Application of an optimization problem in Max-Plus algebra to scheduling problems

J.-L. Bouquard, C. Lenté, J.-C. Billaut*

Laboratoire d’Informatique, Université François Rabelais de Tours, Ecole Polytechnique de l’Université de Tours,
64 avenue Jean Portalis, 37200 Tours, France

Received 15 September 2002; received in revised form 21 March 2003; accepted 21 April 2005
Available online 29 March 2006

Abstract

The problem tackled in this paper deals with products of a finite number of triangular matrices in Max-Plus algebra, and more precisely with an optimization problem related to the product order. We propose a polynomial time optimization algorithm for 2 \times 2 matrices products. We show that the problem under consideration generalizes numerous scheduling problems, like single machine problems or two-machine flow shop problems. Then, we show that for 3 \times 3 matrices, the problem is NP-hard and we propose a branch-and-bound algorithm, lower bounds and upper bounds to solve it. We show that an important number of results in the literature can be obtained by solving the presented problem, which is a generalization of single machine problems, two- and three-machine flow shop scheduling problems. The branch-and-bound algorithm is tested in the general case and for a particular case and some computational experiments are presented and discussed.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Scheduling; Optimization; Max-Plus algebra

1. Introduction

The Max-Plus algebra has been presented by Gondran and Minoux for the computation of longest paths of valued graphs [10], among others [7]. This algebra has a lot of applications to system theory and optimal control [6,20], graph theory [10], Petri nets [8], etc. To our knowledge, just a few papers concern the application of another algebra to scheduling theory [9], and the application of this algebra to scheduling problems (see for instance [5,8,12,16]) has not been extensively studied. In this paper, we use the dioid of Max-Plus matrices and we address the problem to minimize a product of triangular matrices.

A very large literature concerns scheduling problems since the preliminary works of Johnson [14]. Several books already present general survey for these problems as [1,2,21] for the more recent ones. We show in this paper that numerous scheduling problems can be covered by a unique problem in Max-Plus algebra and one of the aims of this paper is to show that Max-Plus algebra is more adapted for solving sequencing problems than the classical (\( \mathbb{R}, +, \times \)) algebra.

* Corresponding author.

E-mail addresses: jean-louis.bouquard@univ-tours.fr (J.-L. Bouquard), christophe.lente@univ-tours.fr (C. Lenté), jean.billaut@univ-tours.fr (J.-C. Billaut).

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
We suppose in the following that we have a set of \( n \) jobs \( \{ J_x \}_{1 \leq x \leq n} \) to schedule on \( m \) machines \((m=1,2,3)\) in the following. Each job \( J_x \) is made up of \( m \) operations and the operation number \( k \) is denoted by \( O_{x,k} \). The processing time of operation \( O_{x,k} \) is equal to \( p_{x,k} \). Preemption is not allowed and a machine can only perform one job at a time.

The paper is organized as follows: in Section 2, we present Max-Plus algebra. In Section 3 we present a general optimization problem in Max-Plus algebra, that involves \( 2 \times 2 \) matrices, and its applications to scheduling problems. In Section 4, we present this optimization problem with \( 3 \times 3 \) matrices, we show that the problem is NP-hard and we propose lower bounds, upper bounds and a general branch-and-bound algorithm to solve it. We present some applications to scheduling problems. This branch-and-bound algorithm is tested for the general problem and for the three-machine flow shop problem in Section 5 and some computational results are discussed.

2. Presentation of Max-Plus algebra

In Max-Plus algebra, we denote the maximum by \( \oplus \) and the addition by \( \otimes \). The first operator, \( \oplus \), is idempotent, commutative, associative and has a neutral element \((-\infty)\) denoted by \( 0 \). The second operator, \( \otimes \), is associative, distributive on \( \oplus \) and has a neutral element \((1)\) denoted by \( 0 \). It is important to note that in Max-Plus algebra more generally in dioids, the first operator does not allow simplification: \( a \oplus b = a \otimes c \oplus b = c \). Furthermore, in \( \mathbb{R}_{\text{max}} \), the second operator \( \otimes \) is commutative, and except \( 0 \), every element is invertible: the inverse of \( x \) is denoted by \( x^{-1} \) or \( \forall /x \). For more convenience, we denote the ordinary subtraction by \( x/y \) instead of \( x \otimes y^{-1} \) and by \( xy \) the product \( x \otimes y \).

It is possible to extend these two operators to \( n \times m \) matrices of elements of \( \mathbb{R}_{\text{max}} \). Let \( A \) and \( B \) be two matrices of size \( m \times m \), operators \( \oplus \) and \( \otimes \) are defined by

\[
\forall (i,j) \in \{1, \ldots, m\}^2 \quad [A \oplus B]_{i,j} = A_{i,j} \oplus B_{i,j},
\]

\[
\forall (i,j) \in \{1, \ldots, m\}^2 \quad [A \otimes B]_{i,j} = \bigoplus_{k=1}^{m} A_{i,k} \otimes B_{k,j}.
\]

It is not difficult to show that \( \mathcal{M}_{m \times m}(\mathbb{R}_{\text{max}}) \), the set of \( m \times m \) matrices in \( \mathbb{R}_{\text{max}} \) is a dioid. Considering a triangular matrix \( A \) of size \( m \times m \), such that \( A_{i,j} = 0 \) \( \forall i > j \), the set of \( m \times m \) triangular matrices, denoted by \( \mathcal{T}_{m \times m}(\mathbb{R}_{\text{max}}) \), is a dioid as well, but \( \otimes \) is not commutative and not every element is invertible. More details can be found in [11].

3. Optimization problem in \( \mathcal{T}_{2 \times 2}(\mathbb{R}_{\text{max}}) \)

3.1. Minimization of the product of matrices

Let us consider a set of \( n \) \( 2 \times 2 \) triangular matrices in Max-Plus algebra \( \mathcal{M} = \{ M(1), M(2), \ldots, M(n) \} \), with \( \forall i, 1 \leq i \leq n; M(i) = \begin{pmatrix} \mu_{1}(i) & \mu_{1,2}(i) \\ 0 & \mu_{2}(i) \end{pmatrix} \).

It follows from the definition that the product of two matrices \( M(i) \) and \( M(i') \) is equal to:

\[
M(i) \otimes M(i') = \begin{pmatrix} \mu_{1}(i)\mu_{1}(i') & \mu_{1}(i)\mu_{1,2}(i') + \mu_{1,2}(i)\mu_{2}(i') \\ 0 & \mu_{2}(i)\mu_{2}(i') \end{pmatrix}.
\]

It is clear that \( M(i) \otimes M(i') \neq M(i') \otimes M(i) \): the product of matrices is not commutative. Furthermore, in the matrix equal to the product of these two matrices, only the top-right term depends on the order of the product.

**Proposition 1.** Given two \( 2 \times 2 \) triangular matrices \( A \) and \( B \), defined by

\[
A = \begin{pmatrix} a_{1} & a_{1,2} \\ 0 & a_{2} \end{pmatrix} \quad B = \begin{pmatrix} b_{1} & b_{1,2} \\ 0 & b_{2} \end{pmatrix}.
\]
We have

\[ A \otimes B \preceq B \otimes A \iff \frac{a_1}{a_{1,2}} \oplus \frac{b_2}{b_{1,2}} \leq \frac{b_1}{b_{1,2}} \oplus \frac{a_2}{a_{1,2}}. \]  

(1)

**Proof.**

\[ A \otimes B \preceq B \otimes A \iff a_1b_{1,2} \oplus a_{1,2}b_2 \leq b_1a_{1,2} \oplus b_{1,2}a_2 \]

\[ \iff \frac{a_1b_{1,2} \oplus a_{1,2}b_2}{b_{1,2}a_{1,2}} \leq \frac{b_1a_{1,2} \oplus b_{1,2}a_2}{a_{1,2}b_{1,2}} \]

\[ \iff \frac{a_1}{a_{1,2}} \oplus \frac{b_2}{b_{1,2}} \leq \frac{b_1}{b_{1,2}} \oplus \frac{a_2}{a_{1,2}}. \]  

\[ \square \]

In the classical algebra, the expression (1) of the priority rule is: \( \max(a_1-a_{1,2}, b_2-b_{1,2}) \leq \max(b_1-b_{1,2}, a_2-a_{1,2}) \), which is equivalent to: \( \min(a_1-a_{1,2}, b_1-b_{1,2}) \leq \min(a_1-a_{1,2}, b_1-b_{1,2}) \).

