Using Max-Plus Algebra to Improve the Analysis of Non-Cyclic Task Models

Haibo Zeng
McGill University, email: haibo.zeng@mcgill.ca

Marco Di Natale
Scuola Superiore S. Anna, email: marco@sssup.it

***Abstract*—Several models have been proposed to represent conditional executions and dependencies among real-time concurrent tasks for the purpose of schedulability analysis. Among them, task graphs with cyclic recurrent behavior, i.e. those modeled with a single source vertex and a period parameter specifying the minimum amount of time that must elapse between successive activations of the source job, allow for efficient schedulability analysis based on the periodicity of the request and demand bound functions (*rbf* and *dbf*). We leverage results from max-plus algebra to identify a recurrent term in *rbf* and *dbf* of general task graph models, even when the execution is neither recurrent nor controlled by a period parameter. As such, the asymptotic complexity of calculating *rbf* and *dbf* is independent from the length of the time interval. Experimental results demonstrate significant improvements on the runtime for system schedulability analysis.**

I. Introduction

Several abstract models have been proposed to represent conditional executions and dependencies among real-time concurrent tasks for the purpose of schedulability analysis. The available models can be classified based on the concept of *task graph*, where vertices represent different kinds of jobs, and edges represent the possible flows of control. Each vertex (job) is characterized by its worst case execution time requirement and relative deadline. Each graph edge is labeled with the minimum separation time between the release of the two vertices (jobs) it connects.

A *single vertex* task graph corresponds to the simplest model of independent tasks [17] activated by periodic or sporadic events. The multiframe [19] and generalized multiframe (GMF) task models [4] assume that worst case execution times are not constant, but are defined according to a cyclic pattern. The corresponding task graph is therefore a *chain of vertices*. The recurring branching task model [3] allows selection points to determine the behavior for a given task instance, in statements such as “if-then-else” and “case”, thus modeling conditional branches and optional (OR-type) executions. The corresponding task graph is a *directed tree*. The recurring real-time task model [5] allows the task graph to be any *directed acyclic graph* (DAG). All the above models satisfy the property of *cyclic recurrent behavior*:

- **recurrent**: the graph has a **unique source vertex**. The completion of a sink vertex automatically releases the source job. This execution pattern may be implicit, or it can be modeled by explicitly adding back edges from the sink vertices to the unique source (as in Figure 1).

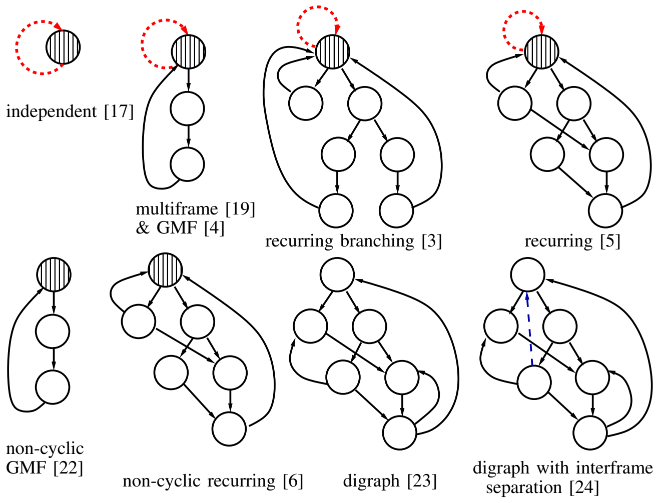

Figure 1. A summary of the proposed task models.

- **cyclic**: a **period** parameter defines the minimum time interval that must elapse between two consecutive releases of the source vertex job.

Figure 1 summarizes the proposed models, where the top row shows the cyclic recurrent models. The unique source vertex is represented in the figure by a *shadowed node*, with the periodicity constraint represented by a *dotted line* as a self loop on it. A *solid line* indicates a precedence constraint associated with minimum inter-release time.

The non-cyclic generalized multiframe model [22] removes the periodicity in the activation pattern of the jobs. More specifically, it is possible to activate any job as long as the minimum separation time with respect to its predecessor has passed. Without considering the (implicit) back edges from the sink vertices to the source vertex, the task graph is still a *directed tree*. The next task graph model is the one proposed in [1] (called Recurring Task Model with Branching and Control variables), or in a similar form in [6] (as the non-cyclic recurring real-time task model). Such a model is a generalization of both the recurring real-time [5] and the non-cyclic GMF [22] models. These three models ([22] [1] [6]) relax the constraint of cyclic execution of the graph, but still assume a recurrent activation model of a source job. In addition, they typically satisfy the **frame separation property**, where the deadline of a job is constrained to be no larger than the inter-release times of its outgoing edges.

The digraph model [23] removes the restriction of recurrence by allowing arbitrary cycles and therefore *arbitrary directed graphs* to represent the release structure of jobs, which significantly increases the expressiveness. Another generalization is to relax the frame separation property to allow arbitrary job deadlines. This model is used for the analysis of implementations of synchronous finite state machines [27]. The model in [24] further extends the digraph model by allowing *global inter-release separation constraints* between non-adjacent job release (denoted by *dashed lines* in Figure 1). The analysis on the resulting task model is **tractable**, i.e., no worse than pseudo-polynomial time for bounded utilization systems. The model of timed automata with tasks [18] is a generalization of all the above models, which allows complex dependencies between job release times and task synchronization. However, schedulability analysis is shown to be very expensive and even undecidable in certain variants of the model [9]. A discussion on the expressiveness and complexity of schedulability analysis for task graph models can be found in [23].

A special subset of task digraphs consists of **strongly connected graphs**, where every node is reachable from any other node. We expect most applications of practical interests to be represented by such graphs, because of the need to bring back the system to a (possibly initial) controlled state, which could be a safe state in case of safety-critical systems. All examples in Figure 1 are strongly connected graphs.

The concepts of **request bound function** (or *rbf*) and **demand bound function** (or *dbf*) have been introduced in [5] for the analysis of task graphs.

Definition 1: For a task τ, the maximum cumulative execution times by its jobs that have their release times within any time interval of length t is defined as its **request bound function** $\tau.rbf(t)$.

Definition 2: For a task τ, the maximum cumulative execution times by its jobs that have their release times **and deadlines** within any time interval of length t is defined as its **demand bound function** $\tau.dbf(t)$.

Intuitively, *rbf* can be used to compute the maximum amount of execution time that can interfere with a task (by adding the *rbf*s of higher priority tasks). *dbf* quantifies the amount of execution time from jobs that are released and must be completed within a given time interval. Schedulability analysis based on these two functions has been proposed for systems with static and dynamic priority scheduling.

The periodicity of the request and demand bound functions for the task graph models is of special interest in this paper. For task models with a *cyclic recurrent behavior*, a repeating pattern of the functions *rbf(t)* and *dbf(t)* allows to compute them for large t based on the values of small t. This makes the complexity of computing $rbf(t)$ and $dbf(t)$ independent from t. However, such a periodicity is only studied for task models with a cyclic recurrent behavior.

Our contributions. In this paper, we study the periodicity of the request and demand bound functions for non-cyclic task models, including the non-cyclic generalized multiframe model [22], the non-cyclic recurring real-time task model [6], the digraph real-time task model [23] and its extension [24]. We present our results using the digraph real-time task model [23], since it is a strict generalization of the models in [22] [6], and an extended digraph task can be transformed into a plain digraph task with the same *rbf* and *dbf* functions [24]. We make a key connection of *rbf* and *dbf* to the matrix power in max-plus algebra [2], and leverage the related research results (e.g. [21]) to prove their **linear periodicity** [2], i.e., they can be represented by a finite aperiodic part and a periodic part repeated infinitely often. The required proof technique is significantly different from [27] (which provides initial results on the periodicity of execution matrix for synchronous state machines), including the consideration of the sequence of maximum elements of matrix power and the required task graph transformation. We also develop efficient algorithms to calculate the periodicity parameters for strongly connected task graphs. The task graph transformation is then used to derive a tight upper bound on *rbf* and *dbf* functions, which improves upon [23].

Many schedulability techniques rely on such a linear periodicity to provide an efficient representation and computation of the system timing behavior. For example, the Modular Performance Analysis (MPA) toolbox [26] based on Real-Time Calculus [25] (and in turn, on the min-plus/max-plus algebra) only supports infinite curves with linear periodicity property [16]. Also, as in [16], linear periodic curves can be safely approximated by the *standard event model* in SymTA/S [14]. Our results **open the possibility of using non-cyclic task models to represent application structures within these schedulability analysis frameworks**.

