



Determining Logistics Route Capacity Using Maximal Solutions of Two-Sided Max-Min Linear System with Python Implementation

Zakia Nur Ramadhani Putri*, Ari Suparwanto, and Sutopo

Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, Indonesia

Abstract

Max-min algebra, equipped with maximum and minimum operations, provides a natural framework for modeling systems with bottleneck constraints, such as logistics and transportation networks. Although an algorithm for solving two-sided linear systems of the form $A \otimes x = B \otimes x$ in max-min algebra has been proposed, no implementation has been made available, limiting its practical use. This paper provides a Python implementation of the algorithm, along with a numerical application to a logistics distribution network. The theoretical results show that the maximal solution can be constructed explicitly for the $1 \times n$ case under three conditions—unconstrained, with a given upper bound, and with a given lower bound—and then generalized to the $m \times n$ case. The algorithm terminates after at most m iterations and produces a unique maximal solution with complexity $O(m^2n)$. The Python implementation makes the algorithm accessible to practitioners, and the logistics example demonstrates how the model can be used to determine optimal road capacities. This work provides a practical computational tool for solving capacity planning problems in logistics and transportation.

Keywords: max-min algebra; two-sided linear system; maximal solution; algorithm; logistics network.

Copyright © 2026 by Authors, Published by CAUCHY Group. This is an open access article under the CC BY-SA License (<https://creativecommons.org/licenses/by-sa/4.0>)

1. Introduction

Max-min algebra is set $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ equipped with two binary operations maximum ($\oplus$) and minimum ($\otimes$). This structure forms a commutative idempotent semiring and provides a natural mathematical framework for modeling systems that involve bottleneck constraints, where the overall capacity of a system is determined by its smallest component. Such problems arise in various fields, including logistics and transportation networks. In these contexts, the minimum operation captures the bottleneck principle, the capacity of a route consisting of multiple segments is limited by the segment with the smallest capacity, while the maximum operation represents the optimal choice among available alternatives. The theoretical foundation of max-min algebra and its applications have been extensively studied since the pioneering work of Cunningham-Green and Raymond on minimax algebra [1], and later developed by Gondran and Minoux in the context of dioids and semirings [2].

The study of linear systems in max-min algebra has received much attention in recent decades. Sanchez [3] pioneered the resolution of composite fuzzy relation equations, which are closely

*Corresponding author. E-mail: zakiaputri119101@gmail.com

related to max-min systems, and later applied these results to medical diagnosis [4]. Santos [5] investigated maximin automata, which indirectly use maximum and minimum as composition operations. Butkovic et al. [6] studied strong linear independence and regularity of matrices in bottleneck algebra, providing foundational results for matrix theory in this structure. Fiedler et al. [7] provided a comprehensive treatment of one-sided systems of the form $A \otimes x = b$, including the discussion of the greatest subsolution and various applications in production scheduling and transportation networks. For two-sided systems of the form $A \otimes x = B \otimes x$, Gavalec and Zimmermann [8] proposed a polynomial algorithm for solving such systems, establishing the theoretical feasibility of obtaining solutions in polynomial time. Subsequently, Krbálek and Pozdílková [9] presented a simpler and more efficient iterative method, which forms the basis of the algorithm examined in this paper.

More recently, researchers have explored several extensions of max-min linear systems, such as optimization with two-sided constraints [10, 11], robustness of interval matrices [12, 13], and methods for max-plus and tropical algebras [14]. Mufid et al. [15] developed a method for max-min ω systems, which generalizes both max-plus and min-plus systems. While these works focus on specific extensions, the basic algorithm for finding maximal solutions of two-sided max-min systems is still an important building block for applications.

The algorithm proposed by Krbálek and Pozdílková [9] is an iterative method for determining the maximal solution of two-sided max-min linear systems. Its theoretical properties have been established, but no implementation has been made available, which makes it hard to use in practice. This paper fills that gap by providing a Python implementation of the algorithm, along with a numerical example from a logistics distribution network to show how its practical application. The mathematical background is included to support the implementation, but the main focus is on the computational tool and its practical use. The rest of this paper is organized as follows. Section 2 presents the mathematical setting and methodology. Section 3 contains the main results, including the derivation of the maximal solution for the $1 \times n$ and $m \times n$ cases, Algorithm 1 with its proof of correctness, and a numerical example. Section 4 concludes the paper. The Python implementation of Algorithm 1 is given in Appendix A.

2. Methods

This study adopts a theoretical–computational framework. The theoretical part establishes the mathematical background and the correctness of the algorithm, while the computational part provides a Python implementation and a numerical example to demonstrate the practical applicability of the model. The study is conducted within the framework of max-min algebra, defined as the structure $(\overline{\mathbb{R}}, \oplus, \otimes)$, where $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, equipped with two binary operations $a \oplus b = \max(a, b)$ and $a \otimes b = \min(a, b)$, for any $a, b \in \overline{\mathbb{R}}$. The operation $\oplus$ is called maximum, and $\otimes$ is called minimum. The structure $(\overline{\mathbb{R}}, \oplus, \otimes)$ forms a commutative idempotent semiring, with $\varepsilon = -\infty$ as the neutral element for $\oplus$ and $\varepsilon' = +\infty$ as the identity element for $\otimes$. The concept of semiring follows from the works of Baccelli et al. [16], Gondran and Minoux [2], and Subiono [17]. We consider matrices $A, B \in \overline{\mathbb{R}}^{m \times n}$ and vectors $x \in \overline{\mathbb{R}}^n$. The matrix-vector product $A \otimes x$ is defined componentwise as [18]:

$$(A \otimes x)_i = \bigoplus_{j=1}^n (a_{ij} \otimes x_j) = \max_{j=1, \dots, n} (\min(a_{ij}, x_j)), \quad i = 1, \dots, m.$$

The relation " $\leq$ " defined on $\overline{\mathbb{R}}$ by $a \leq b \iff a \oplus b = b$ is a partial order on $\overline{\mathbb{R}}$. Moreover, " $\leq$ " is a total order on $\overline{\mathbb{R}}$ [18]. The relation " $\leq$ " is then extended componentwise to vectors and matrices. The problem is the two-sided system $A \otimes x = B \otimes x$, and the goal is to determine its maximal solution.

The algorithm used in this paper was proposed by Krbálek and Pozdílková [9]. It is an iterative method that finds the maximal solution of $A \otimes x = B \otimes x$. Starting from an upper

bound, the algorithm repeatedly reduces the solution vector until a solution is found. The algorithm terminates after at most m iterations and produces a unique maximal solution. The full algorithm, along with its proof of correctness, is presented in Section 3.

To make the algorithm usable in practice, we implement the algorithm in Python 3.9. The implementation uses only the standard library; no external packages are required. The code reads matrices A and B as input and iteratively computes the maximal solution following the algorithm described in Section 3. The full implementation is provided in Appendix A.

To show how the model can be applied, we construct a numerical example from a logistics distribution network. It involves three industrial areas and three transit points, where the system $A \otimes x = B \otimes x$ is used to determine the optimal capacities of access roads from the transit points to the main distribution center. The data used in this example are simulated and serve only for illustrative purposes; they are not based on actual field measurements. The aim of this example is to show the applicability of the model, rather than to serve as a real-world case study.

