MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

Combinatorial Relaxation Algorithm for the Entire Sequence of the Maximum Degree of Minors

Shun SATO

(Communicated by Takayasu MATSUO)

METR 2014–23 August 2014

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.keisu.t.u-tokyo.ac.jp/research/techrep/index.html

Combinatorial Relaxation Algorithm for the Entire Sequence of the Maximum Degree of Minors

Shun Sato

Department of Mathematical Informatics
Graduate School of Information Science and Technology
The University of Tokyo
shun_sato@mist.i.u-tokyo.ac.jp

August, 2014

Abstract

This paper presents an efficient "combinatorial relaxation" algorithm for computing the entire sequence of the maximum degree of minors, whereas the previous algorithms find them separately for a specified order k. The efficiency of the algorithm is based on the discrete concavity related to valuated bimatroids.

1 Introduction

Let $A(x)$ be a rational matrix over a field F, i.e., each entry $A_{ij}(x)$ is a rational function in x over F. Typically, F is the real number field $\mathbb{R}$ or the rational number field $\mathbb{Q}$. We define the degree of a rational function $f(x) = p(x)/q(x)$ by $\deg f = \deg p - \deg q$. For $f(x) = 0$ we put $\deg f = -\infty$ by convention. The maximum degree of the minors of order k is defined as follows:

$$\delta_k(A) := \max\left\{\deg\det A[I,J] \mid I \subseteq \mathrm{Row}(A), J \subseteq \mathrm{Col}(A), |I| = |J| = k\right\}, \tag{1}$$

where $\mathrm{Row}(A)$ is the row-set of A, $\mathrm{Col}(A)$ is the column-set of A, and $A[I,J]$ denotes the submatrix of A with the row-set I and the column-set J. We set $\delta_0(A) = 0$.

Finding the maximum degree of minors is a fundamental problem in engineering. For example, the Smith–McMillan form at infinity and the Kronecker form are closely related to the maximum degree of minors. The former is a normal form of rational matrices often used in control theory (e.g., Commault–Dion [1]), and the latter is a normal form of matrix pencils frequently employed in analyzing DAEs (e.g., Hairer–Wanner [5]). These

normal forms are determined from the entire sequence of the maximum degree of minors.

The aim of this paper is to propose an efficient combinatorial relaxation type algorithm for finding the entire sequence of the maximum degree of minors in rational matrices. We are based on the approach of Murota [9] which gives a method for computing the maximum degree for a specified order. Obviously, we can compute the entire sequence by repeatedly applying the existing algorithm [9] (or its variant [6]). But it can be done much more efficiently by fully utilizing the discrete concavity inherent in this problem. In the present paper, we propose such an algorithm. We also use some techniques for constructing the proposed algorithm and the method to analyze the time complexity devised in Iwata–Takamatsu [7], where mixed polynomial matrices were considered (instead of rational matrices).

All the existing algorithms mentioned above [6, 7, 9] adopted a general framework of "combinatorial relaxation" due to Murota [8]. This framework provides us the practical efficiency of the graph theoretic approach and the accuracy of the numerical methods. The outline of the algorithm for $\delta_k(A)$ for a specified k reads as follows:

Step 1: Solve a weighted bipartite matching problem (as a combinatorial relaxation problem) associated with the rational matrix $A(x)$.

Step 2: Test whether the solution of the combinatorial problem is equal to $\delta_k(A)$ or not. This can be done without knowing $\delta_k(A)$ itself.

Step 3: In case they are not equal, modify $A(x)$ and go back to Step 1.

The heart of the "combinatorial relaxation" lies in Step 3. Here we introduce an auxiliary variable $\hat{\delta}_k(A)$, an estimate of $\delta_k(A)$, as the maximum weight of a matching of size k computed in Step 1. The auxiliary variable $\hat{\delta}_k(A)$ satisfies the following properties:

1. For all k, $\delta_k(A) \leq \hat{\delta}_k(A)$ holds (see Fig. 1);
2. There exists $A'(x)$ such that $\delta_k(A) = \delta_k(A') = \hat{\delta}_k(A')$ holds.

Thanks to the former property, in the modification of the rational matrix $A(x)$ in Step 3, we only have to consider to decrease $\hat{\delta}_k(A)$. The latter property guarantees that the estimated value can in fact achieve the desired value.

This paper is organized as follows: Preliminaries such as the definition of the maximum degree of minors are given in Section 2. In Section 3, we show a primitive algorithm for the entire sequence of the maximum degree of minors. The proposed algorithm is described in Section 4. In Section 5, we illustrate the algorithm through a simple example. Then, we theoretically analyze the time complexity of the proposed algorithm in Section 6.

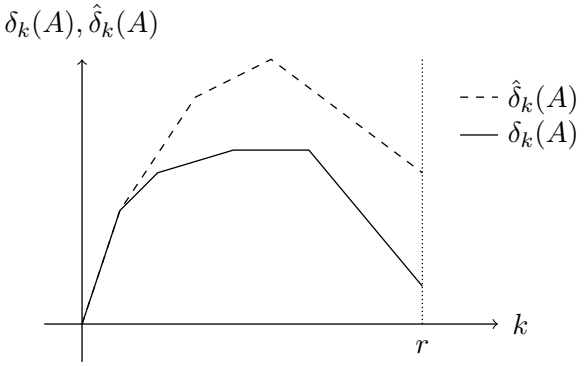

Figure 1: The relation between $\delta_k(A)$'s and $\hat{\delta}_k(A)$'s

2 Preliminaries

2.1 Biproper Equivalence Transformation

We call a rational function $f(x)$ proper if $\deg f \leq 0$, and strictly proper if $\deg f < 0$. A rational matrix is said to be proper if each entry of it is proper. Similarly, a rational matrix is said to be strictly proper if each entry of it is strictly proper. We call a square proper rational matrix $U(x)$ biproper if it is invertible and $U(x)^{-1}$ is proper. For a proper rational matrix $U(x)$, the following two conditions are equivalent (see, e.g., [12]):

1. $U(x)$ is a biproper rational matrix;
2. $\det U(x)$ is a nonzero constant.

A transformation of the form: $U(x)A(x)V(x)$ is called a biproper equivalence transformation if $U(x)$ and $V(x)$ are biproper rational matrices. The values $\delta_k(A)$'s are invariant under biproper equivalence transformations.

