Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 154 (2006) 437–446

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Communication

A strongly polynomial algorithm for solving two-sided linear systems in max-algebra[☆]

Peter Butkovič[a,*], Karel Zimmermann[b,1]

[a]School of Mathematics, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK
[b]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Received 14 September 2005; accepted 26 September 2005
Available online 27 October 2005
Communicated by R.E. Burkard

Abstract

An algorithm for solving $m \times n$ systems of $(max, +)$-linear equations is presented. The systems have variables on both sides of the equations. After $O(m^4n^4)$ iterations the algorithm either finds a solution of the system or finds out that no solution exists. Each iteration needs $O(mn)$ operations so that the complexity of the presented algorithm is $O(m^5n^5)$.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Max-algebra; Two-sided equation; Strongly polynomial algorithm

1. Introduction

Consider an industrial process in which products $P_1, \ldots, P_m$ are prepared in n workshops, each workshop contributing to the completion of each product by producing a semiproduct. It is assumed that every workshop can work for all products simultaneously and that these works start as soon as the workshops open. Let a_{ij} be the duration of the work of the jth workshop needed to complete the semiproduct for P_i $(i = 1, \ldots, m;\ j = 1, \ldots, n)$. Let us denote by x_j the starting time of the jth workshop $(j = 1, \ldots, n)$. Then all semiproducts for P_i $(i = 1, \ldots, m)$ will be ready at time $\max(x_1 + a_{i1}, \ldots, x_n + a_{in})$. Now suppose that independently, k other workshops prepare semiproducts for products $Q_1, \ldots, Q_m$ and the duration and starting times are b_{ij} and y_j, respectively. Then the synchronisation problem is to find starting times of all $n + k$ workshops so that each pair (P_i, Q_i) $(i = 1, \ldots, m)$ is completed at the same time. This task is equivalent to solving the system of equations

$$\max(x_1 + a_{i1}, \ldots, x_n + a_{in}) = \max(y_1 + b_{i1}, \ldots, y_k + b_{ik}) \quad (i = 1, \ldots, m).$$

[☆] Research supported by the LMS Grant 4417.
[*] Corresponding author.
E-mail addresses: p.butkovic@bham.ac.uk (P. Butkovič), Karel.Zimmermann@MFF.CUNI.CZ (K. Zimmermann).
[1] Research supported by the Grant GA ČR 202-03/2060982.

0166-218X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.09.008

If we denote $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$ for $a, b \in \mathbb{R}$ then this system has the form

$$\sum_{j=1,\ldots,n}^{\oplus} a_{ij} \otimes x_j = \sum_{j=1,\ldots,k}^{\oplus} b_{ij} \otimes y_j \quad (i = 1, \ldots, m).$$

Therefore, this system is called a system of (max, +) linear equations. It is easily shown [5] that the above system can be straightforwardly transformed to another one where $k = n$ and $x_j = y_j$ $(j = 1, \ldots, n)$.

Systems of (max, +) or (max, ·)-linear equations were investigated already in the first publications dealing with the introduction of algebraic structures called (max, +)-algebras (sometimes also extremal algebras or path algebras). In these publications, systems of equations with variables only on one side were considered [4,7]. Other systems with a special structure were studied in the context of solving eigenvalue problems in the corresponding algebraic structures or synchronization of discrete event systems [1,3,4]. Using the $(\oplus, \otimes)$-notation [4], the systems have one of the following forms: $A \otimes x = b$, $A \otimes x = x$ or $A \otimes x = x \oplus b$, where A is a given matrix, b is a given vector of an appropriate size and $(\oplus, \otimes) = (\max, +)$. In this article, we consider systems of equations with general (max, +) -linear functions on both sides of the systems (i.e. systems of the form $A \otimes x = B \otimes x$, where A, B are $m \times n$ matrices with finite rational entries). However, we do not use the $(\oplus, \otimes)$-notation in the present paper as the method introduced here is purely numerical and does not use any theoretical results previously obtained in max-algebra.

General two-sided linear systems in max-algebra have been studied in a number of papers, e.g. [2,5,6,8]. A general solution method was presented in [8], however, no complexity bound was given. In [5] a pseudopolynomial algorithm has been developed. In [2] it was shown that the solution set is generated by a finite number of vectors. A general iterative approach suggested in [6] assumes that finite upper and lower bounds for all variables are given. The iterative method presented in [6] makes it possible to find an approximation of the maximum solution of the given system, which satisfies the given lower and upper bounds or to find out that no such solution exists. In the latter case, the method does not answer the question whether no solution exists only because of the lower and upper bounds or whether there exists no solution of the given system at all (independently of any lower or upper bound). Besides, the method in [6] is constructed for general equations with residuated functions on both sides and cannot take advantage of the special structure of (max, +)-linear systems with finite entries. We show that the method suggested in this article answers the solvability question after at most $O(m^4n^4)$ iterations and each iteration has a complexity $O(mn)$ (m equations with n variables), so that the complexity of the suggested algorithm is $O(m^5n^5)$. If the system has a solution, the method finds the maximum solution smaller or equal to a given finite upper bound vector; no lower bound is considered. Since the (max, +)-linear functions are +-homogeneous, it solves the solvability problem completely, because if $x = (x_1, \ldots, x_n)$ is a solution of the system and α an appropriately chosen rational number, then $(x_1 + \alpha, \ldots, x_n + \alpha)$ is smaller or equal than the given upper bound and solves the system too (see Lemma 2.1 below).

