Chapter 2
Max-algebra: Two Special Features

The aim of this chapter is to highlight two special features of max-algebra which make it unique as a modelling and solution tool: the ability to efficiently describe *all* solutions to some problems where it would otherwise be awkward or impossible to do so; and the potential to describe combinatorial problems algebraically.

First we show an example of a problem where max-algebra can help to efficiently find all solutions and, consequently, find a solution satisfying additional requirements (Sect. 2.1).

Then in Sect. 2.2 we show that using max-algebra a number of combinatorial and combinatorial optimization problems can be formulated in algebraic terms. Based on this max-algebra may, to some extent, be considered "an algebraic encoding" of combinatorics [27].

This chapter may be skipped without loss of continuity in reading this book.

2.1 Bounded Mixed-integer Solution to Dual Inequalities: A Mathematical Application

2.1.1 Problem Formulation

A special feature of max-algebra is the ability to efficiently describe the set of all solutions to some problems in contrast to standard approaches, using which we can usually find one solution. Finding all solutions may be helpful for identifying solutions that satisfy specific additional requirements. As an example consider the systems of the form

$$x_i - x_j \geq b_{ij} \quad (i, j = 1, \ldots, n) \tag{2.1}$$

where $B = (b_{ij}) \in \mathbb{R}^{n \times n}$. In [55] the matrix of the left-hand side coefficients of this system is called the *dual network matrix*. It is the transpose of the constraint matrix of a circulation problem in a network (such as the maximum flow or minimum-cost

P. Butkovič, *Max-linear Systems: Theory and Algorithms,*
Springer Monographs in Mathematics 151,
DOI 10.1007/978-1-84996-299-5_2, © Springer-Verlag London Limited 2010

flow problem) and inequalities of the form (2.1) therefore appear as dual inequalities for this type of problems. These facts motivate us to call (2.1) the *system of dual inequalities* (SDI). The aim of this section is to show that using standard max-algebraic techniques it is possible to generate the set of all solutions to (2.1) (which is of size $n^2 \times n$) using n generators. This description enables us then to find, or to prove that it does not exist, a *bounded mixed-integer solution* to the system of dual inequalities, that is, a vector $x = (x_1, \ldots, x_n)^T$ satisfying:

$$\left.\begin{array}{ll} x_i - x_j \geq b_{ij}, & (i, j \in N) \\ u_j \geq x_j \geq l_j, & (j \in N) \\ x_j \text{ integer}, & (j \in J) \end{array}\right\} \tag{2.2}$$

where $u = (u_1, \ldots, u_n)^T, l = (l_1, \ldots, l_n)^T \in \mathbb{R}^n$ and $J \subseteq N = \{1, \ldots, n\}$ are given. We will refer to this problem as BMISDI. Note that without loss of generality u_j and l_j may be assumed to be integer for $j \in J$. This type of a system of inequalities has been studied for instance in [55] where it has been proved that a related mixed-integer feasibility question is *NP*-complete.

We will show that, in general, the application of max-algebra leads to a pseudopolynomial algorithm for solving BMISDI. However, an explicit solution is described in the case when B is integer (but still a mixed-integer solution is wanted). This implies that BMISDI can be solved using $O(n^3)$ operations when B is an integer matrix. Note that when $J = \emptyset$ then BMISDI is polynomially solvable since it is a set of constraints of a linear program. When $J = N$ and B is integer then BMISDI is also polynomially solvable since the matrix of the system is totally unimodular [120].

2.1.2 All Solutions to SDI and All Bounded Solutions

The system of inequalities

$$x_i - x_j \geq b_{ij} \quad (i, j \in N)$$

is equivalent to

$$\max_{j \in N} \left(b_{ij} + x_j\right) \leq x_i \quad (i \in N).$$

In max-algebraic notation this reads

$$\sum\nolimits^{\oplus}_{j \in N} b_{ij} \otimes x_j \leq x_i \quad (i \in N)$$

or in the compact form

$$B \otimes x \leq x. \tag{2.3}$$

Recall that using the notation introduced in Sect. 1.6.2 the set of finite solutions to (2.3) is $V_0^*(B)$.

