symmetry

MDPI

Article

An Efficient Algorithm for Eigenvalue Problem of Latin Squares in a Bipartite Min-Max-Plus System

Mubasher Umer [1], Umar Hayat [1], Fazal Abbas [2,*], Anurag Agarwal [3] and Petko Kitanov [4]

[1] Department of Mathematics, Quaid-i-Azam University, Islamabad 45320, Pakistan; mumer@math.qau.edu.pk (M.U.); umar.hayat@qau.edu.pk (U.H.)
[2] Department of Mathematics and Computer Sciences, Stetson University, DeLand, FL 32723, USA
[3] School of Mathematical Sciences, Rochester Institute of Technology, Rochester, NY 14623, USA; axasma@rit.edu
[4] Department of Mathematics, Wells College, Aurora, NY 13026, USA; pkitanov@gmail.com
* Correspondence: fabbas1@stetson.edu; Tel.: +1-(585)-993-4291

Received: 8 January 2020; Accepted: 12 February 2020; Published: 21 February 2020

Abstract: In this paper, we consider the eigenproblems for Latin squares in a bipartite min-max-plus system. The focus is upon developing a new algorithm to compute the eigenvalue and eigenvectors (trivial and non-trivial) for Latin squares in a bipartite min-max-plus system. We illustrate the algorithm using some examples. The proposed algorithm is implemented in MATLAB, using max-plus algebra toolbox. Computationally speaking, our algorithm has a clear advantage over the power algorithm presented by Subiono and van der Woude. Because our algorithm takes 0.088783 s to solve the eigenvalue problem for Latin square presented in Example 2, while the compared one takes 1.718662 s for the same problem. Furthermore, a time complexity comparison is presented, which reveals that the proposed algorithm is less time consuming when compared with some of the existing algorithms.

Keywords: bipartite min-max-plus systems; eigenvalue and eigenvectors; latin squares

1. Introduction

Time evolution of discrete event dynamic systems can be described through equations composed using three operations the maximum, the minimum and the addition. Such systems are described as min-max-plus systems. The bipartite min-max-plus systems are determined by the union of two sets of equations: one set of equations containing the maximization and the addition; and another set of equations containing the minimization and the addition.

The idea of max-plus algebra was first seen in the 1950s or even at an earlier period. Nowadays, it has been studied intensively. In [1], the authors have investigated the set of invertible linear operators over a subalgebra of max-plus algebra. For a matrix in max-plus algebra, some necessary and sufficient conditions have been established to possess various types of g-inverses including Moore–Penrose inverse [2]. In [3], the authors have characterized the linear operators which preserve the maximal column rank of matrices over max-plus algebra.

Max-plus algebra has many applications in mathematics as well as in different areas such as mathematical physics, optimization, combinatorics, and algebraic geometry [4]. Max-plus algebra is used in machine scheduling, telecommunication networks, control theory, manufacturing systems, parallel processing systems, and traffic control [5–7]. Max-plus algebra is also used in image steganography [8]. Max-plus algebra can be used to model disk events related to synchronization and time delays [9]. In [10] the authors have shown how max-plus algebra can be helpful for

the dynamic programming of algorithms. In [11], the author has described the whole Dutch railway system by a max-plus system.

The eigenproblems for a matrix A of order $n \times n$ are the problems to find out an eigenvalue λ and the eigenvector v, such that $Av = \lambda v$. In this article, we consider eigenproblems for the bipartite min-max-plus systems. Mostly, power algorithm [12], is used to compute the eigenvalue and eigenvectors in an iterative way. Umer et al. [13] developed an efficient algorithm to solve the eigenvalue problem in max-plus systems, which computes the eigenvalue as a maximal cycle mean and use an iterative approach to determine the eigenvectors. In [14], the authors have demonstrated that the presence of eigenvectors (non-trivial) depend on the position of the greatest and the least elements in the Latin squares in a bipartite min-max-plus system. For further study of eigenproblems for discrete event systems, see [10,14–16].

As per our knowledge and from the literature review, one of the open problems in the max-plus algebra is to come up with an efficient and fast algorithm to solve the eigenproblems for these systems. In this paper, a robust algorithm is developed, that can calculate the eigenvectors corresponding to an eigenvalue λ in an iterative way. In particular, we apply this algorithm to find the eigenvectors (trivial and non-trivial) for Latin squares in bipartite min-max-plus systems. A computational comparison of the proposed algorithm and the power algorithm presented by Subiono and van der Woude [12] is given, which shows the efficiency of the proposed algorithm.

The structure of the paper is along these lines. First of all, some preliminary notions and bipartite min-max-plus systems are presented in Section 2, then a robust algorithm is developed for determining the eigenvalue and eigenvectors of such systems. In Section 3, we present these systems for the Latin squares and calculate trivial and non-trivial eigenvectors for such models. The proposed algorithm and the power algorithm presented by Subiono and van der Woude [12], are implemented in MATLAB (using max-plus algebra toolbox for MATLAB [17]). A time complexity comparison of these algorithms has been given at the end of Section 3. Finally, the conclusion are made in Section 4.