2. Formulation of the problem

We will use the following notation, assuming that m and n are given integers:
$N = \{1, \ldots, n\}$, $S = \{1, \ldots, m\}$, $\mathbb{Q}$ is the set of rational numbers,
$\mathbb{Q}^n = \mathbb{Q} \times \cdots \times \mathbb{Q}$ (n-times), $x = (x_1, \ldots, x_n) \in \mathbb{Q}^n$.
Throughout the paper we assume that a_{ij}, b_{ij} $(i \in S, j \in N)$ are given rational numbers.
We denote for all $i \in S$:

$$a_i(x) \equiv \max_{j \in N} (a_{ij} + x_j),$$
$$b_i(x) \equiv \max_{j \in N} (b_{ij} + x_j).$$

The aim of this paper is to develop a strongly polynomial algorithm which finds a solution to the system

$$a_i(x) = b_i(x), \quad i \in S \tag{1}$$

or decides that no such solution exists.

The set of all rational solutions of (1) will be denoted by M. We further define for any $\overline{x} \in \mathbb{Q}^n$:

$$M(\overline{x}) \equiv \{x | x \in M \,\&\, x \leqslant \overline{x}\}. \tag{2}$$

Note that the functions $a_i(x)$, $b_i(x)$ are $+$-homogeneous, i.e. if $\alpha \in \mathbb{Q}$, $x \in M$ then also $(x_1 + \alpha, \dots, x_n + \alpha) \in M$.

Lemma 2.1. *Let $\overline{x} \in \mathbb{Q}^n$ be given. Then*

$$M \neq \emptyset \quad \textit{if and only if} \quad M(\overline{x}) \neq \emptyset.$$

Proof. Since $M(\overline{x}) \subseteq M$, it remains to prove the "only if" part. If $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_n) \in M$, we can find $\alpha \in \mathbb{Q}$ such that $\tilde{x}' = (\tilde{x}_1 + \alpha, \dots, \tilde{x}_n + \alpha) \leqslant \overline{x}$ and since $\tilde{x}' \in M$ we have $\tilde{x}' \in M(\overline{x})$. □

Let $L \subseteq \mathbb{Q}^n$, $\tilde{x} \in L$. Then $\tilde{x}$ is called the *maximum element* of L if $x \leqslant \tilde{x}$ for every $x \in L$.

Remark 2.1. The algorithm presented in the next section will either find the maximum element of $M(\overline{x})$ for a given upper bound $\overline{x}$ or find out that $M(\overline{x})$ is empty (and thus by Lemma 2.1 the set M is empty too). If $\overline{x} \in M(\overline{x})$, then $\overline{x}$ is the maximum element of $M(\overline{x})$. Let us note that the existence of the maximum element in $M(\overline{x})$, if $M(\overline{x}) \neq \emptyset$, was proved in a much more general context [4].

3. The algorithm

Let us assume that $\overline{x} \notin M(\overline{x})$. We may also assume w.l.o.g. that

$$a_i(\overline{x}) \geqslant b_i(\overline{x}), \quad i \in S \tag{3}$$

(otherwise we swap the inequality sides appropriately). Let

$$H(\overline{x}) \equiv \{i \in S | a_i(\overline{x}) > b_i(\overline{x})\}.$$

Since $\overline{x} \notin M(\overline{x})$, the set $H(\overline{x})$ is non-empty. It follows from (3) that $a_i(\overline{x}) = b_i(\overline{x})$ for all $i \in S \backslash H(\overline{x})$.
We denote for all $i \in S$:

$$F_i(x) = \{j \in N | a_{ij} + x_j = a_i(x)\},$$
$$G_i(x) = \{j \in N | b_{ij} + x_j = b_i(x)\}.$$

Variables x_j, $j \in F_i(x)$ and x_j, $j \in G_i(x)$ will be called *active variables* at point x of $a_i(x)$, $b_i(x)$, respectively.
If we want to find an element of $M(\overline{x})$, it is necessary to decrease $a_i(\overline{x})$ for all $i \in H(\overline{x})$; for this it is necessary to decrease all active variables in $a_i(\overline{x})$, $i \in H(\overline{x})$, i.e. all variables x_j, $j \in A(\overline{x})$, where

$$A(\overline{x}) \equiv \bigcup_{i \in H(\overline{x})} F_i(\overline{x}). \tag{4}$$

Besides, we need to ensure that in the process of decreasing x_j, $j \in A(\overline{x})$, the equalities for $i \in S \backslash H(\overline{x})$ will be preserved. If for instance $a_i(\overline{x}) = b_i(\overline{x})$ for some $i \in S \backslash H(\overline{x})$ and $F_i(\overline{x}) \subseteq A(\overline{x})$ and $G_i(\overline{x}) \nsubseteq A(\overline{x})$, then the decrease of x_j, $j \in A(\overline{x})$ would destroy this equality. To preserve it, we have to include further variables in the set of decreased variables, namely variables x_j, $j \in G_i(\overline{x}) \backslash A(\overline{x})$. We will now describe Algorithm I which after at most $n-1$ iterations finds the set $P(\overline{x})$ of indices of all variables, which have to be decreased, if variables x_j, $j \in A(\overline{x})$are decreased and the equalities $a_i(\overline{x}) = b_i(\overline{x})$ for $i \in S \backslash H(\overline{x})$ have to be preserved.

