Transferring the Progress Control Policy for a Class of Discrete Event Systems

Hiroyuki Goto
Department of Management and Information Systems Science
Nagaoka University of Technology
Nagaoka, Niigata, 940-2188 Japan
e-mail: hgoto@kjs.nagaokaut.ac.jp

Takakazu Tsubokawa
Faculty of Management and Information Systems Engineering
Nagaoka University of Technology
Nagaoka, Niigata, 940-2188 Japan
e-mail: s073368@ics.nagaokaut.ac.jp

Abstract—**This research develops an intelligent method of transferring progress control policies for a class of repetitive discrete event systems, whose approach is based on Dioid algebra. The commonly used policies include a method that requires precedence constraints regarding processing order relations, and a method where relative processing start and completion times of all facilities need specified. However, in the conventional methods, these frameworks for handling the state equation differ significantly and not been unified yet. Hence, this research proposes a method of transferring the transition matrices, and accomplishes a method of transferring the progress control policy per job.**

Keywords-**Dioid algebra; adjacency matrix; state-space representation; directed asyclic graph; progress control;**

I. INTRODUCTION

This research focuses on scheduling problems for a class of discrete event systems, and develops a method for progress control of jobs. The primary formulation is based on Dioid algebra [1], [2], a class of discrete mathematics. In the target systems, the installed facilities are repetitively used, the number of maximum jobs that can be processed simultaneously is one or a finite value, and no concurrency constraints are imposed in the facilities. The behavior of these systems can be described by simple linear equations in Dioid algebra. Specifically, using max-plus algebra [2], [3], on which the max and '+' operations are defined as addition and multiplication, the earliest times of event propagation can be represented by a form similar to the state space representation in modern control theory.

In the well-known simple state equation, we assume that all jobs use all the installed facilities, where occupation times in each facility are constant, independent of the job number. These assumptions are somewhat restrictive, with the result that several extensions are needed for application to practical systems. For instance, we sometimes have to address systems with in which, (i) the occupation times in facilities and (ii) the facilities used, differ for each job. With respect to issue (i), [4] and [5] extend the conventional state equation to 'event-varying' system types, using the analogy of time-varying systems in modern control theory. Regarding issue (ii), [6] and [7] adopt the concept of a 'heap model', assimilating an appearance of a Gantt chart expressing a job's schedule.

In executing a single job, the following two methods are commonly used for progress control.

- (A) Precedence constraints and occupation times in the respective facilities are specified, and the processing start and completion times are variable as long as the precedence constraints are adhered to.
- (B) From commencement to completion of a job, all processing start and completion times for using facilities are specified in terms of relative time.

Hereafter, these are referred to as policies (A) and (B), respectively. In policy (A), processing is supposed to start as soon as all required materials are supplied and the relative facility is ready. This policy is suitable for systems where 'bulk processing' is desirable. In contrast, policy (B) is suitable if we wish to finish the job after a certain predetermined time has lapsed.

The difference between these two policies appears in the transition matrices of the state equation. In [4], [5] and [8], policy (A) is adopted to handle systems of event-varying types in which the occupation times in facilities differ by job. However, it is assumed that all jobs use all installed facilities, which means that policy (B) cannot be implemented. By contrast, [6] and [7] address issue (ii) and policy (B), but they can not create schedules based on earliest time. For application to a wider class of practical systems, it is desirable to consider both issues (i) and (ii), and to transfer or mix policies (A) and (B).

This research therefore, proposes a method for calculating the transition matrices for both policies by using common parameter vectors and matrices.

II. PRELIMINARIES

Denoting the real field by $\boldsymbol{R}$, we define a field $\mathcal{D} = \boldsymbol{R} \cup \{-\infty\}$. For $x, y \in \mathcal{D}$, we define the following operators: $x \oplus y = \max(x, y)$, $x \otimes y = x + y$ and $x^{\otimes y} = x \cdot y$. Unit elements for operators $\oplus$ and $\otimes$ are denoted by ε $(= -\infty)$ and $e\,(= 0)$, respectively. If $q \le r$, $\oplus_{l=q}^{r} x_l = \max(x_q, x_{q+1}, \cdots, x_r)$. For matrices, if $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{D}^{m\times n}$ and $\boldsymbol{Z} \in \mathcal{D}^{n\times p}$, $[\boldsymbol{X} \oplus \boldsymbol{Y}]_{ij} = \max([\boldsymbol{X}]_{ij}, [\boldsymbol{Y}]_{ij})$, $[\boldsymbol{X} \otimes \boldsymbol{Z}]_{ij} = \oplus_{l=1}^{n}([\boldsymbol{X}]_{il} + [\boldsymbol{Z}]_{lj})$. We denote the unit matrices for operators $\oplus$ and $\otimes$ by $\boldsymbol{\varepsilon}$ and $\boldsymbol{e}$, respectively. $\boldsymbol{\varepsilon}$ is a matrix in which all elements are ε, and $\boldsymbol{e}$ a matrix in which the diagonal elements are e and off-diagonal elements are ε. Operator $\otimes$ is higher precedence than $\oplus$, and $\otimes$ is abbreviated when no confusion is likely to arise. The Kleene star operator [9] is defined in $\boldsymbol{X} \in \mathcal{D}^{n\times n}$, as follows: $\boldsymbol{X}^{*} =$