2. Bipartite Min-Max-Plus Systems

First of all we denote the set of real numbers and the set of whole numbers by $\mathbb{R}$ and $\mathbb{W}$ respectively, then we define $\mathbb{R}_\epsilon = \mathbb{R} \cup \{\epsilon\}$, $\mathbb{R}_\tau = \mathbb{R} \cup \{\tau\}$, where $\epsilon = -\infty$, $\tau = +\infty$. A matrix or a vector where all components equal to $-\infty$ is represented by $\boldsymbol{\epsilon}$, whereas a matrix or a vector where all components equal to $+\infty$ is represented by $\boldsymbol{\tau}$. $\underline{n}$ denotes the set of first n positive integers, i.e., $\underline{n} = \{1, ..., n\}$. The following scalar operations are introduced as;

$$\begin{aligned} r \oplus s &= \max\{r, s\}, \quad \text{for each } r, s \in \mathbb{R}_\epsilon \\ r \oplus' s &= \min\{r, s\}, \quad \text{for each } r, s \in \mathbb{R}_\tau \\ r \otimes s &= r + s, \quad \text{for each } r, s \in \mathbb{R}. \end{aligned}$$

The algebraic structure $\mathbb{R}_{\min} = (\mathbb{R}_\tau, \oplus', \otimes)$ represents min-plus algebra and $\mathbb{R}_{\max} = (\mathbb{R}_\epsilon, \oplus, \otimes)$ represents max-plus algebra. Since in both $\mathbb{R}_{\min}$ and $\mathbb{R}_{\max}$ the multiplication operator is defined by addition, therefore in both systems the notation for multiplication operator is the same.

The collection of all matrices of order $m \times n$ in min-plus and max-plus algebra is represented as $\mathbb{R}_{\min}^{m \times n}$ and $\mathbb{R}_{\max}^{m \times n}$ respectively. While $\mathbb{R}_{\min}^{m}$ and $\mathbb{R}_{\max}^{m}$ denotes the set of all vectors in min-plus and max-plus algebra respectively. The scalar operations to matrices are extended as follows.

Suppose that $A = [a_{ij}]$, $B = [b_{ij}]$, $U = [u_{ij}]$, $V = [v_{ij}]$ such that $A, B \in \mathbb{R}_{\max}^{m \times n}$; $U, V \in \mathbb{R}_{\min}^{m \times n}$ and $\alpha \in \mathbb{R}$ then

$$\begin{aligned} A \oplus B &= [c_{ij}], \quad \text{where } c_{ij} = \max\{a_{ij}, b_{ij}\}, \\ U \oplus' V &= [w_{ij}], \quad \text{where } w_{ij} = \min\{u_{ij}, v_{ij}\}, \\ \alpha \otimes A &= \alpha \otimes [a_{ij}] = \alpha + [a_{ij}], \end{aligned}$$

$$\text{for } i \in \underline{m},\ j \in \underline{n}.$$

If $A \in \mathbb{R}_{\max}^{m\times r}$, $B \in \mathbb{R}_{\max}^{r\times n}$, $U \in \mathbb{R}_{\min}^{m\times r}$ and $V \in \mathbb{R}_{\min}^{r\times n}$, then

$$\begin{aligned} A \otimes B &= [c_{ij}], \quad \text{where } c_{ij} = \bigoplus_{k=1}^{r}(a_{ik} \otimes b_{kj}) = \max\{a_{ik} + b_{kj}\}, \\ U \otimes' V &= [w_{ij}], \quad \text{where } w_{ij} = \bigoplus_{k=1}^{r}{}'(u_{ik} \otimes v_{kj}) = \min\{u_{ik} + v_{kj}\}, \end{aligned}$$

$$\text{for } k \in \underline{r}, i \in \underline{m},\ j \in \underline{n}.$$

One can see that the notation of the multiplication operator in both systems is different. The addition is defined as maximum (minimum) in $\mathbb{R}_{\max}$ (respectively, $\mathbb{R}_{\min}$). A bipartite min-max-plus system can be represented as:

$$\begin{aligned} u_i(l+1) &= max\{a_{i1} + w_1(l), ..., a_{in} + w_n(l)\} \\ w_j(l+1) &= min\{b_{j1} + u_1(l), ..., b_{jm} + u_m(l)\}, \end{aligned}$$

where $u_i(l), a_{ij} \in \mathbb{R}_\epsilon$; $w_j(l), b_{ji} \in \mathbb{R}_\tau$; $l \in \mathbb{W}$ for all $i \in \underline{m}$; $j \in \underline{n}$. These equations can be written as;

$$\begin{aligned} u_i(l+1) &= \bigoplus_{j=1}^{n}(a_{ij} \otimes w_j(l)), \\ w_j(l+1) &= \bigoplus_{i=1}^{m}{}'(b_{ji} \otimes u_i(l)). \end{aligned}$$