Algorithm I

```
[1] $P(\bar{x}) := A(\bar{x})$.
[2] $E_1 := \{i \in S\backslash H(\bar{x}) | F_i(\bar{x}) \subseteq P(\bar{x})\ \&\ G_i(\bar{x}) \nsubseteq P(\bar{x})\}$,
    $E_2 := \{i \in S\backslash H(\bar{x}) | F_i(\bar{x}) \nsubseteq P(\bar{x})\ \&\ G_i(\bar{x}) \subseteq P(\bar{x})\}$.
[3] If $E_1 \cup E_2 = \emptyset$, then $P(\bar{x})$is the set of indices of variables to be decreased, STOP.
[4] $P(\bar{x}) := P(\bar{x}) \cup \bigcup_{i \in E_1} (G_i(\bar{x})\backslash P(\bar{x})) \cup \bigcup_{i \in E_2} (F_i(\bar{x})\backslash P(\bar{x}))$, go to [2].
```

Note that at the end of Algorithm I $F_i(\bar{x}) \subseteq P(\bar{x})$ if and only if $G_i(\bar{x}) \subseteq P(\bar{x})$ for every $i \notin H(\bar{x})$. Since $A(\bar{x})$ contains at least one index and in each iteration at least one new index from N is added to $P(\bar{x})$, Algorithm I terminates after at most $(n-1)$ iterations.

We will now describe the process of decreasing variables x_j, $j \in P(\bar{x})$.

Let us define $x(t) = (x_1(t), \ldots, x_n(t))$ for $t \geqslant 0$ as follows:

$$\left.\begin{array}{ll} x_j(t) \equiv \bar{x}_j - t & \text{if } j \in P(\bar{x}), \\ x_j(t) \equiv \bar{x}_j & \text{otherwise.} \end{array}\right\} \tag{5}$$

If we increase parameter *t*, variables x_j, $j \in P(\bar{x})$ will be decreased (i.e. $x_j(0) = \bar{x}_j$, $x_j(t) < \bar{x}_j$ for any $t > 0$ and $j \in P(\bar{x})$).

Note that if we increase *t* to a value $\varepsilon > 0$, where ε is sufficiently small, then for all $i \in S$ such that $F_i(\bar{x}) \subseteq P(\bar{x})$ we have

$$F_i(x(t)) = F_i(\bar{x}), \quad G_i(x(t)) = G_i(\bar{x}).$$

The following hold for any $t \in (0, \varepsilon)$:

$$F_i(\bar{x}) \subseteq P(\bar{x}) \Rightarrow a_i(x(t)) = a_i(\bar{x}) - t, \quad G_i(\bar{x}) \subseteq P(\bar{x}) \Rightarrow b_i(x(t)) = b_i(\bar{x}) - t,$$

and $H(x(t)) = H(\bar{x})$.

Let us first assume that $P(\bar{x}) \neq N$ (it will follow from Theorem 4.2 of the next section that $M(\bar{x}) = \emptyset$, if $H(\bar{x}) \neq \emptyset$ and $P(\bar{x}) = N$). Denote

$$\begin{aligned} L_1 &\equiv \{i \in S | F_i(\bar{x}) \subseteq P(\bar{x})\}, \\ L_2 &\equiv \{i \in S | G_i(\bar{x}) \subseteq P(\bar{x})\}, \\ L_3 &\equiv \{i \in H(\bar{x}) |\ G_i(\bar{x}) \nsubseteq P(\bar{x})\}. \end{aligned}$$

We will increase *t* until for the first time at least one of the following cases occurs:

(i) $F_i(x(t)) \neq F_i(\bar{x})$, i.e. $a_i(x(t)) = \alpha_i(\bar{x}) \equiv \max_{j \in N\backslash P(\bar{x})} (a_{ij} + \bar{x}_j)$ for some $i \in L_1$;
(ii) $G_i(x(t)) \neq G_i(\bar{x})$, i.e. $b_i(x(t)) = \beta_i(\bar{x}) \equiv \max_{j \in N\backslash P(\bar{x})} (b_{ij} + \bar{x}_j)$ for some $i \in L_2$;
(iii) $H(x(t)) \neq H(\bar{x})$, i.e. $a_i(x(t)) = \beta_i(\bar{x})$ for some $i \in L_3$.

If $P(\bar{x}) = N$, we will set $\alpha_i(\bar{x}) = \beta_i(\bar{x}) = -\infty$ for all $i \in S$.

We will determine the values t_1, t_2, t_3 at which the cases (i), (ii), (iii) occur.

If $P(\bar{x}) \neq N$ (i.e. $P(\bar{x}) \subset N$), $\alpha_i(\bar{x})$, $\beta_i(\bar{x})$ are always finite. Since in the cases (i) and (ii) $a_i(x(t)) = a_i(\bar{x}) - t$ and $b_i(x(t)) = b_i(\bar{x}) - t$ respectively, we obtain the following formulae for determining t_1, t_2:

$$t_1 = \min_{i \in L_1} (a_i(\bar{x}) - \alpha_i(\bar{x})), \tag{6}$$

$$t_2 = \min_{i \in L_2} (b_i(\bar{x}) - \beta_i(\bar{x})), \tag{7}$$

In the case (iii) we obtain t_3 as follows:

$$t_3 = \min_{i \in L_3} (a_i(\overline{x}) - \beta_i(\overline{x})). \tag{8}$$

We will set $t_i = \infty$ if $L_i = \emptyset$ $(1 \leqslant i \leqslant 3)$.