II. The Digraph Task Model and its Extension

A **digraph real-time task** (DRT) τ is characterized by a directed graph $\mathcal{D}(\tau) = (\mathbb{V}, \mathbb{E})$ where the set of n vertices $\mathbb{V} = \{v_1, v_2, ..., v_n\}$ represents the types of jobs that can be released for task τ. Each vertex $v_i \in \mathbb{V}$ (or type of job) is characterized by an ordered pair $< e(v_i), d(v_i) >$ where $e(v_i)$ and $d(v_i)$ denote its worst case execution time (WCET) and relative deadline, respectively. Edges represent possible flows of control, i.e. the release order of the jobs of τ. An edge $(v_i, v_j) \in \mathbb{E}$ is labeled with a parameter $p(v_i, v_j)$ that denotes the minimum separation time between the releases of v_i and v_j. We assume that relative deadlines and minimum inter-release times are positive *integers* (i.e. $\in \mathbb{N}^+$).

An **event** δ of τ is a pair (t, v) which denotes the release of a job $v \in \mathbb{V}$ at time t. An **event sequence** Δ is a (possibly infinite) sequence of job release events. A **legal** event sequence $\Delta = [(t_1, v_1), (t_2, v_2), ...]$ corresponds to a (potentially infinite) path $(v_1, v_2, ...)$ in $\mathcal{D}(\tau)$, in which

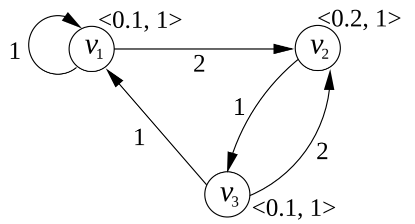

Figure 2. An example digraph real-time task.

$\forall i \geq 1$, $(v_i, v_{i+1}) \in \mathbb{E}$, and the release times satisfy $t_{i+1} \geq t_i + p(v_i, v_{i+1})$. A legal event sequence $\Delta = [(t_i, v_i)]$ is called **urgent** if each t_i is the minimum for Δ to be legal, i.e. for an arbitrarily small $\epsilon > 0$ and any i, $\Delta' = [(t_1, v_1), ..., (t_i - \epsilon, v_i), ...]$ is illegal.

We assume that the task satisfy the **l-MAD** property [4], i.e. for each edge $(v_i, v_j) \in \mathbb{E}$, $d(v_i) \leq d(v_j) + p(v_i, v_j)$. This constraint is less restrictive than the frame separation property, but is sufficient to guarantee that the absolute deadline of the last job is the largest among all jobs in any legal event sequence. The results on arbitrary deadlines are left for future publication due to limited space.

Example: Figure 2 shows an example of a digraph real-time task with 3 vertices (types of jobs), which satisfies the l-MAD property. The event sequence $\Delta = [(5, v_1), (7, v_2), (9, v_3)]$ is legal but not urgent, as the third job (associated with v_3) could be released at time 8.

The **extended digraph real-time task model** (EDRT) generalizes the digraph task model by allowing a set of additional constraints $H(\tau)$ to express a minimum separation time between any two jobs. Each constraint $h = (v_f, v_t, \gamma(v_f, v_t)) \in H(\tau)$ specifies that a minimum time interval $\gamma(v_f, v_t)$ must occur between the releases of v_f and v_t. Each $\gamma(v_f, v_t)$ is also assumed to be a positive integer.

A **task system** Γ consists of a set of independent real time tasks $\tau_1, \tau_2, ..., \tau_m$. We assume that tasks are scheduled on **a uniprocessor with preemptive scheduling**.

III. Schedulability Analysis

We review the schedulability analysis techniques developed for systems scheduled with dynamic or static priority. For **dynamic priority** scheduling, Earliest Deadline First (EDF) is optimal for independent tasks on a preemptive uniprocessor [17].

Theorem 1: ([23]) A task system Γ is schedulable by dynamic priority (EDF) if and only if the sum of the demand bound functions for all tasks over any time interval does not exceed the length of the interval, that is,

$$\forall t \geq 0, \sum_{\tau \in \Gamma} \tau.dbf(t) \leq t \tag{1}$$

For a task system scheduled with **static priority**, schedulability is guaranteed if, for each task, the available cpu time is no smaller than the total execution time required by the task itself and all the higher priority tasks.

Theorem 2: For a task system Γ with static priorities, task $\tau_i \in \Gamma$ is schedulable if and only if

$$\forall t \geq 0, \exists t' \leq t \text{ s.t. } \tau_i.dbf(t) + \sum_{\tau_j \in hp(i)} \tau_j.rbf(t') \leq t' \tag{2}$$

where $hp(i)$ is the set of tasks with priority higher than τ_i.

The above theorem derives from the analysis of recurring real-time tasks presented in [5], as the proof also applies to the extended digraph task model.

In practice, the schedulability of dynamic priority systems is analyzed by checking whether there exists a counterexample to Theorem 1: $\exists t \geq 0$ such that $\sum_{\tau \in \Gamma} \tau.dbf(t) > t$. A similar approach can be derived for static priority systems. Thus, schedulability analysis requires the efficient computation of the *rbf* and *dbf* functions of a task τ over a time interval of given length t. Also, since it is practically impossible to check the schedulability condition for all (integer) $t \geq 0$, an upper bound t_f on such a counterexample is defined in Section VII, which improves upon [23].

For the periodic task model [17], the *rbf* and *dbf* functions of task τ with period p and WCET e are

$$\tau.rbf(t) = \left\lceil \frac{t}{p} \right\rceil e, \quad \tau.dbf(t) = \left\lfloor \frac{t}{p} \right\rfloor e$$

For other cyclic recurrent task graphs, such as the most generic one (the recurring task model [5]), every cycle in the graph contains the unique source node for which two consecutive releases are separated by the period parameter. Such properties lead to a regular repeating pattern of *rbf* and *dbf*. For a recurring task with period p, the maximum execution request for any path from the unique source node to a sink node is denoted as e. Its *rbf* and *dbf* satisfy the following equations for sufficiently large r (where $q = \frac{e}{p}$)

$$\forall j \in \mathbb{N}^+, \begin{cases} \tau.rbf(r + j \cdot p) = \tau.rbf(r) + j \cdot p \cdot q \\ \tau.dbf(r + j \cdot p) = \tau.dbf(r) + j \cdot p \cdot q \end{cases} \tag{3}$$

In the following, we show how **the *dbf* and *rbf* functions are long-term periodic even for task graphs without cyclic recurrent properties**, including the non-cyclic GMF model [22], the non-cyclic recurring task model [6], the digraph real-time task (DRT) model [23], and its extension (EDRT) [24]. An EDRT can be transformed into an equivalent plain digraph task (with the same *rbf* and *dbf*) [24]. We present our results using the *digraph task model*, since it is also a strict generalization of the non-cyclic GMF and non-cyclic recurring task models.

We first introduce in Section IV the background on max-plus algebra and the matrix power sequence under it. In Section V we prove the main theorem of the paper on the periodicity of the *rbf* and *dbf* functions for digraph real-time tasks. In Sections VI and VII we discuss methods for computing the attributes of periodic *rbf* and *dbf* and an improved upper bound on the time interval on which to check feasibility conditions. Section VIII shows the experimental results demonstrating the significant improvements of our analysis over existing methods.

IV. Max-plus Algebra

The max-plus algebra [2] is defined over $\mathbb{R}^* = \mathbb{R} \bigcup \{-\infty\}$ (or in general, any dioid) where the **addition** (denoted by $\oplus$) and **multiplication** (denoted by $\otimes$) operations are defined as

$$a \oplus b = \max(a, b) \qquad a \otimes b = a + b$$

The element $-\infty$ is neutral with respect to $\oplus$, i.e. $a \oplus (-\infty) = a, \forall a \in \mathbb{R}^*$. Likewise, 0 is neutral with respect to $\otimes$, i.e. $a \otimes 0 = a, \forall a \in \mathbb{R}^*$.

A. Matrix and its Power Sequence

For two matrices $\mathbf{A} \in \mathbb{R}^*(m, k)$ and $\mathbf{B} \in \mathbb{R}^*(k, n)$, the result of the multiplication is a matrix $\mathbf{C} \in \mathbb{R}^*(m, n)$, where

$$c_{i,j} = \bigoplus_{l=1}^{k} (a_{i,l} \otimes b_{l,j}) = \max_{l=1}^{k} (a_{i,l} + b_{l,j})$$

The k-th power of a square matrix $\mathbf{A} \in \mathbb{R}^*(n, n)$, denoted as $\mathbf{A}^{(k)}$, is recursively defined as the multiplication of $\mathbf{A}^{(k-1)}$ and $\mathbf{A}^{(1)} = \mathbf{A}$.

$$a_{i,j}^{(k)} = \max_{l=1}^{n} (a_{i,l}^{(k-1)} + a_{l,j})$$

The properties of $\mathbf{A}$ are assessed through its graph $\mathcal{G}(\mathbf{A})$.