3. Results and Discussion

This section presents the theoretical foundations needed for the implementation, followed by a numerical application. We begin with the $1 \times n$ case, then generalize to $m \times n$ matrices. The theoretical results are then used to develop the Python implementation and the logistics application. For the $1 \times n$ case, we consider three scenarios: without constraints, with a lower bound, and with an upper bound.

3.1. Maximal Solutions of $A \otimes x = B \otimes x$ for $1 \times n$ Matrices A, B

This subsection discusses the maximal solution of the system

$$A \otimes x = B \otimes x \quad (1)$$

where A, B are of size $1 \times n$, and $x \in \overline{\mathbb{R}}^n$ without constraints, with an upper bound, and with a lower bound. System 1 can be written as

$$(a_{11} \quad a_{12} \quad \cdots \quad a_{1n}) \otimes \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = (b_{11} \quad b_{12} \quad \cdots \quad b_{1n}) \otimes \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

or equivalently,

$$(a_{11} \otimes x_1) \oplus (a_{12} \otimes x_2) \oplus \cdots \oplus (a_{1n} \otimes x_n) = (b_{11} \otimes x_1) \oplus (b_{12} \otimes x_2) \oplus \cdots \oplus (b_{1n} \otimes x_n).$$

In other words, we seek the maximal vector x satisfying

$$\bigoplus_{j=1}^n (a_{1j} \otimes x_j) = \bigoplus_{j=1}^n (b_{1j} \otimes x_j).$$

3.1.1. Maximal Solution of $A \otimes x = B \otimes x$ for $1 \times n$ Matrices A, B with $x \in \overline{\mathbb{R}}^n$ Without Constraints

This subsection discusses the maximal solution $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ of system 1 where A, B are of size $1 \times n$ and $\bar{x}$ is given without any constraints. Let $N = \{1, 2, \dots, n\}$, and define

$$a = \max_{j=1, \dots, n} (a_{11}, \dots, a_{1n}) = \bigoplus_{j=1}^n a_{1j}, \quad b = \max_{j=1, \dots, n} (b_{11}, \dots, b_{1n}) = \bigoplus_{j=1}^n b_{1j}.$$

There are three possibilities for the values of a and b : $a = b$, $a < b$, or $a > b$.

Lemma 1 and Lemma 2 discuss the maximal solution for the cases $a = b$ and $a > b$, respectively. The case $a < b$ is analogous to the case $a > b$.

Lemma 1. If $a = b$, then $\bar{x} = (\infty, \infty, \dots, \infty)^T$ is the greatest solution of the linear system $A \otimes x = B \otimes x$.

Proof. Substituting $\bar{x}$ into $A \otimes x = B \otimes x$, we obtain

$$A \otimes \bar{x} = \bigoplus_{j=1}^n (a_{1j} \otimes \infty) = \bigoplus_{j=1}^n a_{1j} = a \quad \text{and} \quad B \otimes \bar{x} = \bigoplus_{j=1}^n (b_{1j} \otimes \infty) = \bigoplus_{j=1}^n b_{1j} = b.$$

Since $a = b$, we have $A \otimes \bar{x} = B \otimes \bar{x}$, which means $\bar{x}$ is a solution of $A \otimes x = B \otimes x$. Clearly, $\bar{x}$ is the greatest solution. $\square$

Lemma 2. Let $N_1 = \{j \in N \mid a_{1j} > b\}$. If $a > b$, then $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^T$ is the greatest solution of $A \otimes x = B \otimes x$, where

$$\bar{x}_j = \begin{cases} b, & j \in N_1, \\ \infty, & \text{otherwise,} \end{cases}$$

for $j = 1, 2, \dots, n$.

Proof. Given $N_1 = \{j \in N \mid a_{1j} > b\}$. Since $a > b$, we have $\bigoplus_{j=1}^n a_{1j} > b$. Hence there exists $j \in N$ such that $a_{1j} > b$, so $N_1 \neq \emptyset$. Substituting $\bar{x}$ into $A \otimes x = B \otimes x$, we obtain

$$B \otimes \bar{x} = \bigoplus_{j=1}^n (b_{1j} \otimes \bar{x}_j) = \left(\bigoplus_{j \in N_1} (b_{1j} \otimes b) \right) \oplus \left(\bigoplus_{j \in N \setminus N_1} (b_{1j} \otimes \infty) \right) \leq b.$$

On the other hand, there exists $k \in N$ such that $b_{1k} = b$, and

$$B \otimes \bar{x} = \bigoplus_{j=1}^n (b_{1j} \otimes \bar{x}_j) \geq b_{1k} \otimes \bar{x}_k = b \otimes \bar{x}_k = b$$

so, $B \otimes \bar{x} = b$. Next,

$$A \otimes \bar{x} = \bigoplus_{j=1}^n (a_{1j} \otimes \bar{x}_j) = \left(\bigoplus_{j \in N_1} (a_{1j} \otimes b) \right) \oplus \left(\bigoplus_{j \in N \setminus N_1} (a_{1j} \otimes \infty) \right) = b \oplus \left(\bigoplus_{j \in N \setminus N_1} a_{1j} \right) = b.$$

Thus $A \otimes \bar{x} = B \otimes \bar{x}$, so $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^T$ is a solution of $A \otimes x = B \otimes x$. We now show that $\bar{x}$ is the greatest solution by contradiction. Let x' be a solution. Suppose $\bar{x}$ is not the greatest solution. Then there exists $k \in N$ such that $\bar{x}_k < x'_k$. Clearly $\bar{x}_k \neq \infty$, which means $k \in N_1$ and $a_{1k} > b = \bar{x}_k$. Since $b = \bar{x}_k < x'_k$, we have

$$A \otimes x' = \bigoplus_{j=1}^n (a_{1j} \otimes x'_j) \geq a_{1k} \otimes x'_k > b.$$

On the other hand,

$$B \otimes x' = \bigoplus_{j=1}^n (b_{1j} \otimes x'_j) = \left(\bigoplus_{j \in N_1} (b_{1j} \otimes b) \right) \oplus \left(\bigoplus_{j \in N \setminus N_1} (b_{1j} \otimes \infty) \right) = \bigoplus_{j=1}^n b_{1j} \leq b.$$

Hence $A \otimes x' \neq B \otimes x'$, so x' is not a solution. This contradiction shows that $\bar{x}$ must be the greatest solution. $\square$

3.1.2. Maximal Solution of $A \otimes x = B \otimes x$ for $1 \times n$ Matrices A, B with $x \in \overline{\mathbb{R}}^n$ Given a Lower Bound

This subsection discusses the maximal solution $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ of system 1 where A, B are of size $1 \times n$ with the constraint

$$l = (l_1, l_2, \dots, l_n)^T \leq (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^T = \bar{x}.$$

Lemma 3. Let $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^T$ be the maximal solution of $A \otimes x = B \otimes x$ without constraints.

1. The maximal solution of $A \otimes x = B \otimes x$ with lower bound $l = (l_1, l_2, \dots, l_n)^T$ is $\bar{x}$ if $l \leq \bar{x}$.
2. The system $A \otimes x = B \otimes x$ with lower bound $l = (l_1, l_2, \dots, l_n)^T$ has no solution if $l \not\leq \bar{x}$.

Proof. Let $\bar{x}$ be the maximal solution of $A \otimes x = B \otimes x$ without constraints.

