On the Capacity of a Process with Batch Processing and Setup Times

Zhichao Feng
International Institute of Finance, School of Management,
University of Science and Technology of China, Hefei, Anhui 230026, China
email: zhichao20@ustc.edu.cn

Milind Dawande **Ganesh Janakiraman**
Jindal School of Management,
The University of Texas at Dallas, Richardson, Texas 75080
email: {milind, ganesh}@utdallas.edu

While batch processes are ubiquitous across industries, there is little understanding of the determination of the capacity (i.e., the maximum long-term output rate) of such processes. In this paper, we consider processes with collaboration (activities may require multiple resources simultaneously) and multitasking (the same resource may be needed for multiple activities), and characterize the capacity of an arbitrary deterministic, single-product process with *batch processing* and (activity) *setup times*. Such a process can be viewed as an "intermittent" process in which production occurs in discrete quantities; specifically, in multiples of the batch size. To obtain a corresponding "smooth" process, we use the most natural transformation that eliminates batching and setups, and simply prorates the processing and setup time per flow unit. The main result of our analysis is that *capacity remains unaffected* in this transformation. Then, using recent developments in the literature on process capacity, the capacity of the transformed process can be obtained by computing the fractional chromatic number of an associated graph – this latter problem can be formulated as a linear program.

Key words: process capacity, batch processing, setup times, collaboration, multitasking

1. Introduction

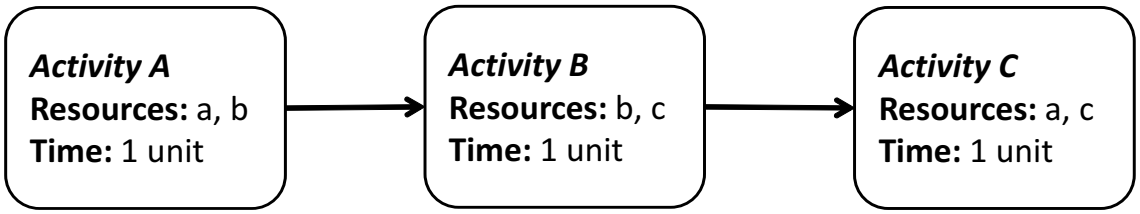
An activity (operation) of a process is said to be implemented in a *batch* mode if it must be executed simultaneously on a collection of flow units. A *batch process* is one in which, motivated typically by economies of scale, one or more activities of the process need to be carried out in batch mode – for each such activity, the number of flow units that need to be processed simultaneously is referred to as the *batch size* of that activity; these sizes can differ across activities. Batch processes are prevalent across manufacturing and service industries, with some prominent examples being in classical manufacturing, agriculture,

food processing, clothing, pharmaceutical formulations, banking, education, and travel; see, e.g., Pinedo (2009), Hopp and Spearman (2011), Anupindi et al. (2012), Pound et al. (2014), Cachon and Terwiesch (2013), Stevenson (2017).

In this paper, we consider an arbitrary deterministic, single-product process with collaboration and multitasking, batch processing, and (activity) setup times; our model for such a process is defined in the next section. Our goal is to characterize the capacity, i.e., the maximum long-term output rate, of such a process. As a byproduct, we will also obtain a practically convenient approach to compute this capacity. The precise definition of process capacity and the statement of our main result (Theorem 1) are also in the next section. At the heart of our derivation of this result is a capacity-preserving transformation from a process with batching and setup times to another process without these two features. Below, we illustrate this transformation on two simple processes.