The above equations can be denoted as;

$$\left.\begin{aligned} u(l+1) = A \otimes w(l), &\quad for\, l \in \mathbb{W} \\ w(l+1) = B \otimes' u(l), &\quad for\, l \in \mathbb{W} \end{aligned}\right\}, \tag{1}$$

where $\mathbb{W}$ is the set of whole numbers and

$$u(l) = \begin{pmatrix} u_1(l) \\ u_2(l) \\ . \\ . \\ . \\ u_m(l) \end{pmatrix} \in \mathbb{R}_\epsilon^m, \quad w(l) = \begin{pmatrix} w_1(l) \\ w_2(l) \\ . \\ . \\ . \\ w_n(l) \end{pmatrix} \in \mathbb{R}_\tau^n$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{pmatrix} \in \mathbb{R}_{\epsilon}^{m\times n},$$

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1m} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nm} \end{pmatrix} \in \mathbb{R}_{\tau}^{n\times m}.$$

Compactly, system (1) can be written by a mapping $\mathcal{M}(\cdot)$, such that

$$x(l+1) = \mathcal{M}(x(l)), \tag{2}$$

$$\text{where } x(l) = \begin{pmatrix} u(l) \\ w(l) \end{pmatrix}, \mathcal{M}\left(\begin{pmatrix} u(l) \\ w(l) \end{pmatrix}\right) = \begin{pmatrix} A \otimes w(l) \\ B \otimes' u(l) \end{pmatrix}, \text{ for } l \in \mathbb{W}.$$

For a system of type (2), the notion of eigenvalue and eigenvectors is defined as follows. A real number $\lambda \in \mathbb{R}$ is said to be an eigenvalue corresponding to an eigenvector $v \in \mathbb{R}^{m+n}$ if

$$\mathcal{M}(v) = \lambda \otimes v. \tag{3}$$

The cycle time vector in a system of type (2) is defined as

$$\eta(A, B) = \lim_{l\to\infty} x(l).$$

The cycle time vector $\eta(A, B) \in \mathbb{R}_{\epsilon}^{n}$ is a vector such that each component is equal to the eigenvalue for the given system of type (2). The cycle time vector is independent of initial vector $x(0)$. In this section, we determine the eigenvalue λ by computing cycle time vector. To compute the eigenvectors, we propose an iterative algorithm. For this purpose, define $A_{\lambda} = -\lambda \otimes A$ and $B_{\lambda} = -\lambda \otimes B$, corresponding to the eigenvalue λ of a system of type (2). Also define

$$\left.\begin{aligned} u^*(l+1) &= A_{\lambda} \otimes w^*(l) \\ w^*(l+1) &= B_{\lambda} \otimes' u^*(l) \end{aligned}\right\} \tag{4}$$

for $l \in \mathbb{W}$, where $u^*(l) \in \mathbb{R}_{\epsilon}^{m}$ and $w^*(l) \in \mathbb{R}_{\tau}^{n}$.

System (4) can be denoted as,

$$x^*(l+1) = \mathcal{N}(x^*(l)) \tag{5}$$

$$\text{where } x^*(l) = \begin{pmatrix} u^*(l) \\ w^*(l) \end{pmatrix}, \mathcal{N}\left(\begin{pmatrix} u^*(l) \\ w^*(l) \end{pmatrix}\right) = \begin{pmatrix} A_{\lambda} \otimes w^*(l) \\ B_{\lambda} \otimes' u^*(l) \end{pmatrix},$$

for $l \in \mathbb{W}$. Let v be a vector, then $\mathcal{N}^l(v)$ represents a vector obtained after applying $\mathcal{N}$ on the vector v by l times. A relation between (2) and (5) is shown in the following theorem.

Theorem 1. *Let λ be an eigenvalue for a system of type (2) and $v = \begin{pmatrix} u(l) \\ w(l) \end{pmatrix}$ be a vector, then*

$$\mathcal{M}(v) = \lambda \otimes \mathcal{N}(v).$$

Proof. Since

$$\begin{aligned}\mathcal{M}(v) &= \begin{pmatrix} A \otimes w(l) \\ B \otimes' u(l) \end{pmatrix} \\ &= \lambda \otimes (-\lambda) \otimes \begin{pmatrix} A \otimes w(l) \\ B \otimes' u(l) \end{pmatrix} \\ &= \lambda \otimes \begin{pmatrix} (-\lambda) \otimes A \otimes w(l) \\ (-\lambda) \otimes B \otimes' u(l) \end{pmatrix} \\ &= \lambda \otimes \begin{pmatrix} A_\lambda \otimes w(l) \\ B_\lambda \otimes' u(l) \end{pmatrix} \\ &= \lambda \otimes \mathcal{N}(v).\end{aligned}$$

□

Corollary 1. *Let v be an eigenvector corresponding to the eigenvalue λ of a system of type (2). Then*

$$\mathcal{N}(v) = v.$$

Proof. By definition, $\mathcal{M}(v) = \lambda \otimes v$. Using Theorem 1, we get, $\lambda \otimes \mathcal{N}(v) = \lambda \otimes v$. Hence $\mathcal{N}(v) = v$. □

