(p') + w(c) \geq w(p)$.

Thus, in a search for the maximal weight of an elementary path from i to j, the non-spiral paths may be left out of consideration. By the remark following Definition 3.1, every spiral path p of length > 1 is either convergent, or divergent, and every path of length 0 or 1 (one-node or one-arc path) is a spiral, which is both convergent and divergent. Therefore, it is sufficient to consider only paths $p \in \mathrm{Sc}(i, j) \cup \mathrm{Sd}(i, j)$. □

4 Computing an eigenvector

In this section we suggest an algorithm for computing one eigenvector of a given Monge matrix A over a max-plus algebra in $O(n^2)$ time.

Theorem 4.1 *There is an algorithm $\mathcal{A}$ which, for a given Monge matrix $A \in G_n$ over a max-plus algebra G, computes an eigenvector of A in $O(n^2)$ time.*

Remark 4.1 Theorem 4.1 can also be used for any matrix which can be permuted to a Monge form by a simultaneous permutation of rows and columns. This can be tested by a modified algorithm of Deineko and Filonenko (1979). In the positive case, the algorithm finds a permutation ϕ such that the permuted matrix A_ϕ is Monge. The simultaneous permutation on rows and columns does not change the eigenvalue $\lambda(A) = \lambda(A_\phi)$. Moreover, the inverse permutation ϕ^{-1} transforms any eigenvector of A_ϕ into an eigenvector of A.

Proof As we have mentioned in section 2, it is sufficient to compute the eigenvalue $\lambda(A)$, then to take the matrix $A_\lambda = -\lambda(A) \otimes A$, find a diagonal element in $\Delta(A_\lambda)$ with weight $\delta_{jj} = 0$ (such an element always exists) and compute the j-th column of the metric matrix $\Delta(A_\lambda)$. Any column vector x computed in this way is an eigenvector of both A_λ and A.

For computing the jth column of the metric matrix $\Delta(A_\lambda)$, the algorithm $\mathcal{A}$ uses the formula given in Theorem 3.2.

It was proved in Gavalec and Plavka (2003) that the eigenvalue $\lambda(A)$ and the matrix A_λ can be computed in $O(n^2)$ time. Moreover, it was shown that all the diagonal elements of the metric matrix $\Delta(A_\lambda)$, which have the weight $\delta_{jj} = 0$, are found during the computation of $\lambda(A)$.

In the remaining part of the proof we shall write A instead of A_λ, i.e. we assume that $\lambda(A) = 0$. It follows from the first paragraph of the proof, that this can be done without any loss of generality. Further, we assume that an index $j \in N$ with $\delta_{jj} = 0$ has been obtained. The algorithm $\mathcal{A}$ is based on a dynamic programming approach. It consists of two parts: the algorithm $\mathcal{A}_c$ computes the maximal weights of all convergent spiral paths from i to j, for all $i \in N$ and the algorithm $\mathcal{A}_d$ does the same for the divergent spiral paths.

The maximal weights of convergent spirals are computed in the state variables $u(i)$, $i \in N$, with the notation $u^{(k)}(i)$ for the value of $u(i)$ in the kth run of the recursion cycle, $k \in \{0, 1, \ldots, n\}$. Similarly, we use state variables $v(i)$ and notation $v^{(k)}(i)$ for the maximal weights of divergent spirals. Let us denote $M_k := \langle j - k, j + k \rangle \cap N$. We remark that $\langle j - k, j + k \rangle$ is the notation for the closed interval with the lower bound $j - k$ and the upper bound $j + k$. This notation is used below as well. The recursion rules in Part I assure that the algorithm $\mathcal{A}_c$

computes, for $i \in M_k$, the values

$$u^{(k)}(i) = \max\{\, w(p);\ \ p \in \mathrm{Sc}(i, j)\,\} \tag{1}$$

and, for $i \in N - M_k$, the values

$$u^{(k)}(i) = \max\{\, w(p);\quad p \in \mathrm{Sc}(i, j), \text{ the first arc in } p \text{ is ending in } M_k\,\} \tag{2}$$

Similarly, the recursion rules in Part II assure that the algorithm $\mathcal{A}_d$ computes, for $i \in M_k$, the values

$$v^{(k)}(i) = \max\{\, w(p);\ \ p \in \mathrm{Sd}(i, j)\,\} \tag{3}$$

and, for $i \in N - M_k$, the values

$$v^{(k)}(i) = \max\{\, w(p);\qquad p \in \mathrm{Sd}(i, j), \text{ the first arc in } p \text{ is ending in } M_k\,\} \tag{4}$$

We shall prove later that the recursive algorithm described below works correctly. As a consequence, the values $\max\big(u^{(k)}(i), v^{(k)}(i)\big)$, $i \in N$ for any k, $k \geq n$, give the j-th column of the metric matrix $\Delta(A_\lambda)$, i.e. an eigenvector of A_λ.