2.2 Valuated Bimatroids

The concept of valuated bimatroid was defined by Murota [11] as a variant of valuated matroid (Dress–Wenzel [2, 3]). A valuated bimatroid is a triple (R, C, w), where R and C are disjoint finite sets and $w : 2^R \times 2^C \to \mathbb{R} \cup \{-\infty\}$ is a map that satisfies a certain exchange axiom (see, e.g., [11, 12]). We define $S \subseteq 2^R \times 2^C$, $S_k \subseteq S$ and $\delta_k \in \mathbb{R}$ as follows:

$$\begin{aligned} S &= \{(I, J) \mid |I| = |J|,\ I \subseteq R,\ J \subseteq C\}, \\ S_k &= \{(I, J) \mid |I| = |J| = k,\ I \subseteq R,\ J \subseteq C\}, \\ \delta_k &= \max\{w(I, J) \mid (I, J) \in S_k\}. \end{aligned}$$

Proposition 1 (Murota [11])**.** *The following inequality holds*:

$$\delta_{k-1} + \delta_{k+1} \leq 2\delta_k \qquad (k = 1, 2, \ldots, r-1). \tag{2}$$

The set $\mathcal{M}_k$ of the maximizers of w is defined as follows:

$$\mathcal{M}_k = \{(I, J) \in S_k \mid w(I, J) = \delta_k\}.$$

Proposition 2 (Murota [11]). *For any $(I_k, J_k) \in \mathcal{M}_k$ with $k = 1, \ldots, r-1$, there exist $(I_l, J_l) \in \mathcal{M}_l$ $(0 \leq l \leq r, l \neq k)$ such that $(\emptyset =) I_0 \subseteq I_1 \subseteq \cdots \subseteq I_{k-1} \subseteq I_k \subseteq I_{k+1} \subseteq \cdots \subseteq I_r$ and $(\emptyset =) J_0 \subseteq J_1 \subseteq \cdots \subseteq J_{k-1} \subseteq J_k \subseteq J_{k+1} \subseteq \cdots \subseteq J_r$.*

It is known that

$$w(I, J) := \deg \det A[I, J] \tag{3}$$

defines a valuated bimatroid [11, 12]. Therefore, $\{\delta_k(A)\}_{k=0}^r$ is a concave sequence by Proposition 1. Proposition 2 means that the maximizers of w have a nesting structure.

3 Existing Combinatorial Relaxation Algorithm

The description of this section is based on [9, 10].

Let $A(x)$ be a rational matrix with the row set R and the column set C. We define $G(A) = (R \cup C, E(A), c)$ as a bipartite graph associated with $A(x)$ where the arc set $E(A)$ and the weight $c : E(A) \to \mathbb{Z}$ are defined as follows:

$$E(A) = \{(i, j) \mid i \in R,\ j \in C,\ A_{ij}(x) \neq 0\}, \tag{4}$$

$$c(i, j) = \deg A_{ij} \quad ((i, j) \in E(A)). \tag{5}$$

Let $\hat{\delta}_k(A)$ be the weight of a maximum weight matching of size k in $G(A)$, i.e.,

$$\hat{\delta}_k(A) = \max\{c(M) \mid M : \text{a matching in } G(A),\ |M| = k\}, \tag{6}$$

where $c(M) = \sum_{(i,j)\in M} c(i, j)$. If there is no matching of size k in $G(A)$, we define $\hat{\delta}_k(A) = -\infty$.

Then,

$$\hat{w}(I, J) := \max\{c(M) \mid M \subseteq E(A), \partial M = I \cup J\}\ (I \subseteq R, J \subseteq C),$$

as well as $w(I, J)$ defined in (3), defines a valuated bimatroid. Therefore, Propositions 1 and 2 hold for $\hat{w}$.

These $\hat{\delta}_k(A)$'s play the role of a combinatorial relaxation of $\delta_k(A)$'s, and the following inequality holds ([9, Theorem 3]):

$$\delta_k(A) \leq \hat{\delta}_k(A). \tag{7}$$

We can test whether $\delta_k(A) = \hat{\delta}_k(A)$ holds or not without knowing $\delta_k(A)$ itself by utilizing the duality of linear programming. The primal-dual pair of linear programming problems associated with weighted bipartite matching problem discussed above are given as follows:

$$\begin{aligned}
\mathrm{PLP}(A,k):\ \text{maximize} \quad & \sum_{e\in E} c_e\xi_e, \\
\text{subject to} \quad & \sum_{\partial e\ni i} \xi_e \le 1 \quad (i\in V), \\
& \sum_{e\in E} \xi_e = k, \\
& \xi_e \ge 0 \quad (e\in E(A));
\end{aligned} \tag{8}$$

$$\begin{aligned}
\mathrm{DLP}(A,k):\ \text{minimize} \quad & \sum_{i\in R} p_i + \sum_{j\in C} q_j + kt \quad (=:\pi(p,q,t)), \\
\text{subject to} \quad & p_i + q_j + t \ge c_{ij} \quad ((i,j)\in E(A)), \\
& p_i \ge 0 \quad (i\in R), \\
& q_j \ge 0 \quad (j\in C).
\end{aligned} \tag{9}$$

Here, ∂e is defined as $\partial e = \{i \in V \mid \exists j \in V, (i,j) = e \text{ or } (j,i) = e\}$. $\mathrm{PLP}(A,k)$ and $\mathrm{DLP}(A,k)$ have an integral optimal solution, and their optimal values are equal to $\hat{\delta}_k(A)$.

We define the active rows $I^* \subseteq R$ and the active columns $J^* \subseteq C$ as follows:

$$I^* = I^*(p) = \{i \in R \mid p_i > 0\}, \tag{10}$$

$$J^* = J^*(q) = \{j \in C \mid q_j > 0\}. \tag{11}$$

Moreover, we define the tight coefficient matrix A^* of A as follows:

$$A^*_{ij} = \lim_{x\to\infty} x^{-p_i-q_j-t} A_{ij}(x). \tag{12}$$

Note that the right-hand side of (12) is a bounded constant because of $p_i + q_j + t \ge c_{ij} = \deg A_{ij}$, and that computing the rank of A^* is relatively easy.

Proposition 3 enables us to test whether $\delta_k(A) = \hat{\delta}_k(A)$ (which we call "tight") or not without knowing $\delta_k(A)$.

Proposition 3 ([9])**.** *Let (p,q,t) be an optimal dual solution, I^* and J^* be the active rows and columns defined by* (10) *and* (11)*, and A^* be the tight coefficient matrix defined by* (12)*. Then, $\delta_k(A) = \hat{\delta}_k(A)$ if and only if the following four conditions are satisfied*:

(r1) $\operatorname{rank} A^*[R,C] \ge k$,

(r2) $\operatorname{rank} A^*[I^*,C] = |I^*|$,

(r3) $\operatorname{rank} A^*[R, J^*] = |J^*|$,

(r4) $\operatorname{rank} A^*[I^*, J^*] \geq |I^*| + |J^*| - k$.