Theorem 2. *Let λ be an eigenvalue of a system of type (2) and let*

$$x^*(l+1) = \mathcal{N}(x^*(l))$$

for $l \in \mathbb{W}$, where $x^*(0)$ is an initial state vector. If $x^*(r) = x^*(s)$ for some integers $r > s \geq 0$, then

$$\mathcal{M}(v) = \lambda \otimes v,$$

where

$$v = x^*(s) \oplus \ldots \oplus x^*(r-1).$$

Proof. From Theorem 1,

$$\begin{aligned}\mathcal{M}(v) &= \lambda \otimes \mathcal{N}(v) \\ &= \lambda \otimes \left\{ \begin{pmatrix} A_\lambda \otimes w^*(s) \\ B_\lambda \otimes' u^*(s) \end{pmatrix} \oplus \ldots \oplus \begin{pmatrix} A_\lambda \otimes w^*(r-1) \\ B_\lambda \otimes' u^*(r-1) \end{pmatrix} \right\} \\ &= \lambda \otimes \{x^*(s+1) \oplus \ldots \oplus x^*(r)\} \\ &= \lambda \otimes v.\end{aligned}$$

□

Now we present Algorithm 1 in the following, which gives the eigenvectors corresponding to the eigenvalue λ. But, first we give three basic assumptions in the following that are necessary for the convergence of the algorithm.

1. For any arbitrary initial vector $x(0)$, the system of type (2), ends up in a periodic behavior, after a finite number of iterations.
2. Eigenvalue of the system exists.
3. Every periodic behavior has the same average weight, equal to eigenvalue.

Algorithm 1 Eigenvectors for systems of type (2)

1. Compute the eigenvalue λ, by calculating the cycle time vector $\eta(A, B)$.
2. Define $A_\lambda = -\lambda \otimes A$, and $B_\lambda = -\lambda \otimes B$.
3. Take an initial state vector $x^*(0)$.
4. Iterate $x^*(l+1) = \mathcal{N}(x^*(l)) = \begin{pmatrix} A_\lambda \otimes w^*(l) \\ B_\lambda \otimes' u^*(l) \end{pmatrix}$, for $l \in \mathbb{W}$, until there are positive integers $r > s \geq 0$, such that $x^*(r) = x^*(s)$.
5. Compute the candidate eigenvector

$$v = x^*(s) \oplus ... \oplus x^*(r-1).$$

6. If $\mathcal{N}(v) = v$ then v is the correct eigenvector and algorithm stops. Else if $\mathcal{N}(v) \neq v$ then go to the following step.
7. Take v as new starting vector and iterate $x^*(l+1) = \mathcal{N}(x^*(l))$, until for some $t \geq 0$, it holds that $x^*(t+1) = x^*(t)$. Finally $x^*(t)$ is an eigenvector.

Theorem 3. *Let λ be an eigenvalue and v be a vector computed as in Algorithm 1 for a system of type (2), then*

$$\mathcal{N}(v) \geq v.$$

Proof. From the given algorithm

$$\begin{aligned} v &= x^*(s) \oplus ... \oplus x^*(r-1) \\ &\geq x^*(l) \quad \text{for all } l \in \{s, ..., r-1\}. \end{aligned}$$

Hence

$$\begin{aligned} \mathcal{N}(v) &\geq \mathcal{N}(x^*(l)) \quad \text{for all } l \in \{s, ..., r-1\} \\ &= x^*(l+1) \quad \text{for all } l \in \{s, ..., r-1\}. \end{aligned}$$

These inequalities imply that

$$\mathcal{N}(v) \geq x^*(s+1) \oplus ... \oplus x^*(r) = v.$$

□

Lemma 1. *Let v be a vector computed as in the above Algorithm 1 for a bipartite min-max-plus system. Let (5) be restarted with $x^*(0) = v$, then*

$$x^*(l+1) \geq x^*(l) \text{ for all } l = 0, 1, 2, ...$$

Proof. Since $x^*(l) = \mathcal{N}^l(v)$, which implies that

$$\begin{aligned} x^*(l+1) &= \mathcal{N}^{l+1}(v) \\ &= \mathcal{N}^l(\mathcal{N}(v)) \\ &\geq \mathcal{N}^l(v) \\ &= x^*(l). \end{aligned}$$

□

Proof of Algorithm 1 . Since the system ends up in a periodic behavior aftera finite number of iterations, therefore step 3 of the algorithm performs. If $\mathcal{N}(v) = v$ then v is the correct eigenvector and clearly algorithm can stop. If $\mathcal{N}(v) \neq v$ then start by considering $x^*(0) = v$ as new initial vector and iterate (5).

By first assumption, there exist integers $r > s \geq 0$, with $x^*(r) = x^*(s)$. By Lemma 1, $x^*(l+1) \geq x^*(l)$ for all $l \in \{s, ..., r-1\}$. Our goal is to show that $x^*(l+1) = x^*(l)$ for all $l \in \{s, ..., r-1\}$. Suppose on contrary that $x^*_{j'}(l'+1) > x^*_{j'}(l')$, for some integers l', j' such that, $s \leq l' \leq r-1$ and $1 \leq j' \leq n$. We obtain that $x^*_{j'}(r) > x^*_{j'}(s)$, which is a contradiction because $x^*(r) = x^*(s)$. Hence $x^*(l+1) = x^*(l)$ for all $l \in \{s, ..., r-1\}$. Which completes the proof. □

