On the job rotation problem

Peter Butkovič and Seth Lewis
School of Mathematics
The University of Birmingham
Edgbaston
Birmingham B15 2TT
United Kingdom

August 26, 2006

Abstract

The job rotation problem (JRP) is the following: Given an $n \times n$ matrix A over $\mathbb{R} \cup \{-\infty\}$ and $k \leq n$, find a $k \times k$ principal submatrix of A whose optimal assignment problem value is maximum. No polynomial algorithm is known for solving this problem if k is an input variable. We analyse JRP and present polynomial solution methods for a number of special cases.

Keywords: principal submatrix, assignment problem, job rotation problem, node disjoint cycles.

AMS-classification: 15A15, 90C27

1 Introduction

One of the classical problems in combinatorial optimization is the (*linear*) *assignment problem* which can be described as follows: A one-to-one assignment between two n-element sets of objects, say $\{A_1, ..., A_n\}$ and $\{B_1, ..., B_n\}$ has to be found. The cost c_{ij} of assigning A_i to B_j is given for every pair (A_i, B_j) and the task is to find an assignment that minimises the total cost. This problem has a convenient matrix formulation: If we store the coefficients c_{ij} in an $n \times n$ matrix C then the assignment problem means to choose n entries of C so that no two are from the same row or column, and their sum is minimal.

The assignment problem has, of course, also a maximising form in which the coefficients represent benefits and the object is to maximise the sum of the benefits. Many solution methods exist for the assignment problem [1], [6], probably the best known being the Hungarian method of computational complexity $O(n^3)$, whose many variants exist in the literature.

The *job rotation problem* is motivated by the following task: Suppose that a company with n employees requires these workers to swap their jobs (possibly on a regular basis) in order to avoid exposure to monotonous tasks (for instance manual workers at an assembly line or ride operators in a theme park). It is also required that to maintain stability of service only a certain number of employees, say k ($k < n$), actually swap their jobs. With each transition old job - new job a coefficient is associated expressing either the cost (for instance for an additional training) or the preference of the worker to this particular change. So the aim is to select k employees and to suggest a plan of the job changes between them so that the sum of the coefficients corresponding to these changes is minimum or maximum.

For any set X and positive integer n the symbol $X^{n\times n}$ will denote the set of all $n \times n$ matrices over X. In most cases we will deal with matrices over $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\}$. By a *principal submatrix* of a square matrix A we understand as usual any submatrix of A whose set of row indices is the same as the set of column indices. A principal submatrix of $A = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ is therefore any matrix of the form

$$\begin{pmatrix} a_{i_1i_1} & a_{i_1i_2} & \dots & a_{i_1i_k} \\ a_{i_2i_1} & a_{i_2i_2} & \dots & a_{i_2i_k} \\ \vdots & \vdots & & \vdots \\ a_{i_ki_1} & a_{i_ki_2} & \dots & a_{i_ki_k} \end{pmatrix}$$

where $1 \leq i_1 < \dots < i_k \leq n$. This matrix will be denoted by $A(i_1, i_2, \dots, i_k)$. Hence the job rotation problem is the problem to find, for a given $n \times n$ matrix A and $k < n$, a $k \times k$ principal submatrix of A for which the optimal assignment problem value is minimal or maximal (the diagonal entries can be set to $+\infty$ or $-\infty$ to avoid an assignment to the same job). For a particular A and k, we shall refer to this problem as $JRP(A, k)$. The task of solving the job rotation problem for all k, we shall refer to as $JRP(A)$ or just JRP. In the rest of the paper, we will discuss the maximisation version of the problem.

Note that there is also a "non-weighted" version of JRP in which it is only given which job moves are feasible. The problem is to decide if it is possible to re-assign / rotate k jobs between the employees, ($k \in N$), where job i can be assigned to job j only if (i, j) is from a given set of feasible transitions. This can obviously be represented by a $\{0, -\infty\}$ matrix C, where a 0 corresponds to a feasible move. Alternatively, this version can be represented by a (non-weighted) digraph $D = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_i, v_j); c_{ij} = 0\}$.

The number of principal submatrices of order k of a matrix of order n is $\binom{n}{k}$. Therefore if k is an input variable, solving the assignment problem for all principal submatrices and then comparing the resulting values would be non-polynomial. If $k \leq n$ is fixed, then the method would be polynomial (though of a high degree in most cases). However, the total number of submatrices of all orders is $\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$ and therefore checking all of them would not solve JRP for all k in polynomial time. In fact, no polynomial method seems to be known for solving this problem, neither is it proved to be NP-hard. In this paper we present a number of cases when JRP is solvable polynomially. Note that there is a randomized polynomial algorithm for solving JRP [5]. It may be interesting to mention that the problem arising by removing the word "principal" from the formulation of the JRP is easily solvable [10].

In Section 2 we will give an overview of known results. Section 3 deals with matrices over $\mathbb{T} = \{-\infty, 0\}$ and Section 4 contains results for matrices over $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\}$. These include the proof that $JRP(A)$ can be solved in polynomial time if this is true for every irreducible diagonal block of the Frobenius normal form of A.

2 Definitions and known results

In the rest of the paper we will assume that $n \geq 1$ is a given integer and we denote by N the set $\{1, \ldots, n\}$. If $A = (a_{ij}) \in \overline{\mathbb{R}}^{n \times n}$ then we denote

$$m(A) = \max_{\pi \in P_n} \sum_{i \in N} a_{i,\pi(i)}$$

where P_n stands for the set of all permutations of the set N. The quantity $\sum\limits_{i \in N} a_{i,\pi(i)}$ will be called the *weight* of π (notation $w(A, \pi)$). Obviously, $m(A)$ is the optimal value of the assignment problem for the matrix A. The set of all optimal permutations will be denoted by $ap(A)$, that is,

$$ap(A) = \{\pi \in P_n; m(A) = \sum_{i \in N} a_{i,\pi(i)}\}.$$

Let us denote for $k = 1, \dots, n$

$$\delta_k(A) = \max_{B \in P_k(A)} m(B),$$

where $P_k(A)$ is the set of all principal submatrices of A of order k. For simplicity we often write just δ_k instead of $\delta_k(A)$. Clearly, $\delta_n = m(A)$ and $\delta_1 = \max(a_{11}, a_{22}, \dots, a_{nn})$. Note that $\delta_k = -\infty$ if $m(B) = -\infty$ for all $B \in P_k(A)$.