1. If $l \leq \bar{x}$, then $\bar{x}$ is a solution of $A \otimes x = B \otimes x$ with lower bound l . It remains to show that $\bar{x}$ is the maximal solution. Let x' be any solution satisfying $l \leq x'$. Suppose $\bar{x} \leq x'$, since x' is a solution without constraints and $\bar{x}$ is the maximal solution without constraints, we must have $x' \leq \bar{x}$. Thus $x' = \bar{x}$, which means $\bar{x}$ is the greatest solution of $A \otimes x = B \otimes x$ with lower bound l .
2. For $l \not\leq \bar{x}$, we prove by contradiction. Since $l \not\leq \bar{x}$, there exists $k \in N$ such that $\bar{x}_k \leq l_k$. Suppose there exists a solution x' satisfying $l \leq x'$. Then x' is also a solution of $A \otimes x = B \otimes x$ without constraints. Since $\bar{x}$ is the greatest solution without constraints, we must have $x' \leq \bar{x}$, so $x'_k \leq \bar{x}_k$ for each $k \in N$. Consequently, $l \leq x' \leq \bar{x}$, which implies $l \leq \bar{x}$. This contradicts $l \not\leq \bar{x}$. Therefore, $A \otimes x = B \otimes x$ with lower bound l has no solution.

□

3.1.3. Maximal Solution of $A \otimes x = B \otimes x$ for $1 \times n$ Matrices A, B with $x \in \overline{\mathbb{R}}^n$ Given an Upper Bound

This subsection discusses the maximal solution $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$ of system 1 where $A = (a_{1j}), B = (b_{1j})$ are of size $1 \times n$ with the constraint

$$\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^T \leq u = (u_1, u_2, \dots, u_n)^T.$$

Let $a = A \otimes u$ and $b = B \otimes u$. There are three possibilities for a and b : $a = b$, $a < b$, or $a > b$. Lemma 4 and Lemma 5 discuss the maximal solution for the cases $a = b$ and $a > b$, respectively. The case $a < b$ is analogous to the case $a > b$.

Lemma 4. If $a = b$, then $u = (u_1, u_2, \dots, u_n)^T$ is the greatest solution of $A \otimes x = B \otimes x$ with upper bound u .

Proof. Substituting u into $A \otimes x = B \otimes x$, we obtain

$$A \otimes u = \bigoplus_{j=1}^n (a_{1j} \otimes u_j) = a \quad \text{and} \quad B \otimes u = \bigoplus_{j=1}^n (b_{1j} \otimes u_j) = b.$$

Since $a = b$, u is a solution of $A \otimes x = B \otimes x$, and because u is the upper bound, it is the greatest solution. □

Lemma 5. Let $N_2 = \{j \in N \mid a_{1j} \otimes u_j > b\}$. If $a > b$, then the greatest solution of $A \otimes x = B \otimes x$ with upper bound u is $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^T$, where

$$\bar{x}_j := \begin{cases} b & j \in N_2, \\ u_j & \text{otherwise,} \end{cases}$$

for $j = 1, \dots, n$.

Proof. Since $a > b$, we have $\bigoplus_{j=1}^n a_{1j} \otimes u_j > b$, which means there exists $j \in N$ such that $a_{1j} \otimes u_j > b$. Hence $N_2 \neq \emptyset$. We consider $\bar{x}_j$ in two cases, for $j \in N_2$, then $\bar{x}_j = b < a_{1j} \otimes u_j \leq u_j$, and for $j \in N \setminus N_2$, then $\bar{x}_j = u_j \leq u_j$. Thus, $\bar{x}_j \leq u_j$ for all $j \in N$. Substituting $\bar{x}$ into $A \otimes x = B \otimes x$, we obtain

$$B \otimes \bar{x} = \bigoplus_{j=1}^n (b_{1j} \otimes \bar{x}_j) \leq \bigoplus_{j=1}^n (b_{1j} \otimes u_j) = b. \quad (2)$$

On the other hand, since $b = \bigoplus_{j \in N} (b_{1j} \otimes u_j)$, there exists $k \in N$ such that $b_{1k} \otimes u_k = b$. We obtain

$$B \otimes \bar{x} = \bigoplus_{j=1}^n (b_{1j} \otimes \bar{x}_j) \geq b_{1k} \otimes \bar{x}_k = b. \quad (3)$$

From (2) and (3), we get $B \otimes \bar{x} = b$. Next, since $\bigoplus_{j \in N_2} (a_{1j} \otimes b) = b$ and $\bigoplus_{j \in N \setminus N_2} (a_{1j} \otimes u_j) \leq b$, then we obtain

$$A \otimes \bar{x} = \bigoplus_{j=1}^n (a_{1j} \otimes \bar{x}_j) = \left(\bigoplus_{j \in N_2} (a_{1j} \otimes b) \right) \oplus \left(\bigoplus_{j \in N \setminus N_2} (a_{1j} \otimes u_j) \right) = b.$$

We have $A \otimes \bar{x} = B \otimes \bar{x}$, so $\bar{x}$ is a solution. It remains to show that $\bar{x}$ is the greatest solution. We prove by contradiction. Let x' be a solution. Suppose $\bar{x}$ is not the greatest solution. Then there exists $k \in N$ such that $\bar{x}_k < x'_k$. Since u is the upper bound, we must have $x'_k \leq u_k$, so $\bar{x}_k \neq u_k$. Thus $k \in N_2$, which implies $a_{1k} \otimes u_k > b = \bar{x}_k$, so $a_{1k} > b$ and $b = \bar{x}_k < x'_k$. Substituting x' into the system, we obtain

$$A \otimes x' = \bigoplus_{j=1}^n (a_{1j} \otimes x'_j) \geq a_{1k} \otimes x'_k > b.$$

On the other hand,

$$B \otimes x' = \bigoplus_{j=1}^n (b_{1j} \otimes x'_j) \leq \bigoplus_{j=1}^n (b_{1j} \otimes u_j) = b.$$

Thus $A \otimes x' \neq B \otimes x'$. Hence x' is not a solution. Therefore, $\bar{x}$ must be the greatest solution. $\square$

The following example illustrates the solution of $A \otimes x = B \otimes x$ for $A, B \in \mathbb{R}^{1 \times n}$ with the constraint $x \leq u$.

Example 1. Consider the system $A \otimes x = B \otimes x$ with $x \leq u$, where $A = \begin{pmatrix} 10 & 8 & 10 \end{pmatrix}$, $B = \begin{pmatrix} 9 & 9 & 9 \end{pmatrix}$, and $u = \begin{pmatrix} 11 & 11 & 11 \end{pmatrix}^T$. We compute

$$a = A \otimes u = \begin{pmatrix} 10 & 8 & 10 \end{pmatrix} \otimes \begin{pmatrix} 11 \\ 11 \\ 11 \end{pmatrix} = 10, \quad \text{and} \quad b = B \otimes u = \begin{pmatrix} 9 & 9 & 9 \end{pmatrix} \otimes \begin{pmatrix} 11 \\ 11 \\ 11 \end{pmatrix} = 9.$$

Since $a > b$, by Lemma 5 we obtain $N_2 = \{j \in N \mid a_{1j} \otimes u_j > 9\} = \{1, 3\}$. Thus,

$$\bar{x}_j = \begin{cases} 9, & j \in \{1, 3\}, \\ u_j, & j = 2, \end{cases}$$

so $\bar{x} = (9, 11, 9)^T$ is a solution of $A \otimes x = B \otimes x$.