The next theorem is straightforwardly deduced from Theorem 1.6.18.

Theorem 2.1.1 *If $B \in \mathbb{R}^{n\times n}$ then*

1. *$V_0^*(B) \neq \emptyset$ if and only if $\lambda(B) \leq 0$.*
2. *If $V_0^*(B) \neq \emptyset$ then*

$$V_0^*(B) = \left\{\Delta(B) \otimes z; z \in \mathbb{R}^n\right\}.$$

We can now use Theorems 2.1.1 and 1.6.25 to describe all bounded solutions to SDI.

Corollary 2.1.2 *The set of all solutions x to SDI satisfying $x \leq u$ is*

$$\left\{\Delta(B) \otimes z; z \leq (\Delta(B))^* \otimes' u\right\}$$

and if this set is nonempty then the vector $\Delta(B) \otimes ((\Delta(B))^* \otimes' u)$ is the greatest element of this set. Hence the inequality

$$l \leq \Delta(B) \otimes \left((\Delta(B))^* \otimes' u\right)$$

is necessary and sufficient for the existence of a solution to SDI satisfying $l \leq x \leq u$.

2.1.3 Solving BMISDI

We start with another corollary to Theorem 2.1.1.

Corollary 2.1.3 *A necessary condition for BMISDI to have a solution is that $\lambda(B) \leq 0$. If this condition is satisfied then BMISDI is equivalent to finding a vector $z \in \mathbb{R}^n$ such that*

$$l \leq \Delta(B) \otimes z \leq u$$

and

$$(\Delta(B) \otimes z)_j \in \mathbb{Z} \quad \text{for } j \in J.$$

In the rest of this subsection we will assume without loss of generality (Theorem 2.1.1) that $\lambda(B) \leq 0$.

Theorem 2.1.4 *Let $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $J \subseteq N$. Let $\tilde{b}$ be defined by*

$$\tilde{b}_j = \lfloor b_j \rfloor \quad \text{for } j \in J,$$
$$\tilde{b}_j = b_j \quad \text{for } j \notin J.$$

Then the following are equivalent:

1. *There exists a $z \in \mathbb{R}^n$ such that $l \leq A \otimes z \leq b$ and*

$$(A \otimes z)_j \in \mathbb{Z} \quad \text{for } j \in J.$$

2. *There exists a* $z \in \mathbb{R}^n$ *such that* $l \leq A \otimes z \leq \tilde{b}$ *and*

$$(A \otimes z)_j \in \mathbb{Z} \quad \text{for} \in J.$$

3. *There exists a* $z \in \mathbb{R}^n$ *such that* $l \leq A \otimes z \leq A \otimes (A^* \otimes' \tilde{b})$ *and*

$$(A \otimes z)_j \in \mathbb{Z} \quad \text{for } j \in J.$$

Proof 1. $\Longleftrightarrow$ 2. is trivial, 2. $\Longleftrightarrow$ 3. follows from Theorem 1.6.25, Corollary 1.6.26 and Lemma 1.1.1. □

Theorem 2.1.4 enables us to compile the following algorithm.

Algorithm 2.1.5 BMISDI
Input: $B \in \mathbb{R}^{n \times n}, u, l \in \mathbb{R}^n$ and $J \subseteq N$.
Output: x satisfying (2.2) or an indication that no such vector exists.

1. $A := \Delta(B), x := u$
2. $x_j := \lfloor x_j \rfloor$ for $j \in J$
3. $z := A^* \otimes' x, x := A \otimes z$
4. If $l \nleq x$ then stop (no solution)
5. If $l \leq x$ and $x_j \in \mathbb{Z}$ for $j \in J$ then stop else go to 2.

