Tropical Abstractions of Max-Plus Linear Systems

Muhammad Syifa'ul Mufid[†], Dieky Adzkiya[‡], and Alessandro Abate[†]

[†]Department of Computer Science, University of Oxford,United Kingdom
{muhammad.syifaul.mufid,alessandro.abate}@cs.ox.ac.uk

[‡]Department of Mathematics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
dieky@matematika.its.ac.id

Abstract. This paper describes the development of finite abstractions of Max-Plus-Linear (MPL) systems using tropical operations. The idea of tropical abstraction is inspired by the fact that an MPL system is a discrete-event model updating its state with operations in the tropical algebra. The abstract model is a finite-state transition system: we show that the abstract states can be generated by operations on the tropical algebra, and that the generation of transitions can be established by tropical multiplications of matrices. The complexity of the algorithms based on tropical algebra is discussed and their performance is tested on a numerical benchmark against an existing alternative abstraction approach.

Keywords: MPL system, tropical algebra, definite form, difference-bound matrix, abstraction, reachability

1 Introduction

Tropical mathematics has been a rapidly growing subject since it was firstly introduced [15]. It has branches in mathematical fields such as tropical geometry [11] and tropical algebra [15]. The latter denotes an algebraic structure that uses max or min for addition and + for multiplication, respectively - hence, it is well known as max-plus or min-plus algebra. In this paper, we use the former operation to define the tropical algebra.

A class of discrete-event system (DES) based on tropical algebra is the Max-Plus-Linear (MPL) one [5]. Models of MPL systems involve tropical operations, namely max and +. The state space of these models represents the timing of events that are synchronised over the max-plus algebra. This means that the next event will occur right after the last of the previous events has finished.

The connections between max-plus algebra and timed systems have been explored in the recent past. First, the dynamics of timed event graphs (a special case of timed Petri nets where all places have exactly one upstream and one downstream transition) can be expressed via MPL systems [5,10]. Second, max-plus polyhedra can be used as data structures in reachability analysis of timed automata [12]: such data structures display a similarity with Difference-Bound Matrices (DBMs) [7].

Finite abstractions of MPL system have been firstly introduced in [3]. These abstraction procedures start by transforming a given MPL system into a Piece-Wise Affine (PWA) model [9]. The PWA model is characterised by several domains (PWA regions) and the corresponding affine dynamics. The resulting abstract states are the partitions corresponding to the PWA regions. Finally, the transition relation between pairs of abstract states depends on the trajectory of the original MPL system. This abstraction technique enables one to perform model checking over an MPL system; one of the applications is safety analysis [3]. Interested readers are referred to [1,3,4] and the VeriSiMPL toolbox [2].

This paper introduces the idea of Tropical Abstractions of MPL systems. The approach is inspired by the fact that an MPL system is a DES that is natively updated via tropical operations. We will show that the abstraction of MPL systems can be established by tropical operations and with algorithms exclusively based on tropical algebra. We argue by experiments that this has clear computational benefits on existing abstraction techniques.

The paper is outlined as follows. Section 2 is divided into three parts. The first part explains the basic of MPL systems including the properties of its state matrix. We introduce the notion of region matrix and of its conjugate, which play a significant role in the abstraction procedures. The notion of definite form and its generalisation are explained in the second part. Finally, we introduce a new definition of DBM as a tropical matrix.

Equipped with these notions, all algorithms of the tropical abstraction procedure are explained in Section 3, which contains the novel contributions of this paper. The comparison of the algorithms performance against the state of the art is presented in Section 4. The paper is concluded with Section 5. The proofs of the propositions are contained in an extended version of this paper [13].

2 Models and Preliminaries

2.1 Max-Plus-Linear Systems

In tropical algebra, $\mathbb{R}_{\max}$ is defined as $\mathbb{R} \cup \{-\infty\}$. This set is equipped with two binary operations, $\oplus$ and $\otimes$, where

$$a \oplus b := \max\{a, b\} \quad \text{and} \quad a \otimes b := a + b,$$

for all $a, b \in \mathbb{R}_{\max}$. The algebraic structure $(\mathbb{R}_{\max}, \oplus, \otimes)$ is a semiring with $\varepsilon := -\infty$ and $e := 0$ as the null and unit element, respectively [5].

The notation $\mathbb{R}_{\max}^{m \times n}$ represents the set of $m \times n$ tropical matrices whose elements are in $\mathbb{R}_{\max}$. Tropical operations can be extended to matrices as follows. If $A, B \in \mathbb{R}_{\max}^{m \times n}, C \in \mathbb{R}_{\max}^{n \times p}$ then

$$[A \oplus B](i,j) = A(i,j) \oplus B(i,j) \text{ and } [A \otimes C](i,j) = \bigoplus_{k=1}^{n} A(i,k) \otimes C(k,j)$$

for all i, j in the corresponding dimension.

Given a natural number m, the tropical power of $A \in \mathbb{R}_{\max}^{n\times n}$ is denoted by $A^{\otimes m}$ and corresponds to $A \otimes \ldots \otimes A$ (m times). As we find in standard algebra, the zero power $A^{\otimes 0}$ is an $n \times n$ identity matrix I_n, where all diagonals and non-diagonals are e and ε, respectively.

An (autonomous) MPL system is defined as

$$x(k+1) = A \otimes x(k), \tag{1}$$

where $A \in \mathbb{R}_{\max}^{n\times n}$ is the matrix system and $x(k) = [x_1(k) \ldots x_n(k)]^\top$ is the state variables [5]. Traditionally, x represents the time stamps of the discrete-events, while k corresponds to an event counter.

Definition 1 (Precedence Graph [5]). *The precedence graph of A, denoted by $\mathcal{G}(A)$, is a weighted directed graph with nodes $1, \ldots, n$ and an edge from j to i with weight $A(i,j)$ if $A(i,j) \neq \varepsilon$.*

Definition 2 (Regular (Row-Finite) Matrix [10]). *A matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is called regular (or row-finite) if there is at least one finite element in each row.*

The following notations deal with a row-finite matrix $A \in \mathbb{R}_{\max}^{n\times n}$. The coefficient $g = (g_1, \ldots, g_n) \in \{1, \ldots, n\}^n$ is called *finite coefficient* iff $A(i, g_i) \neq \varepsilon$ for all $1 \leq i \leq n$. We define the *region matrix* of A w.r.t. the finite coefficient g as

$$A_g(i,j) = \begin{cases} A(i,j), & \text{if } g_i = j \\ \varepsilon, & \text{otherwise.} \end{cases} \tag{2}$$

One can say that A_g is a matrix that keeps the finite elements of A indexed by g. The *conjugate* of A is A^{c}, where

$$A^{\mathsf{c}}(i,j) = \begin{cases} -A(j,i), & \text{if } A(i,j) \neq \varepsilon \\ \varepsilon, & \text{otherwise.} \end{cases} \tag{3}$$

2.2 Definite Forms of Tropical Matrices

The concept of *definite form* over a tropical matrix was firstly introduced in [17]. Consider a given $A \in \mathbb{R}_{\max}^{n\times n}$ and let α be one of the maximal permutations[1] of A. The definite form of A w.r.t. α is $\overline{A}_\alpha$, where

$$\overline{A}_\alpha(i,j) = A(i,\alpha(j)) \otimes A(j,\alpha(j))^{\otimes -1} = A(i,\alpha(j)) - A(j,\alpha(j)). \tag{4}$$

In this paper, we allow for a generalisation of the notion of definite form. We generate the definite form from the finite coefficients introduced above. Notice that the maximal permutation is a special case of finite coefficient $g = (g_1, \ldots, g_n)$ when all g_i are different. Intuitively, the definite form over a finite coefficient g is established by; 1) column arrangement of A using g i.e.