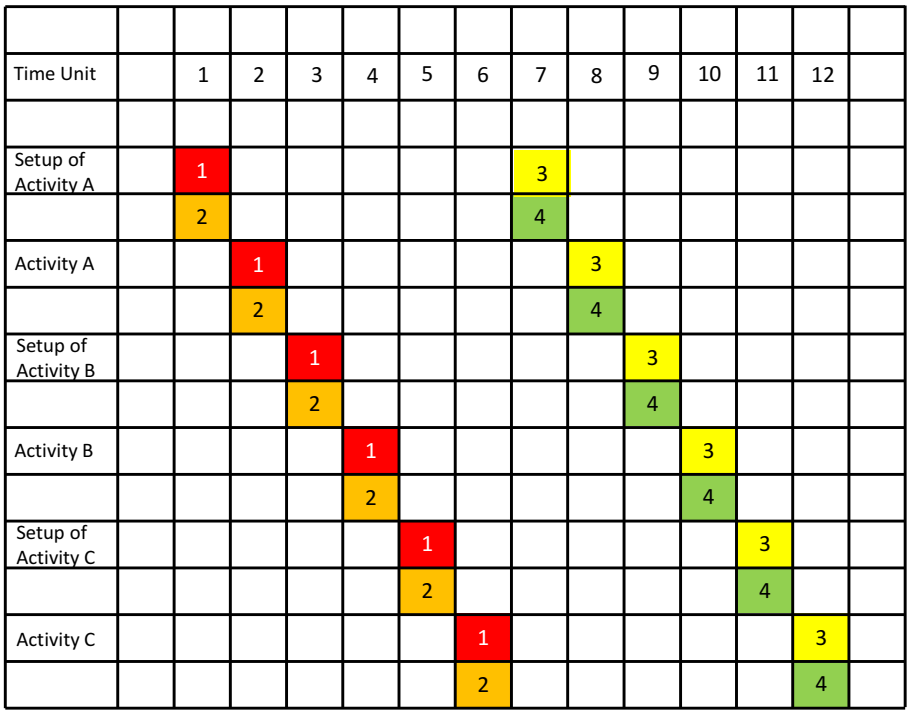
EXAMPLE 1. Consider a process with three activities: A, B, and C. Figure 1 shows the activities of the process as well as their processing times and required resources. Each of the three activities takes one unit of time and we have one unit of each of the three resources, namely, a, b, and c. Let the batch size of each activity be 2 flow units and the

Figure 1 Activities, Resources, and Processing Times in the process in Example 1.



setup time of each activity be 1 unit of time. We assume that the setup of each activity requires the same set of resources as that needed for the activity. Notice that any two of activities A, B, and C share at least one resource in common. Therefore, no two activities or setups can be in progress simultaneously. Thus, in any feasible processing schedule, the total time required for these three activities and their setups is $1+1+1+1+1+1=6$. Consequently, the process capacity is at most $2/6 = 1/3$ units per time unit. In fact, the process capacity is *exactly* $1/3$ units per time unit, since we can obtain a feasible schedule whose throughput (i.e., the long-term average rate of completed units) achieves this upper

Figure 2 Gantt chart of a feasible schedule for the process in Example 1.



bound; one such schedule is shown in Figure 2; each flow unit is depicted using a distinct color in the figure.

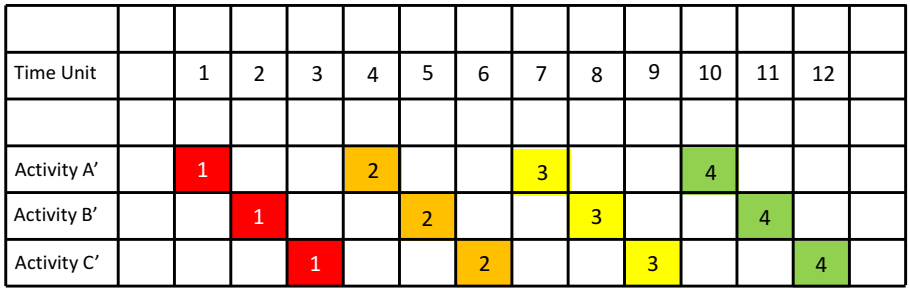
Next, we obtain a corresponding process with *no batching* (i.e., the batch size is 1) and *no setups* (i.e., the setup time of each activity is 0) by considering a natural transformation that simply prorates the processing and setup times per flow unit. The resource requirements of the activities remain unchanged by this transformation. Let A' (resp., B', C') denote the corresponding activities in the transformed process. Then, the processing time of activity A' in the transformed process equals

$$\frac{\text{Setup time of activity } A + \text{Processing time of activity } A}{\text{Batch size of activity } A} = \frac{1+1}{2} = 1.$$

The processing times of activities B' and C' are calculated in a similar manner, and both equal 1. Each of the three activities in the transformed process has a batch size of 1 and no setup time. Similar to the argument above, no two of the three activities A', B', and C', can be in progress simultaneously. Thus, it is again easy to see that the capacity of the transformed process is at most 1/3 units per time unit. Figure 3 shows a feasible schedule whose throughput achieves this upper bound; consequently, the capacity of the