We will now set $\tau \equiv \min(t_1, t_2, t_3)$, and use on the next iteration $x(\tau)$ as the new upper bound. If $H(x(\tau)) \neq \emptyset$, we begin the new iteration with this new upper bound.

If $P(\overline{x}) = N$, we will set by definition $t_i = \infty$ for $i = 1, 2, 3$ and thus also $\tau = \infty$. Let us note that if $P(\overline{x}) \neq N$, it is always $\tau \in (0, \infty)$ and thus $x(\tau) \leqslant \overline{x}$ and $x(\tau) \neq \overline{x}$. It will be proved in the next section that if $H(x(\tau)) = \emptyset$, then $x(\tau)$ is the maximum element of $M(\overline{x})$. Further, it will be proved that if $P(\overline{x}) = N$, then $M(\overline{x}) = \emptyset$. The following algorithm summarizes the previous considerations.

Algorithm II

```
[1] $\overline{y} := \overline{x}$.
[2] If $H(\overline{y}) = \emptyset$ then $\overline{y}$ is the maximum element of $M(\overline{x})$, STOP.
[3] Find $A(\overline{y})$ and $P(\overline{y})$ using Algorithm I (with $\overline{x}$ replaced by $\overline{y}$).
    If $P(\overline{y}) = N$, then $M(\overline{y}) = \emptyset$, STOP.
[4] Define $x(t)$ using (5) and $t_1, t_2, t_3$ using (6), (7), (8) (with $\overline{x}$ replaced by $\overline{y}$ everywhere).
[5] $\tau := \min(t_1, t_2, t_3)$, $\overline{y} := x(\tau)$, go to [2].
```

Remark 3.1. The proofs for steps $\boxed{2}$ (i.e. that $\overline{y}$ is the maximum element of $M(\overline{x})$) and $\boxed{3}$ (i.e. that $P(\overline{y}) = N$ implies $M(\overline{x}) = \emptyset$) will be given in the next section, and the complexity of Algorithm II will be analysed in Section 5.

4. Theoretical background of Algorithm II

Theorem 4.1. *Let $\overline{y}$ be a current upper bound in* Algorithm II, $\overline{y} \notin M(\overline{y})$, $t \in (0, \tau)$, $\tilde{x} \leqslant \overline{y}$ *and* $\tilde{x} \nleqslant x(t)$ *where τ is defined in step* $\boxed{5}$ *of Algorithm* II. *Then* $\tilde{x} \notin M(\overline{y})$.

Proof. Since $H(\overline{y}) \neq \emptyset$, it follows from the description of Algorithm II that for any $t \in (0, \tau)$ and all $i \in S$, if $F_i(\overline{y}) \subseteq P(\overline{y})$ (or, equivalently $G_i(\overline{y}) \subseteq P(\overline{y})$) then the equalities

$$H(x(t)) = H(\overline{y}), \quad F_i(\overline{y}) = F_i(x(t)), \quad G_i(\overline{y}) = G_i(x(t)) \tag{9}$$

hold. Further, we have:

$$a_i(x(t)) > b_i(x(t)), \quad \forall i \in H(\overline{y}), \tag{10}$$

$$a_i(x(t)) = b_i(x(t)), \quad \forall i \in S \backslash H(\overline{y}), \tag{11}$$

$$a_i(x(t)) = a_i(\overline{y}) \quad \text{if } F_i(\overline{y}) \nsubseteq P(\overline{y}), \tag{12}$$

$$b_i(x(t)) = b_i(\overline{y}) \quad \text{if } G_i(\overline{y}) \nsubseteq P(\overline{y}), \tag{13}$$

$$a_i(x(t)) = a_i(\overline{y}) - t \quad \text{if } F_i(\overline{y}) \subseteq P(\overline{y}), \tag{14}$$

$$b_i(x(t)) = b_i(\overline{y}) - t \quad \text{if } G_i(\overline{y}) \subseteq P(\overline{y}). \tag{15}$$

Let us assume now that $t \in (0, \tau)$ is fixed and $\tilde{x} \nleq x(t)$, so that $N_1 \equiv \{j \in N | \tilde{x}_j > x_j(t)\} \neq \emptyset$. Let $\tilde{\alpha}$ be defined as follows:

$$\tilde{\alpha} \equiv \max_{j \in N_1} (\tilde{x}_j - x_j(t)). \tag{16}$$

Let $N_2 \equiv \{j \in N_1 \,|\, \tilde{x}_j - x_j(t) = \tilde{\alpha}\}$. We shall investigate the following two cases:

Case I: $A(\overline{y}) \cap N_2 \neq \emptyset$.

Case II: $A(\overline{y}) \cap N_2 = \emptyset$.

We shall show that in both cases $\tilde{x} \notin M(\overline{y})$.