[1] A permutation α is called maximal if $\bigotimes_{i=1}^{n} A(i,\alpha(i)) = \mathrm{per}(A)$, where $\mathrm{per}(A)$ is the permanent of A [6,17].

$B(\cdot, j) = A(\cdot, g_j)$ and then 2) subtracting each column by the corresponding diagonal element i.e. $\overline{A}_g(\cdot, j) = B(\cdot, j) - B(j, j)$ for all $j \in \{1, \ldots, n\}$.

Furthermore, we define two types of definite forms. We call the definite form introduced in [17] to be a *column-definite* form. We define as an additional form the *row-definite* form ${}_g\overline{A}$. The latter form is similar to the former, except that now the row arrangement is used, namely $B(g_i, \cdot) = A(i, \cdot)$ for all $i \in \{1, \ldots, n\}$. Notice that, in a row arrangement, one could find two or more different rows of A are moved into the same row at B. As a consequence, some rows of B remain empty. In these cases, ε is used to fill the empty rows. For rows with multiple entries, we take the maximum point-wise after subtracting by the corresponding diagonal element.

Example 1. Consider a tropical matrix

$$A = \begin{bmatrix} \varepsilon & 1 & 3 \\ 5 & \varepsilon & 4 \\ 7 & 8 & \varepsilon \end{bmatrix}.$$

and a finite coefficient $g = (2, 1, 1)$. The row-definite form for g is

$$A = \begin{bmatrix} \varepsilon & \textcircled{1} & 3 \\ \textcircled{5} & \varepsilon & 4 \\ \textcircled{7} & 8 & \varepsilon \end{bmatrix} \dashrightarrow \left[\begin{array}{ccc} 5 & \varepsilon & 4 \\ 7 & 8 & \varepsilon \\ \hline \varepsilon & 1 & 3 \\ \hline \varepsilon & \varepsilon & \varepsilon \end{array}\right] \dashrightarrow \left[\begin{array}{ccc} 0 & \varepsilon & -1 \\ 0 & 1 & \varepsilon \\ \hline \varepsilon & 0 & 2 \\ \hline \varepsilon & \varepsilon & \varepsilon \end{array}\right] \dashrightarrow {}_g\overline{A} = \begin{bmatrix} 0 & 1 & -1 \\ \varepsilon & 0 & 2 \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix}.$$

On the other hand, the column-definite form w.r.t. g is

$$A = \begin{bmatrix} \varepsilon & \textcircled{1} & 3 \\ \textcircled{5} & \varepsilon & 4 \\ \textcircled{7} & 8 & \varepsilon \end{bmatrix} \dashrightarrow \left[\begin{array}{c|c|c} 1 & \varepsilon & \varepsilon \\ \varepsilon & 5 & 5 \\ 8 & 7 & 7 \end{array}\right] \dashrightarrow \overline{A}_g = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & 0 & -2 \\ 7 & 2 & 0 \end{bmatrix}.$$

Notice that, the elements at the 3rd row of ${}_g\overline{A}$ are all ε. □

The generation of definite forms is formulated as tropical operations as follows:

Proposition 1. *The column-definite and row-definite form of $A \in \mathbb{R}^{n\times n}_{\max}$ w.r.t. a finite coefficient g are $\overline{A}_g = A \otimes A^{\mathsf{c}}_g$ and ${}_g\overline{A} = A^{\mathsf{c}}_g \otimes A$, respectively.* □

2.3 Difference Bound Matrices as Tropical Matrices

Definition 3 (Difference Bound Matrices). *A DBM in $\mathbb{R}^n$ is the intersection of sets defined by $x_i - x_j \sim_{i,j} d_{i,j}$, where $\sim_{i,j} \in \{>, \geq\}$ and $d_{i,j} \in \mathbb{R} \cup \{-\infty\}$ for $0 \leq i, j \leq n$. The variable x_0 is set to be equal to 0.* □

The dummy variable x_0 is used to allow for the single-variable relation $x_i \sim c$, which can be written as $x_i - x_0 \sim c$. Definition 3 slightly differs from [7] as we use operators $\{>, \geq\}$ instead of $\{<, \leq\}$. The reason for this alteration is to transfer DBMs into the tropical domain.

A DBM in $\mathbb{R}^n$ can be expressed as a pair of matrices (D, S). The element $D(i, j)$ stores the bound variable $d_{i,j}$, while S represents the *sign matrix* of the operator i.e. $S(i, j) = 1$ if $\sim_{i,j} = \geq$ and 0 otherwise. In case of $i = j$, it is more convenient to put $D(i, i) = 0$ and $S(i, i) = 1$, as it corresponds to $x_i - x_i \geq 0$.

Under Definition 3, each DBM D in $\mathbb{R}^n$ is an $(n + 1)$-dimensional tropical matrix. Throughout this paper, we may not include the sign matrix whenever recalling a DBM. Operations and properties in tropical algebra can be used for DBM operations such as intersection, computation of the canonical-form, and emptiness checking. Such DBM operations are key for developing abstraction procedures.

Proposition 2. *The intersection of DBM D_1 and D_2 is equal to $D_1 \oplus D_2$.* □

The sign matrix for $D_1 \oplus D_2$ is determined separately as it depends on the operator of the tighter bound. More precisely, suppose that S_1, S_2 and S are the sign matrices of D_1, D_2 and of $D_1 \oplus D_2$ respectively, then

$$S(i,j) = \begin{cases} S_1(i,j), & \text{if } D_1(i,j) > D_2(i,j) \\ S_2(i,j), & \text{if } D_1(i,j) < D_2(i,j) \\ \min\{S_1(i,j), S_2(i,j)\}, & \text{if } D_1(i,j) = D_2(i,j). \end{cases}$$

Any DBM admits a graphical representation, the potential graph, by interpreting the DBM as a weighted directed graph [14]. Because each DBM is also a tropical matrix, the potential graph of D can be viewed as $\mathcal{G}(D)$.