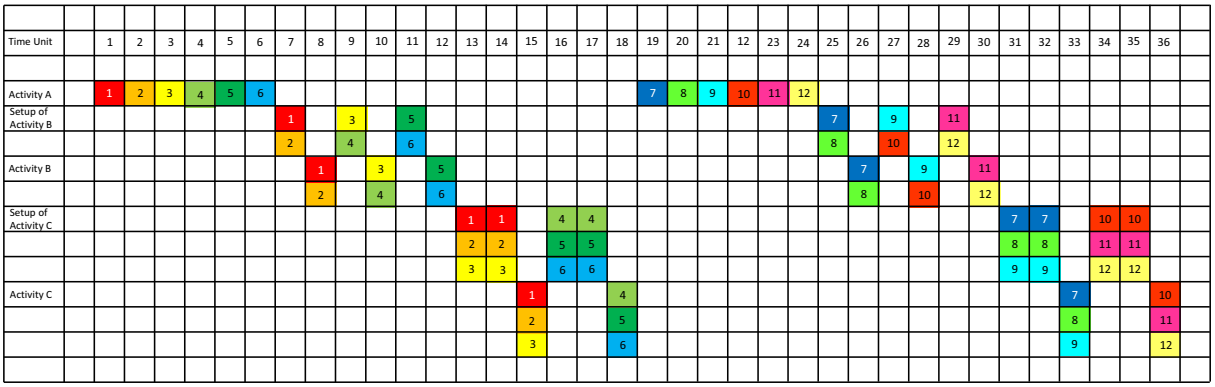
transformed process equals 1/3 units per time unit. Thus, *capacity remains unaffected* by the transformation. Later, in Theorem 1, we will formally establish this result. ■

Figure 3 Gantt chart of a feasible schedule for the transformed process in Example 1.



EXAMPLE 2. Consider now a process that is similar in structure to the one in the previous example, except for the batch sizes and setup times. As before, our process has three activities (A, B, and C) that each take one unit of time and we have one unit of each of three resources (a, b, and c). The batch sizes of activities A, B, and C are 1, 2, and 3, respectively, and the setup times of these activities are 0, 1, and 2, respectively. Arguing as in Example 1, we note that no two activities or setups can be in progress simultaneously. Thus, to obtain 6 units of output, which requires activity A to be executed 6 times, activity B to be executed $6/2 = 3$ times, and activity C to be executed $6/3 = 2$ times, we necessarily need $6 \times 1 + 3 \times (1 + 1) + 2 \times (2 + 1) = 18$ time units. Consequently, the process capacity is at most $6/18 = 1/3$ units per time unit. A feasible schedule whose throughput achieves this upper bound is shown in Figure 4, thus establishing that the capacity of our process is exactly 1/3 units per time unit.

Figure 4 Gantt chart of a feasible schedule for the process in Example 2.



As in Example 1, we apply the prorating transformation to obtain a process with no batching and no setups: the processing time of activity $A^{'} = (0+1)/1 = 1$, the processing time of activity $B^{'} = (1+1)/2 = 1$, and the processing time of activity $C^{'} = (2+1)/3 = 1$. As argued in Example 1, the capacity of this transformed process capacity is 1/3 units per time unit, again illustrating the capacity-preserving nature of our transformation. ■

A process with batching and setups can be viewed as an "intermittent" process in which production occurs in discrete blocks; i.e., in batches, with each activity requiring a setup. The discrete nature of the output in such processes renders the optimization of the long-term throughput quite challenging, with no characterization available in the extant literature. As illustrated above, the process transformation bestows two key advantages. First, the transformation is capacity-preserving and results in a process *without* batching and setups – this is a relatively "smooth" production process. Second, recent work in the literature – particularly, Bo et al. 2019 and Dawande et al. (2020) – has analyzed the computation of the capacity of such processes, allowing us to exploit these developments. The basic idea behind Theorem 1 is to exploit both these advantages to characterize the capacity of deterministic, single-product processes *with* batching and setups.

Before proceeding further, we briefly discuss the connections of our work with the process-capacity literature.