Case I. Let p be any element from $A(\overline{y}) \cap N_2$. It follows from (4) that $p \in A(\overline{y})$ implies the existence of an index $k \in H(\overline{y})$ such that $p \in F_k(\overline{y})$. Let us choose $\tilde{t} \in (0, \tau)$ such that $x_p(\tilde{t}) = \tilde{x}_p$. Clearly $\tilde{t} - \tilde{\alpha} < t$. Let us note that (5) implies $x_p(\tilde{t}) = \overline{y}_p - \tilde{t}$ and $x_p(t) = \overline{y}_p - t$, so that $\tilde{\alpha} = \tilde{x}_p - x_p(t) = \tilde{x}_p - (\overline{y}_p - t) = \tilde{x}_p - \overline{y}_p + t$ and $x_p(\tilde{t}) = \overline{y}_p - \tilde{t} = \tilde{x}_p = x_p(t) + \tilde{\alpha}$. By (5) we have:

$$x_j(\tilde{t}) = \overline{y}_j - \tilde{t}, \quad j \in P(\overline{y}),$$
$$x_j(\tilde{t}) = \overline{y}_j \quad \text{otherwise.}$$

Taking into account the definition of $\tilde{\alpha}$ (see (16)), we obtain for any $j \in N_1$:

$$x_j(\tilde{t}) = x_j(t) + \tilde{\alpha} \geqslant x_j(t) + \tilde{x}_j - x_j(t) = \tilde{x}_j.$$

Since $x_j(\tilde{t}) \geqslant x_j(t) \geqslant \tilde{x}_j$ for all $j \in N \backslash N_1$, it follows that

$$x(\tilde{t}) \geqslant \tilde{x} \tag{17}$$

and

$$a_k(\tilde{x}) \geqslant a_{kp} + \tilde{x}_p = a_{kp} + x_p(\tilde{t}) = a_k(x(\tilde{t})), \tag{18}$$

where the last equality follows from the fact that $p \in F_k(\overline{y}) = F_k(x(\tilde{t}))$. Using relations (18), (10) and (16), we obtain:

$$a_k(\tilde{x}) \geqslant a_k(x(\tilde{t})) > b_k(x(\tilde{t})) \geqslant b_k(\tilde{x}), \tag{19}$$

so that $a_k(\tilde{x}) > b_k(\tilde{x})$ and thus $\tilde{x} \notin M(\overline{y})$.

Case II. Let p be any element of $(P(\overline{y}) \backslash A(\overline{y})) \cap N_2$. Let $N_3 \subseteq N_2$ be the set of indices in N_2 included in $P(\overline{x})$ by Algorithm I first. We can assume w.l.o.g. that $p \in N_3$ (otherwise we could consider instead of p an arbitrary element of N_3). Now, since $p \in (P(\overline{y}) \backslash A(\overline{y})) \cap N_3$, it follows from the description of Algorithm I that there exists an index $r \in S \backslash H(\overline{y})$ such that $F_r(\overline{y}) \subseteq P(\overline{y})$, $G_r(\overline{y}) \subseteq P(\overline{y})$ and either $p \in F_r(\overline{y})$, $p \notin G_r(\overline{y})$ and $N_2 \cap G_r(\overline{y}) = \emptyset$ or $p \notin F_r(\overline{y})$, $p \in G_r(\overline{y})$ and $N_2 \cap F_r(\overline{y}) = \emptyset$. We can assume w.l.o.g. that the latter case occurs. Since $r \in S \backslash H(\overline{y})$, the equation $a_r(x(\tilde{t})) = b_r(x(\tilde{t}))$ holds. It follows further from (16) (note that $\tilde{\alpha} > 0$) and (17):

$$x_j(\tilde{t}) = x_j(t) + \tilde{\alpha} = x_j(t) + \tilde{x}_j - x_j(t) = \tilde{x}_j \quad \text{if } j \in N_2; \tag{20}$$

$$x_j(\tilde{t}) = x_j(t) + \tilde{\alpha} > x_j(t) + \tilde{x}_j - x_j(t) = \tilde{x}_j \quad \text{if } j \in N_1 \backslash N_2; \tag{21}$$

$$x_j(\tilde{t}) = x_j(t) + \tilde{\alpha} > x_j(t) \geqslant \tilde{x}_j \quad \text{if } j \in P(\overline{y}) \backslash N_1; \tag{22}$$

$$x_j(\tilde{t}) = x_j(t) = \overline{y}_j \geqslant \tilde{x}_j \quad \text{if } j \in N \backslash P(\overline{y}). \tag{23}$$

It follows that $\tilde{x}_j \leqslant x_j(\tilde{t})$ for all $j \in N$, which implies $b_{rj} + \tilde{x}_j \leqslant b_{rj} + x_j(\tilde{t})$ for all $j \in N$. Since $p \in G_r(\overline{y}) = G_r(x(\tilde{t}))$ and $x_p(\tilde{t}) = \tilde{x}_p$, we obtain for all $j \in N$:

$$b_{rj} + \tilde{x}_j \leqslant b_{rj} + x_j(\tilde{t}) \leqslant b_{rp} + x_p(\tilde{t}) = b_{rp} + \tilde{x}_p, \tag{24}$$

and thus

$$b_r(\tilde{x}) = \max_{j \in N} (b_{rj} + \tilde{x}_j) = b_{rp} + \tilde{x}_p = b_{rp} + x_p(\tilde{t}) = b_r(x(\tilde{t})). \tag{25}$$

Further, it follows from (20)–(23): $a_{rj} + \tilde{x}_j < a_{rj} + x_j(\tilde{t})$ for all $j \in P(\overline{y})\backslash N_2$ and $a_{rj} + \tilde{x}_j = a_{rj} + x_j(\tilde{t})$ for all $j \in N_2$. Since $N_2 \cap F_r(x(\tilde{t})) = \emptyset$, we have $j \in N_2 \Longrightarrow j \notin F_r(x(\tilde{t}))$. Since $F_r(x(\tilde{t})) = F_r(\overline{y}) \subseteq P(\overline{y})$, it holds that $F_r(x(\tilde{t})) \cap (N\backslash P(\overline{y})) = \emptyset$. Therefore, we obtain:

$$a_{rj} + \tilde{x}_j = a_{rj} + x_j(\tilde{t}) < a_r(x(\tilde{t})), \quad j \in N_2 \cup (N\backslash P(\overline{y})) \tag{26}$$

and according to (21), (22)

$$a_{rj} + \tilde{x}_j < a_{rj} + x_j(\tilde{t}) \leqslant a_r(x(\tilde{t})) \tag{27}$$

otherwise.