Definition 3: The **graph $\mathcal{G}(\mathbf{A})$ of a square matrix** $\mathbf{A} \in \mathbb{R}^*(n, n)$ is a weighted digraph $(\mathbb{V}, \mathbb{E}, w)$ with nodes $\mathbb{V} = \{1, ..., n\}$. Every finite $a_{i,j}$ defines an **edge** $(i, j) \in \mathbb{E}$ weighted by its value $a_{i,j}$. If $a_{i,j} = -\infty$, there is no edge from i to j. A **path** Π in $\mathcal{G}$ is a sequence of nodes $(i_1, i_2, ..., i_{t+1})$ where each (i_k, i_{k+1}) is an edge in $\mathbb{E}$. The **length** $|\Pi|$ of Π is t, the number of edges in the path. If $i_1 = i_{t+1}$, Π is called a **cycle**. The **weight** of a path Π, $w(\Pi)$, is the sum of the weights of its edges. For a cycle c with length $|c| > 0$, its **cycle mean** $\bar{w}(c)$ is the ratio between its weight and length, i.e. $\bar{w}(c) = w(c)/|c|$. The maximum mean of any cycle in $\mathcal{G}(\mathbf{A})$ is denoted as $\lambda(\mathbf{A})$.

$\mathcal{G}(\mathbf{A})$ is **strongly connected** if all its nodes are contained in a common cycle. In this case, $\mathbf{A}$ is defined as **irreducible**.

Definition 4: Given a subset of the vertexes $\mathbb{K} \subseteq \mathbb{V}$ defining a strongly connected component $\mathcal{K} = (\mathbb{K}, \mathbb{E} \bigcap (\mathbb{K} \times \mathbb{K}))$ of $\mathcal{G}(\mathbf{A})$, its maximum cycle mean $\lambda(\mathcal{K})$ is defined as the maximum of $\bar{w}(c)$ where $c \subseteq \mathcal{K}$. $\mathcal{K}$ is called a **highly connected component** of $\mathcal{G}(\mathbf{A})$ if $\lambda(\mathcal{K}) = \lambda(\mathbf{A})$. $HCC^*(\mathcal{G}(\mathbf{A}))$ denotes the set of highly connected components with a cycle. The **high period** of $\mathcal{K} \in HCC^*(\mathcal{G}(\mathbf{A}))$ is defined as

$$\text{hper}(\mathcal{K}) = gcd\{|c| : c \text{ is a cycle in } \mathcal{K}, \bar{w}(c) = \lambda(\mathbf{A})\}$$

Definition 5: An **elementary path** is a path with no cycle. The operation of **cycle deletion** replaces a cycle $(i_1, i_2, ..., i_1)$ with a single node i_1. Given two paths Π and Π', Π' is a **cycle extension** of Π, denoted as $\Pi \subseteq_c \Pi'$, if Π can be created from Π' by a finite number of cycle-deletions.

The **set of paths** of $\mathcal{G}(\mathbf{A})$ from node i to j is denoted as $\mathbb{P}_{\mathcal{G}(\mathbf{A})}(i, j)$. The **set of elementary paths** is denoted as $\mathbb{P}^*_{\mathcal{G}(\mathbf{A})}(i, j)$. The subset of the paths of length t is $\mathbb{P}^t_{\mathcal{G}(\mathbf{A})}(i, j)$. The **power sequence for an elementary path** $\Pi \in \mathbb{P}^*_{\mathcal{G}(\mathbf{A})}(i, j)$ is the sequence of the maximum weights among all the cycle extensions of Π. Each sequence term is

$$a_{\Pi}^{(t)} = \max\{w(\Pi') : \Pi' \in \mathbb{P}^t_{\mathcal{G}(\mathbf{A})}(i, j), \Pi \subseteq_c \Pi'\}$$

Intuitively, each element $a_{i,j}^{(t)}$ of $\mathbf{A}^{(t)}$ defines the path of length t with the maximum weight in $\mathcal{G}(\mathbf{A})$ from node i to j. The following fundamental theorem in max-plus algebra defines the relationship between the power of a matrix $\mathbf{A}$ and the maximum weights of the paths in $\mathcal{G}(\mathbf{A})$, and consequently, the power of elementary paths.

Theorem 3: ([2]) The power sequence of $\mathbf{A} \in \mathbb{R}^*(n, n)$, for all $t \in \mathbb{N}^+$ and all $i, j \in \{1, ..., n\}$ can be computed as

$$\begin{aligned} a_{i,j}^{(t)} &= \max\{w(\Pi') : \Pi' \in \mathbb{P}^t_{\mathcal{G}(\mathbf{A})}(i, j)\} \\ &= \max\{a_{\Pi}^{(t)} : \Pi \in \mathbb{P}^*_{\mathcal{G}(\mathbf{A})}(i, j)\} \end{aligned} \tag{4}$$

B. Linear Periodicity and General Periodicity

Definition 6: A sequence $a^* = \{a^{(t)}\}, t \in \mathbb{N}^+$ is **almost linear periodic**, if there exists a real number $q \in \mathbb{R}$ and a pair of integers r and p such that

$$\forall t > r, \ \ a^{(t+p)} = a^{(t)} + p \cdot q \tag{5}$$

The smallest p with the above property is the **linear period** of a^*, denoted as $p = \text{lper}(a^*)$. q is the **linear factor** of a^*, or $q = \text{lfac}(a^*)$. Finally, the smallest r with the above property is the **linear defect**, or $r = \text{ldef}(a^*)$.

Definition 7: The matrix $\mathbf{A} = (a_{i,j})$ is defined as **almost linear periodic** if the power sequence $a_{i,j}^*$ of each element $a_{i,j}$ in $\mathbf{A}^* = \{\mathbf{A}^{(t)}\}, t \in \mathbb{N}^+$ is almost linear periodic. The matrix $\text{lfac}(\mathbf{A}^*) = (\text{lfac}(a_{i,j}^*))$ is the **linear factor matrix** of $\mathbf{A}$, the number $\text{ldef}(\mathbf{A}) = \max\{\text{ldef}(a_{i,j}^*)\}$ is the **linear defect** of $\mathbf{A}$, and $\text{lper}(\mathbf{A}) = lcm\{\text{lper}(a_{i,j}^*)\}$ is the **linear period** of $\mathbf{A}$.

A matrix is almost linear periodic if it is irreducible (as demonstrated in [2]). Gavalec [11] proposed an $O(n^3)$ algorithm for computing the linear period and factor of an irreducible matrix based on the following theorem:

Theorem 4: ([11]) An irreducible matrix $\mathbf{A} \in \mathbb{R}^*(n, n)$ is almost linear periodic, its linear factor is $\text{lfac}(\mathbf{A}) = \mathbf{Q}$, with $q_{i,j} = \lambda(\mathbf{A})$ for all i, j; its linear period is

$$\text{lper}(\mathbf{A}) = lcm\{\text{hper}(\mathcal{K}) : \mathcal{K} \in HCC^*(\mathcal{G}(\mathbf{A}))\}. \tag{6}$$

Example: Consider the square matrix $\mathbf{A}$ in (7) and its digraph $\mathcal{G}(\mathbf{A})$ in Figure 3.

$$\mathbf{A} = \begin{bmatrix} 0.1 & -\infty & -\infty & 0.1 & -\infty \\ -\infty & 0 & 0.2 & -\infty & -\infty \\ 0.1 & -\infty & 0 & -\infty & 0.1 \\ -\infty & 0 & -\infty & -\infty & -\infty \\ -\infty & 0 & -\infty & -\infty & -\infty \end{bmatrix} \tag{7}$$

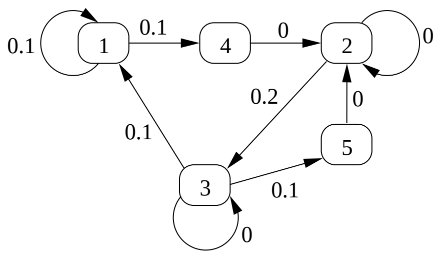

Figure 3. The digraph corresponding to the matrix of Equation (7)

$\mathcal{G}(\mathbf{A})$ is strongly connected, thus $\mathbf{A}$ is irreducible. There are three cycles $(1,1)$, $(2,3,5,2)$, and $(1,4,2,3,1)$ each with a cycle mean of 0.1. Since there is no other cycle with a larger cycle mean, $\lambda(\mathbf{A}) = 0.1$, and the high period of $\mathcal{G}(\mathbf{A})$ (which is a highly connected component itself) is $gcd\{1,2,3\} = 1$. By Theorem 4, $\mathbf{A}$ is almost linear periodic with a linear factor 0.1 and linear period 1. Equation (8) shows how to apply the linear periodicity of $\mathbf{A}$ to compute its power sequence with its linear defect equal to 6.

$$\forall i \geq 0, \mathbf{A}^{(6+i)} = \begin{bmatrix} 0.6 & 0.5 & 0.6 & 0.6 & 0.6 \\ 0.7 & 0.6 & 0.7 & 0.7 & 0.7 \\ 0.6 & 0.5 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.5 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.5 & 0.6 & 0.6 & 0.6 \end{bmatrix} + 0.1 \times i \quad (8)$$

When the matrix is reducible, it can still be almost linear periodic, but deciding whether this is the case has been demonstrated to be an NP-complete problem [11]. However, even when the matrix is not linear periodic, it is still possible to avoid computing the *rbf* function over a long time interval by leveraging the concept of general periodicity ([21]).