1.1. Connections with the Process-Capacity Literature

The simple "bottleneck formula" is widely discussed in Operations Management textbooks to illustrate the calculation of the capacity of a process: The capacity of each resource is first computed by considering that resource in isolation; the bottleneck resources are those with the least capacity, which is defined to be the capacity of the process, see, e.g., Anupindi et al. (2012), Cachon and Terwiesch (2013), Stevenson (2017). It is now well-known that this formula correctly computes process capacity only in processes with special structures – e.g., processes with no collaboration or those with no multitasking. In fact, Gurvich and Van Mieghem (2015) provide a necessary and sufficient condition under which the bottleneck formula correctly computes process capacity. Extending this work, Bo et al. (2019) show that it is strongly NP-hard to compute process capacity and to even approximate it within a reasonable factor. In addition, they obtain a graph-theoretic

characterization of process capacity by connecting it to the fractional chromatic number of the "collaboration graph" of the process, which is defined as follows: Each node of the graph corresponds to an activity of the process and two activities are connected by an edge iff they share at least one common resource. Gurvich and Van Mieghem (2017) investigate how process capacity is affected by prioritizing certain tasks over others. Dawande et al. (2020) define and examine bottleneck structures that are responsible for limiting the capacity of a process to its present value.

As we will show in our analysis of processes with batching, it is sufficient to optimize the production rate of the process over all *cyclic policies*, namely policies that repeatedly execute a fixed sequence of actions, to determine its capacity (similar results have been established in the literature for many other production systems). In this sense, our work is also related to the vast literature on cyclic scheduling problems; see, e.g., the cyclic flow shop problem (Abadi et al. 2000, McCormick et al. 1989), the cyclic job shop problem (Lee and Posner 1997, Matsuo et al. 1991), and the cyclic robotic-cell scheduling problem (Dawande et al. 2002, Crama and Van De Klundert 1997). In this literature, it is typical to restrict attention to a specific class of cyclic schedules (e.g., 1-unit cyclic policies; see, e.g., Geismar et al. 2008, 2006, Kreipl and Pinedo 2004) and obtain an optimal policy over that class.

2. Our Main Result

Our main result is Theorem 1 (Section 2.4) below. In Sections 2.1–2.3, we first define the basic ingredients needed to state this result. The notation we use is adopted from Bo et al. (2019) and Dawande et al. (2020). Section 2.5 provides a brief road map of the proof of Theorem 1.

2.1. A Deterministic, Single-Product Process with Batch Processing and (Activity) Setup Times

Our model of a process is an extension of that in Bo et al. (2019) – the latter did not consider batching and setup times. Therefore, to the extent possible, we retain the same notation. One or more activities of the process may require multiple resources simultaneously (collaboration) and the same resource may be needed for multiple activities (multitasking); see Gurvich and Van Mieghem (2015). A deterministic, single-product process with batching and setup times is represented using a 9-tuple, whose components are as follows:

1. The set of activities required to produce one flow unit of the product is denoted by $V = \{v_1, v_2, \ldots, v_k\}$. Without loss of generality, we assume that, on each flow unit, each activity needs to be performed exactly once.
2. The precedence relationships are captured using a directed acyclic graph $G = (V, A)$, where, for $v_i, v_j \in V$, a directed edge $e_{ij} = (v_i, v_j) \in A$ represents the precedence relationship $v_i \to v_j$ (i.e., activity v_i must precede activity v_j). If there are no precedence relationships, then $A = \varnothing$.
3. The durations of the activities are defined by the function $\tau : V \to \mathbb{N}$, where $\mathbb{N}$ is the set of strictly positive integers.
4. The set of resources is denoted by W.
5. The number of units of resource w is denoted by N_w, $w \in W$.
6. The set of resources needed (simultaneously) to perform activity v is denoted by W_v, $v \in V$. We assume that, to perform activity $v \in V$ on a batch of b_v flow units, one unit of each resource $w \in W_v$ is needed.
7. The indicator $\mathcal{Z}$ denotes whether or not preemption is allowed in the processing of the activities: $\mathcal{Z} = 1$ denotes that preemption is not allowed; 0, otherwise.
8. The batch size of activity v is denoted by b_v, $v \in V$. That is, at any time instant, the number of flow units undergoing activity v is an integral multiple, including zero, of b_v.
9. The setup time of activity v is denoted by s_v, $v \in V$. That is, setup of activity v, which must be scheduled immediately before activity v, takes s_v units of time. We assume that the setup of activity v requires the same set of resources as that needed for activity v.