Example 1. *Consider a system of type (2), given as follows:*

$$A = \begin{bmatrix} 2 & 0 & 3 \\ 1 & 5 & \epsilon \\ \epsilon & 2 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} \tau & 3 & 1 \\ 2 & 4 & 4 \\ 5 & \tau & 0 \end{bmatrix}.$$

Since the cycle time vector is equal to $(2\ 2\ 2\ 2\ 2\ 2)^T$. *Therefore the eigenvalue* $\lambda = 2$.
We get,

$$A_\lambda = \begin{bmatrix} 0 & -2 & 1 \\ -1 & 3 & \epsilon \\ \epsilon & 0 & 2 \end{bmatrix}, \quad B_\lambda = \begin{bmatrix} \tau & 1 & -1 \\ 0 & 2 & 2 \\ 3 & \tau & -2 \end{bmatrix}.$$

Now, take the initial state vector

$$x^*(0) = \begin{pmatrix} u^*(0) \\ w^*(0) \end{pmatrix} \quad with \quad u^*(0) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad w^*(0) = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

The following sequence is obtained, after iterating (5),

$$\begin{array}{ccccccccccc} x^*(0) & \to & x^*(1) & \to & x^*(2) & \to & x^*(3) & \to & x^*(4) & \to & x^*(5) \\ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} & \to & \begin{pmatrix} 2 \\ 3 \\ 3 \\ -1 \\ 0 \\ -2 \end{pmatrix} & \to & \begin{pmatrix} -1 \\ 3 \\ 0 \\ 2 \\ 2 \\ 1 \end{pmatrix} & \to & \begin{pmatrix} 2 \\ 5 \\ 3 \\ -1 \\ -1 \\ -2 \end{pmatrix} & \to & \begin{pmatrix} -1 \\ 2 \\ 0 \\ 2 \\ 2 \\ 1 \end{pmatrix} & \to & \begin{pmatrix} 2 \\ 5 \\ 3 \\ -1 \\ -1 \\ -2 \end{pmatrix} \end{array}.$$

Since $x^*(5) = x^*(3)$, *therefore* $s = 3$, $r = 5$. *To compute a corresponding eigenvector we proceed with the computation of the vector v.*

$$\begin{aligned} v &= x^*(s) \oplus ... \oplus x^*(r-1) \\ &= x^*(3) \ \oplus \ x^*(4) \end{aligned}$$

$$v = \begin{pmatrix} 2 \\ 5 \\ 3 \\ -1 \\ -1 \\ -2 \end{pmatrix} \oplus \begin{pmatrix} -1 \\ 2 \\ 0 \\ 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 5 \\ 3 \\ 2 \\ 2 \\ 1 \end{pmatrix}.$$

Now verify that either v is the correct eigenvector or not. So

$$\mathcal{M}(v) = \begin{pmatrix} 4 \\ 7 \\ 5 \\ 4 \\ 4 \\ 3 \end{pmatrix} = \lambda \otimes v.$$

Which shows that the eigenvector v is the correct eigenvector obtained by Algorithm 1.

3. Latin Squares in Bipartite (Min, Max, Plus)-Systems

A Latin square is a square matrix of order n, with elements from n independent variables over $\mathbb{R}^+$ in such a way that each row and each column is a different permutation of the n variables [18]. In the following, an example of a Latin square of order 4 is given

$$L = \begin{pmatrix} 3 & 2 & 4 & 1 \\ 4 & 1 & 3 & 2 \\ 2 & 4 & 1 & 3 \\ 1 & 3 & 2 & 4 \end{pmatrix}.$$

We consider the Latin squares of size n in a system of type (2). We have four possibilities for the Latin squares in a system of type (2): (1) The entries of both matrices A and B are $\underline{n}$; (2) The entries of matrix A are $\underline{n}_\epsilon$ and the entries of matrix B are $\underline{n}$; (3) The entries of matrix A are $\underline{n}$ and the entries of matrix B are $\underline{n}_\tau$; (4) The entries of matrix A are $\underline{n}_\epsilon$ and the entries of matrix B are $\underline{n}_\tau$, where $\underline{n}_\epsilon = \{1, ..., n-1, \epsilon\}$ and $\underline{n}_\tau = \{1, ..., n-1, \tau\}$.

In this section, we consider Latin squares for systems of type (2) and Algorithm 1 is extended to calculate the eigenvalue and eigenvectors for such type of systems. In [14], authors show that for Latin squares in a system of type (2), the eigenvalue λ is determined as

$$\lambda = \frac{\max(A) + \min(B)}{2}. \tag{6}$$

Now, we propose Algorithm 2 to solve eigenvalue problem for Latin squares in a system of type (2) as follows:

Algorithm 2 Eigenvalue and Eigenvectors for Latin squares in a system of type (2)

1. Compute the eigenvalue as, $\lambda = \dfrac{\max\,(A) + \min(B)}{2}$.
2. Define $A_\lambda = -\lambda \otimes A$, and $B_\lambda = -\lambda \otimes B$.
3. Take a starting vector $x^*(0)$.
4. Iterate (5), until for some integers $r > s \geq 0$, it holds that $x^*(r) = x^*(s)$.
5. Determine the eigenvector as,

$$v = x^*(s) \oplus ... \oplus x^*(r-1).$$

6. If $\mathcal{N}(v) = v$ then v is the correct eigenvector corresponding to the eigenvalue λ and algorithm can stops. Else if $\mathcal{N}(v) \neq v$ then go to the following step.
7. Take v as initial state vector and iterate $x^*(l+1) = \mathcal{N}(x^*(l))$, until for some $t \geq 0$, it holds that $x^*(t+1) = x^*(t)$. Finally $x^*(t)$ is an eigenvector.