Theorem 2.1.6 [30] *The algorithm BMISDI is correct and requires* $O(n^3 + n^2 L)$ *operations of addition, maximum, minimum, comparison and integer part, where*

$$L = \sum\nolimits_{j \in J} \left(u_j - l_j\right).$$

Proof If the algorithm terminates at step 4 then there is no solution by the repeated use of Theorem 2.1.4.

The sequence of vectors x constructed by this algorithm is nonincreasing by Corollary 1.6.26 and hence $x = A \otimes z \leq u$ if it terminates at step 5. The remaining requirements of (2.2) are satisfied explicitly due to the conditions in step 5.

Computational complexity: The calculation of $\Delta(B)$ is $O(n^3)$ by Theorem 1.6.22. Each run of the loop between steps 2 and 5 is $O(n^2)$. In every iteration at least one component of $x_j, j \in J$ decreases by one and the statement now follows from the fact that all x_j range between l_j and u_j. □

Example 2.1.7 Let

$$B = \begin{pmatrix} -2 & 2.7 & -2.1 \\ -3.8 & -1 & -5.2 \\ 1.6 & 3.5 & -3 \end{pmatrix},$$

$u = (5.2, 0.8, 7.4)^T$ and $J = \{1, 3\}$ (l is not specified). The algorithm BMISDI will find:

$$A = \Delta(B) = \begin{pmatrix} 0 & 2.7 & -2.1 \\ -3.6 & 0 & -5.2 \\ 1.6 & 4.3 & 0 \end{pmatrix},$$

$$x = (5, 0.8, 7)^T,$$

$$z = A^* \otimes' x = \begin{pmatrix} 0 & 3.6 & -1.6 \\ -2.7 & 0 & -4.3 \\ 2.1 & 5.2 & 0 \end{pmatrix} \otimes' x = \begin{pmatrix} 4.4 \\ 0.8 \\ 6 \end{pmatrix}$$

and

$$x = A \otimes z = (4.4, 0.8, 6)^T.$$

Now $x_1 \notin \mathbb{Z}$ so the algorithm continues by another iteration: $x = (4, 0.8, 6)^T$,

$$z = A^* \otimes' x = (4, 0.8, 6)^T$$

and

$$x = A \otimes z = (4, 0.8, 6)^T,$$

which is a solution (provided that $l \leq x$ since otherwise there is no solution) to the BMISDI since $x_1, x_3 \in \mathbb{Z}$.

2.1.4 Solving BMISDI for Integer Matrices

In this subsection we prove that a solution to the BMISDI can be found explicitly if B is integer. The following will be useful (the proof below is a simplification of the original proof due to [132]):

Theorem 2.1.8 [30] *Let $A \in \mathbb{Z}^{n\times n}, b \in \mathbb{R}^n$ and $A \otimes x = b$ for some $x \in \mathbb{R}^n$. Let $J \subseteq N$ and $\tilde{b}$ be defined by*

$$\tilde{b}_k = \lfloor b_k \rfloor \quad \text{for } \in J,$$

$$\tilde{b}_k = b_k \quad \text{for } k \notin J.$$

Then there exists an $\tilde{x} \in \mathbb{R}^n$ such that

$$A \otimes \tilde{x} \leq \tilde{b}$$

and

$$(A \otimes \tilde{x})_k = \tilde{b}_k \quad \text{for } k \in J.$$

Proof Without loss of generality assume that $b_k \notin \mathbb{Z}$ for some $k \in J$, then the set

$$S = \{s \in N; a_{ks} + x_s > \lfloor b_k \rfloor \text{ for some } k \in J\}$$

is nonempty and $x_s \notin \mathbb{Z}$ for every $s \in S$ since A is integer. Let $\tilde{x} \in \mathbb{R}^n$ be defined by $\tilde{x}_j = \lfloor x_j \rfloor$ for $j \in S$ and $\tilde{x}_j = x_j$ otherwise. Clearly $\tilde{x} \leq x$ and so $A \otimes \tilde{x} \leq A \otimes x$ by Lemma 1.1.1. Hence $\max_{j \in N}(a_{kj} + \tilde{x}_j) \leq b_k = \tilde{b}_k$ for all $k \notin J$. At the same time $\max_{j \in N}(a_{kj} + \tilde{x}_j) = \lfloor b_k \rfloor = \tilde{b}_k$ for all $k \in J$. □