3.1.4. Interval Solution of $A \otimes x = B \otimes x$ for $1 \times n$ Matrices A, B with $x \in \overline{\mathbb{R}}^n$ Given an Upper Bound

Lemma 6 discusses the interval solution of $A \otimes x = B \otimes x$ with the constraint $\bar{x} \leq u$.

Definition 1. An interval vector $\hat{x}$ is called an interval solution of $A \otimes x = B \otimes x$ with upper bound u if for every $\bar{x} \in \hat{x}$, we have $A \otimes \bar{x} = B \otimes \bar{x}$.

Lemma 6. Let $a > b$ and $N_2 = \{j \in N \mid a_{1j} \otimes u_j > b\}$. The interval vector $\hat{x}$ defined by

$$\hat{x}_j := \begin{cases} [b, b] & j \in N_2, \\ [b, u_j] & \text{otherwise,} \end{cases}$$

for $j = 1, 2, \dots, n$ is an interval solution of $A \otimes x = B \otimes x$ with upper bound u .

Proof. Note that $b \leq \hat{x}_j \leq u_j$ for each $j \in N$. Since for $j \in N_2$, $a_{1j} > b$ implies $a_{1j} \otimes b = b$ and for $j \notin N_2$, $a_{1j} \otimes \hat{x}_j \leq a_{1j} \otimes u_j \leq b$, by direct substitution we have

$$A \otimes \hat{x} = \bigoplus_{j=1}^n (a_{1j} \otimes \hat{x}_j) = \left(\bigoplus_{j \in N_2} (a_{1j} \otimes b) \right) \oplus \left(\bigoplus_{j \in N \setminus N_2} (a_{1j} \otimes \hat{x}_j) \right) = b.$$

Next, we have $B \otimes \hat{x} \leq B \otimes u = b$. On the other hand, since $\hat{x}_j \geq b$, there exists k such that $b_{1k} \otimes u_k = b$, giving $B \otimes \hat{x} \geq b$. Hence, $B \otimes \hat{x} = b$ and $A \otimes \hat{x} = B \otimes \hat{x}$. So, $\hat{x}$ is a solution. $\square$

3.2. Maximal Solution of $A \otimes x = B \otimes x$ for $m \times n$ Matrices A, B

In this subsection, we discuss the maximal solution of system 1 where A, B are of size $m \times n$. The maximal solution for $m \times n$ matrices is constructed by repeatedly applying Lemma 5 to each row, since the system $A \otimes x = B \otimes x$ consists of m equations of size $1 \times n$. Several adjustments are needed:

1. Identify rows where $a_i(x) \neq b_i(x)$.
2. Swap rows where $a_i(x) < b_i(x)$.
3. Choose the smallest right-hand side to ensure the condition for N_2 holds for all rows.

Based on these adjustments, the procedure to determine the maximal solution for $m \times n$ matrices is described as follows.

Let $M = \{1, \dots, m\}$ and $N = \{1, \dots, n\}$. Given the system $A \otimes x = B \otimes x$ with $A, B \in \overline{\mathbb{R}}^{m \times n}$, we begin by initialize $k = 0$, $u = (\infty, \infty, \dots, \infty)^T$, and $\bar{x}^0 = u$.

Step 1. Set $k = k + 1$. If any equation in the system has not yet satisfied equality, swap the left and right sides so that

$$a_i(\bar{x}^{k-1}) \geq b_i(\bar{x}^{k-1}) \tag{4}$$

holds for every $i \in M$.

Step 2. Determine

$$M_k = \{r \in M \mid a_r(\bar{x}^{k-1}) \neq b_r(\bar{x}^{k-1})\} \tag{5}$$

and

$$I_k = \{i \in M_k \mid b_i(\bar{x}^{k-1}) = \min_{r \in M_k} b_r(\bar{x}^{k-1})\}. \tag{6}$$

Then choose an index i_k such that $i_k \in I_k$.

Step 3. Determine the maximal solution $\bar{x}^k$ for row i_k with $\bar{x}^{k-1}$ as the upper bound. The solution is given by Lemma 5:

$$\bar{x}_j^k := \begin{cases} b_k & \text{if } j \in J_k, \\ \bar{x}_j^{k-1} & \text{otherwise,} \end{cases}$$

for $j = 1, 2, \dots, n$, where

$$b_k = b_{i_k}(\bar{x}^{k-1}) = \bigoplus_{j=1}^n (b_{i_k j} \otimes \bar{x}_j^{k-1}), \quad (7)$$

and

$$J_k = \{j \in N \mid a_{i_k j} \otimes \bar{x}_j^{k-1} > b_k\}. \quad (8)$$

Step 4. Check whether $\bar{x}^k$ is a solution of $A \otimes x = B \otimes x$. If yes, $\bar{x}^k$ is the maximal solution and the process stops. Otherwise, return to Step 1 and repeat until a maximal solution is obtained.

The procedure above is formalized in Algorithm 1 below.

Algorithm 1: Determination of the maximal solution of $A \otimes x = B \otimes x$

Input : Matrices $A, B \in \overline{\mathbb{R}}^{m \times n}$

Output : Maximal solution $\bar{x} \in \overline{\mathbb{R}}^n$

```

1  $k \leftarrow 0$ 
2  $u \leftarrow (\infty, \infty, \dots, \infty)^T$ 
3  $\bar{x}^0 \leftarrow u$ 
4 while True do
5    $k \leftarrow k + 1$ 
6   for  $i \leftarrow 1$  to  $m$  do
7     if  $a_i(\bar{x}^{k-1}) < b_i(\bar{x}^{k-1})$  then
8       swap  $a_i$  and  $b_i$  for row  $i$  // Step 1: swap rows
9    $M_k \leftarrow \{r \in M \mid a_r(\bar{x}^{k-1}) \neq b_r(\bar{x}^{k-1})\}$  // Step 2a: identify unsatisfied rows
10  if  $M_k = \emptyset$  then
11    return  $\bar{x}^{k-1}$  // solution found
12   $I_k \leftarrow \{i \in M_k \mid b_i(\bar{x}^{k-1}) = \min_{r \in M_k} b_r(\bar{x}^{k-1})\}$  // Step 2b: select row with
    min RHS
13  choose  $i_k \in I_k$ 
14   $b_k \leftarrow b_{i_k}(\bar{x}^{k-1}) = \bigoplus_{j=1}^n (b_{i_k j} \otimes \bar{x}_j^{k-1})$  // Step 3a: compute  $b_k$ 
15   $J_k \leftarrow \{j \in N \mid a_{i_k j} \otimes \bar{x}_j^{k-1} > b_k\}$  // Step 3b: compute  $J_k$ 
16  for  $j \leftarrow 1$  to  $n$  do
17    if  $j \in J_k$  then
18       $\bar{x}_j^k \leftarrow b_k$  // Step 3c: update solution vector
19    else
20       $\bar{x}_j^k \leftarrow \bar{x}_j^{k-1}$ 
21  if  $\bar{x}^k$  is a solution of  $A \otimes x = B \otimes x$  then
22    return  $\bar{x}^k$  // Step 4: check solution
23 return  $\bar{x}^k$ 

```

To prove the validity of Algorithm 1, we first present Lemma 7 through Lemma 10, which will be used to prove Lemma 11 and Lemma 12. These lemmas establish that Algorithm 1 produces a solution and that this solution is maximal. Finally, Lemma 11 and Lemma 12 are used to prove Theorem 1, which states that Algorithm 1 yields a maximal solution.