Let $A(x)$ be an $m \times n$ Laurent polynomial matrix. A rational function is said to be a Laurent polynomial function if there exists an integer N such that $x^N f(x)$ is a polynomial function. We define $d_{\max}$ and $d_{\min}$ as follows:

$$d_{\max} = d_{\max}(A) = \max\{\deg A_{ij} \mid i \in R,\ j \in C\},$$
$$d_{\min} = d_{\min}(A) = \min\{\operatorname{ord} A_{ij} \mid i \in R,\ j \in C\},$$

where $\operatorname{ord} f$ is defined as follows:

$$\operatorname{ord} f := -\min\{N \in \mathbb{Z} \mid x^N f(x) \text{ is a polynomial}\}.$$

To compute the entire sequence of the maximum degree of minors $\{\delta_k(A)\}_{k=1}^r$ ($r = \operatorname{rank} A$), we can simply execute the existing algorithms [6, 9] r times as follows:

Outline of the Primitive Algorithm for Computing $\{\delta_k(A)\}_{k=1}^r$

Step 0: Set $k := 1$.

Step 1: Find an optimal solution (p, q, t) of DLP(A, k).

Step 2: Test whether $\delta_k(A) = \hat{\delta}_k(A)$ or not using (p, q, t) (Proposition 3). If equality holds, go to Step 4.

Step 3: Modify $A(x)$ to $A'(x)$, and go back to Step 1.

Step 4: Output $\delta_k(A)$, update $k := k + 1$ and go back to Step 1.

In general, we do not know the value of r at the starting point of the algorithm. But we can stop the algorithm as follows: If the rank of the matrix $A(x)$ is equal to $\min\{m, n\}$, the rule of the termination is obvious. Otherwise, we use the inequality $\delta_k(A) \geq k d_{\min}$ ($k \leq r$) to stop the algorithm in Step 1: If $\hat{\delta}_k(A) < k d_{\min}$ holds, then stop with $r := k - 1$.

By virtue of the significant properties of $\delta_k(A)$'s and $\hat{\delta}_k(A)$'s shown in Propositions 1 and 2, the efficiency of this algorithm can be improved, which is the main contribution of this paper.

4 Proposed Algorithm

In this section, we propose an efficient algorithm to compute the entire sequence of the maximum degree of minors. This section is organized as follows: In Section 4.1, we show two key theorems for an efficient computation of the entire sequence of the maximum degree of minors. Section 4.2

provides the outline of the proposed algorithm. The details of each step of the algorithm are developed in Section 4.3–4.6. Then, we show the complete description of the algorithm in Section 4.7. Finally, we give the proofs of the two key theorems (stated in Section 4.1) in Section 4.8.

4.1 Theorems to Improve Efficiency

We show two theorems which come from the significant properties of $\delta_k(A)$'s and $\hat{\delta}_k(A)$'s.

Theorem 1. *Suppose that* $\delta_k(A) = \hat{\delta}_k(A)$ *holds and* (p, q, t) *is a common optimal dual solution of* $\mathrm{DLP}(A, k)$ *and* $\mathrm{DLP}(A, k+1)$*. Let* A^* *be the tight coefficient matrix defined by* (12)*. Then,* $\delta_{k+1}(A) = \hat{\delta}_{k+1}(A) = \delta_k(A) + t$ *if and only if* $\operatorname{rank} A^* \geq k+1$.

Theorem 1 allows us to check if $\delta_{k+1}(A) = \hat{\delta}_{k+1}(A)$ by computing $\operatorname{rank} A^*$ only. This value, $r^* := \operatorname{rank} A^*$, is always greater than k. Furthermore, when $r^* > k+1$, thanks to the next theorem, we obtain all of $\delta_{k+1}(A), \ldots, \delta_{r^*}(A)$ at the same time, i.e., we can skip the computation of $\delta_{k+2}(A), \ldots, \delta_{r^*}(A)$.

Theorem 2. *Let* (p, q, t) *be a common optimal dual solution of* $\mathrm{DLP}(A, k)$ *and* $\mathrm{DLP}(A, k+1)$*. Let* A^* *be the tight coefficient matrix defined by* (12)*. Suppose that* $\delta_k(A) = \hat{\delta}_k(A)$ *holds and* $k < r^*$ *holds. Then, the following equality holds*:

$$\delta_l(A) = \hat{\delta}_l(A) = \delta_k(A) + (l-k)t \quad (l = k+1, \ldots, r^*). \tag{13}$$

Moreover, $\delta_{r^*+1}(A) < \delta_k(A) + (r^* + 1 - k)t$ *holds.*

4.2 The Outline of the Proposed Algorithm

The outline of the proposed algorithm is as follows.

Outline of the Proposed Algorithm for Computing $\{\delta_k(A)\}_{k=1}^r$

Step 0: Compute $\delta_1(A)$ and set $k := 1$.

Step 1: Find a common optimal dual solution (p, q, t) of $\mathrm{DLP}(A, k)$ and $\mathrm{DLP}(A, k+1)$.

Step 2: Test for the tightness, i.e., whether $\delta_{k+1}(A) = \hat{\delta}_{k+1}(A)$ or not, by using (p, q, t) and the tight coefficient matrix A^* (Theorem 1). If the equality holds, go to Step 4. Otherwise, go to Step 3.

Step 3: Modify $A(x)$ to $A'(x)$ such that $\hat{\delta}_{k+1}(A') < \hat{\delta}_{k+1}(A)$ and $\delta_{k+1}(A') = \hat{\delta}_{k+1}(A)$ hold, and go back to Step 1.

Step 4: Output $\delta_{k+1}(A), \ldots, \delta_{r^*}(A)$ (Theorem 2), update $k := r^*$ and go back to Step 1.

The detail of the algorithm is described in Section 4.7. The termination of the algorithm will also be described there.

For Step 0, the initialization, we look for a pair $(i, j) \in R \times C$ that is a maximizer of $\deg A_{ij}$. Then we have $I_1^* = \{i\}$, $J_1^* = \{j\}$, $M_1 = \{(i, j)\}$. The other steps are discussed in Section 4.3–4.6.