Here we find the eigenvalue and eigenvectors for Latin squares in systems of type (2) by using Algorithm 2. First, recall the power algorithm (Algorithm 3) for systems of type (2), proposed by Subiono and van der Woude [12], then we compare it with Algorithm 2.

Algorithm 3 Eigenproblems for Bipartite (Min, Max, Plus)-Systems

1. Define a starting vector $x(0)$.
2. Iterate (2), until there exist a real number c and positive integers $r; s$, such that $r > s \geq 0$ with $x(r) = c \otimes x(s)$.
3. The eigenvalue is obtained as, $\lambda = \dfrac{c}{r-s}$.
4. Determine the eigenvector as, $v = \oplus_{j=1}^{r-s}(\lambda^{\otimes(r-s-j)} \otimes x(s+j-1))$.
5. If $\mathcal{M}(v) = \lambda \otimes v$ then v is the required eigenvector corresponding to the eigenvalue λ and algorithm can stop. If $\mathcal{M}(v) \neq \lambda \otimes v$, then the algorithm has to be continued as follows.
6. Define $x(0) = v$ as new starting vector and restart (2), until for some r, there holds $x(r+1) = \lambda \otimes x(r)$. Then $x(r)$ is a correct eigenvector of system (2).

To illustrate the Algorithms 2 and 3, consider the following example. In this example, we consider a system of type (2), where A and B are Latin Squares with entries in $\underline{n}_\epsilon$ and $\underline{n}_\tau$ respectively.

Example 2. *Let A and B are Latin squares in a system of type (2), given as follows:*

$$A = \begin{bmatrix} 3 & 2 & \epsilon & 1 \\ \epsilon & 1 & 3 & 2 \\ 2 & 3 & 1 & \epsilon \\ 1 & \epsilon & 2 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 3 & \tau & 1 \\ 3 & \tau & 1 & 2 \\ 1 & 2 & 3 & \tau \\ \tau & 1 & 2 & 3 \end{bmatrix}.$$

1. *The eigenvalue λ is given as,*

$$\lambda = \frac{max\,(A) + min(B)}{2} = 2.$$

By Algorithm 2,

$$A_\lambda = \begin{bmatrix} 1 & 0 & \epsilon & -1 \\ \epsilon & -1 & 1 & 0 \\ 0 & 1 & -1 & \epsilon \\ -1 & \epsilon & 0 & 1 \end{bmatrix}, \quad B_\lambda = \begin{bmatrix} 0 & 1 & \tau & -1 \\ 1 & \tau & -1 & 0 \\ -1 & 0 & 1 & \tau \\ \tau & -1 & 0 & 1 \end{bmatrix}.$$

Now, take the initial state vector

$$x^*(0) = \begin{pmatrix} u^*(0) \\ w^*(0) \end{pmatrix} \quad with \quad u^*(0) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \quad w^*(0) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

The following sequence is obtained, after iterating (5),

$$\begin{array}{ccccccccccccc} x^*(0) & \to & x^*(1) & \to & x^*(2) & \to & x^*(3) & \to & x^*(4) & \to & x^*(5) & \to & x^*(6) \\ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} & \to & \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 0 \\ -1 \\ -1 \\ 0 \end{pmatrix} & \to & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} & \to & \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 0 \\ -1 \\ 0 \\ -1 \end{pmatrix} & \to & \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} & \to & \begin{pmatrix} 2 \\ 1 \\ 1 \\ 2 \\ -1 \\ -1 \\ 0 \\ 0 \end{pmatrix} & \to & \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array}.$$

Since $x^*(6) = x^*(0)$, *therefore* $s = 0, r = 6$. *The required eigenvector* v *is computed as,*

$$\begin{aligned} v &= x^*(s) \oplus ... \oplus x^*(r-1) \\ &= x^*(0) \oplus ... \oplus x^*(5) \\ &= \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 0 \\ -1 \\ -1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 0 \\ -1 \\ 0 \\ -1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 2 \\ 1 \\ 1 \\ 2 \\ -1 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 2 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}. \end{aligned}$$

Now verify that either v *is the correct eigenvector or not. So*

$$\mathcal{M}(v) = \begin{pmatrix} 4 \\ 4 \\ 3 \\ 4 \\ 3 \\ 2 \\ 3 \\ 3 \end{pmatrix} = \lambda \otimes v.$$

Which shows that the eigenvector v is the correct eigenvector obtained by Algorithm 2.