For the main application, Theorem 2.1.10 below, it will be convenient to deduce from the statement of Theorem 2.1.8 a property of the greatest solution $\overline{x}$ to $A \otimes x \leq \tilde{b}$ (Corollary 1.6.26):

Corollary 2.1.9 *Under the assumptions of Theorem 2.1.8 and using the same notation, if* $\overline{x} = A^* \otimes' \tilde{b}$ *then*

$$A \otimes \overline{x} \leq \tilde{b}$$

and

$$(A \otimes \overline{x})_k = \tilde{b}_k \quad \text{for } k \in J.$$

Proof The inequality follows from Corollary 1.6.26. Let $\tilde{x}$ be the vector described in Theorem 2.1.8. By Theorem 1.6.25 we have $\tilde{x} \leq \overline{x}$ implying that

$$\tilde{b}_k = (A \otimes \tilde{x})_k \leq (A \otimes \overline{x})_k \leq \tilde{b}_k \quad \text{for } k \in J$$

which concludes the proof. □

Finally, we are prepared to use max-algebra and explicitly describe a solution to BMISDI in the case when B is an integer matrix:

Theorem 2.1.10 *Let* $B \in \mathbb{Z}^{n \times n}$, $\lambda(B) \leq 0$, $A = \Delta(B)$, $b = A \otimes (A^* \otimes' u)$ *and* $\tilde{b}$ *be defined by*

$$\tilde{b}_k = \lfloor b_k \rfloor \quad \text{for } k \in J$$

and

$$\tilde{b}_k = b_k \quad \text{for } k \notin J.$$

Then the BMISDI has a solution if and only if

$$l \leq A \otimes \left(A^* \otimes' \tilde{b}\right),$$

and $\hat{x} = A \otimes (A^* \otimes' \tilde{b})$ *is then the greatest solution (that is,* $y \leq \hat{x}$ *for any solution* y*).*

Proof Note first that A is an integer matrix and we therefore may apply Corollary 2.1.9 to A.

"If": By Corollary 1.6.26 $\hat{x} \leq \tilde{b} \leq b \leq u$. Let us take in Corollary 2.1.9 (and Theorem 2.1.8) $x = A^* \otimes' u$. Then $\hat{x} = A \otimes \overline{x}$ and so $\hat{x}_k \in \mathbb{Z}$ for $k \in J$.

"Only if": Let y be a solution. Then $y = A \otimes w \leq u$ for some $w \in \mathbb{R}^n$, thus by Theorem 1.6.25

$$w \leq A^* \otimes' u$$

and so

$$y = A \otimes w \leq A \otimes \left(A^* \otimes' u\right) = b.$$

Since $y_k \in \mathbb{Z}$ for $k \in J$ we also have

$$A \otimes w = y \leq \tilde{b}.$$

Hence by Theorem 1.6.25

$$w \leq A^* \otimes' \tilde{b}$$

and by Lemma 1.1.1 then

$$l \leq y = A \otimes w \leq A \otimes \left(A^* \otimes' \tilde{b}\right) = \hat{x}.$$

We also have $\hat{x} \leq \tilde{b} \leq b \leq u$ by Corollary 1.6.26 and $\hat{x}_k \in \mathbb{Z}$ for $k \in J$ by Corollary 2.1.9 as above, hence $\hat{x}$ is the greatest solution. □