4.3 Step 1: Construction of an Optimal Dual Solution

At every starting point of Step 1, the following conditions are satisfied as a result of the last loop, or the initialization:

1. $\delta_l(A) = \hat{\delta}_l(A) \quad (l = 1, 2, \ldots, k)$;
2. $\deg \det A[I_k^*, J_k^*] = \sum_{(i,j)\in M_k} \deg A_{ij} = \delta_k(A)$;
3. $I_k^* = \partial^+ M_k$, $J_k^* = \partial^- M_k$.

Here, $\partial^+ M$ and $\partial^- M$ are defined as follows:

$$\partial^+ M = \{i \in R \mid \exists j \in C \text{ s.t. } (i, j) \in M\},$$
$$\partial^- M = \{j \in C \mid \exists i \in R \text{ s.t. } (i, j) \in M\}.$$

In actual computation, we do not need to store I_k^* and J_k^* because they can be easily constructed from M_k. They are explicitly introduced here for a better presentation of the proposed algorithm.

An optimal dual solution can be constructed as follows. As stated in Iwata–Takamatsu [7], we can obtain it by solving a shortest path problem on an auxiliary graph. Consider the auxiliary graph $G_{M_k} = (V, E, \gamma)$ associated with M_k, where

$$V = R \cup C \cup \{u^+\} \cup \{u^-\},$$
$$E = E(A) \cup M^\circ \cup W^+ \cup W^-.$$

Here, u^+ and u^- are new vertices and

$$E(A) = \{(i, j) \mid i \in R,\ j \in C,\ A_{ij}(x) \neq 0\},$$
$$M^\circ = \{(j, i) \mid (i, j) \in M_k\},$$
$$W^+ = \left\{(u^+, i) \mid i \in R \setminus I_k^*\right\},$$
$$W^- = \left\{(j, u^-) \mid j \in C \setminus J_k^*\right\} \cup \{(u^-, j) \mid j \in C\}.$$

We define the arc length $\gamma : E \to \mathbb{Z}$ by

$$\gamma(i, j) = \begin{cases} -\deg A_{ij} & ((i, j) \in E(A)), \\ \deg A_{ji} & ((i, j) \in M^\circ), \\ 0 & ((i, j) \in W^+ \cup W^-). \end{cases}$$

Let $\varphi(v)$ be the length of a shortest path from u^+ to $v \in V$ with arc length γ in G_{M_k}. If there is no path from u^+ to $v \in V$, we set $\varphi(v) = +\infty$. We define (p, q, t) as follows:

$$\begin{cases} p_i = \varphi(i) & (i \in R), \\ q_j = \varphi(u^-) - \varphi(j) & (j \in C), \\ t = -\varphi(u^-). \end{cases} \tag{14}$$

If there is a path from u^+ to u^-, (p, q, t) is an optimal dual solution of $\mathrm{DLP}(A, k)$ as stated in Lemma 1 below. Otherwise, i.e., if there is no path from u^+ to u^-, $k = \operatorname{rank} A(x)$ holds since $k = \text{term-rank}\, A(x) \geq \operatorname{rank} A(x) \geq k$. Here, term-rank A is defined as follows (see, e.g., [12]):

$$\text{term-rank}\, A = \max\{|M| \mid M \text{ is a matching on } G(A)\}.$$

Lemma 1 (cf. [7, Lemma 3])**.** *Let M_k be a maximum weight matching in $G(A)$ with $|M_k| = k$. The triple (p, q, t) defined by* (14) *with respect to G_{M_k} is an optimal dual solution of* $\mathrm{DLP}(A, k)$*. Furthermore, if the weight function is integer valued, then (p, q, t) is an integral solution.*

In this step, we can adopt "reweighting" using the newest optimal dual variable [4, 13].

4.4 Step 2: Test for Tightness

Lemma 2. *Under the condition of Lemma 1, (p, q, t) is an optimal dual solution of* $\mathrm{DLP}(A, k+1)$*.*

Proof. Since the triple (p, q, t) is obviously a feasible solution of $\mathrm{DLP}(A, k+1)$, it is sufficient to show the optimality. The objective function of $\mathrm{DLP}(A, k+1)$ is expressed as

$$\pi_{k+1}(p, q, t) = \sum_{i \in R} p_i + \sum_{j \in C} q_j + (k+1)t = \pi_k(p, q, t) + t.$$

On the other hand, since the dual variable $t = -\varphi(u^-)$ is defined as the maximum length of augmenting paths, the optimal value of $\mathrm{DLP}(A, k+1)$ is larger than the optimal value of $\mathrm{DLP}(A, k)$ by t. Therefore, (p, q, t) is an optimal dual solution of $\mathrm{DLP}(A, k+1)$. □

Lemma 2 means $\hat{\delta}_{k+1}(A) = \delta_k(A) + t$. Since $\hat{\delta}_{k+1}(A) \geq \delta_{k+1}(A) \geq (k+1)d_{\min}$ holds for all integer $k < r$, $\hat{\delta}_{k+1}(A) = \delta_k(A) + t < (k+1)d_{\min}$ implies $k = r$. Therefore, if $t < (k+1)d_{\min} - \delta_k(A)$ holds, we can set $\operatorname{rank} A = k$ and halt.

Since Lemmas 1 and 2 mean that (p, q, t) defined in (14) is a common optimal dual solution of $\mathrm{DLP}(A, k)$ and $\mathrm{DLP}(A, k+1)$, we can adopt Theorem 1 instead of Proposition 3 to test for the tightness, i.e., $\delta_{k+1}(A) = \hat{\delta}_{k+1}(A)$ holds if and only if $\operatorname{rank} A^* > k$.

4.5 Step 3: Matrix Modification

Theorem 1 allows us to focus on the single condition $\operatorname{rank} A^* = k$ whereas the existing algorithms deal with the four cases. When the execution of Step 3 starts, the following conditions are satisfied:

1. $\operatorname{rank} A^* = k$,
2. $\operatorname{rank} A^*[I_l^*, J_l^*] = l \quad (l = 1, 2, \ldots, k)$.

Therefore, there exists a unique $m \times m$ constant matrix U such that

(U1) $U[I_k^*, I_k^*]$ and $U[R \setminus I_k^*, R \setminus I_k^*]$ are identity matrices,

(U2) $U[I_k^*, R \setminus I_k^*] = O$,

(U3) $\text{term-rank}\, UA^* = k$.