In the following, this relation, which is neither transitive nor anti-symmetric, is denoted by \( A \preceq B \). Similarly,

\[ \frac{a_1}{a_{1,2}} \oplus \frac{b_2}{b_{1,2}} \leq \frac{b_1}{b_{1,2}} \oplus \frac{a_2}{a_{1,2}} \]

is denoted by \( A \triangleleft B \).

The optimization problem that we consider is stated as follows: considering a set of \( n \) matrices, \( \mathcal{M} = \{ M(1), M(2), \ldots, M(n) \} \), find a permutation \( \sigma \) such that the product of these matrices in the order of \( \sigma \) minimizes the top-right term of the resulting matrix. This term is the only term that depends on the order of the product. This problem is denoted by \( \otimes \mathcal{M} \).

It is easily proved by induction that the product of \( n \) matrices \( M(\sigma(1)), M(\sigma(2)), \ldots, M(\sigma(n)) \) in this order, denoted by \( \bigotimes_{i=1}^{n} M(\sigma(i)) \), is equal to

\[ \left( \bigotimes_{i=1}^{n} \mu_1(\sigma(i)) + \bigotimes_{i=1}^{n} \mu_1(\sigma(k)) \mu_{1,2}(\sigma(i)) \bigotimes_{k=i+1}^{n} \mu_2(\sigma(k)) \right). \]  

(2)

**Proposition 2.** Obtaining a minimum value of \( \bigotimes_{i=1}^{n} M(\sigma(i)) \) can be done in \( O(n \log(n)) \) time by sorting the matrices according to the priority rule of Proposition 1.

The proof is done through the following lemmas.

**Lemma 1.** There exists a permutation \( \sigma \) such that \( \forall (i, j) \in \{1, \ldots, n\}^2 \ i < j \Rightarrow M(\sigma(i)) \preceq M(\sigma(j)) \).

Such a permutation can be found using an \( O(n \log(n)) \) time algorithm.

**Proof.** Let us consider the following algorithm, called JG.

Begin

| Let \( U = \{ M(i) : \mu_1(i) < \mu_2(i) \} \). |
| Let \( V = \{ M(i) : \mu_1(i) \geq \mu_2(i) \} \). |
| Let \( U^S \) be the set \( U \) sorted in the \( \mu_2(i)/\mu_{1,2}(i) \) non-increasing order. |
| Let \( V^S \) be the set \( V \) sorted in the \( \mu_1(i)/\mu_{1,2}(i) \) non-decreasing order. |
| The concatenation of \( U^S \) and \( V^S \), \( \sigma = U^S; V^S \), is an optimal permutation. |

End

The determination of \( U \) and \( V \) is done in \( O(n) \) time and sorting them is an \( O(n \log(n)) \) time procedure. Consider \( i \) and \( j \in U^S \) with \( i < j \).
By the definition of $U$, we have
\[
\frac{\mu_1(i)}{\mu_{1,2}(i)} < \frac{\mu_2(i)}{\mu_{1,2}(i)}.
\]
Because $U^S$ is sorted, we have
\[
\frac{\mu_2(j)}{\mu_{1,2}(j)} < \frac{\mu_2(i)}{\mu_{1,2}(i)}.
\]
Adding two inequalities is valid in the Max-Plus algebra, so we obtain
\[
\frac{\mu_1(i)}{\mu_{1,2}(i)} + \frac{\mu_2(j)}{\mu_{1,2}(j)} < \frac{\mu_1(j)}{\mu_{1,2}(j)} + \frac{\mu_2(i)}{\mu_{1,2}(i)}.
\]
That is,
\[
\forall (i, j) \in U^S \times U^S \quad (i < j \Rightarrow M(i) \preceq M(j)). \tag{3}
\]
Similarly, we prove that:
\[
\forall (i, j) \in V^S \times V^S \quad (i < j \Rightarrow M(i) \preceq M(j)). \tag{4}
\]
Besides, when $i \in U$ and $j \in V$:
we have
\[
\frac{\mu_1(i)}{\mu_{1,2}(i)} < \frac{\mu_2(i)}{\mu_{1,2}(i)} \quad \text{and} \quad \frac{\mu_2(j)}{\mu_{1,2}(j)} < \frac{\mu_1(j)}{\mu_{1,2}(j)}.
\]
By adding these inequalities we obtain:
\[
\frac{\mu_1(i)}{\mu_{1,2}(i)} + \frac{\mu_2(j)}{\mu_{1,2}(j)} < \frac{\mu_1(j)}{\mu_{1,2}(j)} + \frac{\mu_2(i)}{\mu_{1,2}(i)},
\]
that is,
\[
\forall (i, j) \in U^S \times V^S \quad M(i) \preceq M(j). \tag{5}
\]

The properties (3)–(5) prove this lemma. □

**Lemma 2.** A permutation $\sigma$ that satisfies Lemma 1 is optimal i.e. minimizes the product
\[
M(\sigma) = \bigotimes_{k=1}^{n} M(\sigma(k)).
\]

**Proof.** Without loss of generality, let us assume that the matrices are numbered so that $\sigma = (1, 2, \ldots, n)$.
Let $\tau$ be an optimal permutation. If $\tau \neq \sigma$, then there exist in $\tau$ two consecutive elements $i$ and $j$ with $i < j$ and $\tau = \tau_1 ji \tau_2$.
\[
M(\tau) = M(\tau_1) \otimes M(j) \otimes M(i) \otimes M(\tau_2) \geq M(\tau_1) \otimes M(i) \otimes M(j) \otimes M(\tau_2).
\]
After the permutation of $i$ and $j$ and by iterating the process, we obtain finally the permutation $\sigma$ and show that $M(\tau) \geq M(\sigma)$, proving that $\sigma$ is optimal. □

Algorithm JG is similar but more general than Johnson’s algorithm [14] for the two-machine flow shop problem. This is perhaps one of the reasons why this algorithm has been so extensively used in the flow shop scheduling literature [23].
3.2. Application to scheduling problems

- Consider the $F2||C_{\text{max}}$ flow shop problem. In this problem, each job is composed by two operations, the first operation is performed on machine $M_1$ and the second on machine $M_2$. We assume that the sequences of jobs on $M_1$ and $M_2$ are the same (permutation flow shop). This problem is polynomially solvable using Johnson’s rule [14]. We denote by $C_{[i],k}$ the completion time of the job in position $i$ on $M_k$ and by $p_{i,k}$ its processing time, which is a positive real number. As shown in Fig. 1, we have: $C_{[i],1} = C_{[i-1],1} \otimes p_{[i],1}$ and $C_{[i],2} = (C_{[i-1],2} \oplus C_{[i-1],1} \otimes p_{[i],1}) \otimes p_{[i],2}$.

We associate with each job $J_x$ the matrix $M(J_x)$, called a job-matrix, defined by

$$M(J_x) = \left( \begin{array}{cc} p_{x,1} & p_{x,1}p_{x,2} \\ 0 & p_{x,2} \end{array} \right).$$

Then, we can rewrite the expression for the completion times of job in position $i$ on the two machines as follows:

$$(C_{[i],1} C_{[i],2}) = (C_{[i-1],1} C_{[i-1],2}) \otimes \left( \begin{array}{cc} p_{[i],1} & p_{[i],1}p_{[i],2} \\ 0 & p_{[i],2} \end{array} \right)$$
or

$$C_{[i]} = C_{[i-1]} \otimes M(J_{[i]}).$$

So, we have: $C_{[n]} = C_{[0]} \otimes M(J_{[1]}) \otimes M(J_{[2]}) \otimes \cdots \otimes M(J_{[n]})$ in which $C_{[0]}$ is the vector of availability dates of machines $M_1$ and $M_2$, that is ($\mathbb{I}$).

Let us consider a sequence of jobs $\sigma$. We define by $M(\sigma)$ the matrix:

$$M(\sigma) = \bigotimes_{i=1}^{n} M(\sigma(i)).$$

$M(\sigma)$ is called a permutation-matrix. If we consider permutation $\sigma$, the top-right term of the permutation-matrix $M(\sigma)$ represents the completion time of the last job on machine $M_2$, i.e. the makespan of the sequence. Thus, we can obtain the makespan expression directly from the sequence and the job-matrices: $C(\sigma) = (\mathbb{I} \otimes M(\sigma))$.