Thus $JRP(A,k)$ $(k = 1, ..., n)$ is the problem of finding a matrix $B \in P_k(A)$ such that

$$\delta_k(A) = m(B).$$

Example 1. Let

$$A = \begin{pmatrix} -\infty & 3 & -\infty & -\infty \\ 1 & -\infty & 0 & 2 \\ 5 & 4 & -\infty & 7 \\ -\infty & 6 & -\infty & -\infty \end{pmatrix}.$$

Then it is easily seen that $\delta_1 = -\infty$, $\delta_2 = 8$, $\delta_3 = 13$, and $\delta_4 = -\infty$.

If $A = (a_{ij}) \in \overline{\mathbb{R}}^{n \times n}$ then we denote by $D(A)$ the digraph whose set of nodes is N and arc set is $\{(i,j); a_{ij} > -\infty\}$. For a path $\tau = (i_1, i_2, \dots, i_p)$, let $V(\tau) = \{i_1, i_2, \dots, i_p\}$. For a digraph D, we say paths $\tau_1, \tau_2, \dots, \tau_s$ in D are *pairwise node disjoint* (PND) if $V(\tau_i) \cap V(\tau_j) = \emptyset$ for $i, j = 1, ..., s, i \neq j$.

For $A \in \overline{\mathbb{R}}^{n \times n}$, we define $k_{\max}(A)$ or just $k_{\max}$ as

$$\max\{k \in N; \delta_k(A) > -\infty\}.$$

Since every permutation is a product of cyclic permutations, $k_{\max}(A)$ is the biggest number of nodes in $D(A)$ that can be covered by PND cycles in $D(A)$. Note that we are not using the word "elementary" in connection with cycles as all cycles in this paper are elementary.

Let $A \in \overline{\mathbb{R}}^{n \times n}$ be a symmetric matrix and σ be an arbitrary cycle of length p in $D(A)$. By symmetry, for each arc (i,j) in $D(A)$, (j,i) is also an arc ("counterarc"). If p is even, we define the operation of *splitting* σ as removing alternate arcs from σ, and adding counterarcs (j,i) for each (i,j) that remains from σ, resulting in a collection of $\frac{p}{2}$ PND cycles in $D(A)$ that cover all p nodes from $V(\sigma)$. If σ is a loop, we define the operation of *splitting* σ as removing the arc. If $p \geq 3$ is odd, we define the operation of *splitting* $\sigma - v$ as removing alternate arcs from σ, starting with an incident arc to node v and ending with the other incident arc to node v, and adding counterarcs (j,i) for each (i,j) that remains from σ, resulting in a collection of $\frac{p-1}{2}$ PND cycles in $D(A)$ that cover $p-1$ nodes from $V(\sigma)$, with node v not being covered. We define *splitting* a path with p arcs as deleting alternate arcs on that path starting from the second arc and adding counterarcs to the remaining arcs, to form a collection of $\frac{p}{2}$ 2-cycles if p was even, or $\frac{p+1}{2}$ 2-cycles if p was odd.

The task of finding $k_{\max}$ for a general matrix can be solved in $O(n^3)$ time [8], however we can do better for symmetric matrices:

Theorem 2. ([7]) The task of finding $k_{\max}$ for a symmetric matrix $A \in \overline{\mathbb{R}}^{n\times n}$ is equivalent to the maximum cardinality matching problem in a bipartite graph with $2n$ nodes and can therefore be solved in $O(n^{2.5}/\sqrt{\log n})$ time.

Proof. Let $B(A)$ be the bipartite graph with the bipartition (U,V), where $U = \{u_1,...,u_n\}, V = \{v_1,...,v_n\}$ and set of arcs $\{u_i v_j; a_{ij} > -\infty\}$.

Let M be a matching of maximum cardinality in $B(A)$, $|M| = m$. Obviously $k_{\max} \leq m$ because if $k = k_{\max}$ then there are k finite entries in A, no two in the same row or column, say $a_{i_r\pi(i_r)}, r = 1,...,k$, and so there is a matching of cardinality k in $B(A)$, namely, $\{u_{i_r} v_{\pi(i_r)}; r = 1,...,k\}$.

We now prove $k_{\max} \geq m$. The set of arcs $H = \{(i,j); u_i v_j \in M\}$ in $D(A)$ consists of directed PND elementary paths or cycles, since both the outdegree and indegree of each node in (N,H) is at most one. We will call a path *proper* if it is not a cycle.

Construct from H another set H' as follows (see Figure 1): If all paths in H are cycles then set $H' = H$. Now suppose that at least one proper path exists. Splitting every proper path in (N,H), we obtain a digraph (N,H') which consists of original cycles in (N,H) and a number of cycles of length 2. All cycles in (N,H') are PND.

Each set of PND cycles in $D(A)$ determines a matching in $B(A)$ whose cardinality is equal to the total number of arcs of these cycles. Thus none of the proper paths in (N,H) could have been of odd length, say s, as otherwise the total number of arcs on cycles constructed from this path would be $s+1$, a contradiction with the maximality of M. Hence $|H'| = m$ and thus $k_{\max} \geq m$.