Namely, there exist a matrix U such that

$$UA^* = \begin{bmatrix} I & O \\ \tilde{U} & I \end{bmatrix} A^* = \begin{bmatrix} A^*[I_k^*, C] \\ O \end{bmatrix} \tag{15}$$

holds (strictly speaking, with some appropriate permutation), where I denote the identity matrix of suitable order and $\tilde{U} = U[R \setminus I_k^*, I_k^*]$ is a constant matrix. From (15), $\tilde{U}$ satisfies

$$\tilde{U} A^*[I_k^*, J_k^*] + A^*[R \setminus I_k^*, J_k^*] = O. \tag{16}$$

We can obtain the constant matrix U satisfying (U1), (U2) and (U3) by solving (16); note that $A^*[I_k^*, J_k^*]$ is nonsingular.

Then we consider to modify the matrix $A(x)$ by the constant matrix U satisfying (U1), (U2) and (U3); let us define a modified matrix

$$A'(x) = \operatorname{diag}(x; p) \cdot U \cdot \operatorname{diag}(x; -p) \cdot A(x), \tag{17}$$

where $\cdot$ denotes the usual matrix multiplication.

The right-hand side of the equation (17) can be computed by executing the multiplication of the submatrices as follows:

$$A'(x) = \operatorname{diag}(x; p) \cdot \begin{bmatrix} I & O \\ \tilde{U} & I \end{bmatrix} \cdot \operatorname{diag}(x; -p) \cdot A(x) \tag{18}$$

$$= \begin{bmatrix} I & O \\ \tilde{U} \cdot \operatorname{diag}(x; -p_k^*) & I \end{bmatrix} \begin{bmatrix} A[I_k^*, C](x) \\ A[R \setminus I_k^*, C](x) \end{bmatrix}, \tag{19}$$

where $\operatorname{diag}(x; p) := \operatorname{diag}(x^{p_1}, \ldots, x^{p_m})$ and $p_k^* := (p_i \mid i \in I_k^*)$. Therefore,

$$A'[I_k^*, C](x) = A[I_k^*, C](x), \tag{20}$$

$$A'[R \setminus I_k^*, C](x) = \tilde{U} \cdot \operatorname{diag}(x : -p_k^*) \cdot A[I_k^*, C](x) + A[R \setminus I_k^*, C](x). \tag{21}$$

This modification makes sense, as stated in Theorem 3.

Theorem 3. *The matrix $A'(x)$ defined in* (17) *has the following four properties*:

1. $\delta_l(A') = \delta_l(A) \quad (l = 1, \ldots, r)$;
2. $\deg \det A'[I_k^*, J_k^*] = \delta_k(A')$;
3. $\hat{\delta}_l(A') = \delta_l(A') \quad (l = 1, \ldots, k)$;
4. $\hat{\delta}_{k+1}(A') < \hat{\delta}_{k+1}(A)$.

Proof.

1. Since $U(x) := \mathrm{diag}(x; p) \cdot U \cdot \mathrm{diag}(x : -p)$ is biproper, they are satisfied.
2. $A'[I_k^*, J_k^*](x) = A[I_k^*, J_k^*](x)$ holds from (20). Therefore,
$$\deg \det A'[I_k^*, J_k^*] = \deg \det A[I_k^*, J_k^*] = \delta_k(A) = \delta_k(A').$$
3. By Proposition 2 and the property 2 of this theorem, $\hat{\delta}_l(A[I_k^*, J_k^*]) = \hat{\delta}_l(A)$ and $\hat{\delta}_l(A'[I_k^*, J_k^*]) = \hat{\delta}_l(A')$ hold for all $l \leq k$. From these equalities and $A[I_k^*, J_k^*] = A'[I_k^*, J_k^*]$, we obtain
$$\hat{\delta}_l(A') = \hat{\delta}_l(A'[I_k^*, J_k^*]) = \hat{\delta}_l(A[I_k^*, J_k^*]) = \hat{\delta}_l(A) = \delta_l(A) = \delta_l(A').$$
4. First, we prove that an optimal dual solution (p, q, t) of $\mathrm{DLP}(A, k+1)$ is feasible in $\mathrm{DLP}(A', k+1)$. We define a rational matrix $F(x)$ as follows:
$$F(x) := x^{-t}\mathrm{diag}(x; -p) \cdot A'(x) \cdot \mathrm{diag}(x; -q).$$
By using (17), we obtain
$$\begin{aligned} F(x) &= x^{-t} U \cdot \mathrm{diag}(x; -p) \cdot A(x) \cdot \mathrm{diag}(x; -q) \\ &= U \cdot (A^* + A^{\infty}(x)), \end{aligned}$$
where $A^{\infty}(x)$ is a strictly proper rational matrix. From this, we see $\deg F_{ij} \leq 0$. Then, we obtain $p_i + q_j + t \leq \deg A'_{ij}$ for all $i \in R$ and $j \in C$ by using $\deg F_{ij} = \deg A'_{ij} - p_i - q_j - t$. Therefore, (p, q, t) is a feasible solution of $\mathrm{DLP}(A', k+1)$.

   Next, we prove that (p, q, t) is not optimal. It is sufficient to prove that there exists another feasible solution (p', q', t') such that $\pi_k(p', q', t') < \pi_k(p, q, t)$ holds. We define (p', q', t') as follows:
$$\begin{aligned} p'_i &= \begin{cases} p_i + 1 & (i \in I_k^*), \\ p_i & (i \notin I_k^*), \end{cases} \\ q'_j &= q_j \quad (j \in C), \\ t' &= t - 1. \end{aligned}$$

Clearly, $p'_i \geq 0$ and $q'_j \geq 0$ hold for all $i \in R$ and $j \in C$. We verify that $p'_i + q'_j + t \geq \deg A'_{ij}$ holds for all $i \in R$ and $j \in C$. If $i \in I^*_k$, we obtain

$$p'_i + p'_j + t = p_i + 1 + q_j + t - 1 = p_i + q_j + t \geq \deg A'_{ij}$$

by using the feasibility of (p, q, t) in $\mathrm{DLP}(A', k)$. Otherwise, i.e., when $i \notin I^*_k$, it holds $p_i + q_j + t > \deg A'_{ij}$, and accordingly,

$$p'_i + p'_j + t = p_i + q_j + t - 1 \geq \deg A'_{ij}$$

holds. Furthermore, by the definition of (p', q', t'), $\pi_k(p', q', t')$ is less than $\pi_k(p, q, t)$ by one.