Using these components, a process $\mathcal{P}$ is defined as a 9-tuple:

$$\mathcal{P} = \big(V, A, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v, v \in V\}\big).$$

We clarify the notation used for the main processes we encounter in our analysis: We use $\mathcal{P}$ to denote a process with batching and setups. The superscript 1 (i.e., $\mathcal{P}^1$) signifies a process with "no precedence constraints" while the subscript 1 (i.e., $\mathcal{P}_1$) signifies a process with "no setups". The subscript 0 (i.e., $\mathcal{P}_0$) signifies a process with "no batching and setups". This notation is conveniently summarized in Table 1 below.

Table 1 Notation for the main processes used in the analysis.

Process	Batching	Setup	Precedence constraints
$\mathcal{P}$	✓	✓	✓
$\mathcal{P}_1$	✓	×	✓
$\mathcal{P}_1^1$	✓	×	×
$\mathcal{P}_0$	×	×	✓
$\mathcal{P}_0^1$	×	×	×

2.2. Capacity of Process $\mathcal{P}$

For $j \in \mathbb{N}$, flow unit j refers to the j^{th} flow unit that enters the process since time unit 1. For process $\mathcal{P}$, we define a generic schedule (or policy), $\psi : \mathbb{N}^2 \to (\{0,1\} \times \{0,1\})^{|V|}$, as follows: For $j, t \in \mathbb{N}$, let

$$\psi(j,t) := (I_{j,t}^v, S_{j,t}^v; v \in V),$$

where for $v \in V$,

$$I_{j,t}^v := \begin{cases} 1, & \text{if activity } v \text{ is executed on flow unit } j \text{ during time unit } t, \\ 0, & \text{otherwise.} \end{cases}$$

$$S_{j,t}^v := \begin{cases} 1 & \text{if setup of activity } v \text{ is executed for flow unit } j \text{ during time unit } t, \\ 0 & \text{otherwise.} \end{cases}$$

We consider the following constraints on ψ:

(a) **Resource availability constraints**. For $t \in \mathbb{N}$ and $w \in W$, we have

$$\sum_{j \in \mathbb{N}} \sum_{v : w \in W_v} \left(\frac{I_{j,t}^v}{b_v} + \frac{S_{j,t}^v}{b_v} \right) \leqslant N_w. \tag{1}$$

In words, this constraint imposes that, during any time unit t, the total number of units in use of a resource cannot exceed the total number of the available units of that resource.

(b) **Activity constraints**. For $j \in \mathbb{N}$ and $v \in V$, we have

$$\sum_{t \in \mathbb{N}} I_{j,t}^v = \tau(v) \text{ and } \sum_{t \in \mathbb{N}} S_{j,t}^v = s_v. \tag{2}$$

That is, activity v (resp., setup of activity v) is scheduled on flow unit j exactly once.

(c) **Batch constraints.** For $t \in \mathbb{N}$ and $v \in V$, we have

$$\sum_{j \in \mathbb{N}} \frac{I^v_{j,t}}{b_v} \in \mathbb{Z}_+ \text{ and } \sum_{j \in \mathbb{N}} \frac{S^v_{j,t}}{b_v} \in \mathbb{Z}_+. \tag{3}$$

That is, during any time unit t, any activity v (resp., setup of any activity v) is executed on (resp., for) exactly mb_v flow units, for some $m \in \mathbb{Z}_+$, i.e., an integer number of batches (including zero), each containing b_v flow units. Here, note that we may have multiple copies of the resources in the set W_v that is needed to perform activity $v \in V$. Thus, during any time unit t, the number of units undergoing activity v may, in general, be an integer multiple of b_v.