The complexity statement now follows from [3]. $\square$

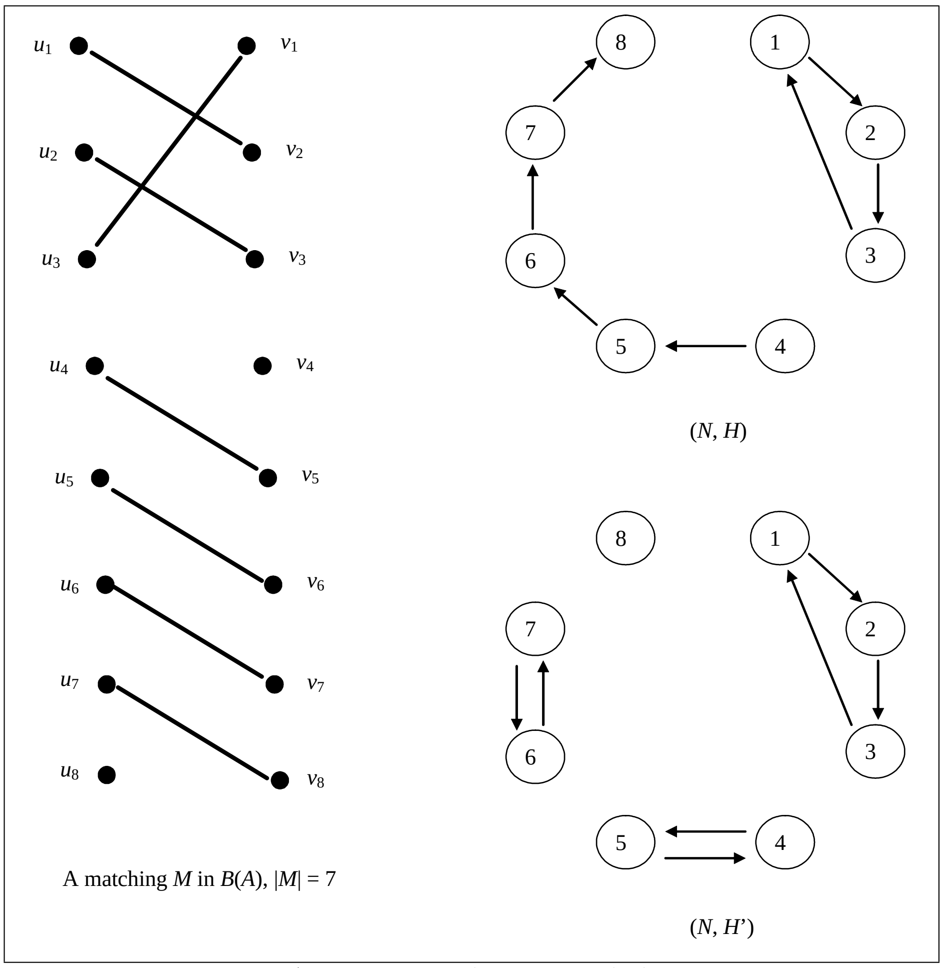

Figure 1: An illustration for the proof of Theorem 2.

We can similarly define $k_{\min}$. It is easily seen that the problem of finding $k_{\min}$ is equivalent to finding a shortest cycle in a digraph and is therefore polynomially solvable [19]. In Example 1, we have $k_{min} = 2$ and $k_{max} = 3$.

As usual a real sequence $g_0, g_1, \ldots$ is called *convex* [*concave*] if

$$\begin{aligned} g_{r-1} + g_{r+1} &\geq 2g_r \\ [g_{r-1} + g_{r+1} &\leq 2g_r] \end{aligned}$$

for all $r = 1, 2, \ldots$.

One class of solvable cases of the JRP is related to the fact that $\delta_k(A)$ for $k = 1, 2, \ldots, n$ are the coefficients of the characteristic polynomial of A in max-algebra as defined in [12]. Max-algebra is the analogue of linear algebra in which the conventional operations of addition and multiplication are replaced by $\oplus$ and $\otimes$ defined as follows: $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$ for $a, b \in \overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty\}$. An account on algebraic properties in max-algebra can be found in [11] and [13]. Note that in recent years max-algebra has been investigated under the name ”tropical algebra” in a number of papers, see for instance [14].

An $O(n^2(m + n \log n))$ algorithm is known [4] for finding so called essential terms of the max-algebraic characteristic polynomial of an $n \times n$ matrix where m is the number of finite entries of A. The algorithm presented in [4] does not explicitly produce the corresponding $k \times k$ principal submatrix but this can easily be identified from the data produced by the algorithm. It then follows that this method solves $JRP(A, k)$ for all $k = 1, \ldots, n$, when all terms are essential or, equivalently when the sequence $\delta_1, \ldots, \delta_n$ is concave [13]. Note that the complexity bound has recently been improved to $O(n(m + n \log n))$ steps [15].

Max-algebraic theory provides various other information for the job rotation problem, one of them being that $\max_{k \in N} \delta_k(A) = m(A')$ where A' is obtained from A after replacing all negative diagonal entries by 0. The corresponding principal submatrix can also be easily identified.

3 JRP for special symmetric matrices over $\{0, -\infty\}$

In this section we show that $JRP(A, k)$, for a symmetric matrix A over $\{0, -\infty\}$, and k even, can be solved in O(1) time, after finding k_{max}. We also describe some cases when this is true for odd values of k. These results can immediately be applied to the question of finiteness of $\delta_k(A)$ for symmetric matrices $A \in \overline{\mathbb{R}}^{n \times n}$.

Let $\mathbb{T} = \{0, -\infty\}$ and $A \in \mathbb{T}^{n \times n}$. Then for all k, the unique finite value for $\delta_k(A)$ is 0. Also, $\delta_k(A) = 0$ if and only if there exist PND cycles in $D(A)$ covering a total of k nodes. Hence, deciding if $\delta_k(A) = 0$ for some matrix $A \in \mathbb{T}^{n \times n}$ is equivalent to deciding whether there exist PND cycles in $D(A)$ covering exactly k nodes.

Theorem 3. If $A \in \mathbb{T}^{n \times n}$ is a symmetric matrix and $\delta_l(A) = 0$ for some $l \in N$, then $\delta_k(A) = 0$ for all even $k \leq l$, and $\delta_k(A) = 0$ for all $k = l - r, \ldots, l$, where r is the number of odd cycles in a collection of PND cycles in $D(A)$ that cover l

nodes.

Proof. Let $\{\sigma_1, \dots, \sigma_t\}$ be a collection of PND cycles in $D(A)$ covering l nodes with σ_i having odd length for $i = 1, \dots, r$ and even length otherwise. By splitting the cycles σ_i for $i = r+1, \dots, t$ if needed, we may assume that all these cycles are 2-cycles. By splitting one by one the cycles $\sigma_i - v_i$ for $v_i \in V(\sigma_i)$ and $i = 1, \dots, r$, we get that $\delta_{l-i} = 0$ for $i = 1, \dots, r$. After these splittings, all remaining cycles are 2-cycles and removing them one by one proves the result. □

Corollary 4. Let $A \in \mathbb{T}^{n\times n}$ be a symmetric matrix. For all even $k \leq k_{max}$, $\delta_k(A) = 0$, and if $\delta_k(A) = 0$ for some odd $k \in N$ then $\delta_{k-1}(A) = 0$.

If A has at least one zero on the main diagonal, (or equivalently, if the digraph $D(A)$ has at least one loop), then we can derive a number of properties:

Theorem 5. If $A \in \mathbb{T}^{n\times n}$ is a symmetric matrix, and there exists a collection of PND cycles in $D(A)$ covering l nodes, at least one of which is a loop, then $\delta_k(A) = 0$ for all $k \leq l$.