□

4.6 Step 4: Outputs and Updates

Recall that the task of this step is to output $\delta_{k+1}(A), \ldots, \delta_{r^*}(A)$ in view of Theorem 2, and then we are going back to Step 1. But in order to start the process of Step 1, we need the corresponding matching M_{r^*} such that

$$\sum_{(i,j)\in M_{r^*}} \deg A_{ij} = \deg \det A[\partial^+ M_{r^*}, \partial^- M_{r^*}] = \delta_{r^*}(A)$$

holds. The key ingredient for obtaining this is the computation of $I^*_{r^*} = \partial^+ M_{r^*}$ and $J^*_{r^*} = \partial^- M_{r^*}$, which can be obtained simultaneously in the calculation of $r^* = \operatorname{rank} A^*$ by the trick below.

0. $M' := M_k$, $R' := \mathrm{Row}(A^*)$.
1. Repeat the following steps k times.
   (a) Choose $(i, j) \in M'$.
   (b) Conduct the row elimination for all rows in $R' \setminus \{i\}$ taking A^*_{ij} as the pivot.
   (c) $M' := M' \setminus \{(i, j)\}$, $R' := R' \setminus \{i\}$.
2. Execute the Gaussian elimination for $A^*[R', C]$.
3. Set $I^*_{r^*}$ and $J^*_{r^*}$ as follows:

$$I^*_{r^*} = \{i \in R \mid \exists j,\ A^*_{ij} \neq 0\},$$
$$J^*_{r^*} = \{j \in C \mid \exists i \in I^*_{r^*},\ j = \min\{j' \mid A^*_{ij} \neq 0\}\}.$$

Then, M_{r^*} is an optimal solution of the weighted bipartite matching problem on $G = (I^*_{r^*} \cup J^*_{r^*}, E, \gamma)$, where

$$E = \{(i, j) \mid i \in I^*_{r^*}, j \in J^*_{r^*}, A_{ij}(x) \neq 0\},$$
$$\gamma(i, j) = \deg A_{ij}(x) \quad ((i, j) \in E).$$

We can construct M_{r^*} efficiently by augmenting paths (we can use M_k as an initial matching).

4.7 Complete Procedure of Proposed Algorithm

Summing up the discussion above, here we show the complete procedure.

Step 0: Initialization

Choose a row i^* and a column j^* that maximize $\deg A_{ij}$. Initialize I_1^*, J_1^*, M_1 as:

$$I_1^* := \{i^*\}, \quad J_1^* := \{j^*\}, \quad M_1 := \{(i^*, j^*)\}.$$

Output $\deg A_{i^*j^*}$ as $\delta_1(A)$, and set $k := 1$. Proceed to Step 1.

Step 1: Construction of an Optimal Dual Solution

1. Construct an optimal dual solution from M_k by solving the shortest path problem on the directed graph G_{M_k}. If there is no path, set $\operatorname{rank} A := k$ and halt.
2. If $t < (k+1)d_{\min} - \delta_k(A)$, then set $\operatorname{rank} A := k$ and halt.

Step 2: Test for Tightness

1. Construct A^* from the optimal dual solution.
2. Compute $r^* := \operatorname{rank} A^*$ by the Gaussian elimination (and construct $I_{r^*}^*$, $J_{r^*}^*$ for a later use).
3. If $\operatorname{rank} A^* = k$, go to Step 3. Otherwise, go to Step 4.

Step 3: Matrix Modification

1. Solve $\tilde{U} \cdot A^*[I_k^*, J_k^*] + A^*[R \setminus I_k^*, J_k^*] = O$ to obtain $\tilde{U}$.
2. Construct $A'(x)$ as follows ($p_k^* := (p_i \mid i \in I_k^*)$):

$$A'[I_k^*, C](x) := A[I_k^*, C](x),$$
$$A'[R \setminus I_k^*, C](x) := \tilde{U} \cdot \operatorname{diag}(x; -p_k^*) \cdot A[I_k^*, C](x) + A[R \setminus I_k^*, C](x).$$

3. Go to Step 1.

Step 4: Outputs and Updates

1. Output the following:

$$\delta_l(A) := \delta_k(A) + (l-k)t \quad (l = k+1, \ldots, r^*).$$

2. If $r^* = \min\{m, n\}$, then halt.
3. Compute M_{r^*} by solving the weighted matching problem (using $I_{r^*}^*$ and $J_{r^*}^*$ obtaind in Step 2).
4. Set $k := r^*$ and go to Step 1

4.8 Proofs of Theorems 1 and 2

Proof of Theorem 1
Since $\delta_k(A) = \hat{\delta}_k(A)$ holds, the following four conditions are satisfied by Proposition 3:

(r1) $\operatorname{rank} A^*[R, C] \geq k$,

(r2) $\operatorname{rank} A^*[I^*, C] = |I^*|$,

(r3) $\operatorname{rank} A^*[R, J^*] = |J^*|$,

(r4) $\operatorname{rank} A^*[I^*, J^*] \geq |I^*| + |J^*| - k$.

On the other hand, again by Proposition 3, $\delta_{k+1}(A) = \hat{\delta}_{k+1}(A)$ holds if and only if the following four conditions are all satisfied:

(r1)′ $\operatorname{rank} A^*[R, C] \geq k + 1$,

(r2)′ $\operatorname{rank} A^*[I^*, C] = |I^*|$,

(r3)′ $\operatorname{rank} A^*[R, J^*] = |J^*|$,

(r4)′ $\operatorname{rank} A^*[I^*, J^*] \geq |I^*| + |J^*| - k - 1$.

Clearly, (r2)′, (r3)′ and (r4)′ are immediately derived from (r2), (r3) and (r4), respectively. Therefore, $\delta_{k+1}(A) = \hat{\delta}_{k+1}(A)$ if and only if $\operatorname{rank} A^* \geq k + 1$. □

Before we prove Theorem 2, we prepare the following lemma.