From a managerial perspective, it is instructive to differentiate the notions of parallel processing and batching. When we have multiple copies of resources in a process without batching, we may be able to simultaneously execute the same activity on multiple flow units. However, each of these flow units requires separate copies of the resources. With batching, however, one copy of a resource can accommodate multiple flow units simultaneously. Specifically, under our batch constraints above, the number of units undergoing an activity simultaneously is restricted to be a multiple of the batch size for that activity. Viewed in this manner, parallel processing and batching are two different environments with no obvious comparison in terms of capacity. In fact, as we will soon see, Theorem 1 defines a transformation through which the capacity of a process with batching is shown to be equal to that of a process without batching.

(d) **Precedence constraints.** For $j \in \mathbb{N}$ and $e_{kl} = (v_k, v_l) \in A$, we have

$$\max\left\{t : t \in \mathbb{N}, I^{v_k}_{j,t} = 1\right\} < \min\left\{t : t \in \mathbb{N}, I^{v_l}_{j,t} = 1\right\}. \tag{4}$$

That is, if we have a precedence relationship $v_k \to v_l$, then on any flow unit j, activity v_k must be scheduled before activity v_l.

For $j \in \mathbb{N}$ and $v \in V$, we have

$$\max\left\{t : t \in \mathbb{N}, S^{v}_{j,t} = 1\right\} < \min\left\{t : t \in \mathbb{N}, I^{v}_{j,t} = 1\right\}. \tag{5}$$

That is, setup of activity v must be scheduled before activity v.

(e) **Non-preemption constraints.** For $j, T \in \mathbb{N}$ and $v \in V$, we have

$$\mathcal{Z} \cdot (I^v_{j,T} + S^v_{j,T}) \cdot \left[\max\left\{t : t \geqslant T, \prod_{i=T}^{t} I^v_{j,i} = 1\right\} - \min\left\{t : t \leqslant T, \prod_{i=t}^{T} S^v_{j,i} = 1\right\} - \tau(v) - s_v\right] = 0, \tag{6}$$

where we take $I^v_{j,t} = S^v_{j,t} = 0$ for $t < 0$ for convenience. That is, if preemption is not allowed ($\mathcal{Z} = 1$) and activity v is on flow unit j during time unit T ($I^v_{j,T} = 1$) or the setup of activity v is executed for flow unit j during time unit T ($S^v_{j,T} = 1$), then there must exist $\tau(v) + s_v$ consecutive time units in $\{T - \tau(v) - s_v + 1, T - \tau(v) - s_v + 2, \cdots, T + \tau(v) + s_v - 1\}$ during which activity v is on flow unit j or setup of activity v is executed for flow unit j.

Constraints (1)–(6) constitute the feasibility constraints on policy ψ. Let Ψ denote the collection of all feasible policies ψ for process $\mathcal{P}$.

For $\psi \in \Psi$ and $T \in \mathbb{N}$, let $N^\psi[1,T]$ represent the number of flow units that enter the process from the start of time unit 1 and are completed by the end of time unit T, under policy ψ. That is,

$$N^\psi[1,T] := \left| \left\{ j : j \in \mathbb{N}, \sum_{t=1}^{T} I^v_{j,t} = \tau(v) \text{ and } \sum_{t=1}^{T} S^v_{j,t} = s_v \text{ for } v \in V \right\} \right|.$$

For $\psi \in \Psi$, let r^ψ represent the process rate (or throughput rate) under ψ. That is,

$$r^\psi := \liminf_{T \to \infty} \frac{N^\psi[1,T]}{T}.$$

Let M denote the least common multiple of b_v, $v \in V$. That is,

$$M = \min\{M' : M' \in \mathbb{N},\ \frac{M'}{b_v} \in \mathbb{N} \text{ for } v \in V\}.$$

The number M, i.e., the least common multiple of all the batch sizes, will be useful later in constructing policies that satisfy the batch constraints. Let $\hat{N}^\psi[1,T] = \lfloor \frac{N^\psi[1,T]}{M} \rfloor$. Then, we have

$$r^\psi = \liminf_{T \to \infty} \frac{N^\psi[1,T]}{T} = \liminf_{T \to \infty} \frac{\hat{N}^\psi[1,T]}{T} \cdot M. \tag{7}$$

The *capacity of process* $\mathcal{P}$, which we denote by $\mu(\mathcal{P})$ is the maximum throughput that can be achieved over all feasible policies, i.e., $\mu(\mathcal{P}) := \sup_{\psi \in \Psi} r^\psi$.