**Proposition 3.** Every instance of the $F2||C_{\text{max}}$ problem is equivalent to an instance of the $\bigotimes M2||M1,2$ problem by setting, for each job $J_x$:

$$M(J_x) = \left( \begin{array}{cc} p_{x,1} & p_{x,1}p_{x,2} \\ 0 & p_{x,2} \end{array} \right).$$

According to proposition 1, it is better to sequence $J_x$ before $J_y$ if $M(J_x) \otimes M(J_y) \leq M(J_y) \otimes M(J_x)$, i.e. if:

$$\frac{p_{x,1}}{p_{x,1}p_{x,2}} \oplus \frac{p_{y,2}}{p_{y,1}p_{y,2}} \leq \frac{p_{y,1}}{p_{y,1}p_{y,2}} \oplus \frac{p_{x,2}}{p_{x,1}p_{x,2}},$$
i.e. after simplification:

$$\frac{1}{p_{y,1}} \oplus \frac{1}{p_{x,2}} \leq \frac{1}{p_{x,1}} \oplus \frac{1}{p_{y,2}}$$
Proposition 4. Every instance of the problem by setting which is equivalent in the classical algebra to: \( \min(p_{x,1}, p_{y,2}) \leq \min(p_{x,2}, p_{y,1}) \). This expression is the well-known Johnson’s rule [14] and, in this case, algorithm JG is exactly Johnson’s algorithm.

This result generalizes the following ones:

- Consider the two-machine flow shop problem with some additional constraints. Each operation \( O_{x,k} \) needs a setup time and a removal time, denoted by \( s_{x,k} \) and \( r_{x,k} \), respectively, and supposed to be sequence-independent and anticipatory, i.e. the setup can be scheduled in anticipation of arriving work. Furthermore, there exists a minimum time lag between the completion time of operation \( O_{x,1} \) and the starting time of its successor \( O_{x,2} \), denoted by \( a_x \) and assumed to be negative or positive integer. We assume that the operations sequences on both machines are the same (permutation flow shop). In the two-machine flow shop context, the problem under consideration can be denoted by \( F2|s_{\text{nsd}}, r_{\text{nsd}}, a, \text{prmu}|C_{\text{max}} \) (see Fig. 2).

\[
M(J_x) = \left( \begin{array}{c}
\frac{r_{x,1}}{a_x p_{x,2} r_{x,2}} + \frac{s_{y,2}}{s_{x,1} p_{x,1} a_x}
\end{array} \right)
\]

We immediately deduce from Proposition 1 a rule to sequence optimally the jobs: \( J_x \) precedes \( J_y \) if

\[
\frac{r_{x,1}}{a_x p_{x,2} r_{x,2}} + \frac{s_{y,2}}{s_{x,1} p_{x,1} a_x} \leq \frac{r_{y,1}}{a_y p_{y,2} r_{y,2}} + \frac{s_{x,2}}{s_{x,1} p_{x,1} a_x}.
\]

In the classical algebra, this relation is

\[
\min(p_{x,1} + a_x + s_{x,1} - s_{x,2}, p_{y,2} + a_y + r_{y,2} - r_{y,1}) \leq \min(p_{x,2} + a_x + r_{x,2} - r_{x,1}, p_{y,1} + a_y + s_{y,1} - s_{y,2}).
\]

This result generalizes the following ones:

- Ref. [18] for a two-machine permutation flow shop problem involving time lags (problem \( F2|a, \text{prmu}|C_{\text{max}} \)),
- Ref. [25] for a two-machine permutation flow shop problem with anticipatory setups (\( F2|s_{\text{nsd}}, \text{prmu}|C_{\text{max}} \)), and [24] for the \( F2|s_{\text{nsd}}, r_{\text{nsd}}, a, \text{prmu}|C_{\text{max}} \) problem,
- Ref. [19] for a two-machine flow shop problem with multiple constraints (problem \( F2|s_{\text{nsd}}, r_{\text{nsd}}, a \geq 0, \text{prmu}|C_{\text{max}} \)).
- Some batch scheduling problems consider that jobs of the same family are partially sequenced, and that setup times are required when a machine switches from processing jobs in one family to jobs in another family [22]. If we consider job availability, i.e. a job becomes available immediately after its processing is completed, these problems can also be solved by JG algorithm, in their simple expression and with setup times, removal times and time lags constraints [4].
- Some single machine scheduling problems can also be solved by application of Proposition 2. For instance, consider the single machine problem in which a release date \( r_x \) is associated with each job \( J_x \). We have: \( C[i] = (r[i] \oslash C[i-1]) \oslash p[i] \). This expression is equivalent to:

\[
(\oslash 1 C[i] = (\oslash 1 C[i-1]) \oslash \left( \begin{array}{c}
r[i] \oslash p[i]
\end{array} \right)
\]

Fig. 2. Makespan expression in the classical algebra for the \( F2|s_{\text{nsd}}, r_{\text{nsd}}, a, \text{prmu}|C_{\text{max}} \) problem.
Thus, we associate with each job $J_x$ a matrix $M(J_x)$ defined by

$$M(J_x) = \begin{pmatrix} 1 & r_x \otimes p_x \\ 0 & p_x \end{pmatrix}$$

and by application of Proposition 1, we deduce that it is better to sequence $J_x$ before $J_y$ if

$$M(J_x) \preceq M(J_y) \iff \frac{1}{r_x p_x} \otimes \frac{p_y}{r_y p_y} \leq \frac{1}{r_y p_y} \otimes \frac{p_x}{r_x p_x} \iff r_y p_y + r_x p_y \leq r_x p_x + r_y p_y.$$

Because $r_y p_y < r_x p_x$ and $r_x p_x < r_x p_x p_y$, we simplify the expression and we deduce:

$$M(J_x) \preceq M(J_y) \iff r_x p_y \leq r_x p_x p_y$$

$$\iff r_x \leq r_y.$$

This leads to the well-known sequencing rule for the $1|r_x|C_{\text{max}}$ problem.

- In the same way, the single machine problem with latency durations, denoted by $1|q_i|C_{\text{max}}$ with $C_{\text{max}} = \max_{1 \leq i \leq n}(C_i + q_i)$, or with due dates and lateness criteria $1||L_{\text{max}}$, can be easily solved using the result of Proposition 1 by defining:

$$M(J_x) = \begin{pmatrix} p_x & p_x \otimes q_x \\ 0 & 1 \end{pmatrix} \quad \text{or} \quad M(J_x) = \begin{pmatrix} p_x & p_x/d_x \\ 0 & 1 \end{pmatrix}.$$

We obtain $M(J_x) \preceq M(J_y)$ if $q_x \geq q_y$, (or $d_x \leq d_y$) another well-known sequencing rule for solving these problems optimally.

- The no-wait constraint can be considered in two-machine flow shop problem, but does not lead to triangular matrices. So, the JG algorithm cannot be applied.

- Notice that a product of matrices in $T_{2 \times 2}(R_{\text{max}})$ cannot model the $F2|a|C_{\text{max}}$ problem if the permutation constraint is not imposed. In such a general problem, a permutation schedule is not dominant and one sequence is involved on each machine.

Conversely, we can translate an instance of the $\otimes M2||M_{1,2}$ problem to some instance of a two-machine flow shop with makespan minimization. We define by $J(M_x)$ the data of the job associated with the matrix $M_x$.

**Proposition 5.** Every instance of the $\otimes M2||M_{1,2}$ problem is equivalent to

- an instance of the $F2|a, \text{prmu}|C_{\text{max}}$ problem by setting, for each matrix $M_x$:

$$J(M_x) = \begin{cases} p_{x,1} = \mu_{x,1}, \\
p_{x,2} = \mu_{x,2}, \\
a_x = \frac{\mu_{x,1} \mu_{x,2}}{\mu_{x,1} \mu_{x,2}}, \end{cases}$$

- an instance of the $F2|r_{\text{nsd}}, \text{prmu}|C_{\text{max}}$ problem by setting, for each matrix $M_x$:

$$J(M_x) = \begin{cases} p_{x,1} = \mu_{x,1}, \\
p_{x,2} = \mu_{x,2}, \\
r_{x,1} = \mu_{x,1}, \\
r_{x,2} = 1, \end{cases}$$

- an instance of the $F2|s_{\text{nsd}}, \text{prmu}|C_{\text{max}}$ problem by setting, for each matrix $M_x$:

$$J(M_x) = \begin{cases} p_{x,1} = \mu_{x,1}, \\
p_{x,2} = \mu_{x,1}, \\
s_{x,1} = 1, \\
s_{x,2} = \frac{\mu_{x,1} \mu_{x,2}}{\mu_{x,1} \mu_{x,2}}. \end{cases}$$

**Proof.** Trivial. \(\square\)
Usually, the data of scheduling problems are not negative, what is not necessarily the case if we follow the previous proposition. Thus, we can prefer the following proposition:

**Proposition 6.** Every instance of the $\otimes M_2| | M_{1,2}$ problem with non-negative data is equivalent to an instance of the $F2| s_{\text{nd}}, r_{\text{nd}}, a, \text{prmu}| C_{\text{max}}$ problem with non-negative data by setting:

\[
J(M_k) = \begin{cases}
p_{x,1} \min = \mathbb{1} \oplus \mu_{x,1,2} / \mu_{x,2}, & p_{x,1} \max = \mu_{x,1} \mu_{x,1,2} / \mu_{x,1} \\
p_{x,1} \in [p_{x,1} \min, p_{x,1} \max], & p_{x,2} = \mu_{x,1,2} / p_{x,1}, \\
r_{x,1} = \mu_{x,1} / p_{x,1}, & r_{x,2} = \mathbb{1}, \\
x_{s,1} = 1, & s_{x,2} = \mu_{x,2} p_{x,1} / \mu_{x,1,2}, \\
x_{a} = 1, &
\end{cases}
\]

if $\mu_{x,1} \mu_{x,2} > \mu_{x,1,2}$, and

\[
J(M_k) = \begin{cases}
p_{x,1} = \mu_{x,1}, & p_{x,2} = \mu_{x,2}, \\
x_{a} = \mu_{x,1,2}, & \\
r_{x,1} = 1, & r_{x,2} = \mathbb{1}, \\
x_{s,1} = 1, & s_{x,2} = \mathbb{1}
\end{cases}
\]

otherwise.

**Proof.** In both cases, it is easy to check that $s_{x,1} p_{x,1} r_{x,1} = \mu_{x,1}$, $s_{x,1} p_{x,1} a_{x} p_{x,2} r_{x,2} = \mu_{x,1,2}$ and $s_{x,2} p_{x,2} r_{x,2} = \mu_{x,2}$. Then, we have to make sure that the data are non-negative ($\geq \mathbb{1}$).

- If $\mu_{x,1} \mu_{x,2} > \mu_{x,1,2}$: $p_{x,1} \min = \mathbb{1} \oplus \mu_{x,1,2} / \mu_{x,2} \geq \mathbb{1}$.

  Besides: $\mu_{x,1} p_{x,1} \min = \mu_{x,1} \oplus \mu_{x,1} \mu_{x,1,2} / \mu_{x,2} \leq \mu_{x,1} \mu_{x,1,2}$ and: $\mu_{x,1,2} p_{x,1} \min \leq \mu_{x,1,2} (1 \oplus \mu_{x,1}) = \mu_{x,1,2} \mu_{x,1}$. By adding these two inequalities: $(\mu_{x,1} \oplus \mu_{x,1,2}) p_{x,1} \min \leq \mu_{x,1,2} \mu_{x,1}$ that is: $p_{x,1} \min \leq \mu_{x,1,2} \mu_{x,1} / (\mu_{x,1} \oplus \mu_{x,1,2}) = p_{x,1} \max$.

  So, we have: $\mathbb{1} \leq p_{x,1} \min \leq p_{x,1} \max$.

  For $p_{x,2}$: $p_{x,2} = \mu_{x,1,2} / p_{x,1} \max \geq \mu_{x,1,2} / p_{x,1} \min = \mathbb{1} \oplus \mu_{x,1,2} / \mu_{x,1} \geq \mathbb{1}$.

  For $r_{x,1}$: $r_{x,1} = \mu_{x,1} / p_{x,1} \max \geq \mu_{x,1} / p_{x,1} \min = \mathbb{1} \oplus \mu_{x,1} / \mu_{x,1,2} \geq \mathbb{1}$.

  For $s_{x,2}$: $s_{x,2} = \mu_{x,2} p_{x,1} / \mu_{x,1,2} \geq \mu_{x,2} p_{x,1} \min / \mu_{x,1,2} = \mathbb{1} \oplus \mu_{x,2} / \mu_{x,1,2} \geq \mathbb{1}$.

  Hence, in this case, all the data are non-negative.

- If $\mu_{x,1} \mu_{x,2} \leq \mu_{x,1,2}$, we have $a_{x} = \mu_{x,1,2} / \mu_{x,1} \mu_{x,2} \geq \mathbb{1}$. □

4. Optimization problem in $\mathcal{F}_{3 \times 3}(\mathbb{R}_{\text{max}})$

4.1. Generalization of the $2 \times 2$ case

Let $\mathcal{M} = \{M(1), M(2), \ldots, M(n)\}$ be a family of $3 \times 3$ triangular matrices defined by, $\forall k, 1 \leq k \leq n$:

\[
M(k) = \begin{pmatrix}
\mu_{1}(k) & \mu_{1,2}(k) & \mu_{1,3}(k) \\
0 & \mu_{2}(k) & \mu_{2,3}(k) \\
0 & 0 & \mu_{3}(k)
\end{pmatrix}
\]

or

\[
M(k)_{i,j} = \begin{cases}
\emptyset & \text{if } i > j \text{ (under the diagonal)} \\
\mu_{i}(k) & \text{if } i = j \text{ (on the diagonal)} \\
\mu_{i,j}(k) & \text{if } i < j \text{ (above the diagonal)}
\end{cases}
\]
We define for each permutation $\sigma$ of $\{1, \ldots, n\}$ the matrix
\[ M(\sigma) = \bigotimes_{k=1}^{n} M(\sigma(k)). \]

The problem is to determine the permutation $\sigma$ which minimizes the top-right term, i.e. the term $\mu_{1,3}(\sigma)$ [17]. This problem is denoted by $MULT\ M2 || M_{1,3}$.

**Proposition 7.** $MULT\ M3 || M_{1,3}$ is NP-complete.

**Proof.** The result is proved by reduction of the $F3|prmu|C_{\text{max}}$ problem. First, let us define the two decision problems:

- **F3**
  
  **Data:** $n, \{p_{x,1}, p_{x,2}, p_{x,3}\}_{1\leq x\leq n}$ and $B$, positive integers.
  
  **Question:** Does there exist a permutation schedule $S$ in the three-machine flow shop problem with $C_{\text{max}}(S) = \max_{1\leq x\leq n} C_{x,3} \leq B$?

- **MULT\ M3**
  
  **Data:** $q$ matrices $M(k)$ with positive integer coefficients $\mu_{1}(k)$, $\mu_{1,2}(k)$, $\mu_{2}(k)$, $\mu_{1,3}(k)$, $\mu_{2,3}(k)$ and $\mu_{3}(k)$, $\forall k$, $1 \leq k \leq q$, $K$ a positive integer.
  
  **Question:** Does there exist a permutation $\sigma$ of $\{1, \ldots, q\}$ such that $[\bigotimes_{k=1}^{q} M(\sigma(k))]_{1,3} \leq K$?

For problem $MULT\ M3$, given a permutation of matrices, the product can be done in polynomial time, so the answer can be verified in polynomial time. Thus, $MULT\ M3$ is in $\mathcal{NP}$.

We set $q = n$, $\mu_{k}(i) = p_{i,k}$, $\forall i, 1 \leq i \leq n$, $\forall k$, $1 \leq k \leq 3$ and $\mu_{k,\ell}(i) = \bigotimes_{j=k}^{\ell} p_{i,j}$, $\forall k$ and $\ell$, $1 \leq k < \ell \leq 3$ and finally we have $K = B$. Then, $[\bigotimes_{i=1}^{n} M(\sigma(i))]_{1,3}$ corresponds to the longest path in a graph that represents exactly the permutation $\sigma$. So, its value is equal to the makespan $C_{\text{max}}(\sigma)$ in the flow shop problem. Thus, the answer to problem F3 is yes if and only if the answer to problem $MULT\ M3$ is yes. □

### 4.2. Branch-and-bound algorithm

In order to solve the $MULT\ M3 || M_{1,3}$ problem, we propose a branch-and-bound algorithm.

A node $s = (\sigma(s), \Omega_{s}, LB(s))$ is defined by

- a partial sequence of matrices, i.e. a permutation of a subset of $\{M_{1}, M_{2}, \ldots, M_{n}\}$, denoted by $\sigma(s)$,
- a set of unscheduled matrices, denoted by $\Omega_{s}$, i.e. a set of matrices that are not involved in $\sigma(s)$,
- a lower bound $LB(s)$.

At each node $s$, a child node $s_{k}$ is built with $\Omega_{s_{k}} = \Omega_{s} \setminus \{M(k)\}$ and $\sigma(s_{k}) = \sigma(s) \setminus \{M_{k}\}$ for each unscheduled matrix $M(k)$, i.e. the concatenation of $\sigma(s)$ and $M_{k}$. We denote by $BB$, the branch-and-bound algorithm implemented with the best-first strategy: the job with the smallest lower bound is explored first, and ties are broken with the job with the smallest upper bound. The lower bounds that are computed at each node and the upper bounds that are computed at the root node are detailed hereafter.