$\oplus_{l=0}^{\infty} \boldsymbol{X}^{l} = \boldsymbol{e} \oplus \boldsymbol{X} \oplus \boldsymbol{X}^{2} \oplus \cdots$. If matrix $\boldsymbol{X}$ is a representation matrix of a weighted DAG (Directed Acyclic Graph), there is an instance of s that satisfies $\boldsymbol{X}^{s-1} \neq \boldsymbol{\varepsilon}$, $\boldsymbol{X}^{s} = \boldsymbol{\varepsilon}$ $(1 \leq s \leq n)$, and the operator can be calculated by adding a finite set of powers of $\boldsymbol{X}$: $\boldsymbol{X}^{*} = \boldsymbol{e} \oplus \boldsymbol{X} \oplus \cdots \oplus \boldsymbol{X}^{s-1}$. Moreover, in a field $\overline{\mathcal{D}} = \underline{\boldsymbol{R}} \cup \{\pm\infty\}$, the following operators are defined. If $x, y \in \mathcal{D}$, $x \wedge y = \min(x, y)$. If $m \leq n$, $\wedge_{l=q}^{r} x_l = \min(x_q, x_{q+1}, \cdots, x_r)$. For matrices $\boldsymbol{X} \in \overline{\mathcal{D}}^{m\times n}$, $\boldsymbol{Z} \in \overline{\mathcal{D}}^{n\times l}$, $[\boldsymbol{X} \odot \boldsymbol{Z}]_{ij} = \wedge_{l=1}^{n} (-[\boldsymbol{X}]_{il} + [\boldsymbol{Z}]_{lj})$, $[\boldsymbol{X}]_{ij}$ = { $-[\boldsymbol{X}]_{ji}$ if $[\boldsymbol{X}]_{ji} \neq \varepsilon$, ε if $[\boldsymbol{X}]_{ji} = \varepsilon$ }.

A. State equation for systems with precedence constraints

We briefly review the method of [4] that adopts policy (A), and implement several extensions. Let the number of installed facilities be n. Assume that each job uses all installed facilities exactly once, and the precedence constraints are represented by a DAG. For the k th job, we denote the occupation times in the facilities by $\boldsymbol{d}(k) \in \mathcal{D}^{n}$, and the list of preceding facilities of facility i $(1 \leq i \leq n)$ by $\mathcal{P}_i(k)$. Moreover, we introduce the following two parameter matrices: $\boldsymbol{P}_k = \mathrm{diag}[\boldsymbol{d}(k)]$,

$$[\boldsymbol{F}_k]_{ij} = \{ e : \text{if } j \in \mathcal{P}_i(k),\ \varepsilon : \text{if } j \notin \mathcal{P}_i(k) \}, \tag{1}$$

where $\boldsymbol{F}_k$ is referred to as the adjacency matrix. Denoting the earliest processing completion times by $\boldsymbol{x}_E(k)$ and the minimum value for the processing completion times for job k by $\boldsymbol{u}(k) \in \mathcal{D}^{n}$, $[\boldsymbol{x}_E(k)]_i$ can be represented as:

$$\begin{aligned}[\boldsymbol{x}_E(k)]_i - [\boldsymbol{d}(k)]_i &= \bigoplus_{j \in \mathcal{P}_i(k)} [\boldsymbol{x}_E(k)]_j \oplus [\boldsymbol{x}(k-1)]_i \oplus [\boldsymbol{u}(k)]_i \\ &= [\boldsymbol{F}_k \boldsymbol{x}_E(k)]_i \oplus [\boldsymbol{x}(k-1)]_i \oplus [\boldsymbol{u}(k)]_i .\end{aligned} \tag{2}$$

The first term of the right-hand side gives the maximum value for the processing completion times in the preceding facilities, the second term indicates no concurrency with respect to the previous job, whilst the third term expresses the minimum time at which processing can start. Now transfer $[\boldsymbol{d}(k)]_i$ in the left-hand side of (2) to the right-hand side, and express it as $\boldsymbol{P}_k$. Since this relationship is true for all i $(1 \leq i \leq n)$, it can be summarized as:

$$\boldsymbol{x}_E(k) = \boldsymbol{P}_k \boldsymbol{F}_k \boldsymbol{x}_E(k) \oplus \boldsymbol{P}_k [\boldsymbol{x}(k-1) \oplus \boldsymbol{u}(k)]. \tag{3}$$

By substituting the entire right-hand side of (3) for the first term, and repeating this, we obtain:

$$\boldsymbol{x}_E(k) = \boldsymbol{A}_k [\boldsymbol{x}(k-1) \oplus \boldsymbol{u}(k)], \tag{4}$$

where $\boldsymbol{A}_k = (\boldsymbol{P}_k \boldsymbol{F}_k)^{*} \boldsymbol{P}_k$. Recall here that there is an instance s $(1 \leq s \leq n)$ that satisfies $(\boldsymbol{P}_k \boldsymbol{F}_k)^{s} = \boldsymbol{\varepsilon}$ if $\boldsymbol{P}_k \boldsymbol{F}_k$ is a representation matrix of the DAG. It should be noted that [4] uses the input matrix $\boldsymbol{B}_0$ and input variables $\boldsymbol{u}(k)$ for stating the earliest processing start times, which is applicable only in cases where the input locations are fixed and independent of the job number. This research, on the other hand, aims to handle more general cases in which the set of facilities used differs per job. Hence, we use $\boldsymbol{u}(k)$ only, without the input matrix.

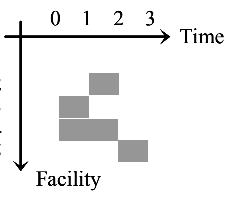

Figure 1. Ganntt chart of a schedule, given by a heap of pieces.