Example 2.1.11 Let

$$B = \begin{pmatrix} -2 & 2 & -2 \\ -3 & -1 & -4 \\ 1 & 3 & -3 \end{pmatrix},$$

$u = (3.5, 0.8, 5.7)^T$ and $J = \{1, 3\}$ (l is not specified). Then we have:

$$A = \Delta(B) = \begin{pmatrix} 0 & 2 & -2 \\ -3 & 0 & -4 \\ 1 & 3 & 0 \end{pmatrix},$$

$$A^* \otimes' u = \begin{pmatrix} 0 & 3 & -1 \\ -2 & 0 & -3 \\ 2 & 4 & 0 \end{pmatrix} \otimes' u = \begin{pmatrix} 3.5 \\ 0.8 \\ 4.8 \end{pmatrix},$$

$$b = A \otimes \left(A^* \otimes' u\right) = \begin{pmatrix} 3.5 \\ 0.8 \\ 4.8 \end{pmatrix},$$

$$\tilde{b} = \begin{pmatrix} 3 \\ 0.8 \\ 4 \end{pmatrix}$$

and

$$\hat{x} = A \otimes \left(A^* \otimes' \tilde{b} \right) = (3, 0.8, 4)^T .$$

By Theorem 2.1.10 $\hat{x}$ is the greatest solution to the BMISDI provided that $l \leq \hat{x}$ (otherwise there is no solution).

2.2 Max-algebra and Combinatorial Optimization

There is a number of combinatorial and combinatorial optimization problems closely related to max-algebra. In some cases max-algebra provides an efficient and elegant algebraic encoding of these problems. Although computational advantages do not necessarily follow from the max-algebraic formulation, for some problems this connection may help to deduce useful information [27].

2.2.1 Shortest/Longest Distances: Two Connections

Perhaps the most striking example is the *shortest-distances problem* which is one of the best known combinatorial optimization problems:

Given an $n \times n$ matrix A of direct distances between n places, find the matrix $\tilde{A}$ of shortest distances (that is, the matrix of the lengths of shortest paths between any pair of places).

It is known that the shortest-distances matrix exists if and only if there are no negative cycles in D_A. For the shortest-distances problem we may assume without loss of generality that all diagonal elements of A are 0.

We could continue from this and show a link to min-algebra; however, to be consistent with the rest of the book we shall formulate these results in max-algebraic terms, similarly as in Example 1.2.3. Hence the considered combinatorial optimization problem is:

Given an $n \times n$ matrix A of direct distances between n places, find the matrix $\tilde{A}$ of longest distances (that is, the matrix of the lengths of longest paths between any pair of places).

We may assume that all diagonal elements of A are 0 and that there are no positive cycles in D_A, thus A is strongly definite. We have seen in Sect. 1.6.2 that $\Gamma(A)$ is exactly $\tilde{A}$. By Proposition 1.6.12 then $\tilde{A} = A^{n-1}$. We have:

Theorem 2.2.1 *If $A \in \mathbb{R}^{n \times n}$ is a strongly definite direct-distances matrix then all matrices A^j $(j \geq n-1)$ are equal to the longest-distances matrix for D_A. Hence, the kth column $(k = 1, \ldots, n)$ of A^j $(j \geq n-1)$ is the vector of longest distances to node k in D_A.*

One benefit of this result is that the longest- (and similarly shortest-) distances matrix for a strongly definite direct-distances matrix A can be found simply by repeated max-algebraic squaring of A, that is,

$$A^2, A^4, A^8, A^{16}, \ldots$$

until a power A^j $(j \geq n-1)$ is reached (see Sect. 1.6.2).

However, there exists another max-algebraic interpretation of the longest-distances problem. We have seen in Proposition 1.6.17 that for a strongly definite matrix A every column v of A^j $(j \geq n-1)$ is an eigenvector of A, that is,

$$A \otimes v = v.$$

Corollary 2.2.2 *If $A \in \mathbb{R}^{n\times n}$ is a strongly definite direct-distances matrix then every vector of longest-distances to a node in D_A is a max-algebraic eigenvector of A corresponding to the eigenvalue* 0.