### 4.3. Lower bounds

The idea of the lower bounds is to focus on two particular lines in the matrix and to forget the third one. We obtain then a relaxed problem which is a $MULT\ M2 || M_{1,2}$ problem, solvable in $O(n \log(n))$ time (see Proposition 2).

Let $u$ and $v$ be two indices, with $1 \leq u < v \leq 3$ and, for a matrix $M$, let $M^{u,v}$ be the matrix obtained by setting every element of $M$ to 0, except the elements $\mu_{u}$, $\mu_{u,v}$ and $\mu_{v}$.

We have: $M \geq M^{u,v}$.

In the following, we denote by $x \wedge y$ the minimum of $x$ and $y$: $x \wedge y = (x^{-1} \oplus y^{-1})^{-1} = (xy/x \oplus y)$ and $J^{u,v}$ denotes the optimal permutation of the $MULT\ M2 || M_{1,2}$ problem associated with matrices $M^{u,v}$. It can be solved using the Proposition 2.
Lemma 3. Let \( L(M, u, v) \in \mathcal{F}_{3 \times 3}(\mathbb{R}_{\max}) \) with \( 1 \leq u < v \leq 3 \) be the matrices defined by

\[
L(M, 1, 2) = \begin{pmatrix}
1 & \frac{\mu_{1,2}}{\mu_1} \\
0 & 1 \\
0 & 0
\end{pmatrix}, \quad L(M, 1, 3) = \begin{pmatrix}
1 & \frac{\mu_{1,3}}{\mu_1} \\
0 & 0 \\
0 & 0
\end{pmatrix},
\]

\[
L(M, 2, 3) = \begin{pmatrix}
\mu_{1,2} & \mu_{1,3} & \mu_{1,3} \\
\mu_2 & \mu_{2,3} & \mu_{2,3} \\
0 & 1 & 0
\end{pmatrix}.
\]

(6)

The symbol \( \times \) indicates that the value can be replaced by an arbitrary number, since these terms will be multiplied by \( 0 \). We have

\[
\forall (u, v), \quad 1 \leq u < v \leq 3 \quad M \geq L(M, u, v) \otimes M^{u,v}.
\]

Proof.

\[
L(M, 1, 2) \otimes M^{1,2} = M^{1,2} \leq M,
\]

\[
L(M, 1, 3) \otimes M^{1,3} = \begin{pmatrix}
0 & 0 & \mu_{1,3} \\
0 & 0 & \mu_{2,3} \\
0 & 0 & \mu_3
\end{pmatrix} \leq M,
\]

\[
L(M, 2, 3) \otimes M^{2,3} = \begin{pmatrix}
0 & \mu_{1,2} \mu_{1,3} & \mu_{1,3} \\
0 & \mu_{2,3} \mu_{2,3} & \mu_{2,3} \\
0 & 0 & 0
\end{pmatrix} \leq M. \quad \Box
\]

Similarly, we have the following lemma.

Lemma 4. Let \( R(M, u, v) \in \mathcal{F}_{3 \times 3}(\mathbb{R}_{\max}) \) with \( 1 \leq u < v \leq 3 \) be the matrix defined by

\[
R(M, 1, 2) = \begin{pmatrix}
1 & \frac{\mu_{1,2}}{\mu_1} & \frac{\mu_{1,3}}{\mu_1} \\
0 & 1 & \mu_{1,2} \mu_{1,3} \\
0 & 0 & \mu_{2,3}
\end{pmatrix},
\]

\[
R(M, 1, 3) = \begin{pmatrix}
1 & \frac{\mu_{1,3}}{\mu_1} \\
0 & 0 \\
0 & 1
\end{pmatrix}, \quad R(M, 2, 3) = \begin{pmatrix}
0 & 0 & \mu_{2,3} \\
0 & 0 & 0
\end{pmatrix}.
\]

(7)

These matrices are such that

\[
\forall (u, v), \quad 1 \leq u < v \leq 3 \quad M \geq M^{u,v} \otimes R(M, u, v).
\]

Proposition 8. For every \( (u, v) \) with \( 1 \leq u < v \leq 3 \), if we denote by \( L_{\min}(M, u, v) \) and \( R_{\min}(M, u, v) \) the matrices defined, respectively, by

\[
[L_{\min}[M, u, v]]_{i,j} = \min_{k=1}^{n}[L(M(k), u, v)]_{i,j}, \quad \forall i, j
\]

[original document page]
and
\[ R_{\min}(M, u, v)|_{i, j} = \min_{k=1}^{n}[R(M(k), u, v)]_{i, j}, \quad \forall i, j \]
then a lower bound of the product \( \bigotimes_{i=1}^{n} M(\sigma(i)) \) is
\[ LB(u, v) = L_{\min}(M, u, v) \otimes M^{u,v}(J^{u,v}) \otimes R_{\min}(M, u, v). \]

Proof.
\[
\bigotimes_{i=1}^{n} M(\sigma(i)) = M(\sigma(1)) \otimes \left( \bigotimes_{i=2}^{n-1} M(\sigma(i)) \right) \otimes M(\sigma(n)) \\
\Rightarrow \bigotimes_{i=1}^{n} M(\sigma(i)) \geq (L(M(\sigma(1)), u, v) \otimes M^{u,v}(\sigma(1))) \otimes \left( \bigotimes_{i=2}^{n-1} M^{u,v}(\sigma(i)) \right) \\
\otimes (M^{u,v}(\sigma(n)) \otimes R(M(\sigma(n), u, v))) \\
\Rightarrow \bigotimes_{i=1}^{n} M(\sigma(i)) \geq L_{\min}(M, u, v) \otimes \left( \bigotimes_{i=1}^{n} M^{u,v}(\sigma(i)) \right) \otimes R_{\min}(M, u, v) \\
\Rightarrow \bigotimes_{i=1}^{n} M(\sigma(i)) \geq L_{\min}(M, u, v) \otimes M^{u,v}(J^{u,v}) \otimes R_{\min}(M, u, v). \quad \square
\]

More explicitly, if \( \text{Cmax}^{2\times2}(J^{u,v}) \) denotes the top-right value of the optimal permutation of the \( 2 \times 2 \) problem based on matrices \( M^{u,v} \), we obtain the three following lower bounds for the top-right value, denoted by \( LB^{u,v} = [LB(u, v)]_{1,3} \) and equal to:

\[
LB^{12} = \min_{i=1}^{n} \left( \frac{\mu_{1,3}(i)}{\mu_{1,2}(i)} \right) \times C_{\text{max}}^{2\times2}(J^{12}) \otimes \min_{i=1}^{n} \left( \frac{\mu_{1,3}(i)}{\mu_{1,1}(i)} \right) \otimes \bigotimes_{i=1}^{n} \mu_{1}(i),
\]

\[
LB^{13} = \text{Cmax}^{2\times2}(J^{13}) \oplus \min_{i=1}^{n} \left( \frac{\mu_{1,3}(i)}{\mu_{1,1}(i)} \right) \otimes \bigotimes_{i=1}^{n} \mu_{1}(i) \oplus \min_{i=1}^{n} \left( \frac{\mu_{1,3}(i)}{\mu_{1,3}(i)} \right) \otimes \bigotimes_{i=1}^{n} \mu_{3}(i),
\]

\[
LB^{23} = \min_{i=1}^{n} \left( \frac{\mu_{1,2}(i)}{\mu_{2,3}(i)} \right) \otimes \text{Cmax}^{2\times2}(J^{23}) \oplus \min_{i=1}^{n} \left( \frac{\mu_{1,3}(i)}{\mu_{3,3}(i)} \right) \otimes \bigotimes_{i=1}^{n} \mu_{3}(i).
\]

In \( LB^{13} \), \( \text{Cmax}^{2\times2}(J^{u,v}) \) dominates the other terms, so finally:
\[ LB^{13} = \text{Cmax}^{2\times2}(J^{13}). \]

4.4. Upper bounds
The permutations \( J^{u,v} \) involved in the lower bounds computation allow to obtain upper bounds, denoted by \( UB^{u,v} \).
\[ \forall (u, v), \quad 1 \leq u < v \leq 3 \quad UB^{u,v} = \text{Cmax}^{3\times3}(J^{u,v}), \]
where \( \text{Cmax}^{3\times3}(\sigma) \) is the top-right value of the product \( \bigotimes_{i=1}^{n} M(\sigma(i)) \).
These permutations only consider \( 2 \times 2 \) matrices of type \( M^{u,v} \), and apply algorithm JG to obtain an optimal permutation for the \( \bigotimes 2 \mid M^{u,v} \) problem. These permutations are defined as follows:

- \( J^{12} \): \( A^{1,2} \preceq B^{1,2} \) if \( A^{1,2} \otimes B^{1,2} \preceq B^{1,2} \otimes A^{1,2} \) \( \iff \frac{a_{1,1}}{a_{1,2}} \oplus \frac{b_{1,1}}{b_{1,2}} \preceq \frac{b_{1,1}}{b_{1,2}} \oplus \frac{a_{1,1}}{a_{1,2}} \).
- \( J^{13} \): \( A^{1,3} \preceq B^{1,3} \) if \( A^{1,3} \otimes B^{1,3} \preceq B^{1,3} \otimes A^{1,3} \) \( \iff \frac{a_{1,1}}{a_{1,3}} \oplus \frac{b_{1,1}}{b_{1,3}} \preceq \frac{b_{1,1}}{b_{1,3}} \oplus \frac{a_{1,1}}{a_{1,3}} \).
- \( J^{23} \): \( A^{2,3} \preceq B^{2,3} \) if \( A^{2,3} \otimes B^{2,3} \preceq B^{2,3} \otimes A^{2,3} \) \( \iff \frac{a_{2,2}}{a_{2,3}} \oplus \frac{b_{2,2}}{b_{2,3}} \preceq \frac{b_{2,2}}{b_{2,3}} \oplus \frac{a_{2,2}}{a_{2,3}} \).
Furthermore, it is possible to combine these conditions, by using operators $\otimes$, $\oplus$ and $\wedge$, in order to obtain new sequencing rules, that consider all the elements of the matrices.

- We define the relation $A \leq B$ by
  \[
  A \leq B \quad \iff \quad \left( \frac{a_1}{a_{1,2}} + \frac{b_2}{b_{1,2}} \right) \otimes \left( \frac{a_1}{a_{1,3}} + \frac{b_3}{b_{1,3}} \right) \otimes \left( \frac{a_2}{a_{2,3}} + \frac{b_3}{b_{2,3}} \right) \leq \left( \frac{b_1}{b_{1,2}} + \frac{a_2}{a_{1,2}} \right) \odot \left( \frac{a_1}{a_{1,3}} + \frac{b_3}{b_{1,3}} \right) \odot \left( \frac{b_2}{b_{2,3}} + \frac{a_3}{a_{2,3}} \right).
  \]

- We define the relation $A \leq B$ by
  \[
  A \leq B \quad \iff \quad \left( \frac{a_1}{a_{1,2}} + \frac{b_2}{b_{1,2}} \right) \odot \left( \frac{a_1}{a_{1,3}} + \frac{b_3}{b_{1,3}} \right) \odot \left( \frac{a_2}{a_{2,3}} + \frac{b_3}{b_{2,3}} \right) \leq \left( \frac{b_1}{b_{1,2}} + \frac{a_2}{a_{1,2}} \right) \wedge \left( \frac{a_1}{a_{1,3}} + \frac{b_3}{b_{1,3}} \right) \wedge \left( \frac{b_2}{b_{2,3}} + \frac{a_3}{a_{2,3}} \right).
  \]

- We define the relation $A \leq B$ by
  \[
  A \leq B \quad \iff \quad \left( \frac{a_1}{a_{1,2}} + \frac{b_2}{b_{1,2}} \right) \odot \left( \frac{a_1}{a_{1,3}} + \frac{b_3}{b_{1,3}} \right) \odot \left( \frac{a_2}{a_{2,3}} + \frac{b_3}{b_{2,3}} \right) \leq \left( \frac{b_1}{b_{1,2}} + \frac{a_2}{a_{1,2}} \right) \oplus \left( \frac{a_1}{a_{1,3}} + \frac{b_3}{b_{1,3}} \right) \oplus \left( \frac{b_2}{b_{2,3}} + \frac{a_3}{a_{2,3}} \right).
  \]

- We define the relation $A \leq B$ by
  \[
  A \leq B \quad \iff \quad a_1 b_{1,3} \oplus a_{1,2} b_{2,3} \oplus a_{1,3} b_3 \leq b_1 a_{1,3} \oplus b_{1,2} a_{2,3} \oplus b_{1,3} a_3.
  \]

With this order between two matrices is given by the minimum top-right element of their products $A \otimes B$ and $B \otimes A$.

These four relations are not transitive. However, $\forall B \in \{\odot, \oplus, \wedge, \odot\}$, these relations are in the form: $A \leq B \iff \psi(A, B) \leq \psi(B, A)$, for some real function $\psi$. Therefore, they are complete, i.e. $\forall B \in \{\odot, \oplus, \wedge, \odot\}, \forall A, \forall B$, either $A \leq B$ or $B \leq A$.

**Lemma 5.** Let us consider a family of matrices $\mathcal{M} = \{M(1), M(2), \ldots, M(n)\}$, $\forall B \in \{\odot, \oplus, \wedge, \odot\}$, there always exists a permutation $\sigma$ of matrices such that $\forall i, 1 \leq i \leq n - 1, M(\sigma(i)) \leq B M(\sigma(i + 1))$.

**Proof.** Trivial for $n = 1$ and $2$. Suppose the proposition is true for $n - 1$. We have $M(\sigma(1)) \leq B M(\sigma(2)) \leq B \ldots \leq B M(\sigma(n - 1))$. If $M(\sigma(n - 1)) \leq B M(n)$, then the permutation $\sigma$ with $\sigma(n) = n$ satisfies the statement of lemma. Otherwise, let $i$ be the smallest index such that $M(n) \leq B M(\sigma(i))$. The permutation $(M(\sigma(1)), \ldots, M(\sigma(i - 1)), M(n), M(\sigma(i)), \ldots, M(\sigma(n - 1)))$ satisfies the statement of lemma. Thus, the proposition is true for $n$. \hfill $\square$

An heuristic algorithm can be derived from this proof, to determine a permutation of matrices. Its complexity is $O(n^2)$. We define four heuristic algorithms, denoted by $UB^\beta$ with $\beta \in \{\odot, \oplus, \wedge, \odot\}$, which implement this algorithm, by considering relation $\leq B$.

**4.5. Application to scheduling problems**

- Obviously, the $F3||C_{\text{max}}$ problem can be solved using the proposed algorithm. The matrix associated with a job $J_x$ is defined by
  \[
  M_x = \begin{pmatrix}
  p_{1,x} & p_{1,x} p_{2,x} & p_{1,x} p_{2,x} p_{3,x} \\
  0 & p_{2,x} & p_{2,x} p_{3,x} \\
  0 & 0 & p_{3,x}
  \end{pmatrix}.
  \]
  The bounds proposed in this section are a generalization of the bounds of Lageweg et al. [15].
• As previously for the two-machine flow shop problem, we can consider non-sequence-dependent setup times ($s_{nsd}$) or removal times ($r_{nsd}$) and time lags between operations ($a$), by setting $s_{x,i}$ (respectively, $r_{x,i}$) the setup time (respectively, the removal time) of the job $J_x$ on the $i$th machine and:

$$
\mu_i(x) = s_{x,i} \otimes p_{x,i} \otimes r_{x,i},
$$

$$
\mu_{i,j}(x) = s_{x,j} \otimes \left( \bigotimes_{k=j}^{i-1} p_{x,k} \otimes a_{x,k} \otimes p_{x,i} \otimes r_{x,i} \right).
$$

The more general problem under consideration can be denoted by $F3|s_{nsd}, r_{nsd}, a, prmu|C_{max}$.

• The constraints of ready times and tails can also be taken into account in a two-machine flow shop. The matrix corresponding to a job $J_x$ in the $F2|r_j|C_{max}$ problem is defined by

$$
M(J_x) = \begin{pmatrix}
1 & r_x p_{1,x} & r_x p_{1,x} p_{2,x} \\
0 & p_{1,x} & p_{1,x} p_{2,x} \\
0 & 0 & p_{2,x}
\end{pmatrix}.
$$

In the same way, the problem $F2||L_{max}$ can be modeled and solved using the proposed algorithms.

• The single machine problem with heads and tails, denoted by $1|r_i,q_i|C_{q_{max}}$, can also be modeled as a $3 \times 3$ triangular matrices problem.

• In a similar way, it is possible to aggregate subsequences of jobs. Thus, we can also model the batch constraint and solve the $F3|batch, prmu|C_{max}$ problem.