Proof. Let $\{\sigma_1, \dots, \sigma_t\}$ be a collection of PND cycles in $D(A)$ covering k_{max} nodes with σ_i having odd length for $i = 1, \dots, r$ and even length otherwise. Assume σ_r is a loop. By splitting the cycles σ_i for $i = r+1, \dots, t$ if needed, we may assume that all these cycles are 2-cycles. By splitting one by one the cycles $\sigma_i - v_i$ for $v_i \in V(\sigma_i)$ and $i = 1, \dots, r-1$, we get that $\delta_{l-i} = 0$ for $i = 1, \dots, r-1$. After these splittings, all remaining cycles except σ_r are 2-cycles. Removing the 2-cycles one by one gives us $\delta_{l-i} = 0$ for odd $i \in \{r+1, \dots, l-1\}$. Removing σ_r and the 2-cycles one by one gives us $\delta_{l-i} = 0$ for even $i \in \{r, \dots, l\}$. □

If $l \in \{k_{max}, k_{max} - 1\}$ in Theorem 5, then we can completely solve the (non-weighted) JRP for this type of matrix:

Theorem 6. If $A \in \mathbb{T}^{n\times n}$ is a symmetric matrix, $l \in \{k_{max}, k_{max} - 1\}$ and there exists a collection of PND cycles in $D(A)$ covering l nodes, at least one of which is a loop, then $\delta_k(A) = 0$ for all $k \leq k_{max}$.

Proof. The statement immediately follows from Theorem 5 and the fact that $\delta_{k_{max}}(A) = 0$. □

Theorem 7. If $A = (a_{ij}) \in \mathbb{T}^{n\times n}$ is a symmetric matrix and $D(A)$ contains a loop, then $\delta_k(A) = 0$ for all $k \leq k_{max}$.

Proof. We assume that (j,j) is a loop in $D(A)$. As $\delta_{k_{max}}(A) = 0$, there exist PND cycles in $D(A)$ $\sigma_1, \sigma_2, \ldots, \sigma_t$ in $D(A)$ covering k_{max} nodes. We need to show there exist PND cycles $\sigma'_1, \sigma'_2, \ldots, \sigma'_{t'}$ in $D(A)$, at least one being a loop, that cover k_{max} or $k_{max} - 1$ nodes.

Clearly if $(j,j) \in \{\sigma_1, \sigma_2, \ldots, \sigma_t\}$ then we can use Theorem 6 and we are done, so assume not. Then j is covered by these cycles, as otherwise, (j,j) together with $\sigma_1, \sigma_2, \ldots, \sigma_t$ would form PND cycles in $D(A)$ covering $k_{max} + 1$ nodes, which contradicts the definition of k_{max}. Hence there exists one cycle $\sigma_r = (j, i_2, i_3, \ldots, i_p, j) \in \{\sigma_1, \sigma_2, \ldots, \sigma_t\}$.

If p is odd, then we can split $\sigma_r - j$, and add (j,j) to the resulting cycles to form PND cycles in $D(A)$ covering k_{max} nodes. If instead p is even, then we can split σ_r, remove the 2-cycle that contains p, and add (j,j) to the remaining cycles to form PND cycles in $D(A)$ covering $k_{max} - 1$ nodes. The result then follows from Theorem 6. □

For $A \in \overline{\mathbb{R}}^{n\times n}$, we define $F = \{k \in N; \delta_k(A) \neq -\infty\}$. By Corollary 4, for symmetric matrices, unless $k_{max} = 1$, the smallest even $k \in F$ is 2. However, the smallest odd value in F is more tricky. We denote this value by k_{oddmin}.

Remark. If there exist PND cycles $\sigma_1, \sigma_2, \ldots, \sigma_t$ in $D(A)$ such that an odd number of nodes is covered, then at least one of the cycles is odd. Hence k_{oddmin} is the length of a shortest odd cycle in $D(A)$. This cycle can be found polynomially [19]. Note that k_{oddmin} does not exist if there is no odd cycle in $D(A)$, and if this is the case, then $\delta_k = -\infty$ for all odd k. For the remainder of this section, we shall assume that k_{oddmin} exists.

Theorem 8. *Let $A \in \mathbb{T}^{n\times n}$ be a symmetric matrix, $\sigma_1, \sigma_2, \ldots, \sigma_t$ be a collection of PND cycles in $D(A)$ covering k' nodes, with at least one having odd length. Then $\delta_k(A) = 0$ for all odd $k \in \{l', \ldots, k'\}$, where l' is the minimum length of the odd cycles in this collection.*

Proof. Without loss of generality, assume the length of σ_t is l'. Then the PND cycles $\sigma_1, \sigma_2, \ldots, \sigma_{t-1}$ in $D(A)$ cover $k' - l'$ nodes, hence $\delta_{k'-l'} = 0$. By Corollary 4, $\delta_k = 0$ for all even $k \in \{0, \ldots, k' - l'\}$. Take an arbitrary even $k \in \{0, \ldots, k' - l'\}$. So $k + l' \in \{l', \ldots, k'\}$. There exist PND cycles $\sigma'_1, \sigma'_2, \ldots, \sigma'_{t'}$ in $D(A)$ covering k nodes other than those in $V(\sigma_t)$. Therefore, $\sigma'_1, \sigma'_2, \ldots, \sigma'_{t'}$ and σ_t are PND cycles in $D(A)$ covering $k + l'$ nodes. Hence the result. □

Corollary 9. Let $A \in \mathbb{T}^{n\times n}$ be a symmetric matrix, $\sigma_1, \sigma_2, \dots, \sigma_t$ be a collection of PND cycles in $D(A)$ covering k_{max} or $k_{max} - 1$ nodes, with at least one having length k_{oddmin}. Then we can decide whether $\delta_k(A)$ is 0 or $-\infty$ for all k in linear time, after finding k_{max} and k_{oddmin}.

Proof. The result follows from Corollary 4 and Theorem 8. □

Theorem 10. If $A \in \mathbb{T}^{n\times n}$ is a symmetric matrix, then $\delta_k(A) = 0$ for all odd $k \in \{k_{oddmin}, \dots, k_{max} - k_{oddmin}\}$.

Proof. As $\delta_{k_{max}} = 0$, there exist PND cycles $\sigma_1, \sigma_2, \dots, \sigma_t$ in $D(A)$ that cover k_{max} nodes. There exists a cycle σ in $D(A)$ of length k_{oddmin}.