2.2.2 Maximum Cycle Mean

The maximum cycle mean of a matrix (denoted $\lambda(A)$ for a matrix A), has been defined in Sect. 1.6.1. As already mentioned, the problem of calculating $\lambda(A)$ was studied independently in combinatorial optimization [106, 109]. At the same time the maximum cycle mean is very important in max-algebra. It is

- the eigenvalue of every matrix,
- the greatest eigenvalue of every matrix,
- the only eigenvalue whose corresponding eigenvectors may be finite.

Moreover, every eigenvalue of a matrix is the maximum cycle mean of some principal submatrix of that matrix.

All these and other aspects of the maximum cycle mean are proved in Chap. 4. Let us mention here a dual feature of the maximum cycle mean (see Corollary 4.5.6 and Theorem 1.6.29):

Theorem 2.2.3 *If $A \in \overline{\mathbb{R}}^{n\times n}$ then*

(a) $\lambda(A)$ *is the greatest eigenvalue of A, that is*

$$\lambda(A) = \max\left\{\lambda \in \overline{\mathbb{R}};\ A \otimes x = \lambda \otimes x, x \in \overline{\mathbb{R}}^n, x \neq \varepsilon\right\}$$

and, dually

(b)

$$\lambda(A) = \inf\left\{\lambda \in \overline{\mathbb{R}};\ A \otimes x \leq \lambda \otimes x, x \in \mathbb{R}^n\right\}.$$

2.2.3 The Job Rotation Problem

Characteristic maxpolynomials of matrices in max-algebra (Sect. 5.3) are related to the following *job rotation problem*. 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, guards in a gallery or ride operators in a theme park). It may also be required that to maintain stability of service only a certain number of employees, say k ($k < n$), actually swap their jobs. With each pair old job−new job a quantity may be associated expressing the cost (for instance for additional training) or the preference of the worker for this particular change. So the aim may be to select k employees and to suggest a schedule of the job swaps between them so that the sum of the parameters corresponding to these changes is either minimum or maximum. This task leads to finding a $k \times k$ principal submatrix of A for which the optimal assignment problem value is minimal or maximal (some entries can be set to $+\infty$ or $-\infty$ to avoid an assignment to the same or infeasible job). More formally, we deal with the *best principal submatrix problem* (BPSM):

Given a real $n \times n$ matrix A, for every $k \leq n$ find a $k \times k$ principal submatrix of A whose optimal assignment problem value is maximal.

Note that solving the assignment problem for all $\binom{n}{k}$ principal submatrices for each k would be computationally difficult since $\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$. No polynomial method for solving BPSM seems to be known, although its modification obtained after removing the word principal is known [73] and is polynomially solvable. This can also be seen from the following simple observation: Let $\widetilde{A}$ be the $(2n-k) \times (2n-k)$ matrix obtained from an $n \times n$ matrix $A \in \mathbb{R}^{n \times n}$ by adding $n-k$ rows and $n-k$ columns ($k < n$) so that the entries in the intersection of these columns are $-\infty$ and the remaining new entries are zero, see Fig. 2.1. If the assignment problem is solved for $\widetilde{A}$ then every permutation selects $2n-k$ entries from $\widetilde{A}$. If A is finite then any optimal (maximizing) permutation avoids selecting entries from the intersection of the new columns and rows. But as it selects $n-k$ elements from the new rows and $n-k$ different elements from the new columns, it will select exactly $2n-k-2(n-k) = k$ elements from A. No two of these k elements are from the same row or from the same column and so they represent a selection of k independent entries from a $k \times k$ submatrix of A. Their sum is maximum as the only elements taken from outside A are zero. So the best $k \times k$ submatrix problem can readily be solved as the classical assignment problem for a special matrix of order $2n-k$.

Unfortunately no similar trick seems to exist, that would enable us to find a best *principal* submatrix.