Definition 8: A sequence $a^* = (a^{(t)}), t \in \mathbb{N}^+$ is defined as **almost generally periodic**, if there exist a pair of integers r and p and a vector $Q(i) \in \mathbb{R}^*, i = 1, ..., p$ such that

$$\begin{aligned} &\forall i = 1, ..., p, \quad \forall t > r, \ \ t \equiv i \pmod{p}, \\ &a^{(t+p)} = a^{(t)} + p \cdot Q(i) \end{aligned} \quad (9)$$

The smallest p with the above property is the **general period** of a^*, or $p = \text{gper}(a^*)$. Q is the **general factor** of a^*, or $Q = \text{gfac}(a^*)$. Finally, the smallest r with the above property is the **general defect**, or $r = \text{gdef}(a^*)$.

Definition 9: Matrix $\mathbf{A}$ is **almost generally periodic** if the power sequence $a^*_{i,j}$ of each element in $\mathbf{A}^* = \{\mathbf{A}^{(t)}\}, t \in \mathbb{N}^+$ is almost generally periodic. $\text{gfac}(\mathbf{A}^*) = (\text{gfac}(a^*_{i,j}))$ is the **general factor matrix** of $\mathbf{A}$, $\text{gdef}(\mathbf{A}) = \max\limits_{i,j} \text{gdef}(a^*_{i,j})$ is its **general defect**, and $\text{gper}(\mathbf{A}) = lcm\{\text{gper}(a^*_{i,j})\}$ is its **general period**.

The following theorem states the applicability of the periodicity property to all matrices.

Theorem 5: ([21]) Every matrix is almost generally periodic over the max-plus algebra.

The problem of computing the general period $\text{gper}(\mathbf{A})$ or general factor matrix $\text{gfac}(\mathbf{A})$ for a given square matrix $\mathbf{A}$ is shown to be NP-hard [21]. But the complexity is expressed in terms of the size of the matrix (or the corresponding graph), and is asymptotically independent from the power of the matrix (or the time interval for the *rbf* and *dbf* functions).

V. Demonstrating the Periodicity of the *rbf* and *dbf*

Existing results on max-plus algebra provide the foundation for original extensions that allow to demonstrate the linear periodicity of the *rbf* and *dbf* for a generic digraph task:

- As the graph weights are a representation of the computing load in a given time (and the elements of the matrix power represent the processor load requested in some time interval), demonstrating the periodicity of the *rbf* and *dbf* requires that the sequence of the maximum element of the matrix powers is also periodic (the periodicity of each element is not enough).
- Next, we provide a task graph transformation (to a unit-weighted digraph or UDRT) that allows to encode the total execution time request of a sequence of graph jobs in a given time interval as elements of a max-plus matrix power sequence. UDRT allows to leverage results from max-plus algebra and formally demonstrate the periodicity of *rbf* and *dbf* for non recurrent graphs.

We now recapture several definitions and lemmas that are useful for our extension to max-plus algebra (Section V-A).

Definition 10: Given an elementary path Π, the set of strongly connected components that includes at least one node of Π is $SCC^*_\Pi(\mathcal{G}(\mathbf{A})) = \{\mathcal{K} : \mathcal{K} \bigcap \Pi \neq \emptyset\}$. The **maximum mean** $\lambda(\Pi)$ of Π is the maximum cycle mean of any element $\mathcal{K}$ in $SCC^*_\Pi(\mathcal{G}(\mathbf{A}))$, and its **period** $\text{lper}(\Pi)$ is defined as the least common multiple of the high periods of highly connected components of $\mathcal{K}$.

$$\begin{aligned} \lambda(\Pi) &= \max\{\lambda(\mathcal{K}) : \mathcal{K} \in SCC^*_\Pi(\mathcal{G}(\mathbf{A}))\} \\ \text{lper}(\Pi) &= lcm\{\text{hper}(\mathcal{K}') : \mathcal{K}' \in HCC^*(\mathcal{K}), \lambda(\mathcal{K}) = \lambda(\Pi)\} \end{aligned}$$

If Π does not share any node with any strongly connected component, then $\lambda(\Pi) = -\infty$.

Lemma 6: ([12] [21]) For an elementary path Π, if its maximum cycle mean $\lambda(\Pi) > -\infty$, then a^*_Π is almost linear periodic with period $\text{lper}(\Pi)$ (or its integer divisor) and factor $\lambda(\Pi)$; otherwise, it is almost generally periodic with a general factor of $-\infty$.

Two other lemmas in [21] provide sufficient conditions for the maximum of two almost linear periodic sequences to be linear periodic.

Lemma 7: ([21]) Consider two almost linear periodic sequences a^* and b^*, where a^* has period p_a and factor q, and b^* has period p_b and the same factor q. Then the sequence $\max(a^*, b^*)$ is almost linear periodic with period p as an integer divisor of $lcm(p_a, p_b)$, and factor q.

Lemma 8: ([21]) Consider two almost linear periodic sequences a^* and b^*. a^* has period p_a and factor q_a. The elements of b^* are **all finite**, with period p_b and factor

$q_b > q_a$. Then the maximum sequence $\max(a^*, b^*)$ is almost linear periodic with period p_b and factor q_b.

A. Periodicity of Largest Element Sequence of Matrix Power

We now prove the maximum element of a matrix power sequence is linear periodic. *We only consider non-trivial matrices (with at least one cycle in the associated graph).* The demonstration requires additional definitions and lemmas as intermediate steps.

Definition 11: The **largest element sequence** of $\mathbf{A} \in \mathbb{R}^*(n,n)$, denoted as $a^*_{\max} = \{a^{(t)}_{\max}\}, \forall t \in \mathbb{N}^+$, is the sequence of the largest element in the power sequence of $\mathbf{A}$, i.e. $a^{(t)}_{\max} = \max\limits_{i,j} a^{(t)}_{i,j}, \forall t \in \mathbb{N}^+$.

Combining the definition and Theorem 3, the largest element sequence is the maximum among the power of all the elementary paths in the graph.

$$\begin{aligned} a^{(t)}_{\max} = & \max_{i,j}\{a^{(t)}_{\Pi'} : \Pi' \in \mathbb{P}^t_{\mathcal{G}(\mathbf{A})}(i,j)\} \\ = & \max\{a^{(t)}_{\Pi} : \Pi \text{ is a primary path in } \mathcal{G}(\mathbf{A})\} \end{aligned} \tag{10}$$

Definition 12: A generally periodic sequence a^* with factor Q_a is **dominated** by a linear periodic sequence b^* with factor q_b, if all elements in b^* are **finite**, and each element in Q_a is less than or equal to q_b.

The following lemma generalizes Lemmas 7 and 8.

Lemma 9: Consider a general periodic sequence a^* (period p_a and factor Q_a) and a dominating linear periodic sequence b^* (period p_b and factor q_b). Then the sequence $\max(a^*, b^*)$ is almost linear periodic with period as an integer divisor of $p = lcm(p_a, p_b)$ and factor q_b.

Proof: By the definitions of periodicity, there exists an integer r such that for a given $i \in \{1, ..., p\}$

$$\forall j \in \mathbb{N}^+, \begin{cases} a^{(r+i+j\cdot p)} = a^{(r+i)} + j \cdot p \cdot Q_a(i) \\ b^{(r+i+j\cdot p)} = b^{(r+i)} + j \cdot p \cdot q_b > -\infty \end{cases}$$

We consider the maximum sequence of $a^{(r+i+j\cdot p)}$ and $b^{(r+i+j\cdot p)}$. There are two cases.