Lemma 7 discusses the interval solution at each cycle.

Lemma 7. Let $\bar{x}^k$ be the solution for row i_k in Step 3 of cycle k . Then $\bar{x}^k$ can be replaced by any value in the interval $\hat{x}^k$ defined by

$$\hat{x}_j^k := \begin{cases} [b_k, b_k], & j \in J_k, \\ [b_k, \bar{x}_j^{k-1}], & \text{otherwise,} \end{cases}$$

for $j = 1, 2, \dots, n$, and we have

$$a_{i_k}(\hat{x}^k) = b_{i_k}(\hat{x}^k).$$

Proof. Given that $\bar{x}^k$ is the maximal solution obtained in Step 3. From Steps 1 and 2, we have $a_{i_k}(\bar{x}^{k-1}) > b_{i_k}(\bar{x}^{k-1})$. By Lemma 6, setting $a = a_{i_k}(\bar{x}^{k-1})$, $b = b_{i_k}(\bar{x}^{k-1}) = b_k$, and $N_2 = J_k$, we obtain that

$$\hat{x}_j^k := \begin{cases} [b_k, b_k], & j \in J_k, \\ [b_k, \bar{x}_j^{k-1}], & \text{otherwise,} \end{cases}$$

for $j = 1, \dots, n$ is a solution of

$$(a_{i_k1} \ a_{i_k2} \ \cdots \ a_{i_kn}) \otimes \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = (b_{i_k1} \ b_{i_k2} \ \cdots \ b_{i_kn}) \otimes \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Thus, $a_{i_k}(\hat{x}^k) = b_{i_k}(\hat{x}^k)$. $\square$

Lemma 8 shows that as the cycle increases, the solution vector becomes smaller or remains stable.

Lemma 8. Let $\bar{x}^k$ be defined as in Step 3 of cycle k , and let $\bar{x}^{k-1}$ be from cycle $(k-1)$. Then

$$\bar{x}^k \leq \bar{x}^{k-1}.$$

Proof. Recall that

$$\bar{x}_j^k := \begin{cases} b_k & \text{if } j \in J_k, \\ \bar{x}_j^{k-1} & \text{otherwise.} \end{cases}$$

For $j \in J_k$, we have $\bar{x}_j^k = b_k < a_{i_kj} \otimes \bar{x}_j^{k-1} \leq \bar{x}_j^{k-1}$, hence $\bar{x}_j^k < \bar{x}_j^{k-1}$. For $j \notin J_k$, we have $\bar{x}_j^k = \bar{x}_j^{k-1}$. Therefore $\bar{x}_j^k \leq \bar{x}_j^{k-1}$ for all j , so $\bar{x}^k \leq \bar{x}^{k-1}$. $\square$

Lemma 9 proves that as the cycle increases, the minimum value on the right-hand side remains stable or increases. Let S_1 be the set of rows that have not yet satisfied the equation in cycle k , and S_2 the set of rows that have already satisfied it:

$$S_1 = \{s \in M \setminus I_k \mid b_s(\bar{x}^{k-1}) > b_k\}, \quad S_2 = \{s \in M \setminus I_k \mid b_s(\bar{x}^{k-1}) \leq b_k\}.$$

These sets will be used in the proofs of Lemma 9 and Lemma 11.

Lemma 9. Let b_k be defined as in Step 3 of cycle k with upper bound $\bar{x}^{k-1}$, and let b_{k+1} be defined in cycle $k+1$ with upper bound $\bar{x}^k$. Then

$$b_{k+1} \geq b_k.$$

Proof. By definition, $b_k = \min_{r \in M_k} b_r(\bar{x}^{k-1})$ and $a_r(\bar{x}^{k-1}) > b_r(\bar{x}^{k-1})$. From Step 3, $\bar{x}_j^k = b_k$ for $j \in J_k$ and $\bar{x}_j^k = \bar{x}_j^{k-1}$ otherwise, with $b_k \leq \bar{x}_j^{k-1}$. Hence, $b_k \leq \bar{x}_j^k$ for all j . For any $s \in S_1$, we have $b_s(\bar{x}^{k-1}) > b_k$. Thus there exists j^* such that $b_{sj^*} \otimes \bar{x}_{j^*}^{k-1} > b_k$, implying $b_{sj^*} > b_k$. Since $\bar{x}_j^k \geq b_k$, we obtain $\min(b_{sj^*}, \bar{x}_{j^*}^k) \geq \min(b_{sj^*}, b_k) = b_k$. Therefore $b_s(\bar{x}^k) \geq b_k$ for every $s \in S_1$. Now $b_{k+1} = \min_{s \in S_1} b_s(\bar{x}^k) \geq b_k$, completing the proof. $\square$

Using Lemma 8 and Lemma 9, we now show in Lemma 10 that once a variable is decreased, its value does not change in subsequent cycles.

Lemma 10. For $j \in J_k$, once $\bar{x}_j^k$ is determined in iteration k , its value remains unchanged for all cycles l with $l > k$ and $l \leq K$.

Proof. Since $j \in J_k$, we have $\bar{x}_j^k = b_k$. Take any $l > k$ and suppose, for contradiction, that $\bar{x}_j^l \neq \bar{x}_j^k$. Then by Step 3, $j \in J_l$ and $\bar{x}_j^l = b_l$. By Lemma 9, $b_l \geq b_k$. If $b_l > b_k$, then $\bar{x}_j^l = b_l > b_k = \bar{x}_j^k$, contradicting Lemma 8 which states $\bar{x}_j^l \leq \bar{x}_j^k$. If $b_l = b_k$, then $\bar{x}_j^l = b_l = b_k = \bar{x}_j^k$, contradicting the assumption $\bar{x}_j^l \neq \bar{x}_j^k$. Hence $\bar{x}_j^l = \bar{x}_j^k$. $\square$

Lemma 11 states that any row that has already satisfied the equation remains satisfied in subsequent cycles.

Lemma 11. *Let $\bar{x}^k$ be defined as in Step 3. Then for every $s \in S_2$,*

$$a_s(\bar{x}^k) = b_s(\bar{x}^k).$$

Proof. For any $s \in S_2$, we know $a_s(\bar{x}^{k-1}) = b_s(\bar{x}^{k-1})$. By Lemma 7, replacing $\bar{x}^{k-1}$ with any value in the interval vector $\hat{x}^{k-1}$ preserves the solution. From Lemma 9 and Lemma 8, we have $b_{k-1} \leq b_k \leq \bar{x}_j^{k-1}$ and $\bar{x}_j^{k-1} \leq \bar{x}_j^{k-2}$ for all $j \in N$. Thus $b_k, \bar{x}_j^{k-1} \in [b_{k-1}, \bar{x}_j^{k-2}]$. Now we show $\bar{x}^k \in \hat{x}^{k-1}$. If $j \in J_{k-1}$, then by Lemma 10 we obtain $\bar{x}_j^k = \bar{x}_j^{k-1} = b_{k-1}$. If $j \notin J_{k-1}$ and $j \in J_k$, then $\bar{x}_j^k = b_k \in [b_{k-1}, \bar{x}_j^{k-2}]$. If $j \notin J_{k-1}$ and $j \notin J_k$, then $\bar{x}_j^k = \bar{x}_j^{k-1} \in [b_{k-1}, \bar{x}_j^{k-2}]$. Hence $\bar{x}^k \in \hat{x}^{k-1}$, so $\bar{x}^k$ is a solution for every row $s \in S_2$. $\square$

Lemma 12 proves that no solution of $A \otimes x = B \otimes x$ is greater than $\bar{x}^k$ for any cycle k .