2.3. A Linear Program

Consider a process $\mathcal{P}_0$ with no batching and no setup times, i.e., $\mathcal{P}_0 = \big(V, A, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v = 1, v \in V\}, \{s_v = 0, v \in V\}\big)$. The *expanded process* $\mathcal{P}_0^{exp}$ corresponding to $\mathcal{P}_0$ is obtained by dividing each activity $v_j \in V$ with $\tau(v_j) > 1$ into $\tau(v_j)$ activities,

say $v_j^1, v_j^2, \ldots, v_j^{\tau(v_j)}$, that each have an activity time of 1 unit and use the same set of resources as the original activity v_j. Thus, if $V = \{v_1, v_2, \ldots, v_k\}$, then the set of activities of $\mathcal{P}_0^{exp}$ is $V^{exp} = \{v_j^1, v_j^2, \ldots v_j^{\tau(v_j)}; j = 1, 2, \ldots k\}$.

We refer to a $|V^{exp}|$-tuple $(n_v)_{v \in V^{exp}}$ as an *independent tuple* of $\mathcal{P}_0^{exp}$ if it satisfies the following:

$$\sum_{v: w \in W_v} n_v \leqslant N_w \text{ for } w \in W, \quad \text{and} \quad n_v \geqslant 0 \text{ is an integer for } v \in V^{exp}.$$

Let $\mathcal{I}$ be the collection of all the independent tuples of $\mathcal{P}_0^{exp}$. It is easy to see that if $I = (n_v)_{v \in V^{exp}} \in \mathcal{I}$, then n_v "copies" of activity v, $v \in V^{exp}$, can be scheduled simultaneously (i.e., in the same time unit). Consider the following "cycle time" LP (whose optimal objective value will be used in Theorem 1 below):

$$\begin{aligned} \chi^*(\mathcal{P}_0^{exp}) = \min \quad & \sum_{I \in \mathcal{I}} x_I \\ \text{s.t.} \quad & \sum_{I \in \mathcal{I}} n_v x_I \geqslant 1 \quad \text{for} \quad v \in V^{exp} \\ & x_I \geqslant 0 \quad \text{for} \quad I \in \mathcal{I}. \end{aligned} \tag{P_χ}$$

The decision variables of the LP P_χ, i.e., $(x_I, I \in \mathcal{I})$, can be interpreted as the long-term proportion of time allocated to each independent tuple $I \in \mathcal{I}$, and its objective value can be interpreted as the corresponding cycle time of process $\mathcal{P}_0^{exp}$ under $(x_I, I \in \mathcal{I})$. The constraints state that each activity $v \in V^{exp}$ is executed at least once on a flow unit. Thus, the capacity of process $\mathcal{P}_0^{exp}$ is the *reciprocal* of the optimal objective value of problem P_χ.

2.4. Statement of the Main Result

THEOREM 1. *Consider an arbitrary deterministic, single-product process with batch processing and setup times:*

$$\mathcal{P} = \big(V, A, \tau, W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v, v \in V\}\big).$$

Define a corresponding process with no batching and no setup times:

$$\mathcal{P}_0(\mathcal{P}) = \big(V, A, \tau', W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v = 1, v \in V\}, \{s_v = 0, v \in V\}\big),$$

where $\tau'(v) = \frac{\tau(v)+s_v}{b_v}$. *Without loss of generality, we scale the unit of time so that* $\tau'(v) \in \mathbb{N}, \forall v \in V$. *Let* $\mathcal{P}_0^{exp}(\mathcal{P})$ *denote the expanded process of* $\mathcal{P}_0(\mathcal{P})$. *Then, we have*

$$\mu(\mathcal{P}) = \mu(\mathcal{P}_0(\mathcal{P})) = \frac{1}{\chi^*(\mathcal{P}_0^{exp}(\mathcal{P}))}. \tag{8}$$