Case 1: $Q_a(i) < q_b$. We define $s_i = \frac{a^{(r+i)} - b^{(r+i)}}{p(q_b - Q_a(i))}$ if $a^{(r+i)} > b^{(r+i)}$; otherwise $s_i = 0$. By simple arithmetics, we can prove that $\forall j \in \mathbb{N}^+$, $\max(a^{(r'_i+j\cdot p)}, b^{(r'_i+j\cdot p)}) = b^{(r'_i+j\cdot p)} = b^{(r'_i)} + j \cdot p \cdot q_b$, where $r'_i = r + i + \left\lceil \frac{s_i}{p} \right\rceil p$.

Case 2: $Q_a(i) = q_b$. The power sequence of a has the same factor as b. Thus, $\forall j \in \mathbb{N}^+$, $\max(a^{(r'_i+j\cdot p)}, b^{(r'_i+j\cdot p)}) = \max(a^{(r'_i)}, b^{(r_i)}) + j \cdot p \cdot q_b$, where $r'_i = r + i$.

Thus, as a generalization of the two cases, $\forall i = 1, ..., p$, $\forall j \in \mathbb{N}^+$, there exists an integer $r' = \max_i(r'_i)$ such that

$$\max(a^{(r'+j\cdot p)}, b^{(r'+j\cdot p)}) = \max(a^{(r')}, b^{(r')}) + j \cdot q_b$$

Hence the proof. ■

We now prove the linear periodicity of the sequence of the largest matrix element. It is done by constructing a subset of the elementary paths whose largest element sequence dominates those of the other paths.

Theorem 10: The sequence of the largest element of any non-trivial square matrix is almost linear periodic.

Proof: For a non-trivial square matrix $\mathbf{A}$, assume its maximum cycle mean $\lambda(\mathbf{A}) > -\infty$. There must exist a cycle c such that $\bar{w}(c) = \lambda(\mathbf{A})$. Without loss of generality, we denote $c = (1, 2, ..., |c|, 1)$. We consider a sequence that is the maximum of the powers of the elementary paths in c: $b^{(t)} = \max\{a^{(t)}_{\Pi} : \Pi \subset c\}$.

The power sequence of a generic elementary path $\Pi \subset c$ is almost linear periodic with factor $\lambda(\mathbf{A})$. By Lemma 7, b^* is also almost linear periodic with the same factor. Moreover, $\forall t \in \mathbb{N}^+$, we can find a path Π' which is the concatenation of j repetitions of $(1, 2, .., |c|)$ and $\Pi = (1, 2, ..., k+1)$, where $j = \lfloor \frac{t}{|c|} \rfloor$ and $k = t - j \cdot |c|$. Π' is a cycle extension of the elementary path Π in c. As $b^{(t)} \geq w(\Pi')$, we have $\forall t \in \mathbb{N}^+, b^{(t)}$ is finite.

By Lemma 6, the power sequences of all other (finitely many) elementary paths are generally periodic with a factor no larger than $\lambda(\mathbf{A})$, thus dominated by b^*. By Lemma 9, the sequence of the largest element (the maximum among b^* and those of the other elementary paths) is almost linear periodic. ■

Example: The sequence of the largest element of the irreducible matrix in Equation (7) can be derived as

$$\forall i \geq 0, \quad a^{(6+i)}_{\max} = 0.7 + 0.1 \times i \tag{11}$$

B. Task Transformation

To apply the above extended results on max-plus algebra, **a task transformation and a refinement of the *rbf* and *dbf* functions are needed**. We first present the transformation of a task digraph into an equivalent digraph where all inter-release times are equal to one[1]. Such a digraph task with unit-weighted edges is defined as a **unit digraph task (UDRT)**. For $\mathcal{D}(\tau) = (\mathbb{V}, \mathbb{E})$, we generate an equivalent UDRT $\mathcal{D}'(\tau') = (\mathbb{V}', \mathbb{E}')$ by the following rules:

- Each vertex $v_i \in \mathbb{V}$ corresponds to k_i vertices $\nu_{i,1}, \nu_{i,2}, ..., \nu_{i,k_i}$ in $\mathbb{V}'$, where
$$k_i = \max(1, \max_{v_j:(v_i,v_j)\in\mathbb{E}} p(v_i, v_j)). \tag{12}$$
- $\nu_{i,1}$ is labeled with an execution time $e(\nu_{i,1}) = e(v_i)$, and all the other new vertices have zero execution time. Deadlines can be assigned arbitrarily (as we are only interested in the *rbf* function of the transformed graph).
- An edge $(\nu_{j,i}, \nu_{j,i+1}) \in \mathbb{E}'$ connects all the newly created nodes for every $i = 1, 2, ..., k_i - 1$.
- Each edge $(v_i, v_j) \in \mathbb{E}$ corresponds to an edge $(\nu_{i,k}, \nu_{j,1})$ in $\mathbb{E}'$ where $k = p(v_i, v_j)$.
- All edges in $\mathbb{E}'$ are labeled as 1.

We now prove the following property of the transformation.

[1]It is sufficient to transform it into a digraph with inter-release times equal to $gcd(p(e) : e \in \mathbb{E})$

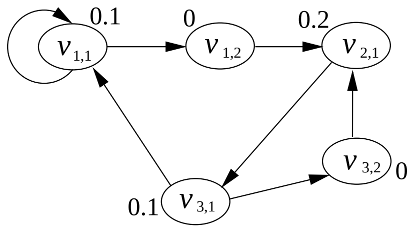

Figure 4. The unit digraph task transformed from Figure 2. The minimum inter-release times are all equal to 1. Deadlines are omitted.

Lemma 11: The request bound functions of a digraph task $\mathcal{D}(\tau)$ and the transformed UDRT $\mathcal{D}'(\tau')$ are the same.

Proof: Let $l = |\Delta|$ be the number of events in an arbitrary legal event sequence $\Delta = [(t_i, v_i)]$ in $\mathcal{D}(\tau)$. We construct an event sequence Δ' in $\mathcal{D}'(\tau')$ by replacing each event (t_i, v_i) except the last one in Δ with a sequence of events $(t_i + k - 1, \nu_{i,k})$ for $k = 1, ..., p(v_i, v_{i+1})$. The last event (t_l, v_l) in Δ is replaced with $(t_l, \nu_{l,1})$. Δ' is legal as the inter-release time between any two events is no smaller than 1. Thus $\tau.rbf(t) \leq \tau'.rbf(t)$. Similarly, we can prove $\tau'.rbf(t) \leq \tau.rbf(t)$. ■

In unit digraph task models, in a legal event sequence (and the corresponding path) containing n events (n nodes), the minimum separation time between the release times of the last and first events is $n-1$. Hence, **the length of the path in τ' represents the inter-release time of the first and last vertices for the corresponding urgent event sequence**.

Example: For the digraph task τ in Figure 2, the transformed unit digraph task τ' is shown in Figure 4. v_1 in τ is transformed to two vertices $\nu_{1,1}$ and $\nu_{1,2}$ in τ', v_2 is transformed to $\nu_{2,1}$, and v_3 to $\nu_{3,1}$ and $\nu_{3,2}$.

We extend the definition of *rbf* by adding the indication of the initial and final jobs of the path on which it is computed. Also, the definition of the refined *rbf* is slightly modified by removing the execution time (with its release time) of the last job. Both are necessary to simplify the formulation of *rbf* in the max-plus algebra, allowing a simple composition operation (as explained below).

Definition 13: Given a pair of vertices v_i and v_j, the **request bound function** of a digraph task τ in the time interval t, denoted as $\tau.rbf(v_i, v_j, t)$, is defined as the maximum sum of execution times of any legal event sequence $[(t_k, v_k), k = 1, ..., l]$ of τ such that

- the vertex corresponding to the first event is $v_1 = v_i$;
- the vertex corresponds to the last event is $v_l = v_j$;
- $t_{l-1} - t_1 \leq t$;
- $\tau.rbf(v_i, v_j, t) = \max \sum_{k=1}^{l-1} e(v_k)$.

If v_j is not reachable from v_i within any time interval of length t, then $\tau.rbf(v_i, v_j, t) = -\infty$. With this definition, the domain for the possible *rbf* values is $\mathbb{R}^* = \mathbb{R} \bigcup \{-\infty\}$.

According to the definition,

$$\tau.rbf(t) = \max_{i,j}\{\tau.rbf(v_i, v_j, t)\} \tag{13}$$

For a UDRT τ, $rbf(v_i, v_j, t)$ is additive, i.e. $\forall i, j, \forall t_1, t_2$,

$$\begin{aligned} & \tau.rbf(v_i, v_j, t_1 + t_2 + 1) \\ = \; & \max_m(\tau.rbf(v_i, v_m, t_1) + \tau.rbf(v_m, v_j, t_2)) \end{aligned} \tag{14}$$

Thus, $rbf(v_i, v_j, t_1 + t_2 + 1)$ of a UDRT τ for a long interval of length $t_1 + t_2 + 1$ can be computed from its values for shorter intervals of length t_1 and t_2. The need for this composition explains the omission of the execution time of the last job, and the sum of the time intervals is incremented by one when concatenating two paths, as the inter-release time of (v_{m-1}, v_m) is not included in the interval t_1 or t_2. Dynamic programming techniques can be used for an efficient calculation of $rbf(v_i, v_j, t)$. In addition, the computation can be represented as a matrix power sequence in max-plus algebra. Thus, we can leverage the related results to find its periodicity.