Lemma 12. *Maximal solution of $A \otimes x = B \otimes x$ does not exceed $\bar{x}^k$ from Step 3 for any cycle k .*

Proof. Let $\tilde{x}$ be any solution. We prove $\tilde{x} \not\geq \bar{x}^k$ by induction. For $k = 1$, by Lemma 5, $\bar{x}^1$ is the maximal solution for row i_1 with upper bound $\bar{x}^0$. Since $\tilde{x}$ is also a solution, we must have $\tilde{x} \not\geq \bar{x}^1$. Assume the statement holds for cycle $k - 1$. Suppose $\tilde{x} > \bar{x}^k$, then there exists $j \in N$ such that $\tilde{x}_j > \bar{x}_j^k$. If $j \notin J_k$, then $\bar{x}_j^k = \bar{x}_j^{k-1}$, so $\tilde{x}_j > \bar{x}_j^{k-1}$, contradicting the induction hypothesis. If $j \in J_k$, then $\bar{x}_j^k = b_{i_k}(\bar{x}^{k-1})$, so $\tilde{x}_j > b_{i_k}(\bar{x}^{k-1})$. By definition of J_k , $a_{i_k j} \otimes \bar{x}_j^{k-1} > b_{i_k}(\bar{x}^{k-1})$, which implies $a_{i_k j} > b_{i_k}(\bar{x}^{k-1})$. Consequently,

$$a_{i_k}(\tilde{x}) = \bigoplus_{j=1}^n a_{i_k j} \otimes \tilde{x}_j > b_{i_k}(\bar{x}^{k-1}).$$

Since $\tilde{x}$ is a solution, $b_{i_k}(\tilde{x}) = a_{i_k}(\tilde{x}) > b_{i_k}(\bar{x}^{k-1})$, so

$$\bigoplus_{j=1}^n b_{i_k j} \otimes \tilde{x}_j > \bigoplus_{j=1}^n b_{i_k j} \otimes \bar{x}_j^{k-1}.$$

Thus there exists $l \in N$ such that $b_{i_k l} \otimes \tilde{x}_l > b_{i_k l} \otimes \bar{x}_l^{k-1}$, implying $\tilde{x}_l > \bar{x}_l^{k-1}$. This again contradicts the induction hypothesis. Hence $\tilde{x} \not\geq \bar{x}^k$, completing the induction. $\square$

Algorithm 1 produces a maximal solution for the linear system $A \otimes x = B \otimes x$, as proved in Theorem 1.

Theorem 1. *The maximal solution $\bar{x}$ of the system $A \otimes x = B \otimes x$ with $A, B \in \mathbb{R}^{m \times n}$ is given by Algorithm 1, with $\bar{x} = \bar{x}^K$.*

Proof. By Lemma 11, $\bar{x}^K$ is a solution of $A \otimes x = B \otimes x$. By Lemma 12, for any solution $\tilde{x}$, we have $\tilde{x} \not\geq \bar{x}^k$ for every cycle k . In particular, for cycle K , $\tilde{x} \not\geq \bar{x}^K$. Hence $\bar{x}^K$ is a maximal solution of $A \otimes x = B \otimes x$. $\square$

In one cycle, Algorithm 1 computes $\bar{x}^k$ as a solution of $A \otimes x = B \otimes x$. Since there are m rows and each row involves operations on n columns, the number of operations per cycle is $m \times n$. Thus, each cycle has complexity $O(mn)$. Lemma 10 shows that the number of cycles is K with

$K \leq m$, meaning the maximum number of cycles required is m . Therefore, the total complexity is $m \times O(mn) = O(m^2n)$.

For the $m \times n$ case, the same result as Lemma 3 holds. That is, if $\bar{x}$ is the maximal solution without constraints, then $\bar{x}$ is the maximal solution with lower bound l if $l \leq \bar{x}$ and it has no solution if $l \not\leq \bar{x}$. The proof is analogous to the $1 \times n$ case.

From Subsections 3.1 and 3.2, the system $A \otimes x = B \otimes x$ always has a solution when there are no constraints or when an upper bound $x \leq u$ is given. For $1 \times n$ without constraints, Lemma 2 explicitly constructs a solution. For $1 \times n$ with an upper bound, Lemma 5 provides a direct formula. For $m \times n$, Algorithm 1 converges by Lemma 10 and produces a solution (Theorem 1). For a lower bound $l \leq x$, the existence of the maximal solution depends on l (Lemma 3). The uniqueness of the maximal solution is given in Theorem 2.

Theorem 2. *The maximal solution of $A \otimes x = B \otimes x$ produced by Algorithm 1 is unique.*

Proof. Let $\bar{x}, \bar{y}$ be two maximal solutions. Define $\bar{z} = \max(\bar{x}, \bar{y})$. Using distributivity,

$$A \otimes \bar{z} = (A \otimes \bar{x}) \oplus (A \otimes \bar{y}) = (B \otimes \bar{x}) \oplus (B \otimes \bar{y}) = B \otimes \bar{z},$$

so $\bar{z}$ is a solution. Since $\bar{x} \leq \bar{z}$ and $\bar{x}$ is maximal, $\bar{x} = \bar{z}$. Similarly, $\bar{y} = \bar{z}$. Hence, $\bar{x} = \bar{y}$. $\square$

3.3. Application to Logistics Distribution Network Using Python

This section presents an application of maximal solutions of two-sided linear systems in max-min algebra to the planning of logistics distribution infrastructure.

Consider a transportation system connecting three industrial areas (A , B , and C) as the origin locations of goods to one main distribution center (T). To reach the distribution center T , each transport vehicle must pass through one of three transit points or toll gates (X , Y , and Z). The roads connecting the industrial areas to the toll gates are existing networks, while the roads connecting the toll gates to the distribution center are special access roads whose capacities are to be determined. The capacities of the existing roads, for both outward and return flows, may have different values. The distribution network scheme is shown in the following figure.

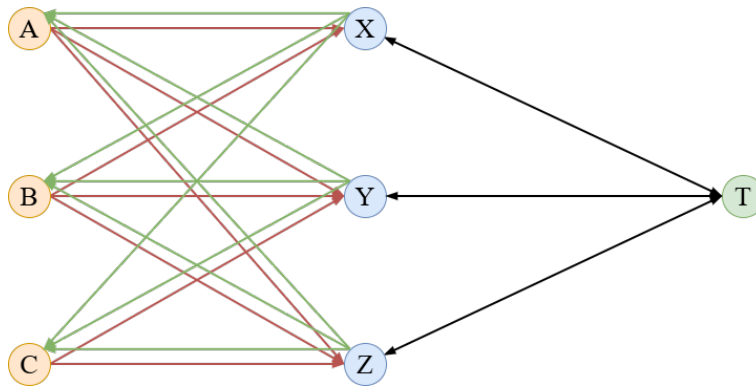


Fig. 1: Schematic diagram of the logistics distribution network

The road capacities (in passenger car units per hour, smp/hour) for each segment are given in Table 1 below.