B. State equation based on a heap model

If the list of facilities used differs per job and the processing start times for each facility are supplied as relative values, the state equation based on the heap model can be utilized. Here, we briefly review the method in [6] that adopts policy (B), and make several extensions.

Suppose the number of facilities is n, and the list of facilities used by job k is $\mathcal{R}(k)$. If the job uses facility i $(1 \leq i \leq n)$, we express this as $i \in \mathcal{R}(k)$, and as $i \notin \mathcal{R}(k)$ otherwise. Set the time $t = t_0$ as the base time, and denote the relative processing start and completion times by $\boldsymbol{a}(k) \in \mathcal{D}^{n}$ and $\boldsymbol{b}(k) \in \mathcal{D}^{n}$, respectively. Then, the following properties hold; if $i \notin \mathcal{R}(k)$, then $[\boldsymbol{a}(k)]_i = [\boldsymbol{b}(k)]_i = \varepsilon$, if $i \in \mathcal{R}(k)$, then $[\boldsymbol{a}(k)]_i \neq \varepsilon$, $[\boldsymbol{b}(k)]_i \neq \varepsilon$, $[\boldsymbol{b}(k)]_i - [\boldsymbol{a}(k)]_i \geq 0$. Normally, the base time is set as $t_0 = \bigwedge_{i \in \mathcal{R}(k)} [\boldsymbol{a}(k)]_i$. For example, if the schedule for job k is given in Fig. 1, $\boldsymbol{a}(k) = [\varepsilon\ 1\ e\ e\ 2]^{T}$, $\boldsymbol{b}(k) = [\varepsilon\ 2\ 1\ 2\ 3]^{T}$. Next, consider matrix $\boldsymbol{M}_k \in \mathcal{D}^{n\times n}$ which has the following properties:

$$[\boldsymbol{M}_k]_{ij} = \begin{cases} [\boldsymbol{b}(k)]_i - [\boldsymbol{a}(k)]_j : \text{if } i \in \mathcal{R}(k) \text{ and } j \in \mathcal{R}(k), \\ e : \quad \text{if } i = j \text{ and } j \notin \mathcal{R}(k), \\ \varepsilon : \quad \text{otherwise.} \end{cases} \tag{5}$$

This matrix can also be expressed as:

$$\boldsymbol{M}_k = \boldsymbol{e} \oplus \boldsymbol{b}(k) \otimes \overline{\boldsymbol{a}(k)}. \tag{6}$$

Denoting the absolute values of the earliest processing completion times of job k by $\boldsymbol{x}_E(k)$,

$$\begin{aligned}[\boldsymbol{x}_E(k)]_i &= \bigoplus_{j \in \mathcal{R}(k)} \{[\boldsymbol{x}(k-1)]_j + [\boldsymbol{b}(k)]_i - [\boldsymbol{a}(k)]_j\} \oplus [\boldsymbol{x}(k-1)]_i \\ &= \bigoplus_{j=1}^{n} [\boldsymbol{M}_k]_{ij} \otimes [\boldsymbol{x}(k-1)]_j = [\boldsymbol{M}_k \boldsymbol{x}(k-1)]_i .\end{aligned}$$

Since this is true for all i $(1 \leq i \leq n)$, if follows that:

$$\boldsymbol{x}_E(k) = \boldsymbol{M}_k \boldsymbol{x}(k-1). \tag{7}$$

Equation (7) represents the earliest processing times for job k considering no-concurrency of facilities with job $(k-1)$.

Next, consider imposing precedence constraints on the processing start time of job k. Assume that processing cannot begin until $\boldsymbol{u}(k) \in \mathcal{D}^{n}$ for various operational or set-

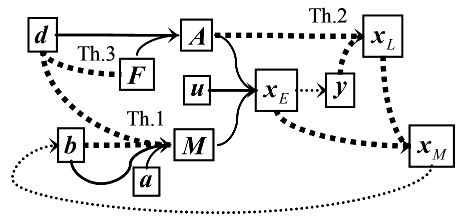

Figure 2. Relationships between the relevant variables, vectors and matrices.

up reasons. For facilities that satisfy $i \notin \mathfrak{R}(k)$, or if there is no constraint on the processing start time, we set $[\boldsymbol{u}(k)]_i = \varepsilon$. According to these assumptions, (7) is extended as:

$$\boldsymbol{x}(k) = \boldsymbol{M}_k[\boldsymbol{x}(k-1) \oplus \boldsymbol{u}(k)], \tag{8}$$

where $\boldsymbol{u}(k)$ is also interpreted as the input variables.

III. Proposed Method

Although policies (A) and (B) differ significantly from each other, the only difference in the state equation is in the transition matrices $\boldsymbol{M}_k$ and $\boldsymbol{A}_k$. This means that we can transfer policies per job by transferring only the transition matrix. Once a method of calculating both transition matrices from common parameters has been developed, such transfer can be accomplished easily. Therefore, this section investigates methods for calculating the transition matrices and converting these matrices to one another, taking the following into consideration:

- Use of facilities: Each job uses all installed facilities, or the list of facilities used differs per job.
- Scheduling policy: Precedence constraints are imposed, or the relative start or completion times are supplied.