For a generic digraph task, Equation (14) does not apply since the inter-release times are not uniformly one. Also, the general $rbf(t)$ function (without constraints on the source and sink vertices) is not additive:

$$\begin{aligned} & \tau.rbf(t_1) + \tau.rbf(t_2) \\ = \; & \max_{i,k}\{\tau.rbf(v_i, v_k, t_1)\} + \max_{l,j}\{\tau.rbf(v_l, v_j, t_2)\} \\ = \; & \max_{i,j,k,l}(\tau.rbf(v_i, v_k, t_1) + \tau.rbf(v_l, v_j, t_2)) \\ \geq \; & \max_{i,j,k=l}(\tau.rbf(v_i, v_k, t_1) + \tau.rbf(v_l, v_j, t_2)) \\ = \; & \tau.rbf(t_1 + t_2 + 1) \end{aligned}$$

C. *Periodicity of* rbf

For a UDRT τ with n vertices, we define the **execution request matrix** as $\mathbf{A} \in \mathbb{R}^*(n, n)$, with elements $a_{i,j} = \tau.rbf(v_i, v_j, 0)$. We also denote $a_{i,j}^{(t+1)} = \tau.rbf(v_i, v_j, t)$. Equation (14) can be rewritten as (denoting $k = t_1 + 1$ and $l = t_2 + 1$, then $k + l = t_1 + t_2 + 2$)

$$\forall i, j, \; \forall k, l \quad a_{i,j}^{(k+l)} = \max_m(a_{i,m}^{(k)} + a_{m,j}^{(l)}) \tag{15}$$

This is exactly the definition of matrix power under the max-plus algebra. In other words, **the *rbf* function of a UDRT τ over a time interval of length t can be expressed as the $(t+1)$-th power of its execution request matrix $\mathbf{A}$**.

Example: Equation (7) computes the execution request matrix $\mathbf{A}$ for the UDRT $\mathcal{D}(\tau')$ in Figure 4. $\mathcal{D}(\tau')$ and $\mathcal{G}(\mathbf{A})$ (Figure 3) are similar, with corresponding vertices: $\nu_{1,1} \Leftrightarrow 1$, $\nu_{2,1} \Leftrightarrow 2$, $\nu_{3,1} \Leftrightarrow 3$, $\nu_{1,2} \Leftrightarrow 4$, and $\nu_{3,2} \Leftrightarrow 5$. Informally, we can derive $\mathcal{G}(\mathbf{A})$ from $\mathcal{D}(\tau')$ by:

- duplicating the topology of $\mathcal{D}(\tau')$;
- adding a self-loop with an edge of weight 0 for each vertex $\nu_{i,1}$. This is to allow the possibility of untight event sequences. Thus, $\mathbf{A}$ is *non-trivial*.
- assigning the edge (u, v) in $\mathcal{G}(\mathbf{A})$ with the same weight as the source node u in $\mathcal{D}(\tau')$.

We now prove the periodicity of *rbf* for a digraph task.

Theorem 12: The *rbf* function of a generic digraph task τ is almost linear periodic, i.e. there exist a real number q and a pair of integers r and p such that

$$\forall t > r,\ \tau.rbf(t+p) = \tau.rbf(t) + p \cdot q \qquad (16)$$

Proof: We consider the unit digraph task τ' obtained from τ and its execution request matrix $\mathbf{A}$. By Equation (13),

$$\tau.rbf(t) = \tau'.rbf(t) = \max_{i,j} a_{i,j}^{(t+1)} = a_{\max}^{(t+1)} \qquad (17)$$

i.e., $\tau.rbf(t)$ is the maximum element of the $(t+1)$-th power of its execution request matrix. The linear periodicity of the *rbf* function of τ follows immediately from Theorem 10. ■

D. Periodicity of dbf

The definition of *dbf* is also extended to include a restriction to a given pair of start and end vertices.

Definition 14: Given a pair of vertices v_i and v_j, the **demand bound function** of a digraph task τ during the time interval t, denoted as $\tau.dbf(v_i, v_j, t)$, is defined as the maximum sum of execution times of any legal event sequence $[(t_k, v_k), k = 1, ..., l]$ of τ such that

- the vertex corresponds to the first event is $v_1 = v_i$;
- the vertex corresponds to the last event is $v_l = v_j$;
- $\forall k = 1, ..., l, d_k + t_k - t_1 \le t$;
- $\tau.dbf_{i,j}(t) = \max \sum_{k=1}^{l} e(v_k)$.

Different from *rbf*, ***dbf*** **includes the execution time and deadline of the last job**, since it is not computed by composition, but starting from the corresponding *rbf*.

Theorem 13: The *dbf* function of a generic digraph task $\mathcal{D}(\tau) = (\mathbb{V}, \mathbb{E})$ is almost linear periodic, i.e. there exist a real number q and a pair of integers r and p such that

$$\forall t > r,\ \tau.dbf(t+p) = \tau.dbf(t) + p \cdot q \qquad (18)$$

Proof: By the **l-MAD** property, the last job in an event sequence always has the latest deadline, thus the execution times of all nodes in the sequence should be included in the *dbf*. For the consideration of $\tau.dbf(v_i, v_j, t)$, the second to last node $v_k : (v_k, v_j) \in \mathbb{E}$ has a release time of $t - d(v_j) - p(v_k, v_j)$. For sufficiently large t, it is

$$\begin{aligned} & \tau.dbf(v_i, v_j, t) \\ = & \max_{v_k:(v_k,v_j)\in\mathbb{E}} \{\tau.rbf(v_i, v_j, t - d(v_j) - p(v_k, v_j))\} + e(v_j) \\ = & \ \tau.rbf(v_i, v_j, t - d(v_j) - \min_{v_k:(v_k,v_j)\in\mathbb{E}} p(v_k, v_j)) + e(v_j) \end{aligned} \qquad (19)$$

Thus, $\tau.dbf(v_i, v_j, t)$ can be computed by simply shifting $\tau.rbf(v_i, v_j, t)$: to the right by $d(v_j) + \min_{v_k:(v_k,v_j)\in\mathbb{E}} p(v_k, v_j)$, and up by $e(v_j)$. This of course does not affect its periodicity. Thus $\tau.dbf(t)$ is almost linear periodic with the same linear factor and period as the corresponding *rbf*. ■

Example: For the digraph task τ in Figure 2, we have $\tau.dbf(v_1, v_1, t) = \tau.rbf(v_1, v_1, t - d(v_1) - \min\{p(v_3, v_1), p_(v_2, v_1)\}) + e(v_1) = \tau.rbf(v_1, v_1, t-2) + 0.1$. From Equation (8), $\forall i \ge 0, \tau.rbf(v_1, v_1, 5+i) = a_{1,1}^{(6+i)} = 0.6 + 0.1 \times i$. Thus $\forall t \ge 7, \tau.dbf(v_1, v_1, t) = 0.1 \times t$. Similarly, we can derive the *dbf* functions for other pairs of source and sink vertices,

$$\forall t \ge 8,\ dbf(t) = \begin{bmatrix} 0 & -0.1 & 0 \\ 0.1 & 0 & 0.1 \\ 0 & -0.1 & 0 \end{bmatrix} + 0.1 \times t \qquad (20)$$

and the *dbf* function is $\forall t \ge 8, \tau.dbf(t) = 0.1 + 0.1 \times t$.

Although we proved the linear periodicity of the *rbf* and *dbf* functions for any digraph task, to the best of our knowledge, there is no general procedure for computing the exact value (or an upper bound) of the general defect if the execution request matrix is reducible (the digraph task is not strongly connected).