Part I – algorithm $\mathcal{A}_c$

1. Initialization ($k = 0$): $u(j) := 0$ and $u(i) := a_{ij}$ for every $i \in N$, $i \neq j$.

2. For $k = 1, 2, 3, \ldots,$ the steps 3 – 5 below are repeated until $j - k \leq 1$, or $j + k \geq n$.

3. Set $u^+ := u(j + k + 1)$, $u^- := u(j - k - 1)$,

$$u(j + k + 1) := \max\big(u^+,\ a_{j+k+1,\,j-k-1} + u^-\big)$$
$$u(j - k - 1) := \max\big(u^-,\ a_{j-k-1,\,j+k+1} + u^+\big)$$

4. For $i = j + k + 2, j + k + 3, \ldots, n,$ set $u^\star := u(i)$,

$$u(i) := \max\big(u^\star,\ a_{i,\,j-k-1} + u(j - k - 1)\big)$$
$$u(j - k - 1) := \max\big(u(j - k - 1),\ a_{j-k-1,\,i} + u^\star\big)$$

5. For $i = j - k - 2, j - k - 3, \ldots, 1,$ set $u^\star := u(i)$,

$$u(i) := \max\big(u^\star,\ a_{i,\,j+k+1} + u(j + k + 1)\big)$$
$$u(j + k + 1) := \max\big(u(j + k + 1),\ a_{j+k+1,\,i} + u^\star\big)$$

Part II – algorithm $\mathcal{A}_d$

1. Initialization ($k = 0$): $v(j) := 0$ and $v(i) := a_{ij}$ for every $i \in N$, $i \neq j$.

2. For $k = 1, 2, 3, \ldots,$ the steps 3 – 8 below are repeated until $j - k \leq 1$ and $j + k \geq n$.

3. If $j-k \leq 1$, then steps 4 – 5 are omitted.

4. Set $v(j-k-1) := \max\big(v(j-k-1),\ a_{j-k-1,1} + v(1)\big)$.

5. For $i = 2, 3, \ldots, j-k-2$, set $v^\star := v(i)$,

$$v(i) := \max\big(v^\star,\ a_{i,j-k-1} + v(j-k-1)\big)$$
$$v(j-k-1) := \max\big(v(j-k-1),\ a_{j-k-1,i} + v^\star\big)$$

6. If $j+k \geq n$, then steps 7 – 8 are omitted.

7. Set $v(j+k+1) := \max\big(v(j+k+1),\ a_{j+k+1,n} + v(n)\big)$.

8. For $i = n-1, n-2, \ldots, j+k+2$, set $v^\star := v(i)$,

$$v(i) := \max\big(v^\star,\ a_{i,j+k+1} + v(j+k+1)\big)$$
$$v(j+k+1) := \max\big(v(j+k+1),\ a_{j+k+1,i} + v^\star\big)$$

It is easy to see that the computational complexity of both parts of the algorithm $\mathcal{A}$ is $O(n^2)$.

In the rest of the proof, we show that the algorithms $\mathcal{A}_c$, $\mathcal{A}_d$ work properly. In other words, we shall prove that the conditions (1), (2) are preserved by the algorithm $\mathcal{A}_c$ and the conditions (3), (4) are preserved by $\mathcal{A}_d$.

At step 1, both algorithms do, in principle, the same. We have $k = 0$ and $M_0 = \{j\}$. The equation (1) is satisfied by the output of the algorithm $\mathcal{A}_c$ in the first run with $k = 0$, because then the only elementary path (and also the only convergent spiral) from j to j is the zero-length path consisting of a single node j. The value of this path is equal to 0 (sum of the empty set of its arc-values). The equation (2) is also satisfied, because for every $i \in N - M_0$, i.e. for $i \neq j$, the only elementary path from i to j, with its first arc ending in M_0, is the one-arc path (i, j), the weight of which is equal to a_{ij}. It can be verified in a similar way that the equations (3), (4) are fulfilled by the output values $v^{(0)}(i)$, $i \in N$, given by the algorithm $\mathcal{A}_d$.