The canonical-form of a DBM D, denoted as $\mathsf{cf}(D)$, is a DBM with the tightest possible bounds [7]. The advantage of the canonical-form representation is that emptiness checking can be evaluated very efficiently. Indeed, for a canonical DBM (D, S), if there exist $0 \leq i \leq n$ such that $D(i, i) > 0$ or $S(i, i) = 0$ then the DBM corresponds to an empty set. Computing $\mathsf{cf}(D)$ is done by the all-pairs shortest path (APSP) problem over the corresponding potential graph [7, 14]. (As we alter the definition of the DBM, it is now equal to all-pairs longest path (APLP) problem). One of the prominent algorithms is Floyd-Warshall [8] which has a cubic complexity w.r.t. its dimension.

On the other hand, in tropical algebra sense, $[D^{\otimes m}](i, j)$ corresponds to the maximal total weights of a path with length m from j to i in $\mathcal{G}(D)$. Furthermore, $[\bigoplus_{m=0}^{n+1} D^{\otimes m}](i, j)$ is equal to the maximal total weights of a path from j to i. Thus, $\bigoplus_{m=0}^{n+1} D^{\otimes m}$ is indeed the solution of APLP problem. Proposition 3 provides an alternative computation of the canonical form of a DBM D based on tropical algebra. Proposition 4 relates non-empty canonical DBMs with the notion of definite matrix. A tropical matrix A is called definite if $\mathrm{per}(A) = 0$ and all diagonal elements of A are zero [6].

Proposition 3. *Given a DBM D, the canonical form of D is $\mathsf{cf}(D) = \bigoplus_{m=0}^{n+1} D^{\otimes m}$, where n is the number of variables excluding x_0.* □

Proposition 4. *Suppose D is a canonical DBM. If D is not empty then it is definite.* □

3 MPL Abstractions Using Tropical Operations

3.1 Related Work

The notion of abstraction of an MPL system has been first discussed in [3]. The procedure starts by transforming the MPL system characterised by $A \in \mathbb{R}_{\max}^{n\times n}$ into a PWA system [3, Alg. 2], and then considering the partitions associated to the obtained PWA system [3, Alg. 6]. The abstract states associated to the partitions are represented by DBMs. The transitions are then generated using one-step forward-reachability analysis [3]: first, the image of each abstract state w.r.t. the MPL system is computed; then, each image is intersected with partitions associated to other abstract states; finally, transition relations are defined for each non-empty intersection. This procedure is summarised in [3, Alg. 7].

The computation of image and of inverse image of a DBM is described in [1]. These computations are used to perform forward and backward reachability analysis, respectively. The worst-case complexity of both procedures is $O(n^3)$, where n is the number of variables in D excluding x_0. A more detailed explanation about image and inverse image computation of a DBM is in Section 3.3.

3.2 Generation of the Abstract States

We begin by recalling the PWA representation of an MPL system characterised by a row-finite matrix $A \in \mathbb{R}_{\max}^{n\times n}$. It is shown in [9] that each MPL system can be expressed as a PWA system. The PWA system comprises of convex domains (or PWA regions) and has correspondingly affine dynamics. The PWA regions are generated from the coefficient $g = (g_1, \ldots, g_n) \in \{1, \ldots, n\}^n$. As shown in [3], the PWA region corresponding to coefficient g is

$$R_g = \bigcap_{i=1}^{n} \bigcap_{j=1}^{n} \{\mathbf{x} \in \mathbb{R}^n | x_{g_i} - x_j \geq A(i,j) - A(i,g_i)\}\,. \tag{5}$$

Notice that, if g is not a finite coefficient, then R_g is empty. However, a finite coefficient might lead to an empty set. Recall that the DBM R_g in (5) is not always in canonical form.

Definition 4 (Adjacent Regions [3, Def. 3.10]). *Two non-empty regions generated by* (5) *R_g and $R_{g'}$ are called adjacent, denoted by $R_g > R_{g'}$, if there exists a single $i \in \{1, \ldots, n\}$ such that $g_i > g'_i$ and $g_j = g'_j$ for each $j \neq i$.* $\square$

The affine dynamic of a non-empty R_g is

$$x_i(k+1) = x_{g_i}(k) + A(i, g_i), \quad i = 1, \ldots, n. \tag{6}$$

Notice that Equation (6) can be expressed as $x(k+1) = A_g \otimes x(k)$, where A_g is a region matrix that corresponds to a finite coefficient g. As mentioned before, a PWA region R_g is also a DBM. The DBM R_g has no dummy variable x_0. For simplicity, we are allowed to consider R_g as a matrix, that is $R_g \in \mathbb{R}_{\max}^{n\times n}$. We show that R_g is related to the row-definite form w.r.t. g.

Proposition 5. *For each finite coefficient g, $R_g = {}_g\overline{A} \oplus I_n$.* □

Algorithm 1 provides a procedure to generate the PWA system from a row-finite $A \in \mathbb{R}^{n\times n}_{\max}$. It consists of: 1) generating region matrices (line 3) and their conjugates (line 4), 2) computing the row-definite form (line 5), and 3) emptiness checking of DBM R_g (lines 6-7). The first two steps are based on tropical operations while the last one is using the Floyd-Warshall algorithm.

Algorithm 1: Generation of the PWA system using tropical operations

```
   Input  : $A \in \mathbb{R}^{n\times n}_{\max}$, a row-finite tropical matrix
   Output: R,A, a PWA system over $\mathbb{R}^n$
           where R is a set of regions and A represent a set of affine dynamics
 1 for $g \in \{1,\dots,n\}^n$ do
 2     if $g$ is a finite coefficient then
 3         generate $A_g$ according to (2)
 4         generate $A^{\mathsf{c}}_g$ from $A_g$ according to (3)
 5         $R_g := (A^{\mathsf{c}}_g \otimes A) \oplus I_n$
 6         $R_g := \mathsf{cf}(R_g)$
 7         if $R_g$ is not empty then
 8             $\mathbf{R} := \mathbf{R} \cup \{R_g\}, \mathbf{A} := \mathbf{A} \cup \{A_g\}$
 9         end
10     end
11 end
```

The complexity of Algorithm 1 depends on line 6; that is $O(n^3)$. The worst-case complexity of Algorithm 1 is $O(n^{n+3})$ because there are n^n possibilities at line 1. However, we do not expect to incur this worst-case complexity, especially when a row-finite A has several ε elements in each row.

In [3], the abstract states are generated via refinement of PWA regions. Notice that, for each pair of adjacent regions R_g and $R_{g'}$, $R_g \cap R_{g'} \neq \emptyset$. The intersection of adjacent regions is removed from the region with the lower index. Mathematically, if $R_g > R_{g'}$ then $R_{g'} := R_{g'} \setminus R_g$.