VI. Computing the Periodicity Parameters for Strongly Connected Digraphs

In this section, we outline efficient algorithms to compute the periodicity parameters for *strongly connected task digraphs*. In such digraph tasks, the sequence of each element in the irreducible execution request matrix $\mathbf{A}$ is almost linear periodic with period $p = lcm\{\text{hper}(\mathcal{K}) : \mathcal{K} \in HCC^*(\mathcal{G}(\mathbf{A}))\}$ and factor $q = \lambda(\mathbf{A})$ (Theorem 4). By Lemma 7, the sequence of the largest element is linear periodic with the same period and factor. The linear factor q (the maximum cycle mean in $\mathcal{G}(\mathbf{A})$) can be computed using the algorithm in [15], with complexity $O(|\mathbb{V}'||\mathbb{E}'|)$, where $|\mathbb{V}'|$ and $|\mathbb{E}'|$ are the number of vertices and edges in the transformed UDRT respectively. p is computed using the algorithm in Theorem 3.6 of [11]. The dominating step in complexity is the metric matrix (the matrix of all-pairs longest paths) of $\mathbf{A} - \lambda(\mathbf{A})$, with complexity $O(|\mathbb{V}'|^3)$.

Since $a_{\max}^{(t)} = \mathbf{0}^T \otimes \mathbf{A} \otimes \mathbf{0}$ (where $\mathbf{0} = (0, 0, ..., 0)^T$), the defect of the linear dynamical system $\mathbf{A}^* \otimes \mathbf{0}$ provides an upper bound to the defect of $a_{\max}^*$ [13]. The linear defect of a linear dynamical system can be computed with complexity $O(|\mathbb{V}'|^3(\log r + \log p))$ [13] (where r and p are its linear defect and period). This is done by iteratively computing the matrix power and finding the first r that satisfies $\mathbf{A}^{(r+p)} \otimes \mathbf{0} = \mathbf{A}^{(r)} \otimes \mathbf{0} + p \times q$. Upper bounds on r can be calculated more efficiently ([13] [7]) with complexity $O(|\mathbb{V}'|^2)$ on top of the algorithms to compute p and q.

In practice, the transformed UDRT can have a much larger size than the original digraph task, and the computation of the periodicity parameters (p, q, and r) might be quite expensive. For a possibly more efficient implementation, we consider algorithms that operate on the original digraph task (or a more compact transformation than the UDRT):

- Because of the correspondence between the paths in the digraph task and its UDRT, $\lambda(\mathbf{A})$ is the utilization (the asymptotic maximum value of the ratio execution request/demand rate) of the digraph task. This is the maximum cost to time ratio [8], which can be computed with complexity $O(|\mathbb{V}|^2|\mathbb{E}|)$, where $|\mathbb{V}|$ and $|\mathbb{E}|$ are the number of nodes and edges in the original digraph task.

- The computation of the metric matrix for $\mathbf{A}-\lambda(\mathbf{A})$ can be performed in $O(|\mathbb{E}|^3)$ time, where $|\mathbb{E}|$ is the number of edges in the original graph. This is because in the UDRT, every path to $\nu_{i,k}$ $(k > 1)$ goes through $\nu_{i,1}$.
- The linear dynamical system $\mathbf{A}^{(t)} \otimes \mathbf{0} = (a_j^{(t)} = \max_i rbf(v_i, v_j, t-1))$ is the vector of the *rbf* functions ending in v_j. Such *rbf* functions can be computed using the same algorithm proposed in [23], and the linear defect is estimated with complexity $O((|\mathbb{V}| + |\mathbb{E}|) \cdot r)$.

These algorithms avoid the need to construct and operate on the possibly large UDRT graph, and allow to compute (often more efficiently) the linear periodicity parameters on the original graph. In our experiments, we use them for the comparison with the analysis in [23]. We leave for future work the study on the best combination of these algorithms with the ones operating on the UDRT ([11] [13] [7]).

VII. An Improved Upper Bound on t_f

We now derive a tight linear upper bound on *rbf* and *dbf*, and consequently, a bound for t_f, the time limit for checking the feasibility conditions in (1) and (2).

We use the previously developed task transformation and the power sequence on its execution request matrix. For the digraph task τ, the graph of its execution request matrix is $\mathcal{G}(\mathbf{A}) = (\mathbb{V}, \mathbb{E}, w)$. We consider $\mathcal{G}'(\mathbf{A}) = \mathcal{G}(\mathbf{A}) - \lambda(\mathbf{A})$, i.e. $\mathcal{G}'(\mathbf{A}) = (\mathbb{V}, \mathbb{E}, w')$ where $w'(e) = w(e) - \lambda(\mathbf{A})$ for all $e \in \mathbb{E}$. $\mathcal{G}'(\mathbf{A})$ shares the same topology as $\mathcal{G}(\mathbf{A})$, thus for any path Π in $\mathcal{G}(\mathbf{A})$, there exists the same path in $\mathcal{G}'(\mathbf{A})$ but with weight

$$w'(\Pi) = w(\Pi) - |\Pi| \cdot \lambda(\mathbf{A}) \tag{21}$$

Hence the maximum cycle mean of $\mathcal{G}'(\mathbf{A})$ is zero, and the longest path in $\mathcal{G}'(\mathbf{A})$ is well defined.

We now derive a linear upper bound on the *rbf* function.

Theorem 14: The linear upper bounds on *rbf* and *dbf* are

$$\forall t \geq 0, \begin{cases} \tau.rbf(t) \leq W + \lambda(\mathbf{A}) + t \cdot \lambda(\mathbf{A}) \\ \tau.dbf(t) \leq X + t \cdot \lambda(\mathbf{A}) \end{cases} \tag{22}$$

where $W = \max\{w'(\Pi) | \Pi \text{ is a path in } \mathcal{G}'(\mathbf{A})\}$, $X = W + \lambda(\mathbf{A}) + \min\{\max_{(v_k,v_j)\in\mathcal{D}(\tau)}\{e(v_j) - (d(v_j) + p(v_k, v_j)) \cdot \lambda(\mathbf{A})\}, -\lambda(\mathbf{A}) \cdot d_{\min}\}$, and $d_{\min} = \min_{v_j\in\mathcal{D}(\tau)} d(v_j)$.

Proof: Combining (21) with (17) and (10), we have

$$\begin{aligned} & \tau.rbf(t) \\ = \; & \max\{w(\Pi) | \Pi \text{ is a path in } \mathcal{G}(\mathbf{A}), |\Pi| = t+1\} \\ \leq \; & \max\{w'(\Pi) + (t+1)\lambda(\mathbf{A}) | \Pi \text{ is a path in } \mathcal{G}'(\mathbf{A})\} \\ = \; & W + \lambda(\mathbf{A}) + t \cdot \lambda(\mathbf{A}) \end{aligned}$$

The bound on *dbf* is derived by combining the bound on *rbf* with (19) and the fact that $dbf(t) \leq rbf(t - d_{\min})$. ■

W can be computed in $O(|\mathbb{V}|^3)$ time ($\mathbb{V}$ is the number of nodes in $\mathcal{G}'(\mathbf{A})$) using the Floyd-Warshall algorithm [10].

The bound in [23] is $\tau.dbf(t) \leq S + t \cdot \lambda(\mathbf{A})$ where $S = \sum_{v_i\in\mathcal{D}(\tau)} e(v_i)$ is the sum of the WCETs of the vertexes in $\mathcal{D}(\tau)$. **Our linear bound is always tighter than [23].** Informally, as there is no positive cycle in $\mathcal{G}'(\mathbf{A})$, there must exist a longest path in $\mathcal{G}'(\mathbf{A})$ which is elementary. Thus, W is no greater than the maximum sum of the positive edge weights in any elementary path (where each node will appear at most once). Also, the task transformation guarantees that the edge weight in $\mathcal{G}(\mathbf{A})$ equals the WCET of its source node in $\mathcal{D}(\tau)$. Hence, $W \leq \sum_{v_i\in\mathcal{D}(\tau)} \max\{e(v_i) - \lambda(\mathbf{A}), 0\}$. This implies $W \leq S - \lambda(\mathbf{A})$, and $X \leq W + \lambda(\mathbf{A}) \leq S$.

Example: For the example in Figure 2, the linear bound on the *rbf* is $0.2 + 0.1 \times t$. For *dbf*, it is $0.1 + 0.1 \times t$. As a comparison, the bound on *dbf* from [23] is $0.4 + 0.1 \times t$.