Delete all nodes in $V(\sigma)$ from $\sigma_1, \sigma_2, \dots, \sigma_t$, as well as incident arcs. As the cycles were PND and each node was incident to precisely two arcs, up to $2k_{oddmin}$ arcs have been deleted. Therefore this leaves a total of at least $k_{max} - 2k_{oddmin}$ arcs within the remaining PND cycles and paths that have arisen from deleting the arcs from the cycles. Split any paths into 2-cycles. We now have PND cycles in $D(A)$ covering at least $k_{max} - 2k_{oddmin}$ arcs, and therefore at least $k_{max} - 2k_{oddmin}$ nodes, none of which are nodes on σ.

Therefore, by Corollary 4, for all even $i \le k_{max} - 2k_{oddmin}$, there exist PND cycles $\sigma'_1, \sigma'_2, \dots, \sigma'_{t'}$ in $D(A)$ covering i nodes, but none on σ. So for all even $i \le k_{max} - 2k_{oddmin}$, we have PND cycles $\sigma'_1, \sigma'_2, \dots, \sigma'_{t'}$ and σ in $D(A)$ which cover $i + k_{oddmin}$ nodes. Hence the result. □

Remark. Note that $\{k_{oddmin}, \dots, k_{max} - k_{oddmin}\} \neq \emptyset \iff k_{oddmin} \le \dfrac{k_{max}}{2}$.

Corollary 11. Let $A \in \mathbb{T}^{n\times n}$ be a symmetric matrix, with PND cycles in $D(A)$ covering k_{max} or $k_{max} - 1$ nodes, one having odd length of at most $k_{max} - k_{oddmin}$. Then we can decide whether $\delta_k(A)$ is 0 or $-\infty$ for all k in linear time, after finding k_{max} and k_{oddmin}.

Proof. The result follows from Corollary 4, Theorem 8 and Theorem 10. □

Corollary 12. Let $A \in \mathbb{T}^{n\times n}$ be a symmetric matrix, with PND cycles in $D(A)$ covering k_{max} or $k_{max} - 1$ nodes, two having odd length. Then we can decide whether $\delta_k(A)$ is 0 or $-\infty$ for all k in linear time, after finding k_{max}.

Proof. The result follows from Corollary 11 and the fact that the length of at least one of the odd cycles is at most $\frac{k_{max}}{2} \le k_{max} - k_{oddmin}$. □

Remark. Note that solving an assignment problem for $A = (a_{ij}) \in \mathbb{T}^{n\times n}$ is equivalent to deciding whether the permanent of the matrix $B = (b_{ij})$ is positive where B is defined by $b_{ij} = 1$ if $a_{ij} = 0$ and $b_{ij} = 0$ otherwise. Therefore the statements in Section 3 solve in special cases the question: Given $A \in \{0,1\}^{n\times n}$, and $k \le n$, is there a $k\times k$ principal submatrix of A whose permanent is positive?

4 JRP for special matrices over $\overline{\mathbb{R}}$

Recall that a matrix $A \in \overline{\mathbb{R}}^{n\times n}$ is called *irreducible* if $D(A)$ is strongly connected or $n = 1$. If A, B are square matrices and A can be obtained from B by simultaneous permutations of the rows and columns then we say that A and B are *equivalent*, notation $A \sim B$. Clearly, $\sim$ is an equivalence relation, and $\delta_k(A) = \delta_k(B)$ if $A \sim B$. It is known [18] that every matrix A can be transformed in linear time to an equivalent matrix B in the Frobenius normal form, that is

$$B = \begin{pmatrix} B_{11} & B_{12} & \dots & B_{1p} \\ & B_{22} & \dots & B_{2p} \\ & & \ddots & \vdots \\ -\infty & & & B_{pp} \end{pmatrix},$$

in which all diagonal blocks are irreducible.

In this section we study JRP for matrices over $\overline{\mathbb{R}}$. First we present some solvable special cases and then we show that $JRP(A)$ for $A \in \overline{\mathbb{R}}^{n\times n}$ can be solved in polynomial time if this is true for every diagonal block of the Frobenius normal form of A.

4.1 Pyramidal matrices

If $A = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ and $k \in N$ then the principal submatrix $A(1, ..., k)$ is called a *main principal submatrix* of A, notation $A[k]$. If for all $i, j, r, s \in N$

$$\max(i, j) < \max(r, s) \Longrightarrow a_{ij} \ge a_{rs}, \tag{1}$$

then A is called *pyramidal*.

Theorem 13. If $A = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ is pyramidal then $\delta_k(A) = m(A[k])$.

Proof. Let $A(l_1, \dots l_k)$ be an arbitrary principal submatrix, where $1 \le l_1 <$

$\cdots < l_k \leq n$. Note that

$$i \leq l_i, \text{ for all } i \leq k.$$

Therefore

$$\max(i,j) \leq \max(l_i, l_j), \text{ for } i,j \leq k.$$

If equality does not hold for some i and j, then by (1) we have,

$$a_{ij} \geq a_{l_i,l_j}.$$

If equality does hold for some i and j, then let $l_t = \max(l_i, l_j)$. Note that $i < j \Leftrightarrow l_i < l_j$. So we have $t = \max(i,j)$ and therefore $l_t = t$. Hence $l_{t-1} = t-1, \ \ldots, \ l_1 = 1$. In this case

$$a_{ij} = a_{l_i,l_j}.$$

Either way, $a_{ij} \geq a_{l_i,l_j}$ holds. Therefore

$$\begin{aligned} m(A(l_1, \ldots l_k)) &\leq m(A(1, \ldots k)) \\ &= m(A[k]) \\ &= \delta_k(A), \end{aligned}$$

as $A(l_1, \ldots l_k)$ was arbitrary. Hence the result. $\square$

Example 14. Consider the matrix

$$A = \left(\begin{array}{cccc} \multicolumn{1}{c|}{9} & \multicolumn{1}{c|}{8} & \multicolumn{1}{c|}{4} & 3 \\ \cline{1-1} 8 & \multicolumn{1}{c|}{6} & \multicolumn{1}{c|}{5} & 4 \\ \cline{1-2} 5 & 4 & \multicolumn{1}{c|}{4} & 3 \\ \cline{1-3} 3 & 2 & 3 & 1 \end{array} \right).$$

The indicated lines help to check that A is pyramidal. Hence by Theorem 13 we find:

$$\begin{aligned} \delta_1(A) &= m(A[1]) = 9 \\ \delta_2(A) &= m(A[2]) = 16 \\ \delta_3(A) &= m(A[3]) = 20 \\ \delta_4(A) &= m(A[4]) = m(A) = 22. \end{aligned}$$

Remark. Matrices that are not pyramidal, may become such after simultaneously permuting rows and columns. It follows from (1) that the diagonal entries of the matrix must be in descending order for (1) to be satisfied. Once rows and columns have been simultaneously permuted in this way, additional simultaneous row and column permutations may be needed between rows and columns which have a diagonal entry equal to another diagonal entry.