5. Computational experiments

The lower bounds, the upper bounds and the branch-and-bound algorithm have been implemented and tested on randomly generated instances. Two types of instances have been generated:

• instances for the $\bigotimes M3|M_{1,3}$ general problem: all the parameters of the triangular matrices are randomly generated,

• three-machine flow shop instances: the processing times of each job are randomly generated and the matrix $M_x$ associated with job $J_x$ is generated according to the definition given in Section 4.5.

5.1. General problem

The number of matrices $n$ belongs to $\{5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ for the comparison of the bounds. The parameters of each matrix are randomly generated in $[1, 100]$. One thousand instances are generated for each value of $n$. The results are presented in Table 1. Column $LB^z$ and $UB^\beta$ with $x \in \{12, 13, 23\}$ and $\beta \in \{12, 13, 23, \otimes, \wedge, W\}$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$LB^{12}$</th>
<th>$LB^{13}$</th>
<th>$LB^{23}$</th>
<th>$UB^{12}$</th>
<th>$UB^{13}$</th>
<th>$UB^{23}$</th>
<th>$UB^{\otimes}$</th>
<th>$UB^\wedge$</th>
<th>$UB^\otimes$</th>
<th>$UB^W$</th>
<th>$A (%)$</th>
<th>$LB = UB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>237(65)</td>
<td>855(540)</td>
<td>232(73)</td>
<td>175(60)</td>
<td>309(34)</td>
<td>155(47)</td>
<td>402(79)</td>
<td>227(56)</td>
<td>306(94)</td>
<td>381(124)</td>
<td>10.9</td>
<td>210</td>
</tr>
<tr>
<td>10</td>
<td>349(57)</td>
<td>840(345)</td>
<td>339(70)</td>
<td>93(55)</td>
<td>158(73)</td>
<td>95(60)</td>
<td>382(231)</td>
<td>156(92)</td>
<td>254(169)</td>
<td>180(102)</td>
<td>8.5</td>
<td>123</td>
</tr>
<tr>
<td>15</td>
<td>402(84)</td>
<td>810(226)</td>
<td>402(77)</td>
<td>40(26)</td>
<td>136(69)</td>
<td>57(43)</td>
<td>401(288)</td>
<td>154(115)</td>
<td>274(201)</td>
<td>150(100)</td>
<td>5.8</td>
<td>142</td>
</tr>
<tr>
<td>20</td>
<td>418(79)</td>
<td>809(196)</td>
<td>430(68)</td>
<td>29(22)</td>
<td>160(95)</td>
<td>32(21)</td>
<td>469(343)</td>
<td>132(95)</td>
<td>244(174)</td>
<td>146(90)</td>
<td>4.4</td>
<td>151</td>
</tr>
<tr>
<td>30</td>
<td>439(71)</td>
<td>803(180)</td>
<td>431(76)</td>
<td>13(7)</td>
<td>153(97)</td>
<td>16(10)</td>
<td>484(371)</td>
<td>117(84)</td>
<td>253(187)</td>
<td>144(103)</td>
<td>2.8</td>
<td>140</td>
</tr>
<tr>
<td>40</td>
<td>470(96)</td>
<td>782(148)</td>
<td>429(79)</td>
<td>12(8)</td>
<td>148(94)</td>
<td>5(3)</td>
<td>469(369)</td>
<td>110(79)</td>
<td>287(226)</td>
<td>128(92)</td>
<td>2.0</td>
<td>153</td>
</tr>
<tr>
<td>50</td>
<td>473(86)</td>
<td>785(135)</td>
<td>439(80)</td>
<td>4(2)</td>
<td>175(113)</td>
<td>6(3)</td>
<td>483(379)</td>
<td>106(73)</td>
<td>279(215)</td>
<td>107(82)</td>
<td>1.6</td>
<td>168</td>
</tr>
<tr>
<td>60</td>
<td>506(108)</td>
<td>742(89)</td>
<td>481(77)</td>
<td>4(2)</td>
<td>186(131)</td>
<td>4(3)</td>
<td>460(355)</td>
<td>94(67)</td>
<td>282(222)</td>
<td>129(94)</td>
<td>1.2</td>
<td>192</td>
</tr>
<tr>
<td>70</td>
<td>478(80)</td>
<td>751(98)</td>
<td>495(98)</td>
<td>4(3)</td>
<td>178(112)</td>
<td>3(2)</td>
<td>467(361)</td>
<td>98(63)</td>
<td>297(231)</td>
<td>126(83)</td>
<td>1.0</td>
<td>180</td>
</tr>
<tr>
<td>80</td>
<td>463(83)</td>
<td>765(110)</td>
<td>483(97)</td>
<td>2(2)</td>
<td>187(127)</td>
<td>4(3)</td>
<td>488(364)</td>
<td>92(64)</td>
<td>276(194)</td>
<td>132(92)</td>
<td>0.9</td>
<td>176</td>
</tr>
<tr>
<td>90</td>
<td>505(95)</td>
<td>756(79)</td>
<td>479(87)</td>
<td>0(0)</td>
<td>194(119)</td>
<td>2(0)</td>
<td>476(370)</td>
<td>83(45)</td>
<td>279(223)</td>
<td>141(95)</td>
<td>0.7</td>
<td>194</td>
</tr>
<tr>
<td>100</td>
<td>502(100)</td>
<td>749(90)</td>
<td>473(86)</td>
<td>1(1)</td>
<td>217(157)</td>
<td>1(0)</td>
<td>482(356)</td>
<td>84(58)</td>
<td>302(216)</td>
<td>95(58)</td>
<td>0.6</td>
<td>192</td>
</tr>
</tbody>
</table>
Table 2
Computational results for the branch-and-bound evaluation for general problem instances

<table>
<thead>
<tr>
<th>$n$</th>
<th>$v_0$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_{&gt;}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>60</td>
<td>99</td>
<td>41</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>53</td>
<td>54</td>
<td>34</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>25</td>
<td>37</td>
<td>49</td>
<td>3</td>
<td>3</td>
<td>83</td>
</tr>
<tr>
<td>20</td>
<td>34</td>
<td>30</td>
<td>60</td>
<td>10</td>
<td>1</td>
<td>65</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>25</td>
<td>49</td>
<td>26</td>
<td>0</td>
<td>74</td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>24</td>
<td>54</td>
<td>1</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>37</td>
<td>30</td>
<td>41</td>
<td>25</td>
<td>2</td>
<td>65</td>
</tr>
<tr>
<td>40</td>
<td>34</td>
<td>25</td>
<td>45</td>
<td>28</td>
<td>2</td>
<td>66</td>
</tr>
</tbody>
</table>

indicate the number of instances, for which the corresponding lower bound and the corresponding upper bound give the best result. In brackets, we indicate the number of instances for which the method is the only one to give the best result. Column $\Delta$ is the average deviation between the best upper bound and the best lower bound. It is defined by

$$\Delta = \frac{\min \{UB^0\} - \max \{LB^2\}}{\max \{LB^2\}}.$$  

The column $LB = UB$ indicates the number of instances for which the best lower bound is equal to the best upper bound.

The proposed lower bounds are quite equivalent, even if $LB^{13}$ seems to perform better. But due to their complexity in $O(n \log(n))$ time, and because they all can return the best lower bound, the three lower bounds are computed at each node of the branch-and-bound algorithm.

The upper bound $UB^{13}$ performs best than $UB^{12}$ and $UB^{23}$, but the upper bounds $UB^\otimes$ and $UB^\odot$ return the better results. Because of their complexity and because all the upper bounds may give the best solution, especially with less than 30 matrices, all the upper bounds are implemented at the root node of the branch-and-bound algorithm. We can notice that for this problem, the bounds are not tightened, since the average deviation is greater than 2.8% for less than 30 jobs. For a number of jobs greater than 60, the average deviation is less than 1%, which is a good result.

The number of matrices $n$ belongs to {5, 10, 15, 20, 25, 30, 35, 40} for the branch-and-bound algorithm evaluation and 200 instances are generated for each value of $n$. Computational results are presented in Table 2. Column $v_0$ indicates the number of instances solved at the root node, $v_1$ the number of instances solved with a number of nodes comprised between 2 and 10, $v_2$ the number of instances solved with a number of nodes comprised between 11 and 100, $v_3$ the number of instances solved with a number of nodes comprised between 101 and 1000, $v_4$ the number of instances solved with a number of nodes comprised between 1001 and 10000, and $v_{>}$ the number of instances not solved in less than 10 000 nodes.

The results show that the problems are difficult to solve for $n$ greater than or equal to 15. We can notice that generally, if the problem is not solved in less than 1001 nodes, it is not solved in less than 10 000 nodes, too.