With the bounds on *rbf* and *dbf* in (22), we can compute the maximum length t_f of the time interval to be checked by using the results in [5], where the utilization of task τ_i is the maximum cycle mean $\lambda_i(\mathbf{A})$ in its digraph. For task τ_j in system Γ with static priority, if the total utilization is less than 1, i.e. $\sum_{\tau_i\in\Gamma} \lambda_i(\mathbf{A}) < 1$, then

$$t_f < \frac{X_j + \sum_{\tau_i\in hp(j)}(W_i + \lambda_i(\mathbf{A}))}{1 - \lambda_j(\mathbf{A}) - \sum_{\tau_i\in hp(j)} \lambda_i(\mathbf{A})} \tag{23}$$

For systems with dynamic priority (EDF), it is

$$t_f < \frac{\sum_{\tau_i\in\Gamma} X_i}{1 - \sum_{\tau_i\in\Gamma} \lambda_i(\mathbf{A})} \tag{24}$$

VIII. Experimental Results

In this section, we evaluate the improvements on the efficiency of schedulability analysis compared to the previous method in [23]. We generate applications consisting of random sets with $n = 5$ to 40 tasks. Each task is a random graph with 1 to 15 nodes. For each n, 1000 sets are generated and then examined for schedulability. The average (in- and out-) degree of the nodes is 3.5, and 77.4% of the task graphs are strongly connected. The base period of each task is generated by the product of one to three factors, each randomly drawn from the harmonic sets (2, 4), (6, 12), (5, 10). The inter-release time is scaled by a factor randomly extracted from the set $\{1, 2, 4, 5, 10\}$. The deadline of a node is the minimum inter-release time among those of its out-going edges, thereby enforcing the frame separation property. The execution times of the task are selected such that its utilization is uniformly distributed. For comparison, we assume the system is scheduled using EDF, the only policy considered in [23].

We compare the performance of our algorithms, applied to the original digraph tasks, with the analysis in [23]. Figure 5 shows the improvement for systems with 20 tasks on t_f, the runtime for calculating the *dbf* function, and the total runtime. It also plots the percentage of strongly connected digraph tasks with $r < t_f$ where t_f is the improved bound computed in Equation (24), i.e., the tasks that we explore the linear periodicity to calculate the *dbf* function.

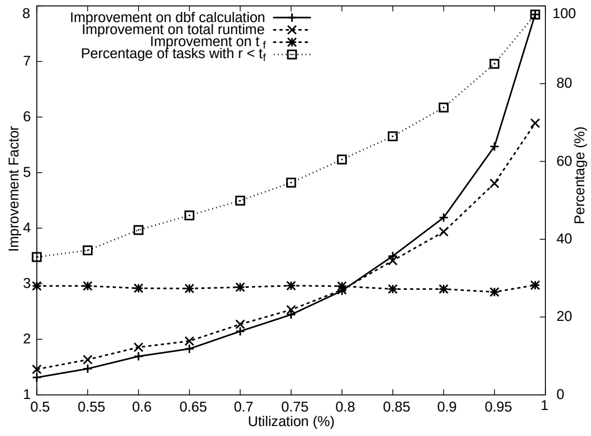

Figure 5. The improvement compared to the analysis in [23] and the percentage of tasks with $r < t_f$ (number of tasks $n = 20$).

As shown in Figure 5, the computed t_f is 3 times smaller for all utilizations (50% to 99%). The speedup on the total runtime and the portion for the calculation of *dbf* function is a factor of around 1.8 at 60% utilization and then increases even further, leveraging the improvements in the computed t_f value. For a very high (99%) utilization, nearly 97% of the strongly connected digraph tasks have a linear defect smaller than t_f. Figure 6 plots the improvement on the total runtime vs. n, the number of tasks in the system. The speedup is almost independent from n, since t_f is uniformly reduced about 3 times for all tasks, and the linear periodicity of *rbf* and *dbf* only depends on the graph structure.

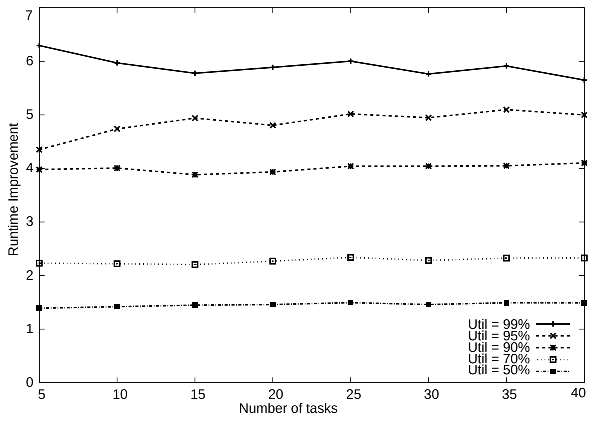

Figure 6. The runtime improvement vs. the number of tasks in the system.

IX. Conclusion

In this paper, by leveraging results from max-plus algebra, we extend the concept of periodicity of execution requests to task models without cyclic recurrent behavior, including digraph task model and its extension. This essentially makes the asymptotic complexity of calculating $rbf(t)$ and $dbf(t)$ independent from the time interval t. We also provide polynomial time algorithms for the computation of the linear periodicity parameters for strongly connected graphs. Experimental results demonstrate that such a property can be used to improve the efficiency of schedulability analysis for real-time systems captured by these task models.

References

[1] M. Anand, A. Easwaran, S. Fischmeister, and I. Lee, "Compositional feasibility analysis of conditional real-time task models," *Proc. Symposium on Object Oriented Real-Time Distributed Computing*, 2008.

[2] F. Baccelli, G. Cohen, G. Olsder, and J. Quadrat, "Synchronization and Linearity: An Algebra for Discrete Event Systems," *Wiley*, 1992.

[3] S. Baruah, "Feasibility analysis of recurring branching tasks," *Proc. 10th Euromicro Workshop on Real-Time Systems*, 1998.

[4] S. Baruah, D. Chen, S. Gorinsky, and A. Mok, "Generalized multiframe tasks," *Real-Time Syst.*, 17(1):5–22, 1999.

[5] S. Baruah, "Dynamic- and static-priority scheduling of recurring real-time tasks," *Real-Time Syst.*, 24(1):93–128, 2003.

[6] S. Baruah, "The non-cyclic recurring real-time task model," *Proc. 31st IEEE Real-Time Systems Symposium*, 2010.

[7] B. Charron-Bost, M. Függer, and T. Nowak, "On the Transience of Linear Max-Plus Dynamical Systems," *Computing Research Repository (CoRR) abs/1111.4600*, 2011.

[8] A. Dasdan, S. S. Irani, and R. K. Gupta. "Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems," *Proc. Design Automation Conference*, 1999.

[9] E. Fersman, P. Krcal, P. Pettersson, and W. Yi, "Task automata: Schedulability, decidability and undecidability," *Information and Computation*, 205(8):1149–1172, August 2007.

[10] R. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, 5(6):345, June 1962.

[11] M. Gavalec, "Linear matrix period in max-plus algebra," *Linear Algebra and its Applications*, 307(1-3):167–182, 2000.

[12] M. Gavalec, "Polynomial algorithm for linear matrix period in max-plus algebra," *Central Europ. J. Oper. Res.*, 8:247–258, 2000.

[13] M. Hartmann and C. Arguelles, "Transience bounds for long walks," *Mathematics of Operations Research*, 24:414–439, May 1999.

[14] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System Level Performance Analysis - the SymTA/S Approach," *IEE Proc. Computers and Digital Techniques*, 152(2):148–166, 2005.

[15] R. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Math*, 23(3):309–311, 1978.

[16] S. Künzli, A. Hamann, R. Ernst, and L. Thiele, "Combined Approach to System Level Performance Analysis of Embedded Systems," *Proc. Conference on HW/SW Codesign and System Synthesis*, 2007.

[17] C. L. Liu and J. W. Layland, "Scheduling algorithms for multi-programming in a hard-real-time environment," *J. ACM*, 20:46–61, January 1973.

[18] C. Norström, A. Wall, and W. Yi, "Timed automata as task models for event-driven systems," *Proc. 6th Conference on Real-Time Computing Systems and Applications*, 1999.

[19] A. K. Mok and D. Chen, "A multiframe model for real-time tasks," *Proc. 17th IEEE Real-Time Systems Symposium*, 1996.

[20] M. Molnárová, "Computational complexity of nachtigall's representation," *Optimization*, 52:93–104, 2003.

[21] M. Molnárová, "Generalized matrix period in max-plus algebra," *Linear Algebra and its Applications*, 404:345–366, 2005.

[22] N. T. Moyo, E. Nicollet, F. Lafaye, and C. Moy, "On schedulability analysis of non-cyclic generalized multiframe tasks," *Proc. 22nd Euromicro Conference on Real-Time Systems*, 2010.

[23] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," *Proc. 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2011.

[24] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "On the tractability of digraph-based task models," *Proc. 23rd Euromicro Conference on Real-Time Systems*, 2011.

[25] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," *Proc. IEEE International Symposium on Circuits and Systems*, 2000.

[26] E. Wandeler and L. Thiele, "Real-Time Calculus (RTC) Toolbox," *http://www.mpa.ethz.ch/Rtctoolbox*, 2006.

[27] H. Zeng and M. Di Natale, "Schedulability Analysis of Periodic Tasks Implementing Synchronous Finite State Machines," *Proc. 23rd Euromicro Conference on Real-Time Systems*, 2012.