A. Scope of this research

In Fig. 2, we depict the relationships that are clarified by this research. For simplicity, suffixes for vectors and matrices (k), $*_k$ are omitted. $\boldsymbol{y}(k) \in \mathfrak{D}^n$ is used to supply the scheduled output times, also handled as the input variables. $\boldsymbol{x}_L(k) \in \mathfrak{D}^n$ returns the latest processing ‘start’ times, and $\boldsymbol{x}_M(k) \in \mathfrak{D}^n$ is expected to supply the processing ‘completion’ times as required parameters. Each arrow depicts that the downstream parameters and variables are obtained by using those upstream. The thick dashed arrows represent the relationships that are examined in this research. For the thin dotted arrows, we assume that both the upstream and downstream values are equivalent. ’Th.’ Denotes a theorem that is examined in a subsequent subsection. For example, the transition matrix $\boldsymbol{A}_k$ is obtained from $\boldsymbol{d}(k)$ and $\boldsymbol{F}_k$, and $\boldsymbol{M}_k$ from $\boldsymbol{a}(k)$ and $\boldsymbol{b}(k)$.

B. Transition matrix for cases with fixed relative times

The transition matrix $\boldsymbol{M}_k$ in (6) is a function of the processing start and completion times of job k, $\boldsymbol{a}(k)$ and $\boldsymbol{b}(k)$, respectively. On the other hand, occupation times $\boldsymbol{d}(k)$ in facilities are not explicitly included in the representation. This means we should supply parameters or matrices other than those for the transition matrix $\boldsymbol{A}_k$. This is not suitable for practical operations. Accordingly, we develop a method for deriving the transition matrix $\boldsymbol{M}_k$ from $\boldsymbol{b}(k)$ and $\boldsymbol{d}(k)$, by which both transition matrices can be calculated. Now, consider a matrix $\boldsymbol{Q}_k \in \mathfrak{D}^{n\times n}$ that satisfies the following:

$$[\boldsymbol{Q}_k]_{ij} = \begin{cases} [\boldsymbol{b}(k)]_i - [\boldsymbol{a}(k)]_j & : \text{if } i \in \mathfrak{R}(k) \text{ and } j \in \mathfrak{R}(k) \\ \varepsilon & : \text{ otherwise.} \end{cases} \tag{9}$$

Noting that $[\boldsymbol{b}(k)]_i - [\boldsymbol{a}(k)]_j \geq 0$ holds true if $j \in \mathfrak{R}(k)$, the transition matrix $\boldsymbol{M}_k$ in (5) can be rewritten as $\boldsymbol{M}_k = \boldsymbol{e} \oplus \boldsymbol{Q}_k$.

Next, consider representing $\boldsymbol{Q}_k$ with only $\boldsymbol{b}(k)$ and $\boldsymbol{P}_k$. This is given by a theorem below.

Theorem 1: $\boldsymbol{Q}_k \in \mathfrak{D}^{n\times n}$ is computed using the following relationship:

$$\boldsymbol{Q}_k = \boldsymbol{b}(k) \otimes \overline{\boldsymbol{b}(k)} \otimes \boldsymbol{P}_k . \tag{10}$$

Proof. For simplicity, suffixes (k), $*_k$ are omitted. In (10), the (i, j) th element of the right-hand side is transformed into:

$$\begin{aligned} [\boldsymbol{b} \otimes \bar{\boldsymbol{b}} \otimes \boldsymbol{P}]_{ij} &= \textstyle\bigoplus_{l=1}^{n} \{[\boldsymbol{b} \otimes \bar{\boldsymbol{b}}]_{il} + [\boldsymbol{P}]_{lj}\} \\ &= [\boldsymbol{b} \otimes \bar{\boldsymbol{b}}]_{ij} + [\boldsymbol{d}]_j = [\boldsymbol{b}]_i + [\bar{\boldsymbol{b}}]_j + [\boldsymbol{d}]_j . \end{aligned} \tag{11}$$

If $j \in \mathfrak{R}(k)$, $[\boldsymbol{b}]_j \neq \varepsilon$ and $[\bar{\boldsymbol{b}}]_j = -[\boldsymbol{b}]_j$ are followed and indicates: $[\boldsymbol{b}]_i + [\bar{\boldsymbol{b}}]_j + [\boldsymbol{d}]_j = [\boldsymbol{b}]_i - [\boldsymbol{a}]_j$. Moreover, if $i \notin \mathfrak{R}(k)$, this yields $\varepsilon - [\boldsymbol{a}]_j = \varepsilon$.

On the other hand, if $j \notin \mathfrak{R}(k)$, $[\bar{\boldsymbol{b}}]_j = [\boldsymbol{b}]_j = \varepsilon$ holds true and indicates: $[\boldsymbol{b}]_i + [\bar{\boldsymbol{b}}]_j + [\boldsymbol{d}]_j = \varepsilon$. These results imply that (11) holds true, which proves the proposition.
■

Consequently, the transition matrix $\boldsymbol{M}_k$ can be computed from $\boldsymbol{b}(k)$ and $\boldsymbol{P}_k$, in the following manner:

$$\boldsymbol{M}_k = \boldsymbol{e} \oplus \boldsymbol{b}(k) \otimes \overline{\boldsymbol{b}(k)} \otimes \boldsymbol{P}_k .$$

There are several ways of supplying $\boldsymbol{b}(k)$. The most acceptable is dependent on the target system, and several common methods are evaluated in the next subsection.

C. Schedules in mid-facilities

Various methods can be used to supply the processing completion times. In policy (A) that requires precedence constraints, the following two are typically used.

- The earliest times: Supplying the input time for the uppermost facility, the remaining facilities begin processing as soon as their inputs are available.
- The latest times: Supplying the output time for the lowermost facility, the remaining facilities begin

processing at the latest time that will not cause the estimated output time to be delayed.