Lemma 3. *Let (p, q, t) be a common optimal solution of* DLP(A, k) *and* DLP$(A, k+1)$. *Let A^* be the tight coefficient matrix defined in* (12). *Suppose that $\delta_k(A) = \hat{\delta}_k(A)$ and $k < r^*(= \operatorname{rank} A^*)$ hold. Then, the following identity holds true:*

$$\delta_{r^*}(A) = \delta_k(A) + (r^* - k)t. \tag{22}$$

Proof. We start by proving that $\delta_{r^*}(A) \geq \delta_k(A) + (r^* - k)t$ holds. Since $\operatorname{rank} A^* = r^*$, there exist $I \subseteq R$, $J \subseteq C$ such that $|I| = |J| = r^*$ and $A^*[I, J]$ is nonsingular. Then, we see

$$\begin{aligned}
\delta_{r^*}(A) &\geq \deg \det A[I, J] \\
&= \deg \left\{ x^{r^* t} \cdot \det \operatorname{diag}(x; p_I) \cdot \det(A^*[I, J] + A^\infty[I, J](x)) \right. \\
&\qquad \left. \times \det \operatorname{diag}(x; p_J) \right\} \\
&= \deg \det(A^*[I, J] + A^\infty[I, J](x)) + \delta_k(A) + (r^* - k)t \\
&\geq \delta_k(A) + (r^* - k)t.
\end{aligned}$$

On the other hand, it is easy to see that $\delta_{r^*} \leq \delta_k(A) + (r^* - k)t$ by the concavity (Proposition 1). Therefore, the equality holds. □

Proof of Theorem 2
We prove (13) first. From Theorem 1, Lemma 3 and Proposition 1, we see

$$\delta_l(A) = \delta_k(A) + (l-k)q \quad (l = k+1, \ldots, r^*). \tag{23}$$

On the other hand, $\hat{\delta}_l(A) \leq \hat{\delta}_k(A) + (l-k)t$ holds because of the concavity. From this and the inequality $\delta_l(A) \leq \hat{\delta}_l(A)$, we obtain

$$\hat{\delta}_l(A) = \delta_k(A) + (l-k)t \quad (l = k+1, \ldots, r^*).$$

This proves (13).

Next, we prove $\delta_{r^*+1}(A) < \delta_k(A) + (r^*+1-k)t$. If (p, q, t) is not an optimal dual solution of $\mathrm{DLP}(A, r^*+1)$, the inequality obviously holds. Otherwise, i.e., when (p, q, t) is an optimal solution of $\mathrm{DLP}(A, r^*+1)$, we can use Theorem 1. Since $\operatorname{rank} A^* = r^*$, we obtain

$$\delta_{r^*+1}(A) < \hat{\delta}_{r^*+1}(A) = \delta_k(A) + (r^*+1-k)t.$$

□

5 Illustrative Example

In this section, we illustrate the proposed algorithm by an example.

Let $A(x)$ be the matrix pencil

$$A(x) = \begin{bmatrix} x+1 & x+3 & x+2 \\ x+2 & x+6 & x+4 \\ x+1 & x+3 & x+1 \\ 2 & 1 & 3 \end{bmatrix}. \tag{24}$$

Then, we label the sets of rows and columns as $R = \{r_1, r_2, r_3, r_4\}$ and $C = \{c_1, c_2, c_3\}$, respectively.

Step 0: Initialization

From (24), we see that (r_1, c_1) is a maximizer of $\deg A_{ij}(A)$. Accordingly, we set $I_1^* := \{r_1\}$, $J_1^* := \{c_1\}$, $M_1 := \{(r_1, c_1)\}$ and $\delta_1(A) := 1$. We set $k := 1$ and proceed to Step 1.

Step 1: Construction of Optimal Dual Solutions

The auxiliary graph G_{M_1} (Section 4.3) is defined as Fig. 2.

In G_{M_1}, the length $\varphi(i)$ of the shortest path from u^+ to $i \in V$ is computed as follows:

$$\varphi(r_1) = \varphi(r_2) = \varphi(r_3) = \varphi(r_4) = 0,$$
$$\varphi(c_1) = \varphi(c_2) = \varphi(c_3) = \varphi(u^-) = -1.$$

Therefore, an optimal dual solution is

$$p = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}, \quad q = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \quad t = 1. \tag{25}$$

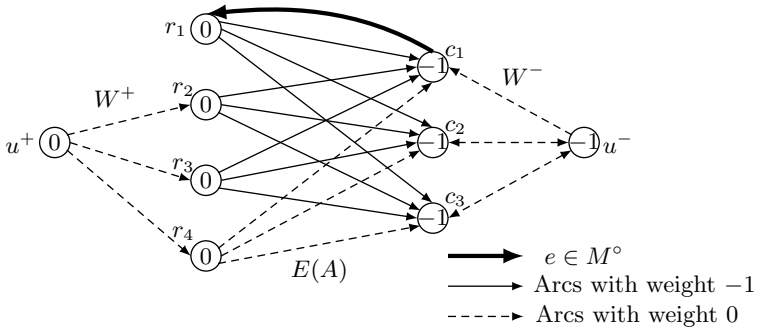

Figure 2: Example: Auxiliary graph G_{M_1}

Step 2: Test for Tightness

With the optimal dual solution (25), A^* is defined as follows:

$$A^* := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{26}$$

Since $\operatorname{rank} A^* = 1 = k$, we proceed to Step 3.

Step 3: Matrix Modification

The condition (16) for the 3×1 matrix $\tilde{U}$ now reads

$$\tilde{U} \cdot 1 + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

from which we see $\tilde{U} = [\ -1 \quad -1 \quad 0\ ]^\top$. Then, the rational matrix $A(x)$ can be modified as follows:

$$\begin{aligned} A'(x) &:= \begin{bmatrix} A[I_1^*, C] \\ \tilde{U} \cdot x^{p_{r_1}} \cdot A[I_1^*, C](x) + A[R \setminus I_1^*, C](x) \end{bmatrix} \\ &= \begin{bmatrix} x+1 & x+3 & x+2 \\ 1 & 3 & 2 \\ 0 & 0 & -1 \\ 2 & 1 & 3 \end{bmatrix}. \end{aligned}$$

Next, we repeat the same steps (Step 1 and 2) to confirm that the modification makes sense.

Step 1 and 2:

Similarly, we can obtain an optimal dual solution:

$$p = [\ 1 \quad 0 \quad 0 \quad 0\ ], \quad q = [\ 0 \quad 0 \quad 0\ ], \quad t = 0. \tag{27}$$

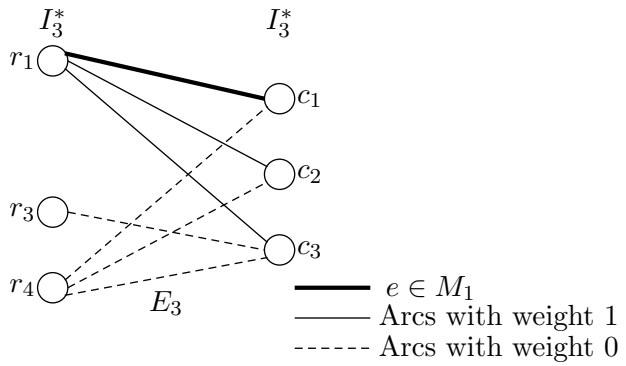

Figure 3: Example: $G = (I_3^* \cup J_3^*, E_3, c)$

Then, A^* can be modified to

$$A^* := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 2 \\ 0 & 0 & -1 \\ 2 & 1 & 3 \end{bmatrix}. \tag{28}$$

To compute rank A^*, we execute the Gaussian Elimination as stated in Section 4.6:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 3 & 2 \\ 0 & 0 & -1 \\ 2 & 1 & 3 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 3 \\ 0 & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix}. \tag{29}$$

Since rank $A^* = 3$, we jump to Step 4.