2. *For Algorithm 3, take the initial vector*

$$x(0) = \begin{pmatrix} u(0) \\ w(0) \end{pmatrix} \quad with \quad u(0) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \quad w(0) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

The following sequence is obtained, after iterating (2)

$$\begin{matrix} x(0) & \rightarrow & x(1) & \rightarrow & x(2) & \rightarrow & x(3) & \rightarrow & x(4) & \rightarrow & x(5) & \rightarrow & x(6) \end{matrix}$$
$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 4 \\ 4 \\ 3 \\ 3 \\ 2 \\ 1 \\ 1 \\ 2 \end{pmatrix} \rightarrow \begin{pmatrix} 5 \\ 4 \\ 4 \\ 5 \\ 4 \\ 4 \\ 5 \\ 5 \end{pmatrix} \rightarrow \begin{pmatrix} 7 \\ 8 \\ 7 \\ 8 \\ 6 \\ 5 \\ 6 \\ 5 \end{pmatrix} \rightarrow \begin{pmatrix} 9 \\ 9 \\ 8 \\ 8 \\ 9 \\ 8 \\ 8 \\ 9 \end{pmatrix} \rightarrow \begin{pmatrix} 12 \\ 11 \\ 11 \\ 12 \\ 9 \\ 9 \\ 10 \\ 10 \end{pmatrix} \rightarrow \begin{pmatrix} 12 \\ 13 \\ 12 \\ 13 \\ 13 \\ 12 \\ 13 \\ 12 \end{pmatrix}.$$

Since $x(6) = 12 \otimes x(0)$. It follows that $s = 0, r = 6$, and $c = 12$. The corresponding eigenvector v is obtained as

$$\begin{aligned} v &= \oplus_{j=1}^{r-s} (\lambda^{\otimes(r-s-j)} \otimes x(s+j-1)) \\ &= \oplus_{j=1}^{6} (\lambda^{\otimes(6-j)} \otimes x(j-1)) \\ &= \lambda^{\otimes(5)} \otimes x(0) \oplus \lambda^{\otimes(4)} \otimes x(1) \oplus \lambda^{\otimes(3)} \otimes x(2) \oplus \lambda^{\otimes(2)} \otimes x(3) \oplus \lambda \otimes x(4) \oplus x(5) \\ &= \begin{pmatrix} 10 \\ 11 \\ 10 \\ 11 \\ 11 \\ 10 \\ 11 \\ 10 \end{pmatrix} \oplus \begin{pmatrix} 12 \\ 12 \\ 11 \\ 11 \\ 10 \\ 9 \\ 9 \\ 10 \end{pmatrix} \oplus \begin{pmatrix} 11 \\ 10 \\ 10 \\ 11 \\ 10 \\ 10 \\ 11 \\ 11 \end{pmatrix} \oplus \begin{pmatrix} 11 \\ 12 \\ 11 \\ 12 \\ 10 \\ 9 \\ 10 \\ 9 \end{pmatrix} \oplus \begin{pmatrix} 11 \\ 11 \\ 10 \\ 10 \\ 11 \\ 10 \\ 10 \\ 11 \end{pmatrix} \oplus \begin{pmatrix} 12 \\ 11 \\ 11 \\ 12 \\ 9 \\ 9 \\ 10 \\ 10 \end{pmatrix} \end{aligned}$$

Finally, we obtain

$$v = \begin{pmatrix} 12 \\ 12 \\ 11 \\ 12 \\ 11 \\ 10 \\ 11 \\ 11 \end{pmatrix}.$$

Which is the correct eigenvector.

Using Algorithm 3 in Example 2, system ends up in a periodic behaviour after 6th iteration and we get $s = 0$, $r = 6$ and $c = 12$. If we consider $x(0) = (0\,0\,0\,0\,0\,0\,0\,0)^T$ in Example 2, then we get $s = 0$,

$r = 2$ and $c = 4$. So in case of Algorithm 3, one can get different value of c for different initial state vectors, while in Algorithm 2, we need not a real number c.

Remark 1. *The main focus in this paper is to develop an efficient algorithm to find out the eigenvectors for systems of type* (2). *Also, we have made computational experiment to compare the Algorithms 2 and 3 for Latin squares. We have implemented both algorithms in MATLAB for* $r \in A_1$, $s \in A_2$ *and* $c \in A_3$ *such that* $|A_i| = n_i, i = 1, 2, 3$. *Here* $|A_i|$ *represents the cardinality of each set* A_i. *In case of Algorithm 3, it stops if there exists a positive integers* $r > s \geq 0$ *and a real number c with* $x(r) = x(s) \otimes c$, *while in that of Algorithm 2, it stops if for some integers* $r > s \geq 0$ *it holds* $x^*(r) = x^*(s)$. *Therefore, the time complexity of Algorithm 3 is* $\mathcal{O}(n_1 n_2 n_3)$, *while the time complexity of Algorithm 2 is* $\mathcal{O}(n_1 n_2)$. *Thus the proposed algorithm takes less time than the compared one.*

The computational efficiency of the proposed method can be seen by comparing the runtimes of Algorithm 2 and 3. Both algorithms are executed under the same Laptop environment with 1.70 *GHz CPU,* 4 *GB RAM, MATLAB (R2015a) simulation software. We computed the time of calculating eigenvalue and eigenvector for Example 2 by both algorithms. The time consumptions of Algorithm 2 and 3 are* 0.088783 *s and* 1.718662 *s respectively, which clearly reveals that the proposed algorithm is faster than the other one. Also in case of Algorithm 2, we obtain the eigenvector v by a simple formula, given as* $v = x^*(s) \oplus ... \oplus x^*(r-1)$ *however, while using that Algorithm 3 an eigenvector is obtained as*

$$v = \oplus_{j=1}^{r-s} (\lambda^{\otimes(r-s-j)} \otimes x(s+j-1)).$$

which is quite difficult compared to Algorithm 2.