4.2 Monge and Hankel matrices

A matrix A will be called *diagonally dominant* if $id \in ap(A)$. (Note that throughout the paper id stands for the identity permutation.) A matrix $A = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ is called *Monge* if $a_{ij} + a_{rs} \geq a_{is} + a_{rj}$ for all $i, j, r, s \in N$, $i \leq r$, $j \leq s$. It is well known [6] that every Monge matrix A is diagonally dominant. It is also easily seen that a principal submatrix of a Monge matrix is also Monge. Hence $JRP(A, k)$ for Monge matrices is readily solved by finding the k biggest diagonal entries of A.

For a given sequence $\{g_r \in \overline{\mathbb{R}}; r = 1, \ldots, 2n-1\}$, the *Hankel* matrix is the matrix $H = (h_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ where $h_{ij} = g_{i+j-1}$. Hankel matrices generated by convex sequences are Monge [9]. Therefore, for these matrices, JRP is readily solved. However, no efficient method seems to exist for Hankel matrices in general.

In this subsection we show that finiteness of $\delta_k(H)$ can be easily decided for any Hankel matrix H. Since Hankel matrices are symmetric, we can use some of the results of Section 3.

Theorem 15. If $\{g_r \in \overline{\mathbb{R}}; r = 1, \ldots, 2n-1\}$ is the sequence generating Hankel matrix $H = (h_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ and $g_r \neq -\infty$ for some odd r, then $\delta_k(H) \neq -\infty$ for all $k \leq k_{max}$.

Proof. Let $C = (c_{ij})$ be defined by $c_{ij} = 0$ if $h_{ij} \neq -\infty$ and $c_{ij} = -\infty$ otherwise. Assume $g_r \neq -\infty$ for some odd r. So $(\exists i)$ $c_{ii} \neq -\infty$, i.e. $(\exists i)$ $c_{ii} = 0$. We now use Theorem 7 to give us $\delta_k(C) = 0$ for all $k \leq k_{max}$. Then as $\delta_k(C) = 0$ if and only if $\delta_k(H) \neq -\infty$, the theorem follows. □

Theorem 16. If a matrix $A = (a_{ij}) \in \overline{\mathbb{R}}^{n\times n}$ is any matrix such that $a_{ij} = -\infty$ if $i + j$ is even, then $\delta_k(A) = -\infty$ for all odd k.

Proof. Assume $A = (a_{ij})$ is a matrix such that $a_{ij} = -\infty$ if $i + j$ is even. If a_{ij} is finite then $i + j$ is odd. So i and j must be of different parities.

Let σ be the cyclic permutation $i_1 \to i_2 \to \cdots \to i_p \to i_1$ of arbitrary length p such that $w(A, \sigma) \neq -\infty$.

As $w(A, \sigma) \neq -\infty$, then $a_{i_j i_{j+1}} \neq -\infty$ for all j. So i_j and i_{j+1} must be of different parities for all j. This means elements in the sequence $i_1, i_2, \ldots, i_p, i_1$ alternate between even and odd. Thus p must be an even number, i.e. there are no cyclic permutations σ of odd length of finite weight. Hence the result. $\square$

If A is symmetric, then together with Corollary 4, this gives us:

Theorem 17. If $A = (a_{ij}) \in \overline{\mathbb{R}}^{n \times n}$ is a symmetric matrix such that $a_{ij} = -\infty$ if $i + j$ is even, then $\delta_k(A) \neq -\infty$ for all even $k \leq k_{max}$, and $\delta_k(A) = -\infty$ for all odd k.

A certain type of Hankel matrix satisfies Theorem 17. Rewriting it for this type of matrix gives:

Theorem 18. If $\{g_r \in \overline{\mathbb{R}}; r = 1, \ldots, 2n-1\}$ is the sequence generating Hankel matrix H and $g_r = -\infty$ for all odd r, then $\delta_k(H) \neq -\infty$ for all even $k \leq k_{max}$ and $\delta_k(H) = -\infty$ for all odd k.

Combining Theorem 15 and Theorem 18 enables us to decide whether $\delta_k(H)$ is finite or not for any Hankel matrix H.

Theorem 19. If $\{g_r \in \overline{\mathbb{R}}; r = 1, \ldots, 2n-1\}$ is the sequence generating Hankel matrix H then

1. $\delta_k(H) \neq -\infty$ for all even $k \leq k_{max}$,
2. $\delta_k(H) = -\infty$ for all odd k if $g_r = -\infty$ for all odd r, and
3. $\delta_k(H) \neq -\infty$ for all odd $k \leq k_{max}$ if $g_r \neq -\infty$ for some odd r.

4.3 Block diagonal matrices

Let

$$A = \mathrm{blockdiag}(A_1, A_2, \ldots, A_p) = \begin{pmatrix} A_1 & & & -\infty \\ & A_2 & & \\ & & \ddots & \\ -\infty & & & A_p \end{pmatrix}.$$

$D(A_i)$ is a subgraph of $D(A)$ for every $i = 1, \ldots, p$. Every $D(A_i)$ is disjoint from any $D(A_j)$, $j \neq i$. So any cycle in $D(A)$ has nodes entirely within one of these disjoint subgraphs, and it is not possible to have a cycle in $D(A)$ with arcs corresponding to elements from more than one of the matrices $A_1, \ldots, A_p$.

We now show how to solve $JRP(A)$ for $A = \text{blockdiag}(A_1, A_2, \ldots, A_p)$ in polynomial time, as long as we can solve $JRP(A_j)$ in polynomial time, for $j = 1, \ldots, p$. This is shown in an algorithm called JRPBLOCKDIAG (see Figure 2).

Let $n(j)$ be the order of A_j, $j = 1, \ldots, p$. Assume that we have solved $JRP(A_j)$. This may have been done in polynomial time if A_j is one of the special types of matrix previously mentioned in this paper.

So for $j = 1, \ldots p$ and $r = 1, \ldots, k$ we are able to find $\delta_r(A_j)$ and also a principal submatrix $B_{jr} \in (A_j)_r$ (where $(A_j)_r$ is the set of all $r \times r$ principal submatrices of A_j) and permutation $\pi_{jr} \in ap(B_{jr})$ such that $w(B_{jr}, \pi_{jr}) = \delta_r(A_j)$.