Instead of removing the intersection of adjacent regions, the partition of PWA regions can be established by choosing the sign matrix for R_g i.e. S_g. As we can see in (5), all operators are $\geq$. Thus, by (5), $S_g(i,j) = 1$ for all $i,j \in \{1,\dots,n\}$. In this paper, we use a rule to decide the sign matrix of R_g as follows

$$S_g(i,j) = \begin{cases} 1, \text{ if } R_g(i,j) > 0 \text{ or} \\ \qquad R_g(i,j) = 0 \text{ and } i \leq j, \\ 0, \text{ if } R_g(i,j) < 0 \text{ or} \\ \qquad R_g(i,j) = 0 \text{ and } i > j. \end{cases} \tag{7}$$

This rule guarantees empty intersection for each pair of region.

Algorithm 2 is a modification of Algorithm 1 by applying rule in (7) before checking the emptiness of R_g. Notation $R_g := (R_g, S_g)$ in line 7 is to emphasise that DBM R_g is now associated with S_g. It generates the partitions of PWA regions which represent the abstract states of an MPL system characterised by

$A \in \mathbb{R}_{\max}^{n\times n}$. The worst-case complexity of Algorithm 2 is similar to that of Algorithm 1.

Algorithm 2: Generation of partition from region of PWA system by tropical operations

```
    Input  : $A \in \mathbb{R}_{\max}^{n\times n}$, a row-finite tropical matrix
    Output: R,A, a PWA system over $\mathbb{R}^n$
            where R is a set of regions and A represent a set of affine dynamics
 1  for $g \in \{1,\dots,n\}^n$ do
 2      if $g$ is a finite coefficient then
 3          generate $A_g$ according to (2)
 4          generate $A_g^{\mathsf{c}}$ from $A_g$ according to (3)
 5          $R_g := A_g^{\mathsf{c}} \otimes A$
 6          generate sign matrix $S_g$ from $R_g$ according to (7)
 7          $R_g := (R_g, S_g)$
 8          $R_g := \mathsf{cf}(R_g)$
 9          if $R_g$ is not empty then
10              $\mathbf{R} := \mathbf{R} \cup \{R_g\}, \mathbf{A} := \mathbf{A} \cup \{A_g\}$
11          end
12      end
13  end
```

Remark 1. The resulting R_g in Algorithm 1 and Algorithm 2 is an n-dimensional matrix which represents a DBM without dummy variable x_0. This condition violates Definition 3. To resolve this, the system matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is extended into $(n+1)$-dimensional matrix by adding the 0th row and column

$$A(0,\cdot) = [0 \;\; \varepsilon \;\; \dots \;\; \varepsilon], \quad A(\cdot,0) = A(0,\cdot)^\top .$$

As a consequence, the finite coefficient g is now an $(n+1)$-row vector $g = (g_0, g_1, \dots, g_n)$ where g_0 is always equal to 0. For the rest of this paper, all matrices are indexed starting from zero. □

As explained in [3], each partition of PWA regions is treated as an abstract state. Therefore, the number of abstract states is equivalent to the cardinality of partitions. Suppose $\hat{R}$ is the set of abstract states, then $\hat{R}$ is a collection of all non-empty R_g generated by Algorithm 2.

3.3 Computation of Image and Inverse Image of DBMs

This section describes a procedure to compute the image of DBMs w.r.t. affine dynamics. First, we recall the procedures from [1]. Then, we develop new procedures based on tropical operations.

The image of a DBM D is computed by constructing a DBM $\mathbf{D}$ consisting of D and its corresponding affine dynamics. The DBM $\mathbf{D}$ corresponds to variables $x_1, x_2, \dots,$ and their primed version $x'_1, x'_2, \dots,$. Then, the canonical-form DBM cf($\mathbf{D}$) is computed. The image of D is established by removing all inequalities with non-primed variables in cf($\mathbf{D}$). This procedure has complexity $O(n^3)$ [1].

Example 2. Let us compute the image of $D = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \geq 6, x_1 - x_3 > -1, x_2 - x_3 \geq 2\}$ w.r.t. its affine dynamics $x_1' = x_2 + 1, x_2' = x_1 + 5, x_3' = x_1 + 2$. The DBM generated from D and the affine dynamics is $\mathbf{D} = \{[\mathbf{x}^\top \ (\mathbf{x}')^\top]^\top \in \mathbb{R}^6 | x_1 - x_2 \geq 6, x_1 - x_3 > -1, x_2 - x_3 \geq 2, x_1' - x_2 = 1, x_2' - x_1 = 5, x_3' - x_1 = 2\}$. The canonical-form representation of $\mathbf{D}$ is cf($\mathbf{D}$) $= \{[\mathbf{x}^\top \ (\mathbf{x}')^\top]^\top \in \mathbb{R}^6 | x_1 - x_2 \geq 6, x_1 - x_3 \geq 8, x_2 - x_3 \geq 2, x_1' - x_1 \leq -5, x_1' - x_2 = 1, x_1' - x_3 \geq 3, x_2' - x_1 = 5, x_2' - x_2 \geq 11, x_2' - x_3 \geq 13, x_3' - x_1 = 2, x_1' - x_2' \leq -10, x_1' - x_3' \leq -7, x_2' - x_3' = 3\}$. The image of D over the given affine dynamics is generated by removing all inequalities containing x_1, x_2 or x_3, i.e. $\{\mathbf{x}' \in \mathbb{R}^3 | x_1' - x_2' \leq -10, x_1' - x_3' \leq -7, x_2' - x_3' = 3\}$. $\square$

The above procedure can be improved by manipulating DBM D directly from the affine dynamics. By (6), one could write $x_i' = x_{g_i} + A_g(i, g_i)$ where x_i and x_i' represent the current and next variables, respectively. For each pair (i, j), we have $x_i' - x_j' = x_{g_i} - x_{g_j} + A_g(i, g_i) - A_g(j, g_j)$. This relation ensures that the bound of $x_i' - x_j'$ can be determined uniquely from $x_{g_i} - x_{g_j}$ and $A_g(i, g_i) - A_g(j, g_j)$.

Proposition 6. *The image of a DBM D w.r.t. affine dynamics $x_i' = x_{g_i} + A_g(i, g_i)$ for $1 \leq i \leq n$ is a set $D' = \bigcap_{i=1}^n \bigcap_{j=1}^n \{\mathbf{x}' \in \mathbb{R}^n | x_i' - x_j' = x_{g_i} - x_{g_j} + A_g(i, g_i) - A_g(j, g_j)\}$, where the bound of $x_{g_i} - x_{g_j}$ is taken from D.* $\square$

Example 3. We compute the image of $D = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \geq 6, x_1 - x_3 > -1, x_2 - x_3 \geq 2\}$ with the same affine dynamics $x_1' = x_2 + 1, x_2' = x_1 + 5, x_3' = x_1 + 2$. From the affine dynamics and D, we have $x_1' - x_2' = x_2 - x_1 - 4 \leq -10$, $x_1' - x_3' = x_2 - x_1 - 1 \leq -7$, and $x_2' - x_3' = 3$ which yields a set $\{\mathbf{x}' \in \mathbb{R}^3 | x_1' - x_2' \leq -10, x_1' - x_3' \leq -7, x_2' - x_3' = 3\}$. $\square$

Algorithm 3 shows a procedure to generate the image of (D, S) w.r.t. the affine dynamics represented by $\mathbf{x}' = A_g \otimes \mathbf{x}$. It requires DBM (D, S) located in a PWA region R_g. This means that there is exactly one finite coefficient g such that $(D, S) \subseteq R_g$. The complexity of Algorithm 3 is in $O(n^2)$ as the addition step at 4 line has complexity of $O(1)$.