5.2. Flow shop instances

The results of the bounds for the $F3||C_{\max}$ instances are presented in Table 3. The processing times are randomly generated in [1,100].

As for the general problem, the three proposed lower bounds are quite equivalent, and have been implemented in the branch-and-bound algorithm. The upper bounds $UB^{13}$, $UB^\otimes$, $UB^\odot$ and $UB^W$ often return the best solution. However, $UB^{13}$ and $UB^\otimes$ are never the only heuristics to return the best solution. Since these algorithms can be implemented in $O(n \log(n))$ time, all the upper bounds are implemented at the root node of the branch-and-bound algorithm. We can notice that for flow shop instances, the bounds are tightened, since the average deviation between the best lower bound and the best upper bound is always smaller than 2.5%. Furthermore, in average, for 58% of the generated instances, the upper bound is equal to the lower bound, and so returns the optimal solution.

Computational results for the branch-and-bound evaluations are presented in Table 4.
Table 3
Comparison of bounds for flow shop instances

<table>
<thead>
<tr>
<th>n</th>
<th>$LB^{12}$</th>
<th>$LB^{13}$</th>
<th>$LB^{23}$</th>
<th>$UB^{12}$</th>
<th>$UB^{13}$</th>
<th>$UB^{23}$</th>
<th>$UB^\otimes$</th>
<th>$UB^\land$</th>
<th>$UB^W$</th>
<th>$\Delta$ (%)</th>
<th>$LB = UB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>363(42)</td>
<td>811(445)</td>
<td>321(20)</td>
<td>184(16)</td>
<td>748(0)</td>
<td>177(11)</td>
<td>810(4)</td>
<td>331(78)</td>
<td>748(0)</td>
<td>811(0)</td>
<td>2.2</td>
</tr>
<tr>
<td>10</td>
<td>476(23)</td>
<td>747(252)</td>
<td>482(24)</td>
<td>45(6)</td>
<td>589(0)</td>
<td>53(3)</td>
<td>645(21)</td>
<td>246(180)</td>
<td>589(0)</td>
<td>657(78)</td>
<td>2.5</td>
</tr>
<tr>
<td>15</td>
<td>559(22)</td>
<td>694(196)</td>
<td>508(23)</td>
<td>24(1)</td>
<td>605(0)</td>
<td>15(0)</td>
<td>641(43)</td>
<td>256(199)</td>
<td>605(0)</td>
<td>618(56)</td>
<td>1.9</td>
</tr>
<tr>
<td>20</td>
<td>580(17)</td>
<td>683(157)</td>
<td>550(15)</td>
<td>15(1)</td>
<td>604(0)</td>
<td>11(1)</td>
<td>631(37)</td>
<td>245(217)</td>
<td>604(0)</td>
<td>608(59)</td>
<td>1.6</td>
</tr>
<tr>
<td>30</td>
<td>559(13)</td>
<td>699(135)</td>
<td>576(19)</td>
<td>3(0)</td>
<td>628(0)</td>
<td>6(0)</td>
<td>645(40)</td>
<td>233(202)</td>
<td>628(0)</td>
<td>626(61)</td>
<td>1.1</td>
</tr>
<tr>
<td>40</td>
<td>622(9)</td>
<td>661(91)</td>
<td>605(13)</td>
<td>0(0)</td>
<td>623(0)</td>
<td>0(0)</td>
<td>616(37)</td>
<td>247(229)</td>
<td>623(0)</td>
<td>600(64)</td>
<td>1.0</td>
</tr>
<tr>
<td>50</td>
<td>634(6)</td>
<td>672(87)</td>
<td>598(4)</td>
<td>1(0)</td>
<td>644(0)</td>
<td>0(0)</td>
<td>601(27)</td>
<td>227(216)</td>
<td>644(0)</td>
<td>616(71)</td>
<td>0.8</td>
</tr>
<tr>
<td>70</td>
<td>617(9)</td>
<td>698(79)</td>
<td>591(6)</td>
<td>1(0)</td>
<td>672(0)</td>
<td>0(0)</td>
<td>671(42)</td>
<td>200(184)</td>
<td>672(0)</td>
<td>646(51)</td>
<td>0.6</td>
</tr>
<tr>
<td>90</td>
<td>639(8)</td>
<td>649(61)</td>
<td>636(7)</td>
<td>0(0)</td>
<td>646(0)</td>
<td>0(0)</td>
<td>615(37)</td>
<td>192(182)</td>
<td>673(0)</td>
<td>643(54)</td>
<td>0.6</td>
</tr>
<tr>
<td>100</td>
<td>634(3)</td>
<td>670(83)</td>
<td>605(5)</td>
<td>0(0)</td>
<td>688(0)</td>
<td>0(0)</td>
<td>656(36)</td>
<td>192(184)</td>
<td>688(0)</td>
<td>634(49)</td>
<td>0.4</td>
</tr>
</tbody>
</table>

We can notice that for flow shop instances, the problem is more easy to solve than the general problem, since more than 92% of the generated instances are solved in less than 1001 nodes, and more than 86% of the instances are solved in less than 101 nodes. Furthermore, problems with $n = 40$ jobs are not difficult to solve, since more than 94% of the instances are solved in less than 1001 nodes.

Some other scheduling problems have been solved using the proposed algorithms. Of course, solving the $1|r_i,q_j|C_{\text{max}}$ problem without the lower bound of Jackson [13] and without the branching scheme of Carlier [3] cannot lead to an efficient algorithm. The proposed algorithms are generic and can be used to solve problems that have never been considered, or for which there is not too much literature, but they are not competitive in comparison with dedicated algorithms.

6. Conclusion

We defined two optimization problems in Max-Plus algebra, related to the minimization of a product of triangular matrices. The first problem concerns $2 \times 2$ matrices, an $O(n \log(n))$ time algorithm is proposed to solve it optimally. This problem is a generalization of numerous sequencing problems like single machine problems or flow shop problems and the proposed algorithm generalizes the corresponding algorithms known in the literature. The second problem concerns $3 \times 3$ triangular matrices. This problem is shown to be $\mathcal{NP}$-hard and we propose lower bounds, upper bounds and a branch-and-bound algorithm to solve it. This problem is also a generalization of sequencing problems like single machine problems or flow shop problems and the bounds we propose generalize some bounds of the literature. The bounds and the branch-and-bound algorithm have been implemented and tested. Computational experiments show

Table 4
Computational results for the branch-and-bound evaluation for flow shop instances

<table>
<thead>
<tr>
<th>n</th>
<th>$BB$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$v_0$</td>
</tr>
<tr>
<td>5</td>
<td>109</td>
</tr>
<tr>
<td>10</td>
<td>94</td>
</tr>
<tr>
<td>15</td>
<td>112</td>
</tr>
<tr>
<td>20</td>
<td>110</td>
</tr>
<tr>
<td>25</td>
<td>111</td>
</tr>
<tr>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>35</td>
<td>115</td>
</tr>
<tr>
<td>40</td>
<td>127</td>
</tr>
</tbody>
</table>

We defined two optimization problems in Max-Plus algebra, related to the minimization of a product of triangular matrices. The first problem concerns $2 \times 2$ matrices, an $O(n \log(n))$ time algorithm is proposed to solve it optimally. This problem is a generalization of numerous sequencing problems like single machine problems or flow shop problems and the proposed algorithm generalizes the corresponding algorithms known in the literature. The second problem concerns $3 \times 3$ triangular matrices. This problem is shown to be $\mathcal{NP}$-hard and we propose lower bounds, upper bounds and a branch-and-bound algorithm to solve it. This problem is also a generalization of sequencing problems like single machine problems or flow shop problems and the bounds we propose generalize some bounds of the literature. The bounds and the branch-and-bound algorithm have been implemented and tested. Computational experiments show...
the efficiency of the bounds for solving flow shop type instances of the problem, but the general problem cannot be efficiently solved optimally using the proposed algorithms.

This work can be extended to $4 \times 4$ matrices and more generally to $m \times m$ matrices. For instance, three-machine flow shop problems with release dates or due dates or two-machine flow shop problems with release dates and due dates can be modeled using $4 \times 4$ matrices. In the same way, general lower bounds and upper bound can be obtained. A future research direction is to consider a new class of optimization problems in Max-Plus algebra, with non-triangular matrices definitions, that can also be applied to scheduling problems. Another possibility is to consider other objective functions like “sum” criteria, or problems that are not solved by finding a unique sequence, like job shop or open shop problems.

Acknowledgements

The authors would like to thank the two anonymous referees for their helpful comments on earlier drafts of this paper.

References