Let $D_j = D(A_j)$. For each block A_j, we have the following information: For $r = 1, \ldots, k$, the permutation π_{jr} in D_j gives cycles of total length r and total weight $\delta_r(A_j)$.

We will use S, a set of pairs to tell us which submatrix to select and which elements from within it to select. We do this by assigning pairs (j, r) to S. A pair (j, r) tells us that by choosing B_{jr} and π_{jr} we select a total of r elements from B_{jr} and give a total sum of $\delta_r(A_j)$.

There are p stages to the algorithm. At each stage information is collected and then stored within a set of triples called M_j. Each triple has the form (S, w, k), where S is as described above, w is the total weight of elements selected by using the information in S, and k is the total number of elements selected by using the information in S.

M_0 is set to $\{(\emptyset, 0, 0)\}$ at Stage 0. For $j = 1, \ldots, p$, at Stage j, the information found from A_j (i.e. $\delta_1(A_j), \delta_2(A_j), \ldots, \delta_{n(j)}(A_j)$) and the information from Stage $j - 1$ (i.e. M_{j-1}) is combined to produce M_j. We start by copying all triples from M_{j-1} to M_j. Next, if we can find a triple (S, w, k) (of the form described above) by combining the information found from A_j and M_{j-1} that is not in M_j, then we add (S, w, k) to M_j. Otherwise, if w is larger than the second coordinate of any triple in M_j having third component equal to k, then we replace that triple with (S, w, k) in M_j.

We now give the algorithm, called JRPBLOCKDIAG, and then discuss the correctness and complexity of this algorithm.

```
Algorithm JRPBLOCKDIAG
Input: $A = \text{blockdiag}(A_1, A_2, \ldots, A_p) \in \overline{\mathbb{R}}^{n\times n}$.
Output: For $k = 1, \ldots, n$, $\delta_k(A)$, and if $\delta_k(A)$ is finite, then also $k$ independent
entries of a $k \times k$ principal submatrix of $A$ whose total is $\delta_k(A)$.

  1. Set $M_0 = \{(\emptyset, 0, 0)\}$
  2. For $j = 1$ to $p$ :
      (a) For $r = 1$ to $n(j)$ :
           i. Find $\delta_r(A_j)$.
          ii. Find $B_{jr} \in (A_j)_r$ and $\pi_{jr} \in ap(B_{jr})$ such that $w(B_{jr}, \pi_{jr}) = \delta_r(A_j)$.
      (b) Set $M_j = M_{j-1}$
      (c) For each element $(S, w, l) \in M_{j-1}$ :
          For each $r = 1$ to $\min(\sum_{t=1}^{j} n(t) - l, n(j))$ with $\delta_r(A_j)$ finite :
           i. If $\nexists (S', w', l + r) \in M_j$, then add $(S \cup \{(j, r)\}, w + \delta_r(A_j), l + r)$
              to $M_j$.
          ii. If $\exists (S', w', l + r) \in M_j$ and $w' < w + \delta_r(A_j)$, then remove
              $(S', w', l + r)$ from $M_j$ and add $(S \cup \{(j, r)\}, w + \delta_r(A_j), l + r)$ to
              $M_j$.
  3. For $k = 1$ to $n$ :
      If $\exists (S, w, k) \in M_p$, then return $\delta_k(A) = w$, and for $i = 1, \ldots, r$ and all
      $(j, r) \in S$, return the element of $A$ that corresponds to the $(i, \pi_{jr}(i))$ entry
      of $B_{jr}$. Else return $\delta_k(A) = -\infty$.
```

Figure 2: An algorithm for solving JRP for block diagonal matrices.

Lemma 20.

1. If $(S, w, k) \in M_j$ in Step 3 of the algorithm, then
   (a) $S \subseteq \{1, \ldots, p\} \times \{1, \ldots, n\}$,
   (b) $\sum_{(i,s)\in S} \delta_s(A_i) = w$,
   (c) $\sum_{(i,s)\in S} s = k$,
   (d) If $(S', w', k) \in M_j$, then $S' = S$ and $w' = w$,
   (e) If $S' \subseteq \{1, \ldots, p\} \times \{1, \ldots, n\}$, $w' = \sum_{(i,s)\in S'} \delta_s(A_i)$ and $\sum_{(i,s)\in S'} s = k$ then $w' \leq w$.

2. If $S \subseteq \{1,\dots,p\} \times \{1,\dots,n\}$, and $\sum\limits_{(i,s)\in S} s = k \leq n$, then in Step 3 of the algorithm, $\exists (S', w', k) \in M_j$ where $w \leq w'$.

Proof. Statements 1(a)-1(c) are proved by induction on j, and hold automatically for $j = 0$. For $j > 0$, assume $(S, w, k) \in M_j$. We have two cases to consider:

Case 1: If $\nexists (j, r) \in S$, then $(S, w, k) \in M_{j-1}$, so 1(a)-1(c) follow by induction.

Case 2: If $\exists (j, r) \in S$, then $(S - \{(j, r)\}, w - \delta_r(A_j), k - r) \in M_{j-1}$. Note that $r \leq n$, as the third coordinate of this lies between 0 and $n - r$. So again 1(a)-1(c) follow by induction.

To prove 1(d), we use the fact that each element of M_j has a unique third component due to the way Step 2(c) of the algorithm was constructed.

To prove 2, we use induction on $\max\limits_{(h,s)\in S} h$.

To prove 1(e), note that by 2, $\exists (S'', w'', k) \in M_j$, with $w' \leq w''$. By 1(d), we see that $S'' = S$ and $w'' = w$, therefore $w' \leq w$, and 1(e) follows. $\square$

From part 2 of Lemma 20, we see that if we have an $S \subseteq \{1,\dots,p\} \times \{1,\dots,k\}$ with $\sum\limits_{(i,s)\in S} s = k$ and $\sum\limits_{(i,s)\in S} \delta_s(A_i) = w$, then $\exists (S^*, w^*, k) \in M_p$ (which gives at least as much total weight w^* as w does). Then from part 1(e) of Lemma 20, we see that if $(S^*, w^*, k) \in M_p$, then no other first coordinate satisfying $\sum\limits_{(i,s)\in S} s = k$ will provide a bigger total weight than w^*. Selecting elements of A that correspond to the $(i, \pi_{jr}(i))$ entry of B_{jr} for all $i = 1,\dots,r$ and all $(j, r) \in S$ and adding them up will give w^*, which is the highest possible value, so $w = \delta_k(A)$.