Finally, it follows from (25)–(27):

$$a_r(\tilde{x}) = \max_{j\in N}(a_{rj} + \tilde{x}_j) < a_r(x(\tilde{t})) = b_r(x(\tilde{t})) = b_r(\tilde{x}),$$

so that $a_r(\tilde{x}) < b_r(\tilde{x})$ and thus $\tilde{x} \notin M(\overline{y})$. This completes the proof. □

For $z = (z_1, \ldots, z_n) \in \mathbb{Q}^n$, $\alpha \in \mathbb{Q}$, we denote $z[\alpha] = (z_1 + \alpha, \ldots, z_n + \alpha)$.

Theorem 4.2. *If on some current iteration of* Algorithm II $\overline{y} \notin M(\overline{y})$ *and* $P(\overline{y}) = N$, *then* $M(\overline{x}) = \emptyset$.

Proof. Assume that $P(\overline{y}) = N$, so that $\tau = \infty$. Let z be any element of $M(\overline{x})$. Then there exists an $\overline{\alpha}$ such that $z[\overline{\alpha}] \leqslant \overline{y}$ and $z[\overline{\alpha}] \neq \overline{y}$. Since $a_i(x)$, $b_i(x)$ are +-homogeneous and $a_i(z) = b_i(z)$ for all $i \in S$, it is also $a_i(z[\alpha]) = b_i(z[\alpha])$ for all $i \in S$ and for any $\alpha \in \mathbb{Q}$. Therefore, also $a_i(z[\overline{\alpha}]) = b_i(z[\overline{\alpha}])$ for all $i \in S$, so that $z[\overline{\alpha}] \in M(\overline{y})$. Since $z[\overline{\alpha}] \neq \overline{y}, \tau = \infty$, there exists $t \in (0, \infty)$ such that $z[\overline{\alpha}] \nleqslant x(t)$, where $x(t)$ is defined as in Algorithm II (since $P(\overline{y}) = N$, it means in this case that $x_j(t) = \overline{y}_j - t$ for all $j \in N$). Then $z[\overline{\alpha}] \notin M(\overline{y})$ by Theorem 4.1. This contradiction completes the proof of Theorem 4.2. □

Theorem 4.3. *Let* $\overline{y}$ *be a current upper bound in* Algorithm II, $\overline{y} \notin M(\overline{y})$. *Let* τ *be defined as in Algorithm* II *and* $P(\overline{y}) \neq N$. *If* $x(\tau) \in M(\overline{y})$ *then* $x(\tau)$ *is the maximum element of* $M(\overline{x})$.

Proof. Let z be any element of $M(\overline{x})$ satisfying $z \nleqslant x(\tau)$. Since $a_i(x)$, $b_i(x)$ are +-homogeneous for all $i \in S$, $z[\alpha] \in M(\overline{x})$ for all $\alpha \leqslant 0$. Therefore, we can find $\tilde{\alpha} \leqslant 0$ such that $z[\tilde{\alpha}] \leqslant \overline{y}$, $z(\tilde{\alpha}) \neq \overline{y}$ and $z(\tilde{\alpha}) \nleqslant x(\tau)$. Let $x(\tau)$ be defined as in Algorithm II. Suppose that p is an index from N such that $z_p[\tilde{\alpha}] > x_p(\tau)$. Let us choose $\tilde{t} \in (0, \tau)$ satisfying $z_p[\tilde{\alpha}] > x_p(\tilde{t}) = \overline{y}_p - \tilde{t} > x_p(\tau)$.Then $z[\tilde{\alpha}] \nleqslant x(\tilde{t})$ and therefore due to Theorem 4.1 $z[\tilde{\alpha}] \notin M(\overline{y})$. On the other hand, since $z \in M(\overline{x})$ and $a_i(x)$, $b_i(x)$ are +-homogeneous, we have for any $i \in S$:

$$a_i(z[\tilde{\alpha}]) = a_i(z) + \tilde{\alpha} = b_i(z) + \tilde{\alpha} = b_i(z[\tilde{\alpha}]).$$

Since at the same time $z[\tilde{\alpha}] \leqslant \overline{y}$, we have $z[\tilde{\alpha}] \in M(\overline{y})$. This contradiction shows that $z \leqslant x(\tau)$ for any $z \in M(\overline{x})$. □

Remark 4.1. As another consequence of Theorem 4.1 at least one of the variables x_j, $j \in N$ has never been decreased if $M(\overline{x}) \neq \emptyset$ and $x(\tau)$ is the solution obtained in the last iteration of Algorithm II. Indeed, if $x_j(\tau) < \overline{x}_j$ for all $j \in N$, then there exists $\varepsilon > 0$ such that $x_j(\tau) + \varepsilon < \overline{x}_j$ for all $j \in N$ and $(x_1(\tau) + \varepsilon, \ldots, x_n(\tau) + \varepsilon) \in M(\overline{x})$, which would lead to a contradiction similarly as in Theorem 4.3.