Step 4: Outputs and Updates

From the elimination (29), we see $I_3^* := \{r_1, r_3, r_4\}$ and $J_3^* := C$. By solving the weighted bipartite matching problem on $G = (I_3^* \cup J_3^*, E_3, c)$ (Fig. 3), where $E_3 \subseteq E$ is defined by $E_3 = E(A) \cap (I_3^* \times J_3^*)$. Then we obtain $M_3 := \{(r_1, c_1), (r_3, c_3), (r_4, c_2)\}$. As a result, $\delta_2(A)$ and $\delta_3(A)$ are calculated as follows:

$$\delta_2(A) = \delta_1(A) + 1 \cdot 0 = 1,$$
$$\delta_3(A) = \delta_1(A) + 2 \cdot 0 = 1.$$

Since $k = \text{rank}\, A^* = 3$ and $|C| = 3$, the algorithm terminates.

6 Complexity Analysis

Table 1 shows the time complexity of each operation. Then the time complexity of the proposed algorithm can be evaluated as follows.

Table 1: Time complexity of each operation (m: number of rows, n: number of columns, r: $\operatorname{rank} A$, $d_{\max}$: $\max \deg A_{ij}$, $d_{\min}$: $\min \operatorname{ord} A_{ij}$, d: $d_{\max} - d_{\min}$)

Step	Operation	Complexity	Method
Step 0	Initialization	$O(mn)$	Max Search
Step 1	Optimal Dual Solution	$O(dn^2r)$	Shortest Path
Step 2	Calculation of Rank	$O(dmnr^2)$	Gaussian Elimination
Step 3	Construction of $\tilde{U}$	$O(dmr^3)$	LU Decomposition
	Matrix Modification	$O(d^2mnr^3)$	Matrix Multiplication
Step 4	Construction of M_{r^*}	$O(r^3)$	Shortest Path

Theorem 4. *For an $m \times n$ Laurent polynomial matrix, $A(x)$, the proposed algorithm runs in $O(dnr(n + dmr^2))$ time.*

Proof. The theorem follows from Table 1.

In Step 0, we search for a maximizer of $\deg A_{ij}$, which can be done in $O(mn)$ time.

In Step 1, we construct an optimal dual solution by solving a single source shortest path problem. Since we can reweight the arc length to be nonnegative, it can be done in $O((n+m)^2)$. Since the value of t decreases at least by one each time and the difference between the optimal q's of $\mathrm{DLP}(A, 1)$ and of $\mathrm{DLP}(A, r)$ is at most rd, Step 1 can be done in $O(dr)$ in the worst case. Therefore, Step 1 runs in $O(dn^2r)$ time.

Step 2 runs in $O(mnr) \times O(dr)$ time, because the Gaussian elimination can be done in $O(mnr)$ time and it is executed $O(dr)$ times (the number of iterations of Step 2 is equal to that of Step 1).

In Step 3, we can construct $\tilde{U}$ by the LU decomposition in $O(mr^2)$ time in each execution. Therefore, it can be done in $O(dmr^3)$ time. In the phase of matrix modification, we execute the multiplication of an $(m-s) \times s$ polynomial matrix and an $s \times n$ polynomial matrix when $k = s$. That multiplication of matrices can be substituted by executing the multiplications of constant matrices $(D+1)$ times ([7, Lemma 8]), where $D := \max_k \hat{\delta}_k(A)$. Since $\hat{\delta}_k(A) \leq kd_{\max}$ holds for all k, $D = O(dr)$ is satisfied. Then, it can be done in $O(d^2mnr^3)$.

We solve the single source shortest path problem $(r-1)$ times in the whole of Step 4. Moreover, the arc length of the auxiliary graph can be reweighted to be nonnegative. Therefore, Step 4 runs in $O(r^3)$. □

Acknowledgements Thanks are due to Kazuo Murota for helpful comments on the manuscript.

References

[1] Commault, C., Dion, J. M.: Structure at infinity of linear multivariable systems: A geometric approach. IEEE Trans. Automat. Control. **AC-27**, 693–696 (1982)

[2] Dress, A. W. M., Wenzel, W.: Valuated matroid: A new look at the greedy algorithm. Appl. Math. Lett. **3**(2), 33–35 (1990)

[3] Dress, A. W. M., Wenzel, W.: Valuated matroids. Adv. Math. **93**, 214–250 (1992)

[4] Edmonds, J., Karp, R. M.: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of ACM. **19**, 248–264 (1972)

[5] Hairer E., Wanner G.: Solving Ordinary Differential Equations II. Springer, Berlin (1991)

[6] Iwata, S., Murota, K., Sakuta, I.: Primal-dual combinatorial relaxation algorithm for the maximum degree of subdeterminants. SIAM J. Sci. Comput. **17**, 993–1012 (1996)

[7] Iwata, S., Takamatsu, M.: Computing the maximum degree of minors in mixed polynomial matrices via combinatorial relaxation. Algorithmica **66**, 346–368 (2013)

[8] Murota, K.: Computing Puiseux-series solutions to determinantal equations via combinatorial relaxation. SIAM J. Comput. **19**, 1132–1161 (1990)

[9] Murota, K.: Combinatorial relaxation algorithm for the maximum degree of subdeterminants: Computing Smith–Mcmillan form at infinity and structural indices in Kronecker form. Appl. Algebra Eng. Commun. Comput. **6**, 251–273 (1995)

[10] Murota, K.: Computing the degree of determinants via combinatorial relaxation. SIAM J. Comput. **24**, 765–796 (1995)

[11] Murota, K.: Finding optimal minors of valuated bimatroids. Appl. Math. Lett. **8**, 37–42 (1995)

[12] Murota, K.: Matrices and Matroids for Systems Analysis. Springer, Berlin (2000)

[13] Tomizawa, N.: On some techniques useful for solution of transportation network problems. Networks **1**, 173–194 (1971)