Algorithm 3: Computation of the image of DBM D w.r.t. $\mathbf{x}' = A_g \otimes \mathbf{x}$

```
  Input  : $(D, S)$, a DBM in $\mathbb{R}^n$
           $g$, the corresponding finite coefficient such that $(D, S) \subseteq R_g$
           $A_g$, a region matrix which represents the affine dynamics
  Output: $(D', S')$, image of $D$ w.r.t. $\mathbf{x}' = A_g \otimes \mathbf{x}$
1 Initialize $(D', S')$ with $\mathbb{R}^n$
2 for $i \in \{0, \ldots, n\}$ do
3     for $j \in \{0, \ldots, n\}$ do
4         $D'(i, j) := D(g_i, g_j) + A_g(i, g_i) - A_g(j, g_j)$
5         $S'(i, j) := S(g_i, g_j)$
6     end
7 end
```

As an alternative, we also show that the image of a DBM can be computed by tropical matrix multiplications with the corresponding region matrix A_g.

Proposition 7. *The image of DBM D in $\mathbb{R}^n$ w.r.t. the affine dynamics $\boldsymbol{x}' = A_g \otimes \boldsymbol{x}$ is $D' = A_g \otimes D \otimes A_g^{\mathsf{c}}$.* □

The procedure to compute the image of DBM D w.r.t. MPL system can be viewed as the extension of Algorithm 3. First, the DBM D is intersected with each region of the PWA system. Then, for each nonempty intersection we apply Algorithm 3. The worst-case complexity is $O(|\hat{R}|n^2)$.

In [1], the procedure to compute the inverse image of D' w.r.t. affine dynamics involves: 1) constructing DBM $\mathbf{D}$ that consists of D' and its corresponding affine dynamics, 2) generating the canonical form of $\mathbf{D}$ and 3) removing all inequalities with primed variables. The complexity of computing the inverse image using this procedure is $O(n^3)$ as it involves the emptiness checking of a DBM [1].

Example 4. Let us compute the inverse image of $D' = \{\mathbf{x}' \in \mathbb{R}^3 | x_1' - x_2' \leq -10, x_1' - x_3' \leq -7, x_2' - x_3' = 3\}$ w.r.t. affine dynamics $x_1' = x_2 + 1, x_2' = x_1 + 5, x_3' = x_1 + 2$. The DBM generated from D' and the affine dynamic is $\mathbf{D}' = \{[\mathbf{x}^\top\ (\mathbf{x}')^\top]^\top \in \mathbb{R}^6 | x_1' - x_2' \leq -10, x_1' - x_3' \leq -7, x_2' - x_3' = 3, x_1' - x_2 = 1, x_2' - x_1 = 5, x_3' - x_1 = 2\}$. The canonical-form of $\mathbf{D}$ is cf($\mathbf{D}$) $= \{[\mathbf{x}^\top\ (\mathbf{x}')^\top]^\top \in \mathbb{R}^6 | x_1 - x_2 \geq 6, x_1' - x_1 \leq -5, x_1' - x_2 = 1, x_1' - x_3 \geq 3, x_2' - x_1 = 5, x_2' - x_2 \geq 11, x_3' - x_1 = 2, x_3' - x_2 \geq 8, x_1' - x_2' \leq -10, x_1' - x_3' \leq -7, x_2' - x_3' = 3\}$. The inverse image of D' over the given affine dynamic is computed by removing all inequalities containing x_1', x_2' or x_3', i.e. $\{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \geq 6\}$. □

The inverse image of D' can be established by manipulating D' from the affine dynamics. Notice that, from (6), we have $x_{g_i} - x_{g_j} = x_i' - x_j' + A_g(j, g_j) - A_g(i, g_i)$. Unlike the previous case, it is possible that $x_{g_i} - x_{g_j}$ has multiple bounds. This happens because there is a case $g_{i_1} = g_{i_2}$ but $i_1 \neq i_2$. In this case, the bound of $x_{g_i} - x_{g_j}$ is taken from the tightest bound among all possibilities.

Proposition 8. *The inverse image of DBM D' w.r.t. affine dynamics $x_i' = x_{g_i} + A_g(i, g_i)$ for $i \in \{1, \ldots, n\}$ is a set $D = \bigcap_{i=1}^n \bigcap_{j=1}^n \{\boldsymbol{x}' \in \mathbb{R}^n | x_{g_i} - x_{g_j} = x_i' - x_j' + A_g(j, g_j) - A_g(i, g_i)\}$ where the bound of $x_i' - x_j'$ is taken from D'.* □

Algorithm 4 shows the steps to compute the inverse image of DBM D' over the affine dynamics $\mathbf{x}' = A_g \otimes \mathbf{x}$. It has similarity with Algorithm 3 except it updates the value of $D(g_i, g_j)$ and $S(g_i, g_j)$ for all $i, j \in \{0, \ldots, n\}$. The variables b and s in lines 4-5 represent the new bound of $x_{g_i} - x_{g_j}$; that is, $x_{g_i} - x_{g_j} \geq b$ if $s = 1$ and $x_{g_i} - x_{g_j} > b$ if $s = 0$. If the new bound is larger then it replaces the old one. In case of they are equal, we only need to update the operator.

Similar to Algorithm 3, Algorithm 4 has complexity in $O(n^2)$. In tropical algebra, the procedure of Algorithm 4 can be expressed as tropical matrix multiplications using a region matrix and its conjugate.

Proposition 9. *The inverse image of DBM D' in $\mathbb{R}^n$ w.r.t. affine dynamic $\boldsymbol{x}' = A_g \otimes \boldsymbol{x}$ is $D = (A_g^{\mathsf{c}} \otimes D' \otimes A_g) \oplus I_{n+1}$.* □

The procedure to compute the inverse image of DBM D' w.r.t. MPL system can be viewed as the extension of Algorithm 4. First, we compute the inverse image of DBM D' w.r.t. all affine dynamics. Then each inverse image is intersected with the corresponding PWA region. The worst-case complexity is $O(|\hat{R}|n^2)$.