In addition to these, the following method is particularly useful for application in practical systems.

- The mid times: Midway between the earliest and latest times.

Denote the earliest 'completion' times, latest 'start' times and mid 'completion' times of job k by $\boldsymbol{x}_E(k)$, $\boldsymbol{x}_L(k)$ and $\boldsymbol{x}_M(k)$, respectively. As an example of the mid time, the definition below would be natural and easy to handle:

$$[\boldsymbol{x}_M(k)]_i = \{[\boldsymbol{x}_E(k)]_i + ([\boldsymbol{x}_L(k)]_i + [\boldsymbol{d}(k)]_i)\}/2. \tag{12}$$

Recalling that the latest time is assigned to the start time, the processing time must be added to calculate the latest completion time.

Supply the input and output times with $\boldsymbol{u}(k) \in \mathscr{D}^n$ and $\boldsymbol{y}_+(k) \in \overline{\mathscr{D}}^n$, respectively. It is known that $\boldsymbol{x}_E(k)$ and $\boldsymbol{x}_{L+}(k)$ are dual and are calculated as follows [4]:

$$\boldsymbol{x}_E(k) = \boldsymbol{A}_k \otimes \boldsymbol{u}(k), \tag{13}$$

$$\boldsymbol{x}_{L+}(k) = \boldsymbol{A}_k^T \odot \boldsymbol{y}_+(k), \tag{14}$$

If $i \notin \mathscr{R}(k)$, we supply the following values as input and output variables: $[\boldsymbol{u}(k)]_i = \varepsilon$, $[\boldsymbol{y}_+(k)]_i = +\infty$. For the corresponding earliest completion and latest start times, these values are returned:

$$[\boldsymbol{x}_E(k)]_i = \varepsilon,\ [\boldsymbol{x}_{L+}(k)]_i = +\infty. \tag{15}$$

Since (14) is defined in $\overline{\mathscr{D}}^n$, the elements of $\boldsymbol{x}_{L+}(k)$ and $\boldsymbol{y}_+(k)$ that have $+\infty$ must be inverted before using (12). This operation can be accomplished by introducing a new operator. However, it would be better if we can calculate the latest times by using operators and rules that have been already defined. Hence, we derive another formula by which $\boldsymbol{x}_L(k)$ is calculated in field $\mathscr{D}^n$. This is accomplished by considering the next theorem.

Theorem 2: If we supply the output times as $\boldsymbol{y}(k) \in \mathscr{D}^n$, the latest processing start times for job k are computed using the following formula:

$$\boldsymbol{x}_L(k) = \overline{\overline{\boldsymbol{y}(k)} \otimes \boldsymbol{A}_k}, \tag{16}$$

where $\boldsymbol{x}_L(k)$ returns $[\boldsymbol{x}_L(k)]_i = \varepsilon$ if $i \notin \mathscr{R}(k)$, and $[\boldsymbol{x}_L(k)]_i \in \boldsymbol{R}$ if $i \in \mathscr{R}(k)$. With respect to $\boldsymbol{y}(k)$, set $[\boldsymbol{y}(k)]_j = \varepsilon$ if $j \notin \mathscr{R}(k)$ and there is at least one instance j that satisfies $[\boldsymbol{y}(k)]_j \in \boldsymbol{R}$ and $j \in \mathscr{R}(k)$.

Proof. For simplicity, suffixes (k), $*_k$ are omitted. The i th element of (16) can be expanded as:

$$\left[\overline{\overline{\boldsymbol{y}} \otimes \boldsymbol{A}}\right]_i = \overline{\bigoplus_{l=1}^n ([\overline{\boldsymbol{y}}]_l + [\boldsymbol{A}]_{li})} \tag{17}$$

$$= \begin{cases} \varepsilon: \text{ if } [\overline{\boldsymbol{y}}]_l + [\boldsymbol{A}]_{li} = \varepsilon \text{ for all } l\ (1 \le l \le n), \\ -\bigoplus_{l=1}^n ([\overline{\boldsymbol{y}}]_l + [\boldsymbol{A}]_{li}): \text{ otherwise.} \end{cases}$$

On the other hand, the i th element of (14) is expanded as:

$$[\boldsymbol{A}^T \odot \boldsymbol{y}_+]_i = \bigwedge_{l=1}^n (-[\boldsymbol{A}^T]_{il} + [\boldsymbol{y}_+]_l)$$

$$= \begin{cases} +\infty: \text{ if } -[\boldsymbol{A}]_{li} + [\boldsymbol{y}_+]_l = +\infty \text{ for all } l, \\ -\bigoplus_{l=1}^n ([\boldsymbol{A}]_{li} + [\overline{\boldsymbol{y}_+}]_l): \text{ otherwise.} \end{cases} \tag{18}$$

First, examine the case where $i \notin \mathscr{R}(k)$. In this case, the property of the transition matrix $\boldsymbol{A}$ follows $[\boldsymbol{A}]_{li} = \varepsilon$ for all l $(1 \le l \le n, l \ne i)$. For the case $l = i$, $[\boldsymbol{A}]_{li} \in R$, $[\boldsymbol{y}]_l = +\infty$ and $[\boldsymbol{y}]_l = [\boldsymbol{y}]_l = \varepsilon$ hold true. Thus, $[\boldsymbol{y}]_l + [\boldsymbol{A}]_{li} = [\boldsymbol{y}_+]_l + [\boldsymbol{A}]_{li} = \varepsilon$ is true for all l $(1 \le l \le n)$, which indicates that:

$$\left[\overline{\overline{\boldsymbol{y}} \otimes \boldsymbol{A}}\right]_i \text{ and } [\boldsymbol{A}^T \odot \boldsymbol{y}_+]_i,$$

return ε and $+\infty$, respectively.