At further steps, the work of algorithms in Parts I and II is formally different, but the main idea remains the same. We shall now analyze the run of $\mathcal{A}_c$ in Part I in more detail. The condition tested in step 2 indicates when the computation of the output vector u is finished. If $j-k \leq 1$, then every node $i \in N$ with $i \leq j$ belongs to M_k (and its value $u^{(k)}(i)$ is final, by the equation (1)). Therefore, for every $i \in N - M_k$, the inequality $j+k < i$ must hold, and in every convergent spiral $p = (i_0, i_1, \ldots, i_r)$ from $i = i_0$ to $j = i_r$, the first arc (i_0, i_1) ends in a node i_1 satisfying $1 \leq i_1 \leq j$, i.e. in M_k. Thus, in view of equation (2), the value $u^{(k)}(i)$ is final, and the computation stops. A similar argument is used for the condition $j+k \geq n$.

Step 3 is entered only if $1 < j-k$ and $j+k < n$. Then the nodes $j+k+1$ and $j-k-1$ will be prepared for being added to M_k. For this purpose, the old values $u^{(k)}(j+k+1)$, $u^{(k)}(j-k-1)$ are stored in temporary variables u^+, u^-

and new values $u^{(k+1)}(j-k-1)$, $u^{(k+1)}(j+k+1)$ are computed in step 3 and later repeatedly updated by the recursions in steps 4 and 5. The recursions use a temporary variable $u^\star$ to store the old values $u^{(k)}(i)$ while computing the new values $u^{(k+1)}(i)$ for $i > j+k+1$ and for $i < j-k-1$ and while updating the above mentioned values for $i = j+k+1$ and $i = j-k-1$. The formulas used in steps 3 – 5 easily follow from equations (1), (2) and from Definition 3.1. Namely, before we add $j+k+1$ and $j-k-1$ to M_k in the process of creating M_{k+1}, we have to ensure that equation (1) is satisfied for $i = j+k+1$ and for $j-k-1$ (from the previous run of $\mathcal{A}_c$ we only know that $j+k+1$ and $j-k-1$ fulfill equation (2)). The value $u^{(k)}(j+k+1)$ must be modified by considering the weights $w(p)$ for all paths $p \in \mathrm{Sc}(j+k+1, j)$ the first arc of which is not ending in M_k. Hence the first arc must be ending either in $j-k-1$, or in some vertex $i = j-k-2, j-k-3, \ldots, 1$ (in view of Definition 3.1, further arcs of the considered closed spiral p lie in M_k). The first case is considered in step 3 and the second case in step 5. The value $u^{(k)}(j+k+1)$ is processed analogously in steps 3 and 4.

When the computation in steps 3 – 5 is complete, then equations (1) and (2) hold for all $i \in M_{k+1}$ ($i \in N - M_{k+1}$). Thus, the recursion continues with the value of k increased by 1.

We have shown that the algorithm $\mathcal{A}_c$ in Part I gives the desired result. The correctness proof of $\mathcal{A}_d$ in Part II works quite analogously with slight changes implied by the difference between the concepts of convergent and divergent spirals. □

References

Aizenshtat VS, Kravchuk DN (1968) Algorithms for finding extreme of a linear form on the set of all cycles in special cases. Dokl. Akad. Nauk BSSR 12:401–404 (in Russian)

Aizenshtat VS, Maksimovich EP (1978) Special cases of traveling salesman problems (in Russian). Kibernetika 4:80–83

Aizenshtat VS, Maksimovich EP (1979) Special cases of traveling salesman problems (English translation). Cybernetics 14:565–569

Burkard RE, Deineko VG, van Dal R, van der Veen J, Woeginger GJ (1998) Well-solvable special cases of the TSP: a survey. SIAM Rev 40 (1998), 496–546

Burkard RE, Klinz B, Rudolf R (1996) Perspectives of Monge properties in Optimization, Discrete Appl. Math. 70:95–161

Cuninghame-Green RA (1979) Minimax Algebra, lecture notes in economics and mathematical Systems, vol 166, Springer, Berlin Heidelberg New York

Deineko VG, Filonenko VL (1979) On the reconstruction of specially structured matrices, Aktualnyje Problemy EVM, programmirovanije (Russian). Dnepropetrovsk, DGU, 1979

Gavalec M, Plavka J (2003) An $O(n^2)$ algorithm for maximum cycle mean of Monge matrices in max-algebra. Discrete Appl Math 127:651–656

Karp RM (1978) A characterization of the minimum cycle mean in a digraph. Discrete Math 23:309–311

Lawler EL (1976) Combinatorial optimization: networks and matroids. Holt, Rinehart and Winston, New York

Wilber R (1988) The concave least-weight subsequence problem revisited. J Algorithms 9: 418–425

Zimmermann U (1981) Linear and combinatorial Optimization in Ordered algebraic structures. North Holland, Amsterdam