Algorithm 4: Computation of the inverse image of DBM D' w.r.t. $\mathbf{x}' = A_g \otimes \mathbf{x}$

```
Input  : $(D', S')$, a DBM in $\mathbb{R}^n$
         $g$, the corresponding finite coefficient such that $(D, S) \subseteq R_g$
         $A_g$, a region matrix which represents the affine dynamics
Output: $(D, S)$, inverse image of $D$ w.r.t. $\mathbf{x}' = A_g \otimes \mathbf{x}$
1  Initialize $(D, S)$ with $\mathbb{R}^n$
2  for $i \in \{0, \dots, n\}$ do
3      for $j \in \{0, \dots, n\}$ do
4          $b := D'(i, j) + A_g(j, g_j) - A_g(i, g_i)$
5          $s := S'(i, j)$
6          if $b > D(g_i, g_j)$ then
7              $D(g_i, g_j) := b$
8              $S(g_i, g_j) := s$
9          else if $b = D(g_i, g_j)$ then
10             $S(g_i, g_j) := \min\{s, S(g_i, g_j)\}$
11         end
12     end
13 end
```

3.4 Generation of the Abstract Transitions

As we mentioned before, the transition relations are generated by one-step forward-reachability analysis, and involve the image computation of each abstract state. Suppose $\hat{R} = \{\hat{r}_1, \dots, \hat{r}_{|\hat{R}|}\}$[2] is the set of abstract states generated by Algorithm 2. There is a transition from $\hat{r}_i$ to $\hat{r}_j$ if $\mathsf{Im}(\hat{r}_i) \cap \hat{r}_j \neq \emptyset$, where $\mathsf{Im}(\hat{r}_i) = \{A \otimes \mathbf{x} | \mathbf{x} \in \hat{r}_i\}$ which can be computed by Algorithm 3. Notice that, each abstract state corresponds to an unique affine dynamics.

The procedure to generate the transitions is summarized in Algorithm 5. It spends most time for emptiness checking at line 5. Therefore, the worst-case complexity is in $O(n^3|\hat{R}|^2)$, where n is the dimension of A in Algorithm 2.

Example 5. Matrix A in Example 1 has 8 finite coefficients. The abstract states generated by Algorithm 2 are $\hat{r}_1 = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \geq 1, x_1 - x_3 \geq 3, x_2 - x_3 \geq 2\}, \hat{r}_2 = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 < 1, x_1 - x_3 > -1, x_2 - x_3 \geq 2\}, \hat{r}_3 = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \leq -3, x_1 - x_3 \leq -1, x_2 - x_3 \geq 2\}, \hat{r}_4 = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \geq 1, x_1 - x_3 > -1, x_2 - x_3 < 2\}, \hat{r}_5 = \{\mathbf{x} \in \mathbb{R}^3 | -3 < x_1 - x_2 < 1, -1 < x_1 - x_3 < 3, -2 < x_2 - x_3 < 2\}, \hat{r}_6 = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 \geq 1, x_1 - x_3 \leq -1, x_2 - x_3 \leq -2\}$, and $\hat{r}_7 = \{\mathbf{x} \in \mathbb{R}^3 | x_1 - x_2 < 1, x_1 - x_3 \leq -1, x_2 - x_3 < 2\}$, which correspond to finite coefficients $(2, 1, 1), (2, 1, 2), (2, 3, 2), (3, 1, 1), (3, 1, 2)$, $(3, 3, 1)$, and $(3, 3, 2)$, respectively. The only finite coefficient that leads to an empty set is $(2, 3, 1)$. Figure 1 shows the illustrations of abstract states and transition relations. □

[2] $\hat{R}$ is the collection of non-empty R_g. We use small letter $\hat{r}_i$ for sake of simplicity.

Algorithm 5: Generation of the transitions via one-step forward reachability.

```
  Input  : $\hat{R} = \{\hat{r}_1, \ldots, \hat{r}_{|\hat{R}|}\}$, the set of abstract states generated by Algorithm 2
  Output: $T \subseteq \hat{R} \times \hat{R}$, a transition relation
1 Initialize $T$ with an empty set
2 for $i \in \{1, \ldots, |\hat{R}|\}$ do
3     for $j \in \{1, \ldots, |\hat{R}|\}$ do
4         compute $\mathsf{Im}(\hat{r}_i)$ by Algorithm 3
5         if $\mathsf{Im}(\hat{r}_i) \cap \hat{r}_j \neq \emptyset$ then
6             $T := T \cup \{(\hat{r}_i, \hat{r}_j)\}$
7         end
8     end
9 end
```

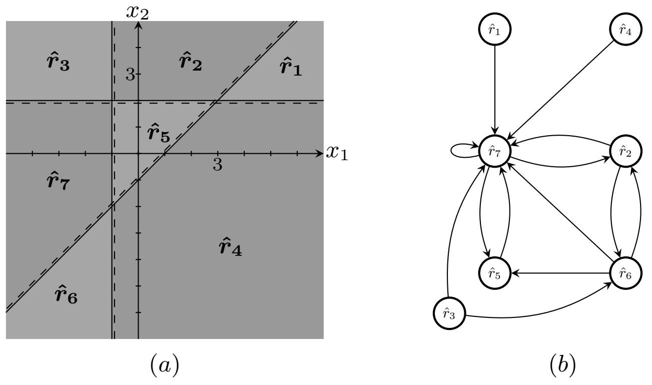

Fig. 1: (a) Plot of partitions (and corresponding abstract states), projected on the plane $x_3 = 0$. The solid and dashed lines represent $\geq$ and $>$, respectively. (b) Transition relations among abstract states.

4 Computational Benchmarks

We compare the run-time of abstraction algorithms in this paper with the procedures in VeriSiMPL 1.4 [2]. For increasing n, we generate matrices $A \in \mathbb{R}^{n\times n}_{\max}$ with two finite elements in each row, with value ranging between 1 and 100. The location and value of the finite elements are chosen randomly. The computational benchmark has been implemented on the Oxford University ARC server [16].

We run the experiments for both procedures (VeriSiMPL 1.4 and Tropical) using MATLAB R2017a with parallel computing. Over 10 different MPL systems for each dimension, Table 1 shows the running time to generate the abstract states and transitions. Each entry represents the average and maximal values.