Theorem 21. If $A = \text{blockdiag}(A_1, A_2, \dots, A_p) \in \overline{\mathbb{R}}^{n\times n}$ and we can solve $JRP(A_i, k)$ in $O(t)$ time, for all $i = 1,\dots,p$ and $k = 1,\dots,n$, then we can solve $JRP(A)$ in $O(n(n + t))$ time.

Proof. Correctness follows from Lemma 20. For the time bound, notice that the size of each set M_{j-1} is no greater than n, because there is at most one element in M_{j-1} with the same third component (by 1(d) of Lemma 20). Each update operation of Step 2(c) can be done in constant time for each r and each M_{j-1}, and must be repeated for all $O(n)$ elements of M_{j-1} and $O(n(j))$ times for the r loop. Steps 1, and 2(b) require one operation each so can be performed in constant time. Assume that for each r, Step 2(a) can be performed in $O(t)$

time. The whole of Step 2 is carried out for $j = 1, \ldots, p$. It is easily seen that Step 3 can be done in $O(n^2)$ time. So algorithm JRPBLOCKDIAG runs in time $\sum_{j=1}^{p} O(n(j))t + \sum_{j=1}^{p} O(n)O(n(j)) + O(n^2) = O(n(n+t))$. □

Corollary 22. If in Theorem 21, t is polynomial in n, that is, if $JRP(A_i, k')$ can be solved in polynomial time, for all $i = 1, \ldots, p$ and $k' = 1, \ldots, k$, then for a block diagonal matrix A, $JRP(A)$ can be solved in polynomial time.

Any matrix that can be obtained by permuting the rows and/or columns of the matrix containing zeros on the main diagonal and $-\infty$ elsewhere, will be called a *permutation matrix.* Any matrix that can be obtained by permuting the rows and/or columns of a matrix containing finite entries on the main diagonal and $-\infty$ elsewhere, will be called a *generalized permutation matrix.* It is known [8] that $JRP(A)$ can be solved in $O(n^2)$ time, for a permutation matrix A. With the same time complexity, this is also true for generalized permutation matrices:

Corollary 23. For any generalized permutation matrix $A \in \overline{\mathbb{R}}^{n\times n}$, $JRP(A)$ can be solved in $O(n^2)$ time.

Proof. Generalized permutation matrices are a special type of block diagonal matrix. Each block contains only one cycle, therefore we can solve JRP for each block in linear time, and hence use Theorem 21 to give the result. □

Any element of a matrix that does not lie on a finite cycle may be set to $-\infty$ without affecting δ_k for any $k \in N$. Hence if B is in Frobenius normal form, then we may set all elements of off-diagonal blocks in B to $-\infty$. Therefore if we define $C_i = B_{ii}$, for $i = 1, \ldots, p$, i.e.

$$C = \text{blockdiag}(C_1, C_2, \ldots, C_p),$$

then we have $\delta_k(C) = \delta_k(A)$ for all $k \in N$. We have derived the following:

Theorem 24. For any $A \in \overline{\mathbb{R}}^{n\times n}$, if we can solve JRP for all diagonal blocks of the Frobenius normal form of A in polynomial time, then we can solve $JRP(A)$ in polynomial time (by converting it to a block diagonal matrix and using the JRPBLOCKDIAG algorithm of Figure 2).

Acknowledgements

The authors are grateful to anonymous referees for suggestions that enabled a substantial improvement to the presentation of the paper.

References

[1] Ahuja, R.K., Magnanti, T., and Orlin J.B., *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, New Jersey, 1993

[2] Baccelli, F.L., G. Cohen, G.-J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity.* (J.Wiley and Sons, New York, 1992).

[3] Burkard, R.E., Selected topics in assignment problems., *Discrete Applied Mathematics* **123** (2002), 257-302.

[4] Burkard, R.E. and P. Butkovič, Finding all essential terms of a characteristic maxpolynomial, *Discrete Applied Mathematics* **130** (2003), 367-380.

[5] Burkard, R.E. and P. Butkovič, Max algebra and the linear assignment problem, *Math. Program., Ser.B* **98** (2003), 415-429.

[6] Burkard, R.E. and E. Çela, Linear Assignment Problems and Extensions, in *Handbook of Combinatorial Optimization* (P.M.Pardalos and D.-Z.Du, Eds.), Supplement Volume A, Kluwer Academic Publishers, 75-149, 1999.

[7] Burkard, R.E. and R.A. Cuninghame-Green, private communication.

[8] Butkovič, P., On the complexity of computing the coefficients of max-algebraic characteristic polynomial and characteristic equation. *Kybernetika* **39** (2003), No. 2, 129–136.

[9] Butkovič, P. and R.A. Cuninghame-Green, The linear assignment problem for special matrices. *IMA journal of Management Mathematics* **15** (2004), 1–12.

[10] Butkovič, P. and L. Murfitt, Calculating essential terms of a characteristic maxpolynomial. *CEJOR* **8** (2000), 237–246.

[11] Cuninghame-Green, R.A., *Minimax Algebra. Lecture Notes in Economics and Math. Systems* **166**, (Springer, Berlin, 1979).

[12] Cuninghame-Green, R.A., The characteristic maxpolynomial of a matrix. *J. Math. Analysis and Applications* **95** (1983), 110–116.

[13] Cuninghame-Green, R.A., Minimax Algebra and Applications in: *Advances in Imaging and Electron Physics*. Vol. **90**, pp. 1–121 (Academic Press, New York, 1995).

[14] M. Develin, F. Santos and B. Sturmfels: On the rank of a tropical matrix, arXiv:math.CO/0312114.

[15] Gassner, E., Variants of the Assignment and of the Transportation Problem, PhD thesis, Graz, 2004.

[16] Gondran M., and M. Minoux, Linear Algebra of Dioïds: a survey of recent results. *Annals of Discrete Mathematics* **19** (1984), 147-164.

[17] Lewis, S. C., On the Best Pricipal Submatrix Problem, PhD thesis, University of Birmingham, in preparation.

[18] Rosen, K. H. (ed), Handbook of discrete and combinatorial mathematics, Boca Raton; London : CRC Press, 2000.

[19] van Leeuwen, J. (ed), Graph Algorithms, in Handbook of theoretical computer science - Vol.A : Algorithms and complexity, 525-631, Amsterdam; Oxford : Elsevier; Cambridge, MA : MIT Press, 1990.

[20] Zimmermann, U., *Linear and Combinatorial Optimization in Ordered Algebraic Structures*. Annals of Discrete Mathematics **10** (North Holland, Amsterdam, 1981).