Table 1: Forward and return road capacity

Forward road capacity			Return road capacity		
Symbol	Route	Capacity (smp/hour)	Symbol	Route	Capacity (smp/hour)
a_{11}	$A \rightarrow X$	1500	b_{11}	$X \rightarrow A$	1000
a_{12}	$A \rightarrow Y$	800	b_{12}	$Y \rightarrow A$	900
a_{13}	$A \rightarrow Z$	1200	b_{13}	$Z \rightarrow A$	1300
a_{21}	$B \rightarrow X$	900	b_{21}	$X \rightarrow B$	800
a_{22}	$B \rightarrow Y$	1600	b_{22}	$Y \rightarrow B$	1100
a_{23}	$B \rightarrow Z$	1100	b_{23}	$Z \rightarrow B$	1200
a_{31}	$C \rightarrow X$	1300	b_{31}	$X \rightarrow C$	1400
a_{32}	$C \rightarrow Y$	700	b_{32}	$Y \rightarrow C$	1000
a_{33}	$C \rightarrow Z$	1700	b_{33}	$Z \rightarrow C$	1500

We aim to determine the maximal capacity of the special access roads from each toll gate to the distribution center, namely $x = (x_1, x_2, x_3)$, where x_1, x_2, x_3 respectively denote the two-way road capacities from X, Y, Z to T in smp/hour. The purpose of determining these capacities is to ensure that the incoming and outgoing vehicle flows remain balanced. With such balance achieved, logistics circulation at the distribution center will run smoothly and the risk of congestion or truck accumulation in the distribution center area can be avoided. Based on Table 1, the maximal road capacities for each area toward the distribution center are as follows.

1. Area A

(a) Forward capacity = $\max(\min(a_{11}, x_1), \min(a_{12}, x_2), \min(a_{13}, x_3))$

(b) Return capacity = $\max(\min(b_{11}, x_1), \min(b_{12}, x_2), \min(b_{13}, x_3))$

2. Area B

(a) Forward capacity = $\max(\min(a_{21}, x_1), \min(a_{22}, x_2), \min(a_{23}, x_3))$

(b) Return capacity = $\max(\min(b_{21}, x_1), \min(b_{22}, x_2), \min(b_{23}, x_3))$

3. Area C

(a) Forward capacity = $\max(\min(a_{31}, x_1), \min(a_{32}, x_2), \min(a_{33}, x_3))$

(b) Return capacity = $\max(\min(b_{31}, x_1), \min(b_{32}, x_2), \min(b_{33}, x_3))$

To avoid vehicle congestion at the distribution center T , the forward and return capacities must be balanced for each area. Thus, we obtain the following two-sided linear system:

$$\max(\min(a_{11}, x_1), \min(a_{12}, x_2), \min(a_{13}, x_3)) = \max(\min(b_{11}, x_1), \min(b_{12}, x_2), \min(b_{13}, x_3))$$

$$\max(\min(a_{21}, x_1), \min(a_{22}, x_2), \min(a_{23}, x_3)) = \max(\min(b_{21}, x_1), \min(b_{22}, x_2), \min(b_{23}, x_3))$$

$$\max(\min(a_{31}, x_1), \min(a_{32}, x_2), \min(a_{33}, x_3)) = \max(\min(b_{31}, x_1), \min(b_{32}, x_2), \min(b_{33}, x_3))$$

or equivalently, $A \otimes x = B \otimes x$ with

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}, \quad \text{and} \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Applying Algorithm 1 to the system above, we obtain the maximal solution

$$x = (1300, 1200, 1500)^T.$$

This means that the special access roads from toll gates X , Y , and Z to the distribution center T should be built with capacities of 1300 smp/hour, 1200 smp/hour, and 1500 smp/hour, respectively. With these capacities, the forward and return flows for each industrial area are balanced. Consequently, no congestion or truck accumulation occurs at the distribution center, and the logistics system operates efficiently.

4. Conclusion

This paper provides a Python implementation of the algorithm for determining the maximal solution of two-sided linear systems in max-min algebra, along with a numerical application to a logistics distribution network. The theoretical results show that for the $1 \times n$ case, the maximal solution can be explicitly constructed under three scenarios: without constraints, with a given upper bound, and with a given lower bound. These results are then generalized to the $m \times n$ case, where the algorithm terminates after at most m iterations and produces a unique maximal solution with complexity $O(m^2n)$. The implementation makes the algorithm accessible to practitioners, and the logistics example demonstrates how the model can be used to determine road capacities. While the example uses simulated data, the approach is applicable to capacity planning problems in logistics and transportation. Future work may include applying the algorithm to real-world data or extending the implementation to other programming languages.

CRedit Authorship Contribution Statement

Zakia Nur Ramadhani Putri: Methodology, Formal Analysis, Software, Validation, Investigation, Writing – Original Draft Preparation, Visualization. **Ari Suparwanto:** Conceptualization, Supervision, Writing – Review & Editing. **Sutopo:** Conceptualization, Supervision, Writing – Review & Editing.

Declaration of Generative AI and AI-assisted technologies

During the preparation of this manuscript, the authors used ChatGPT (OpenAI) for writing assistance, language refinement, and proofreading. The AI tool was employed to assist in structuring the manuscript, improving sentence flow, and checking grammatical consistency. The authors have thoroughly reviewed and edited the AI-generated suggestions and take full responsibility for the final content of the publication.

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

This research received no external funding. The authors thank the Department of Mathematics, Universitas Gadjah Mada, for providing the necessary facilities and support during this study.

Data and Code Availability

The Python source code implementing Algorithm 1 is provided in Appendix A of this manuscript. The numerical example in Section 3 uses simulated data constructed solely for illustrative purposes, and all relevant data are included within the text. The code is also available from the corresponding author upon reasonable request.

References

- [1] Cuninghame-Green and A. Raymond. *Minimax algebra*. Springer Science & Business Media, 1979. DOI: [10.1007/978-3-642-48708-8](https://doi.org/10.1007/978-3-642-48708-8).
- [2] M. Gondran and M. Minoux. *Graphs, dioids and semirings: new models and algorithms*. Springer, 2008. DOI: [10.1007/978-0-387-75450-5](https://doi.org/10.1007/978-0-387-75450-5).
- [3] E. Sanchez. “Resolution of composite fuzzy relation equations”. In: *Information and Control* 30.1 (1976), pp. 38–48. DOI: [10.1016/S0019-9958\(76\)90446-0](https://doi.org/10.1016/S0019-9958(76)90446-0).