Next, consider the case for $i \in \mathscr{R}(k)$. Recalling the assumptions, the latest processing start time in facility i is affected by at least one estimated output time in a downstream facility. This indicates that there is at least one instance l which satisfies: $[\boldsymbol{A}]_{li} \in R$, $[\boldsymbol{y}]_l \in R$ and $[\boldsymbol{y}_+]_l \in R$. This yields:

$$([\overline{y}]_l + [\boldsymbol{A}]_{li}) = ([\overline{\boldsymbol{y}_+}]_l + [\boldsymbol{A}]_{li}) \in R. \tag{19}$$

The set of l which satisfies (19) is interpreted as those that have an external output, and located downstream of facility i or facility i itself. With respect to other facilities, they do not have an external output or are located upstream of facility i. This implies: $[\boldsymbol{y}]_l = [\boldsymbol{y}_+]_l = \varepsilon$ or $[\boldsymbol{A}]_{li} = \varepsilon$, which yields: $([\boldsymbol{y}]_l + [\boldsymbol{A}]_{li}) = ([\boldsymbol{y}_+]_l + [\boldsymbol{A}]_{li}) = \varepsilon$.

Accordingly, the lower cases of (17) and (18) return the same finite value. This is equivalent to the latest start time in facility i.

■

A primary feature of (16) is that the calculation is closed in the field $\mathscr{D}^n$. This would make the relevant computations simpler.

Next, a method for determining the base time for $\boldsymbol{b}(k)$ is considered. In (10), the 'difference' of two elements of $\boldsymbol{b}(k)$ plays a more important role than the absolute value. Thus we can set $[\boldsymbol{u}(k)]_i = e$ for a facility $i \in \mathscr{R}(k)$ with an external input, without loss of generality. In a similar way, we can set $[\boldsymbol{y}(k)]_i = e$ for a facility $i \in \mathscr{R}(k)$ with an external output. However, note that the base times for $\boldsymbol{u}(k)$ and $\boldsymbol{y}(k)$ must be the same if we supply $\boldsymbol{b}(k)$ with $\boldsymbol{x}_M(k)$ in (12).

D. Commonality of required parameters

From the previous discussions, we are now in a position to calculate the transition matrices for both policies (A) and (B) once $\boldsymbol{d}(k)$ and $\boldsymbol{F}_k$ have been prepared. The former vector represents the processing times whilst the latter matrix represents the precedence constraints. This is confirmed by Fig.2.

First, the transition matrix $\boldsymbol{A}_k$ for policy (A) is obtained in a straightforward way by supplying $\boldsymbol{d}(k)$ and $\boldsymbol{F}_k$. Next, supply the input variables $\boldsymbol{u}(k)$ with a base time 0, and calculate the earliest processing completion times $\boldsymbol{x}_E(k)$ using (13). Extracting required elements from these, we set the output variables $\boldsymbol{y}(k)$. Then, calculate the latest processing start times $\boldsymbol{x}_L(k)$ using (16). After the mid times $\boldsymbol{x}_M(k)$ have been set using (12), we equate those to the processing completion times $\boldsymbol{b}(k)$. In the final stage, the transition matrix $\boldsymbol{M}_k$ is obtained using $\boldsymbol{b}(k)$ and $\boldsymbol{d}(k)$.

E. Change of policy

We now develop a method to obtain the transition matrix $\boldsymbol{A}_k$ for policy (A), from parameters $\boldsymbol{a}(k)$ and $\boldsymbol{b}(k)$ that is suitable for policy (B). Since $\boldsymbol{d}(k)$ represent the occupation times in the respective facilities, the next relationship is obtained:

$$\boldsymbol{d}(k) = \overline{\overline{\boldsymbol{b}(k)} \otimes \mathrm{diag}[\boldsymbol{a}(k)]}. \tag{20}$$

The correctness of this formula is confirmed as follows; if $i \in \mathcal{R}(k)$, the right-hand side of (20) is expanded as:

$$\overline{\oplus_{l=1}^{n} [\overline{\boldsymbol{b}(k)}]_l + \mathrm{diag}[\boldsymbol{a}(k)]_{li}}$$
$$= \overline{[\overline{\boldsymbol{b}(k)}]_i + [\boldsymbol{a}(k)]_i} = [\boldsymbol{b}(k)]_i - [\boldsymbol{a}(k)]_i .$$

On the other hand, if $i \notin \mathcal{R}(k)$, the right-hand side is $\overline{\varepsilon + \varepsilon} = \overline{\varepsilon} + \varepsilon = \varepsilon$. Subsequently, $\boldsymbol{P}_k \ (= \mathrm{diag}[\boldsymbol{d}(k)])$ is calculated. This can also be calculated with $\boldsymbol{a}(k)$ and $\boldsymbol{b}(k)$ as :

$$\boldsymbol{P}_k = \overline{\overline{\mathrm{diag}[\boldsymbol{b}(k)]} \otimes \mathrm{diag}[\boldsymbol{a}(k)]}.$$

Next, consider a procedure to obtain $\boldsymbol{F}_k$. We interpret $[\boldsymbol{F}_k]_{ij}$ as the transition time between two facilities, rather than having a precedence constraint. The time between two arbitrary facilities is identified using the following theorem.