4. Conclusions

The eigenproblem of bipartite min-max-plus systems for Latin squares has been discussed in this work. The iterative Algorithm 1 was developed for the computation of the eigenvectors (trivial and nontrivial) for systems of type (2). In particular, this algorithm was extended to compute the eigenvalue and eigenvectors for Latin squares in a system of type (2). At the end, a computational comparison has been given. Experimental result shows that the proposed Algorithm 2 is much more computationally efficient than the compared one. Of course, Algorithm 2 is only for Latin squares in a system of type (2), while Algorithm 3 may be used for all bipartite min-max-plus systems. Similarly, one can derive an algorithm to calculate the eigenvalue and eigenvectors for separated min-max-plus systems.

Author Contributions: The individual contribution and responsibilities of the authors were as follows: U.H., M.U. and F.A. designed and proposed the research to A.A. and P.K. During the implementation of algorithms F.A., A.A. and P.K. provided useful advices to M.U. and U.H. All authors participated in the writing the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We thank the referees for their valuable suggestions and helpful remarks.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Song, S.Z.; Kang, K.T.; Jun, Y.B. Invertible commutativity preservers of matrices over max algebra. *Czechoslovak Math. J.* **2006**, *56*, 1185–1192. [CrossRef]
2. Kang, K.T.; Song, S.Z. Regular matrices and their generalized inverses over the max algebra. *Linear Multilinear Alg.* **2015**, *63*, 1649–1663. [CrossRef]
3. Song, S.Z.; Kang, K.T. Column ranks and their preservers of matrices over max algebra. *Linear Multilinear Alg.* **2003**, *51*, 311–318. [CrossRef]
4. Halburd, R.G.; Southall, N.J. Tropical nevanlinna theory and ultra-discrete equations. *Int. Math. Res. Not.* **2009**, *5*, 887–911.

5. Cuninghame-Green, R.A. Lecture notes in economics and mathematical systems. In *Minimax Algebra*; Springer: New York, NY, USA, 1979.
6. De Schutter, B. On the ultimate behavior of the sequence of consecutive powers of a matrix in the max-plus algebra. *Linear Alg. Appl.* **2000**, *307*, 103–117. [CrossRef]
7. Gaubert, S. Methods and applications of (max,+) linear algebra. In *Annual Symposium on Theoretical Aspects of Computer Science;* Springer: Berlin/Heidelberg, Germany, 1997; pp. 261–282.
8. Santoso, K.A.; Suprajitno, H. On max-plus algebra and its application on image steganography. *Sci. World J.* **2018**, 6718653. [CrossRef]
9. Kubo, S.; Nishinari, K. Applications of max-plus algebra to flow shop scheduling problems. *Discret. Appl. Math.* **2018**, *247*, 278–293. [CrossRef]
10. Braker, J.G.; Olsder, G.J. The power algorithm in max algebra. *Linear Alg. Appl.* **1993**, *182*, 67–89. [CrossRef]
11. Subiono. On Classes of Min-Max-Plus Systems and Their Application. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2000.
12. Subiono; van der Woude, J. Power algorithms for (max,+)- and bipartite (min,max,+)-systems. *Discret. Event Dyn. Syst.* **2000**, *10*, 369–389. [CrossRef]
13. Umer, M.; Hayat, U.; Abbas, F. An efficient algorithm for nontrivial eigenvectors in max-plus algebra. *Symmetry* **2019**, *11*, 738. [CrossRef]
14. Subiono; Mufid, M. S.; Adzkiya, D. Eigenproblems of latin squares in bipartite (min, max, plus)-systems. *Discret. Event Dyn. Syst.* **2016**, *26*, 657–668. [CrossRef]
15. Akian, M.; Gaubert, S.; Nitica, V.; Singer, I. Best approximation in maxplus semimodules. *Linear Alg. Appl.* **2011**, *435*, 3261–3296. [CrossRef]
16. Garca-Planas, M.I.; Magret, M.D. Eigenvectors of permutation matrices. *Adv. Pure Math.* **2015**, *5*, 390–394. [CrossRef]
17. Stańczyk, J. Max-Plus Algebra Toolbox for Matlab. 2016. Available online: http://gen.up.wroc.pl/stanczyk/mpa/ (accessed on 16 February 2020).
18. McKay, B.D.; Wanless, I.M. On the number of Latin squares. *Ann. Comb.* **2005**, *9*, 334–344. [CrossRef]