With regards to the generation of abstract states, the tropical algebra based algorithm is much faster than VeriSiMPL 1.4. As the dimension increases, we see an increasing gap of the running time. For a 12-dimensional MPL system over

Table 1: Generation of abstract states and transitions.

n	VeriSiMPL 1.4.		Tropical	
	time for generating abstract states	time for generating transitions	time for generating abstract states	time for generating transitions
3	$\{7.51, 9.82\}$[ms]	$\{0.13, 0.21\}$[sec]	$\{4.04, 8.39\}$[ms]	$\{0.12, 0.17\}$[sec]
4	$\{11.29, 15.58\}$[ms]	$\{0.20, 0.29\}$[sec]	$\{5.23, 16.10\}$[ms]	$\{0.17, 0.22\}$[sec]
5	$\{18.51, 28.19\}$[ms]	$\{0.20, 0.21\}$[sec]	$\{5.16, 6.89\}$[ms]	$\{0.19, 0.20\}$[sec]
6	$\{49.22, 55.10\}$[ms]	$\{0.21, 0.22\}$[sec]	$\{9.99, 11.44\}$[ms]	$\{0.20, 0.21\}$[sec]
7	$\{90.88, 118.94\}$[ms]	$\{0.24, 0.26\}$[sec]	$\{15.88, 20.67\}$[ms]	$\{0.22, 0.24\}$[sec]
8	$\{0.21, 0.28\}$[sec]	$\{0.32, 0.44\}$[sec]	$\{0.04, 0.04\}$[sec]	$\{0.27, 0.38\}$[sec]
9	$\{0.52, 0.69\}$[sec]	$\{0.72, 1.07\}$[sec]	$\{0.07, 0.10\}$[sec]	$\{0.60, 0.91\}$[sec]
10	$\{1.25, 1.88\}$[sec]	$\{2.62, 4.48\}$[sec]	$\{0.14, 0.17\}$[sec]	$\{2.38, 4.22\}$[sec]
11	$\{3.87, 5.14\}$[sec]	$\{17.62, 29.44\}$[sec]	$\{0.35, 0.39\}$[sec]	$\{17.17, 28.88\}$[sec]
12	$\{8.34, 14.22\}$[sec]	$\{1.20, 2.24\}$[min]	$\{0.61, 0.71\}$[sec]	$\{1.10, 2.19\}$[min]
13	$\{26.17, 45.17\}$[sec]	$\{5.05, 10.45\}$[min]	$\{1.21, 1.37\}$[sec]	$\{4.98, 10.40\}$[min]
14	$\{1.81, 4.24\}$[min]	$\{41.14, 112.09\}$[min]	$\{0.06, 0.07\}$[min]	$\{40.61, 110.06\}$[min]
15	$\{10.29, 23.18\}$[min]	$\{2.63, 7.57\}$[hr]	$\{0.11, 0.17\}$[min]	$\{2.57, 7.65\}$[hr]

10 independent experiments, the time needed to compute abstract states using tropical based algorithm is less than 1 second. In comparison, average running time using VeriSiMPL 1.4 for the same dimension is 8.34 seconds.

For the generation of transitions, the running time of tropical algebra-based algorithm is slightly faster than that of VeriSiMPL 1.4. We remind that the procedure to generate transitions involves the image computation of each abstract state. In comparison to the 2nd and 4th columns of Table 1, Table 2 shows the running time to compute the image of abstract states. Each entry represents the average and maximum of running time. It shows that our proposed algorithm for computing the image of abstract states is faster than VeriSiMPL 1.4.

Table 2: Computation of the image of abstract states.

n	VeriSiMPL 1.4.	Tropical
3	$\{0.84, 1.13\}$[ms]	$\{0.16, 0.23\}$[ms]
4	$\{1.13, 1.76\}$[ms]	$\{0.13, 0.20\}$[ms]
5	$\{1.53, 2.40\}$[ms]	$\{0.14, 0.16\}$[ms]
6	$\{5.32, 6.68\}$[ms]	$\{0.18, 0.20\}$[ms]
7	$\{11.22, 15.19\}$[ms]	$\{0.31, 0.44\}$[ms]
8	$\{26.05, 46.94\}$[ms]	$\{0.71, 1.19\}$[ms]
9	$\{70.31, 92.87\}$[ms]	$\{2.37, 3.37\}$[ms]
10	$\{153.07, 183.08\}$[ms]	$\{4.06, 6.57\}$[ms]
11	$\{380.01, 477.94\}$[ms]	$\{5.58, 8.19\}$[ms]
12	$\{0.79, 1.13\}$[sec]	$\{0.02, 0.03\}$[sec]
13	$\{1.96, 3.13\}$[sec]	$\{0.03, 0.04\}$[sec]
14	$\{5.51, 9.60\}$[sec]	$\{0.06, 0.16\}$[sec]
15	$\{14.33, 23.82\}$[sec]	$\{0.49, 0.87\}$[sec]

We also compare the running time algorithms when applying forward- and backward-reachability analysis. We generate the forward reach set [3, Def 4.1] and backward reach set [3, Def 4.3] from an initial and a final set, respectively. In more detail, suppose $\mathcal{X}_0$ is the set of initial conditions; the forward reach set $\mathcal{X}_k$ is defined recursively as the image of $\mathcal{X}_{k-1}$, namely $\mathcal{X}_k = \{A \otimes \mathbf{x} \mid \mathbf{x} \in \mathcal{X}_{k-1}\}$. On the other hand, suppose $\mathcal{Y}_0$ is a set of final conditions. The backward reach set $\mathcal{Y}_{-k}$ is defined via the inverse image of $\mathcal{Y}_{-k+1}$, $\mathcal{Y}_{-k} = \{\mathbf{y} \in \mathbb{R}^n \mid A \otimes \mathbf{y} \in \mathcal{Y}_{-k+1}\}$, where n is the dimension of A.

We select $\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^n \mid 0 \leq x_1, \ldots, x_n \leq 1\}$ and $\mathcal{Y}_0 = \{\mathbf{y} \in \mathbb{R}^n \mid 90 \leq y_1, \ldots, y_n \leq 100\}$ as the sets of initial and final conditions, respectively. The experiments have been implemented to compute $\mathcal{X}_1, \ldots, \mathcal{X}_N$ and $\mathcal{Y}_{-1}, \ldots, \mathcal{Y}_{-N}$ for $N = 10$. Notice that it is possible that the inverse image of $\mathcal{Y}_{-k+1}$ results in an empty set: in this case, the computation of backward reach sets is terminated, since $\mathcal{Y}_{-k} = \ldots = \mathcal{Y}_{-N} = \emptyset$. (If this termination happens, it applies for both VeriSiMPL 1.4 and the algorithms based on tropical algebra.)

Table 3 reports the average computation of PWA system and reach sets over 10 independent experiments for each dimension. In general, algorithms based on tropical algebra outperform those of VeriSiMPL 1.4. For a 15-dimensional MPL system, the average time to generate PWA system using VeriSiMPL 1.4 is just over 20 seconds. In comparison, the computation time for tropical algorithm is under 5 seconds.