Theorem 3: If the occupation times $\boldsymbol{P}_k \in \mathcal{D}^{n\times n}$ and processing completion times $\boldsymbol{b}(k) \in \mathcal{D}^n$ for job k are supplied, a matrix $\boldsymbol{T}_k \in \mathcal{D}^{n\times n}$ that returns the transition times between two arbitrary facilities is given by:

$$\boldsymbol{T}_k = \overline{\boldsymbol{Q}_k}, \tag{21}$$

where $\boldsymbol{Q}_k$ represents the matrix used in Theorem 1. Moreover, if $i \notin \mathcal{R}(k)$ or $j \notin \mathcal{R}(k)$, $[\boldsymbol{T}_k]_{ij}$ returns ε.

Proof. From (9), if $i \in \mathcal{R}(k)$ and $j \in \mathcal{R}(k)$, it follows that:

$$[\overline{\boldsymbol{Q}_k}]_{ij} = -([\boldsymbol{b}(k)]_j - [\boldsymbol{a}(k)]_i) = [\boldsymbol{a}(k)]_i - [\boldsymbol{b}(k)]_j .$$

The right-hand side indicates the difference between the processing start time in facility i and the processing completion time in facility j, which is equivalent to the transition time from facility j to i. Accordingly, if $i \in \mathcal{R}(k)$ and $j \in \mathcal{R}(k)$, $[\boldsymbol{T}_k]_{ij} = [\overline{\boldsymbol{Q}_k}]_{ij}$ is satisfied.

Next, if $i \notin \mathcal{R}(k)$ or $j \notin \mathcal{R}(k)$ because (9) indicates that $[\boldsymbol{Q}_k]_{ij} = \varepsilon$, $[\boldsymbol{T}_k]_{ij} = [\overline{\boldsymbol{Q}_k}]_{ij} = \varepsilon$ follows directly. Thus the transition times between two arbitrary facilities can be calculated using (21).

■

Note that $\boldsymbol{T}_k$ in (21) is not a representation matrix of the DAG. This is because order relations are not taken into consideration in the heap model. Thus additional information regarding order relations must be supplied for transforming $\boldsymbol{T}_k$ into the transition matrix $\boldsymbol{F}_k$ for policy (A). Denoting the list of preceding facilities of facility i $(1 \le i \le n)$ by $\mathcal{P}_i(k)$, $\boldsymbol{F}_k$ is determined by: $[\boldsymbol{F}_k]_{ij} = \{ [\boldsymbol{T}_k]_{ij} : \text{if } j \in \mathcal{P}_i(k),\ \varepsilon : \text{otherwise}\}$.

IV. Numerical Example

A numerical example of a simple system is presented to confirm the effectiveness of the proposed methods. Fig. 3 depicts the schedules for two jobs in a system with five facilities. In the initial stage, assume that jobs $k=1$ and $k=2$ are scheduled based on policies (A) and (B), respectively. The arrows represent precedence constraints. Bars with thick lines on either the left or right side mean that the corresponding facilities are attached to an external input or output, respectively. Since job $k=2$ is based on the heap model, the precedence constraints are ignored.

For job $k=1$, the adjacency matrix $\boldsymbol{F}_1$ and the occupation times $\boldsymbol{d}(1)$ are supplied as follows:

$$\boldsymbol{F}_1 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & e & \varepsilon \end{bmatrix}, \ \boldsymbol{d}(1) = \begin{bmatrix} \varepsilon \\ 2 \\ 3 \\ 1 \\ 1 \end{bmatrix}. \tag{22}$$

For job $k=2$, the processing start and completion times for using all the facilities, $\boldsymbol{a}(2)$ and $\boldsymbol{b}(2)$, respectively, are supplied in the following manner:

$$\boldsymbol{a}(2) = [8\ 6\ \varepsilon\ 7\ 10]^T,\ \boldsymbol{b}(2) = [10\ 7\ \varepsilon\ 9\ 11]^T. \tag{23}$$

First, the transition matrix for policy (B) is obtained from the parameters in (22). The external input is attached to facility 2, and we supply the input vector $\boldsymbol{u}(1)$ as $\boldsymbol{u}(1) = [\varepsilon\ e\ \varepsilon\ \varepsilon\ \varepsilon]^T$. Since there is no job at $k=0$, it follows that $\boldsymbol{x}(0) = \boldsymbol{\varepsilon}$. Hence, the transition matrix $\boldsymbol{A}_1$ and the earliest processing completion times $\boldsymbol{x}_E(1)$ are calculated by utilizing (1) and (4), as follows:

$$\boldsymbol{A}_1 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 5 & 3 & \varepsilon & \varepsilon \\ \varepsilon & 3 & \varepsilon & 1 & \varepsilon \\ \varepsilon & 6 & 4 & 2 & 1 \end{bmatrix}, \ \boldsymbol{x}_E(1) = \begin{bmatrix} \varepsilon \\ 2 \\ 5 \\ 3 \\ 6 \end{bmatrix}.$$

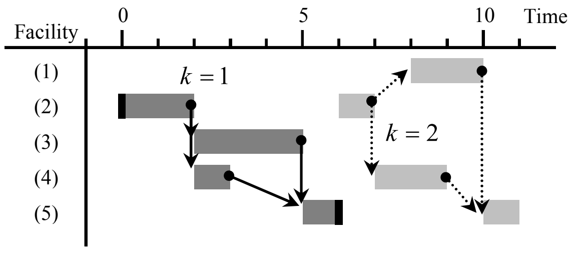

Figure 3. Initial schedule for two jobs.