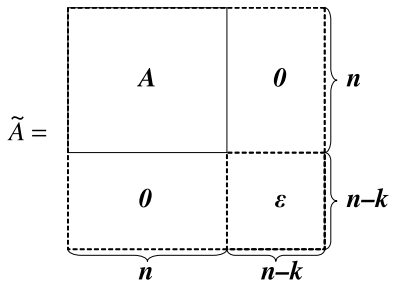

Fig. 2.1 Solving the best submatrix problem

Let us denote by δ_k the optimal value in the assignment problem for a best principal submatrix of order k ($k = 1, \ldots, n$). It will be proved in Sect. 5.3 that $\delta_1, \ldots, \delta_n$ are coefficients of the max-algebraic characteristic polynomial of A. It is not known whether the problem of finding all these quantities is an *NP*-complete or polynomially solvable problem (see Chap. 11). However, in Sect. 5.3.3 we will present a polynomial algorithm, based on the max-algebraic interpretation, for finding some and in some cases all these coefficients. Note that there is an indication that the problem of finding all coefficients is likely to be polynomially solvable as the following result suggests:

Theorem 2.2.4 [20] *If the entries of $A \in \mathbb{R}^{n\times n}$ are polynomially bounded, then the best principal submatrix problem for A and all k, $k \leq n$, can be solved by a randomized polynomial algorithm.*

2.2.4 Other Problems

In the table below (where SD stands for "strongly definite") is an overview of combinatorial or combinatorial optimization problems that can be formulated as max-algebraic problems [27]. The details of most of these links will be presented in the subsequent chapters.

Max-algebra	**Combinatorics (0-1 entries)**	**Combinatorial Optimization**
maper(A)	Term rank	Optimal value to the assignment problem
$A \otimes x = b$		
$\exists x$	Set covering	
$\exists! x$	Minimal set covering	
$\Gamma(A)$ if A SD	Transitive closure	Longest distances matrix
$A \otimes x = \lambda \otimes x$		
λ		Maximum cycle mean
x		Balancing coefficients
x if A SD	Connectivity to a node	Longest distances
x if A SD		Scaling to normal form
GM regularity	$\nexists$ even directed cycle 0-1 sign-nonsingularity	All optimal permutations of the same parity
Strong regularity	Digraph acyclic	Unique optimal permutation
Characteristic polynomial	$\exists$ exact cycle cover $\exists$ principal submatrix with > 0 permanent	Best principal submatrix (JRP)

2.3 Exercises

Exercise 2.3.1 The assignment problem for $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ can be described as a (conventional) linear program

$$f(x) = \sum_{i,j\in N} a_{ij}x_{ij} \longrightarrow \max$$

s.t.

$$\sum_{j\in N} a_{ij}x_{ij} = 1, \quad i \in N,$$

$$\sum_{i\in N} a_{ij}x_{ij} = 1, \quad j \in N,$$

$$x_{ij} \geq 0.$$

Its dual is

$$g\,(u, v) = \sum_{i\in N} u_i + \sum_{j\in N} v_j \longrightarrow \min$$

s.t.

$$u_i + v_j \geq a_{ij}, \quad i, j \in N.$$

Show using max-algebra that $f^{\max} = g^{\min} = \text{maper}(A)$.

(Hint: First show that $f \leq g$ and then prove the rest by using the results on the eigenproblem for strongly definite matrices.)

Exercise 2.3.2 A matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is called pyramidal if $a_{ij} \geq a_{rs}$ whenever $\max(i, j) < \max(r, s)$. Prove that $\delta_k = \text{maper}(A_k)$, where A_k is the principal submatrix of A determined by the first k row and column indices. [See [37].]

http://www.springer.com/978-1-84996-298-8

Max-linear Systems: Theory and Algorithms

Butkovič, P.

2010, XVIII, 274 p., Hardcover
ISBN: 978-1-84996-298-8