Table 3: Reachability analysis.

n	VeriSiMPL 1.4.			Tropical		
	time for generating PWA system	time for generating forward reach sets	time for generating backward reach sets	time for generating PWA system	time for generating forward reach sets	time for generating backward sets
3	2.55[ms]	11.37[ms]	5.73[ms]	1.70[ms]	8.33[ms]	5.63[ms]
4	4.31[ms]	9.87[ms]	27.00[ms]	1.37[ms]	7.72[ms]	28.48[ms]
5	9.23[ms]	11.77[ms]	3.62[ms]	1.88[ms]	9.25[ms]	2.89[ms]
6	23.44[ms]	18.49[ms]	9.76[ms]	3.80[ms]	13.81[ms]	7.35[ms]
7	49.59[ms]	35.68[ms]	21.53[ms]	7.84[ms]	32.02[ms]	17.92[ms]
8	108.75[ms]	85.27[ms]	34.05[ms]	16.84[ms]	73.63[ms]	28.62[ms]
9	0.25[sec]	0.18[sec]	0.09[sec]	0.03[sec]	0.17[sec]	0.07[sec]
10	0.48[sec]	0.28[sec]	0.17[sec]	0.08[sec]	0.25[sec]	0.14[sec]
11	1.19[sec]	0.77[sec]	1.35[sec]	0.18[sec]	0.76[sec]	1.13[sec]
12	2.52[sec]	1.14[sec]	0.88[sec]	0.38[sec]	1.01[sec]	0.70[sec]
13	7.02[sec]	3.96[sec]	2.78[sec]	1.09[sec]	3.56[sec]	1.95[sec]
14	8.15[sec]	5.54[sec]	4.61[sec]	1.54[sec]	5.24[sec]	2.98[sec]
15	20.60[sec]	19.23[sec]	12.39[sec]	4.21[sec]	18.37[sec]	7.16[sec]
16	46.92[sec]	60.19[sec]	36.00[sec]	9.62[sec]	58.70[sec]	20.41[sec]
18	2.98[min]	3.91[min]	2.61[min]	0.83[min]	3.83[min]	1.35[min]
20	15.74[min]	21.03[min]	15.21[min]	4.84[min]	20.86[min]	7.51[min]

Tropical algorithms also show advantages to compute reach sets. As shown in Table 3, the average computation time for forward and backward-reachability analysis is slightly faster when using tropical procedures. There is evidence that the average time to compute the backward reach sets decreases as the dimension increases. This happens because the computation is terminated earlier once there is a $k \leq N$ such that $\mathcal{Y}_{-k} = \emptyset$. Notice that, this condition occurs for both VeriSiMPL 1.4 and the new algorithms based on tropical algebra.

We summarise the worst-complexity of abstraction procedures via VeriSiMPL 1.4 and our proposed algorithms in Table 4 – recall that in VeriSiMPL 1.4 the generation of abstract states involves two procedures: the generation of PWA systems and the refinement of PWA regions.

Table 4: The worst-case complexity of abstraction procedures.

Procedures	VeriSiMPL 1.4	Tropical
Generating the PWA systems	$O(n^{n+3})$	$O(n^{n+3})$
Generating the abstract states	$O(n^{n+3})$ and $O(n^{2n+1})$	$O(n^{n+3})$
Computing the image of DBMs	$O(n^3)$	$O(n^2)$
Computing the inverse image of DBMs	$O(n^3)$	$O(n^2)$
Generating the abstract transitions	$O(n^3\|\hat{R}\|^2)$	$O(n^3\|\hat{R}\|^2)$

5 Conclusions

This paper has introduced the concept of MPL abstractions using tropical operations. We have shown that the generation of abstract states is related to the row-definite form of the given matrix. The computation of image and inverse image of DBMs over the affine dynamics has also been improved based on tropical algebra operations.

The procedure has been implemented on a numerical benchmark and compared with VeriSiMPL 1.4. Algorithm 2 has shown a strong advantage to generate the abstract states especially for high-dimensional MPL systems. Algorithms (Algorithm 3, 4, and 5) for the generation of transitions and for reachability analysis also display an improvement.

For future research, the authors are interested to extend the tropical abstractions for non-autonomous MPL systems [5], with dynamics that are characterised by non-square tropical matrices.

Acknowledgements The first author is supported by Indonesia Endowment Fund for Education (LPDP), while the third acknowledges the support of the Alan Turing Institute, London, UK.

References

1. D. Adzkiya. *Finite Abstractions of Max-Plus-Linear Systems: Theory and Algorithms.* PhD thesis, Delft University of Technology, 2014.
2. D. Adzkiya and A. Abate. VeriSiMPL: Verification via biSimulations of MPL models. In K. Joshi, M. Siegle, M. Stoelinga, and P. D'Argenio, editors, *Proc. 10th International Conference on Quantitative Evaluation of Systems (QEST'13)*, volume 8054 of *Lecture Notes in Computer Science*, pages 253–256, Hiedelberg, 2013. Springer.
3. D. Adzkiya, B. De Schutter, and A. Abate. Finite abstractions of max-plus-linear systems. *IEEE Transactions on Automatic Control*, 58(12):3039–3053, 2013.

4. D. Adzkiya, B. De Schutter, and A. Abate. Computational techniques for reachability analysis of max-plus-linear systems. *Automatica*, 53:293–302, 2015.
5. F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and linearity: an algebra for discrete event systems*. John Wiley & Sons Ltd, 1992.
6. P. Butkovič. Max-algebra: the linear algebra of combinatorics? *Linear Algebra and its applications*, 367:313–335, 2003.
7. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proc. International Conference on Computer Aided Verification*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212, Hiedelberg, 1989. Springer.
8. R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.
9. W. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.
10. B. Heidergott, G. J. Olsder, and J. Van der Woude. *Max Plus at work: modeling and analysis of synchronized systems: a course on Max-Plus algebra and its applications*. Princeton University Press, 2014.
11. I. Itenberg, G. Mikhalkin, and E. I. Shustin. *Tropical algebraic geometry*, volume 35. Springer Science & Business Media, 2009.
12. Q. Lu, M. Madsen, M. Milata, S. Ravn, U. Fahrenberg, and K. G. Larsen. Reachability analysis for timed automata using max-plus algebra. *The Journal of Logic and Algebraic Programming*, 81(3):298–313, 2012.
13. M. S. Mufid, D. Adzkiya, and A. Abate. Tropical abstractions of max-plus-linear systems, 2018. arXiv:1806.04604.
14. M. Péron and N. Halbwachs. An abstract domain extending difference-bound matrices with disequality constraints. In *Proc. 8th International Conference on Verification, Model-checking, and Abstract Intepretation (VMCAI'07)*, volume 43497 of *Lecture Notes in Computer Science*, pages 268–282. Springer, January 2007.
15. J.-E. Pin. Tropical semirings. *Idempotency*, pages 50–69, 1998.
16. A. Richards. University of Oxford Advanced Research Computing, 2015. Zenodo.10.5281/zenodo.22558.
17. S. Sergeev. Max-plus definite matrix closures and their eigenspaces. *Linear algebra and its applications*, 421(2-3):182–201, 2007.