Remark 4.2. As we have already mentioned, if we use the $(\oplus, \otimes)$-notation as e.g. in [4], the algorithm suggested here solves systems of m $(\oplus, \otimes)$-linear equations with n variables of the form $A \otimes x = B \otimes x$, where A, B are given (m, n)-matrices with finite entries over the $(\mathbb{Q} \cup \{-\infty\}, \oplus, \otimes)$-algebra with $(\oplus, \otimes) = (\max, +)$. The algorithm can be straightforwardly applied in any other $(E, \oplus, \otimes)$-algebra, where $E \subseteq (\mathbb{R} \cup \{\infty\} \cup \{-\infty\})$, E is a semigroup with respect to $\oplus$ and group with respect to $\otimes$, and the order relation $\leqslant$ is defined by $\alpha \leqslant \beta \Longleftrightarrow \alpha \oplus \beta = \beta$ for any $\alpha, \beta \in E$. Such cases are e.g. $(\mathbb{Z} \cup \{-\infty\}, \max, +)$-algebra, where $\mathbb{Z}$ is the set of integers, or $(\mathbb{R}_+ \cup \{\infty\}, \min, \cdot)$-algebra, where $\mathbb{R}_+$ is the set of positive reals [6,9].

5. Computational complexity of the algorithm

Now we analyze the computational complexity C of Algorithm II. This algorithm runs in loops between steps 2 and 5 and will terminate at either step 2 or 3. It calls Algorithm I as a subroutine in step 3. It is easily seen that Algorithm I and all individual steps of Algorithm II are $O(mn)$ and thus $C = O(mnq)$ where q is the number of iterations of Algorithm II, that is the number of repetitions of the loop 2–5. We now derive an upper bound for q in terms of m and n.

The iterations between steps 2 and 5 run at three different levels, which we refer to as macroiterations, miniiterations and microiterations depending on how the value of τ in step 4 is attained. We will introduce this terminology precisely below. A vector found by Algorithm II at the end of any iteration will be called a *current solution* (during the next iteration). Let $\bar{x} \in \mathbb{Q}^n$ be a current solution. A variable x_j will be called *primary* (in that iteration) if $j \in A(\bar{x})$. The expressions $a_{ij} + x_j$, $b_{ij} + x_j$ will be called *terms*. If x_j is primary, then we call the corresponding term primary.

The set of all iterations between two attainments of τ by t_3, that is between two successive changes of the set $H(\cdot)$ will be called a *macroiteration*. When $H(\cdot)$ is changing at least one of the inequalities becomes an equality. Since the algorithm maintains existing equalities, once a row index moves from $H(x)$ to $S - H(x)$ for some current solution x it will not appear in $H(y)$ for any future current solution y and hence the number of macroiterations is at most m. In what follows we consider the algorithm performance restricted to one macroiteration, thus the set $H(x)$ will be the same for all current solutions x during this macroiteration. We may therefore denote it by H.

If the value of τ is attained by t_1 for some $i \in H$, then at least one new variable becomes primary. If a variable becomes primary, it will remain primary at least until the end of the current macroiteration. The set of all iterations between two successive attainments of τ by t_1 for some $i \in H$ will be called a *miniiteration*. Obviously, there can be at most $n - 1$ miniiterations in one macroiteration (if all variables become primary for some current solution x then $P(x) = N$ and thus $M(x) = \emptyset$) and the total number of all miniiterations is at most $\ell = m(n - 1)$.

We will now estimate the number of iterations in one miniiteration.

Consider a miniiteration, that is assume that the set of primary variables is fixed. The symbol $\bar{x}$ will stand for the current solution at the beginning of the miniiteration, while x will denote the (changing) current solution during the miniiteration. We will refer to the individual equations by their row numbers. Since $A(x) \subseteq P(x)$ for every x it is clear that all primary variables will change by the same value in each iteration within this miniiteration. The following cases at the beginning of the miniiteration are considered.

(I) If there is no side of any equation $i \in S - H$ in which all active variables are primary then none of these equations is restrictive in the choice of τ (that is τ will not be attained by either t_1 or t_2 for any $i \in S - H$) and therefore all primary variables will decrease in one iteration

(a) either by τ attained by t_1 for some $i \in H$, in which case the current miniiteration ends,
(b) or by τ attained by t_3 for some $i \in H$, in which case the current macroiteration ends,
(c) or by τ attained by t_2 for some $i \in H$. This last subcase can only happen at most once for each inequality within one miniiteration, thus at most $k \leqslant m$ times in any miniiteration.

So the total number of iterations in this case is $C_1 \leqslant k$.

(II) There are equations $i \in S - H$ in which on exactly one side all active variables are primary, but there is no equation in which all active variables on both sides are primary. Let us assume without loss of generality that in all these equations all active variables on the left are primary (but on the right some active variables are non-primary). Let $R(x) \subseteq S - H$ be the set of all such equations for a current solution x. We will call an equation *i regular* if $i \in R(x)$. If $i \in R(\bar{x})$ then in every iteration $G_i(x)$ either grows or remains unchanged, while $i \in R(x)$. During a miniiteration, some of the regular equations may stop being regular, when on the left-hand side a non-primary variable becomes active. No new equations become regular during a miniiteration.

All non-primary variables active on the right-hand side of at least one regular equation will be called *secondary*, and the set of indices of all secondary variables will be denoted by $Q(x)$, that is

$$Q(x) = \bigcup_{i \in R(x)} G_i(x) - A(x).$$

There are two ways a miniiteration can finish: Either all regular equations stop being regular — then we have the situation described in (I) and the miniiteration therefore ends after additional at most k iterations (see the case I(c)). Or all non-primary variables become secondary in which case $P(x)=N$ and thus $M(x)=\emptyset$.