Noting that the external output is attached to facility 5, the estimated output time $\boldsymbol{y}(1)$ is supplied by extracting only the fifth element of $\boldsymbol{x}_E(1)$: $\boldsymbol{y}(1) = [\varepsilon\ \varepsilon\ \varepsilon\ \varepsilon\ 6]^T$. With the help of (16), $\boldsymbol{x}_L(1)$ is obtained as: $\boldsymbol{x}_L(1) = [\varepsilon\ e\ 2\ 4\ 5]^T$. Only facility 4 has a float time of two time units. For fixing the relative times, we supply mid times $\boldsymbol{x}_M(1)$ here. Using (12), $\boldsymbol{x}_M(1)$ is obtained which is set as the fixed processing completion times $\boldsymbol{b}(1)$:

$$\boldsymbol{x}_M(1) = [\varepsilon\ 2\ 5\ 4\ 6]^T \equiv \boldsymbol{b}(1)\,. \tag{24}$$

Using (10), (20) and (23), the transition matrix for policy (B), $\boldsymbol{M}_1\ \ (= \boldsymbol{e} \oplus \boldsymbol{Q}_1)$, can be calculated as:

$$\boldsymbol{M}_1 = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 2 & e & -1 & -3 \\ \varepsilon & 5 & 3 & 2 & e \\ \varepsilon & 4 & 2 & 1 & -1 \\ \varepsilon & 6 & 4 & 3 & 1 \end{bmatrix}. \tag{25}$$

Next, the transition matrix for policy (A) is obtained from the parameters in (23). With the help of (20), $\boldsymbol{d}(2) = [2\ 1\ \varepsilon\ 2\ 1]^T$ is obtained. Then, using Theorem 1, the transition matrix $\boldsymbol{M}_2$ is calculated as:

$$\boldsymbol{M}_2 = \begin{bmatrix} 2 & 4 & \varepsilon & 3 & e \\ -1 & 1 & \varepsilon & e & -3 \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ 1 & 3 & \varepsilon & 2 & -1 \\ 3 & 5 & \varepsilon & 4 & 1 \end{bmatrix}. \tag{26}$$

In addition, with the help of (21), we calculate the transition times between two arbitrary facilities represented by $\boldsymbol{T}_2$:

$$\boldsymbol{T}_2 = \begin{bmatrix} -2 & 1 & \varepsilon & -1 & -3 \\ -4 & -1 & \varepsilon & -3 & -5 \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ -3 & e & \varepsilon & -2 & -4 \\ e & 3 & \varepsilon & 1 & -1 \end{bmatrix}.$$

Furthermore, additional information regarding the precedence constraints is supplied to extract the necessary elements of $\boldsymbol{T}_2$, followed by setting all other elements to ε . Here, assume that the dashed arrows in Fig. 3 represent the precedence constraints. Accordingly, only four elements of the adjacency matrix, $(1,2)$, $(4,2)$, $(5,1)$ and $(5,4)$, are extracted to reflect the precedence constraints:

$$\boldsymbol{F}_2 = \begin{bmatrix} \varepsilon & 1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon & 1 & \varepsilon \end{bmatrix}.$$

This matrix can be interpreted as the adjacency matrix taking into account the transition times, which can be confirmed directly from Fig. 3.

As the above shows, by utilizing the proposed methods, policy (A) can be transferred to policy (B), and the reverse is also possible. Thus, we now have an intelligent method of transferring progress control policies for each job.

V. CONCLUSION

This research has focused on scheduling problems for a class of repetitive discrete event systems, and proposed a method for transferring two scheduling policies to control the progress of jobs. With the proposed method, we can select two policies, (A) and (B) for each job; the former policy is based on precedence constraints whilst the latter allows the completion of a job after a predetermined time has lapsed. We proposed a method for calculating two transition matrices by supplying a common parameter vector and matrix.

REFERENCES

[1] S. Lahaye, J. Boimond, and L. Hardouin, “Linear periodic systems over dioids,” *Discrete Event Dynamical Systems: Theory and Applications*, vol. 14, no. 2, pp. 133–152, 2004.

[2] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat, *Synchronization and Linearity*. New York: John Wiley & Sons, 1992. [Online]. Available: http://maxplus.org

[3] B. Heidergott, G. J. Olsder, and L. Woude, *Max Plus at Work: Modeling and Analysis of Synchronized Systems*. New Jersey: Princeton University Pr., 2006.

[4] H. Goto, “Dual representation of event-varying max-plus linear systems,” *International Journal of Computational Science*, vol. 1, no. 3, pp. 225–242, 2007.

[5] H. Goto, “Dual representation and its online scheduling method for event-varying DESs with capacity constraints,” *International Journal of Control*, vol. 81, no. 4, pp. 651–660, 2008.

[6] S. Gaubert and J. Mairesse, “Modeling and analysis of timed petri nets using heaps of pieces,” *IEEE Transactions on Automatic Control*, vol. 44, no. 4, pp. 683–698, 1999.

[7] J. Mairesse and L. Vuillon, “Asymptotic behavior in a heap model with two pieces,” *Theoretical Computer Science*, vol. 270, no. 1–2, pp. 525–560, 2002.

[8] G. Schullerus and V. Krebs, “Diagnosis of batch processes based on parameter Estimation of discrete event models,” *Proceedings of the European Control Conference*, pp. 1612–1617, 2001.

[9] B. Cottenceau, L. Hardouin, J. Boimond, and J. Ferrier, “Model reference control for timed event graphs in dioids,” *Automatica*, vol. 37, no. 9, pp. 1451–1458 , 2001.