- [4] E. Sanchez. “Inverses of fuzzy relations. Application to possibility distributions and medical diagnosis”. In: *Fuzzy Sets and Systems* 2.1 (1979), pp. 75–86.
- [5] E. S. Santos. “Maximin automata”. In: *Information and Control* 13.4 (1968), pp. 363–377. DOI: [10.1016/S0019-9958\(68\)90864-4](https://doi.org/10.1016/S0019-9958(68)90864-4).
- [6] P. Butkovič, K. Cechlárová, and P. Szabó. “Strong linear independence in bottleneck algebra”. In: *Linear Algebra and its Applications* 94 (1987), pp. 133–155. DOI: [10.1016/0024-3795\(87\)90085-1](https://doi.org/10.1016/0024-3795(87)90085-1).
- [7] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn, and K. Zimmermann. *Linear optimization problems with inexact data*. Springer Science & Business Media, 2006. DOI: [10.1007/0-387-32698-7](https://doi.org/10.1007/0-387-32698-7).
- [8] M. Gavalec and K. Zimmermann. “Solving systems of two-sided (max, min)–linear equations”. In: *Kybernetika* 46.3 (2010), pp. 405–414. URL: <http://eudml.org/doc/33986>.
- [9] P. Krbálek and A. Pozdílková. “Maximal solutions of two-sided linear systems in max–min algebra”. In: *Kybernetika* 46.3 (2010), pp. 501–512. URL: <http://eudml.org/doc/33988>.
- [10] H. Myšková and J. Plavka. “Optimizing the max-min function with a constraint on a two-sided linear system”. In: *AIMS Mathematics* 9.4 (2024), pp. 7791–7809. DOI: [10.3934/math.2024378](https://doi.org/10.3934/math.2024378).
- [11] P. Li. “Linear optimization over the approximate solutions of a system of max-min equations”. In: *Fuzzy Sets and Systems* 485 (2024), p. 108946. DOI: [10.1016/j.fss.2024.108946](https://doi.org/10.1016/j.fss.2024.108946).
- [12] H. Myšková and J. Plavka. “AE and EA versions of X-robustness for interval circulant matrices in max–min algebra”. In: *Linear and Multilinear Algebra* 71.15 (2022), pp. 2441–2461. DOI: [10.1080/03081087.2022.2105783](https://doi.org/10.1080/03081087.2022.2105783).
- [13] J. Plavka. “Strong X-robustness of interval max-min matrices”. In: *Kybernetika* 58.1 (2022), pp. 50–68. DOI: [10.14736/kyb-2022-1-0050](https://doi.org/10.14736/kyb-2022-1-0050).
- [14] Y. Ooga, Y. Nishida, and Y. Watanabe. “On max-plus two-sided linear systems whose solution sets are min-plus linear”. In: *Linear Algebra and its Applications* 694 (2024), pp. 283–306. DOI: [10.1016/j.laa.2024.04.014](https://doi.org/10.1016/j.laa.2024.04.014).
- [15] M. S. Mufid, E. Patel, and S. Sergeev. “Solving linear equations over maxmin- ω systems”. In: *Linear Algebra and its Applications* 679 (2023), pp. 1–32. DOI: [10.1016/j.laa.2023.10.012](https://doi.org/10.1016/j.laa.2023.10.012).
- [16] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *Synchronization and linearity*. Vol. 1. Wiley New York, 1992.
- [17] Subiono. *Aljabar Min-Max Plus dan Terapannya*. Surabaya: Jurusan Matematika, Institut Teknologi Sepuluh Nopember, 2015.
- [18] A. Rudhito. “Aljabar Max-Min Interval”. In: *Prosiding Seminar Nasional Penelitian, Pendidikan, dan Penerapan MIPA*. Yogyakarta, 2013.

A. Python Implementation of Algorithm 1

```
import numpy as np

# Function to print a separator line
def print_separator(title=""):
    print(f"\n{'='*50}")
    if title:
        print(f"{title}")
        print(f"{'='*50}")
```

```
4859
4860 print("Enter the number of rows and columns")
4871 m = int(input("Number of rows: "))
4882 n = int(input("Number of columns: "))
4893
4904 print("Enter elements of matrix A")
4915 A = [[float(input(f"a_{i+1},{j+1}=")) for j in range(n)] for i in range
492 (m)]
4936 print("Enter elements of matrix B")
4947 B = [[float(input(f"b_{i+1},{j+1}=")) for j in range(n)] for i in range
495 (m)]
4968
4979 print(f"\nA={np.array(A)}")
4980 print(f"B={np.array(B)}")
4991
5002 k = 0
5013 x = [float('inf')] * n
5024
5035 print_separator("START ALGORITHM")
5046
5057 while k < m + 1:
5068     k += 1
5079
5080     # ===== Step 1 (swap sides) =====
5091     for i in range(m):
5102         a_i = max(min(A[i][j], x[j]) for j in range(n))
5113         b_i = max(min(B[i][j], x[j]) for j in range(n))
5124         if a_i <= b_i:
5135             A[i], B[i] = B[i], A[i]
5146
5157     # ===== Step 2 (M_k and I_k) =====
5168     rhs = []
5179     lhs = []
5180     M_k = []
5191     I_k = []
5202
5213     for i in range(m):
5224         a_i = max(min(A[i][j], x[j]) for j in range(n))
5235         b_i = max(min(B[i][j], x[j]) for j in range(n))
5246
5257         if a_i != b_i:
5268             M_k.append(i)
5279
5280         rhs.append(b_i)
5291         lhs.append(a_i)
5302
5313     # Display Step 1 & 2 results
5324     print_separator(f"ITERATION {k}")
5335     print(f"A={np.array(A)}")
5346     print(f"B={np.array(B)}")
5357     print(f"lhs(a_i)={lhs}")
5368     print(f"rhs(b_i)={rhs}")
5379     print(f"\nM_{k}={ [i+1 for i in M_k] }")
5380
5391     # Check if M_k is empty (solution found)
5402     if not M_k:
5413         print(f"SOLUTION FOUND!")
5424         print(f"x={np.array(x)}")
```

```

5435         break
5446
5457     # Compute b_min and I_k
5468     b_min = min(rhs[i] for i in M_k)
5479     for i in M_k:
5480         if rhs[i] == b_min:
5491             I_k.append(i)
5502
5513     i_k = I_k[0]
5524     b_k = b_min
5535
5546     print(f"I_{k}={ [i+1 for i in I_k] }")
5557     print(f"b_{k}={b_k}")
5568     print(f"i_{k}={i_k+1}")
5579
5580     # ===== Step 3 (J_k and update x) =====
5591     J_k = []
5602     for j in range(n):
5613         if min(A[i_k][j], x[j]) > b_k:
5624             J_k.append(j)
5635
5646     # Update x
5657     for j in J_k:
5668         x[j] = b_k
5679
5680     print(f"J_{k}={ [j+1 for j in J_k] }")
5691     print(f"x={np.array(x)}")
5702
5713     # ===== Step 4 (Check solution) =====
5724     ai_new = []
5735     bi_new = []
5746     for i in range(m):
5757         a_i = max(min(A[i][j], x[j]) for j in range(n))
5768         b_i = max(min(B[i][j], x[j]) for j in range(n))
5779         ai_new.append(a_i)
5780         bi_new.append(b_i)
5791
5802     print(f"Solution check: ai={ai_new}, bi={bi_new}")
5813
5824     if all(ai_new[i] == bi_new[i] for i in range(m)):
5835         print_separator("SOLUTION FOUND")
5846         print(f"x={np.array(x)}")
5857         break
5868
5879 print_separator("END OF ALGORITHM")
588

```

Listing 1: Python implementation of Algorithm 1