We now give (a very rough) upper bound on the number of iterations in one miniiteration. Variables x_j, $j\in P(x)$ that are neither primary nor secondary are called *tertiary*. Variables x_j, $j\in N-P(x)$ are called *ordinary*. We will use the same adjective for the terms $a_{ij}+x_j$, $b_{ij}+x_j$ as for the corresponding variable x_j.

Lemma 5.1. *At every iteration there is a pair of terms, one of which is secondary* (v_{s}) *and the other one ordinary* (v_{o}) *such that*

(a) $v_{\mathrm{s}}>v_{\mathrm{o}}$ *at the beginning of this iteration*;
(b) $v_{\mathrm{s}}=v_{\mathrm{o}}$ *at the end of this iteration*;
(c) $v_{\mathrm{s}}\leqslant v_{\mathrm{o}}$ *at all iterations following this iteration while* v_{s} *stays secondary*.

Proof. (a) and (b) follow immediately from the way $P(x)$ and τ are found by Algorithms I and II. Part (c) is due to the fact that all x_j, $j\in P(x)$ change by the same value and that indices of secondary variables are in $P(x)$ in every iteration of a miniiteration, whereas indices of variables that are neither primary nor secondary may be in $P(x)$ in some but not necessarily all iterations. $\square$

It follows from this Lemma that at every iteration a new secondary term decreases to the level of an ordinary term and will never exceed it while the corresponding variable remains secondary. The number of secondary and ordinary terms may change from iteration to iteration, but neither of them exceeds

$$u=2(n-1)(m-k-1)+(n-1)k$$

(2 sides, at most $n-1$ non-primary variables, at most $m-k-1$ non-regular equations $i\in S-H$ and k right-hand sides for $i\in H$). Thus, after at most $u^2=\mathrm{O}(n^2m^2)$ iterations the remaining secondary variables can decrease without being restricted by other than regular equations, thus in the next iteration $\tau=t_2$ for some i regular, yielding at least one new secondary variable in at least one regular equation. The set of iterations between two consecutive such attainments of τ will be called a *microiteration*. Since the number of secondary variables is at most $n-1$, there can be at most $(n-1)$ microiterations in one miniiteration.

Hence the number of iterations in one miniiteration in this case is

$$C_2\leqslant(n-1)\mathrm{O}(m^2n^2)=\mathrm{O}(m^2n^3).$$

(III) There are equations in which all active variables on both sides are primary ("primary equations"). We will assume without loss of generality that there is an equation in which all active variables on the left-hand side (but not on the right) are primary, as this would be achieved in at least one equation after at most k iterations (each being determined by one of the k right-hand side "stoppages" in inequalities $i\in H$). We now count the number of iterations similarly as in (II), while all primary equations remain primary (in that period they have no effect on the number of iterations). If a primary equation stops being primary but all active variables on one side are still primary then we re-start counting the number of iterations as in (II). This re-starting can take place at most $m-k$ times. If a primary equation stops being primary and neither side has all active variables primary then no re-starting is necessary.

So the total number of operations in a miniiteration in this case is

$$C_3\leqslant(m-k)\mathrm{O}(m^2n^3)=\mathrm{O}(m^3n^3).$$

Hence $q=\ell\mathrm{O}(m^3n^3)=\mathrm{O}(m^4n^4)$ and $C=\mathrm{O}(m^5n^5)$.

6. Conclusions

In this paper a strongly polynomial algorithm for solving systems of two-sided (max, +) linear equations is presented. It gives an answer to the question of polynomial solvability of such systems which was open for more than 30 years. Further research may concentrate on the reduction of the number of iterations, thus reducing the overall complexity of the method. Polynomial solvability of a range of related problems in max-algebra and its applications in machine-scheduling is likely to be deduced.

References

[1] F.L. Baccelli, G. Cohen, G.J. Olsder, J.P. Quadrat, Synchronization and Linearity, An Algebra for Discrete Event Systems, Wiley, Chichester, 1992.

[2] P. Butkovič, G. Hegedüs, An elimination method for finding all solutions of the system of linear equations over an extremal algebra, Ekonom. mat. Obzor 20 (1984) 203–215.

[3] K. Cechlárová, Efficient computation of the greatest eigenvector in fuzzy algebra, Tatra Mt. Math. Publications (1997) 73–79.

[4] R.A. Cuninghame-Green, Minimax Algebra, Lecture Notes in Economics and Mathematical Systems, vol.166, Springer, Berlin, 1985.

[5] R.A. Cuninghame-Green, P. Butkovič, The equation $Ax = By$ over (max, $+$), Theoret. Comput. Sci. 293 (1991) 3–12.

[6] R.A. Cuninghame-Green, K. Zimmermann, Equation with residual functions, Comment. Math. Univ. Carolin. 42 (2001) 729–740.

[7] N.N. Vorobjov, Extremal algebra of positive matrices, Elektronische Datenverarbeitung und Kybernetik 3 (1967) 39–71 (in Russian).

[8] E.A. Walkup, G. Boriello, A general linear max-plus solution technique, in: J. Gunawardena (Ed.), Idempotency, Cambridge, 1988, pp. 406–415.

[9] U. Zimmermann, Linear and combinatorial optimization in ordered algebraic structures, Ann. Discrete Math. 10 (1981).