Remark 1: The second equality in (8) follows from Bo et al. (2019) – that paper shows that the capacity of a process without batching and setups can be obtained by solving the corresponding "cycle time" LP defined above; in particular, we have $\mu(\mathcal{P}_0(\mathcal{P})) = \mu(\mathcal{P}_0^{exp}(\mathcal{P})) = \frac{1}{\chi^*(\mathcal{P}_0^{exp}(\mathcal{P}))}$. Bo et al. (2019) also obtain a graphical interpretation of $\chi^*(\mathcal{P}_0^{exp}(\mathcal{P}))$: this value is related to the fractional chromatic number of the collaboration hypergraph of process $\mathcal{P}_0(\mathcal{P})$; we avoid providing the details here and refer the reader to Bo et al. (2019) and Dawande et al. (2020). ■

Remark 2 (Comments on the simple bottleneck approach for computing process capacity): The simple "bottleneck formula", often used in textbooks to illustrate the computation of the capacity of a process, is defined as follows: First, we compute the capacity of each resource by considering that resource in isolation. Then, the bottleneck resources are those with the least capacity, which is defined to be the capacity of the process.

For processes *without* batching and setups, the following is known in the literature about this bottleneck formula:

- The capacity computed by the bottleneck formula is only an upper bound on the true capacity of the process. In general, this formula does not correctly compute process capacity. In the worst case, the bottleneck formula can be significantly inaccurate. In fact, there are examples for which the ratio of the capacity of a bottleneck resource (obtained by the bottleneck formula) to the capacity of the process is arbitrarily large (Gurvich and Van Mieghem 2015).

These observations trivially also hold for processes with batching and setups too, since the class of such processes also includes processes without batching and setups.

- There are special types of processes in which the bottleneck formula correctly computes process capacity. For example, (i) processes in which each resource is dedicated to only one activity (**No multitasking**) and (ii) processes in which each activity requires

only one resource (**No collaboration**); see Gurvich and Van Mieghem (2015) and Bo et al. (2019) for more-general characterizations of when the bottleneck formula correctly computes process capacity.

The same results also hold for processes with batching and setups, since our capacity-preserving transformation only affects the processing times of the activities and does not affect the resource(s) needed by each activity. Consequently, for these special types of processes, we can simply apply the capacity-preserving transformation and then use the bottleneck formula on the transformed process to correctly compute capacity. ■

2.5. A Brief Road Map of the Proof of Theorem 1

Theorem 1 provides us with a convenient approach to calculate the capacity of a process with batching and (activity) setup times. To be able to exploit the (known) analysis and results in the literature on process capacity (for processes *without* batching and setup times), our approach is to convert any process with batching and setup times to a "capacity-equivalent" process without these two features; that is, the two processes have the same capacity.

To this end, we first establish the following two important properties: For the purpose of determining the capacity of an arbitrary process with batching but no setup times, we can (a) restrict attention to cyclic policies; i.e., policies that repeatedly execute a fixed (and finite) processing schedule, and (b) ignore the precedence constraints. These two properties have been established in the literature (see, e.g., Bo et al. 2019) for processes *without* batching and setup times. Thus, our contribution is in proving the generalization to batch processing. In the course of this analysis, an auxiliary result we establish is that it is sufficient to restrict attention to the class of *first-come, first-served* (FCFS) policies; i.e., policies in which activities of the process are executed on the flow units in the order of their arrival. The FCFS structure makes it convenient to construct cyclic policies satisfying the batch constraints. We provide the basic reasoning behind these properties below.

Under an arbitrary policy for a process satisfying the precedence and non-preemption constraints, for each activity, we can rearrange the order of the flow units on which that activity is executed, so that the activities are scheduled on a first-come first-serve basis. In this manner, starting with an arbitrary policy, we construct an FCFS policy with equal

or higher throughput. Further, note that any acyclic, FCFS policy can be shown to be the limit of an infinite sequence of cyclic, FCFS policies. Consequently, we can restrict attention to the class of cyclic, FCFS policies. For an arbitrary cyclic, FCFS policy that violates the precedence constraints, we can keep the same steady-state schedule of activities and carefully re-number the flow units to construct a cyclic, FCFS policy that satisfies the precedence constraints and achieves the same throughput.

Next, the following two crucial steps (which we establish using properties (a) and (b) above) bring us to the point where we can utilize the available analysis in the literature:

(i) For an arbitrary process with batching and setup times, we obtain a capacity-equivalent process with batching but no setup times. Specifically, for any feasible policy in the former process, we obtain a feasible policy in the latter process with the same long-term throughput, and vice-versa.

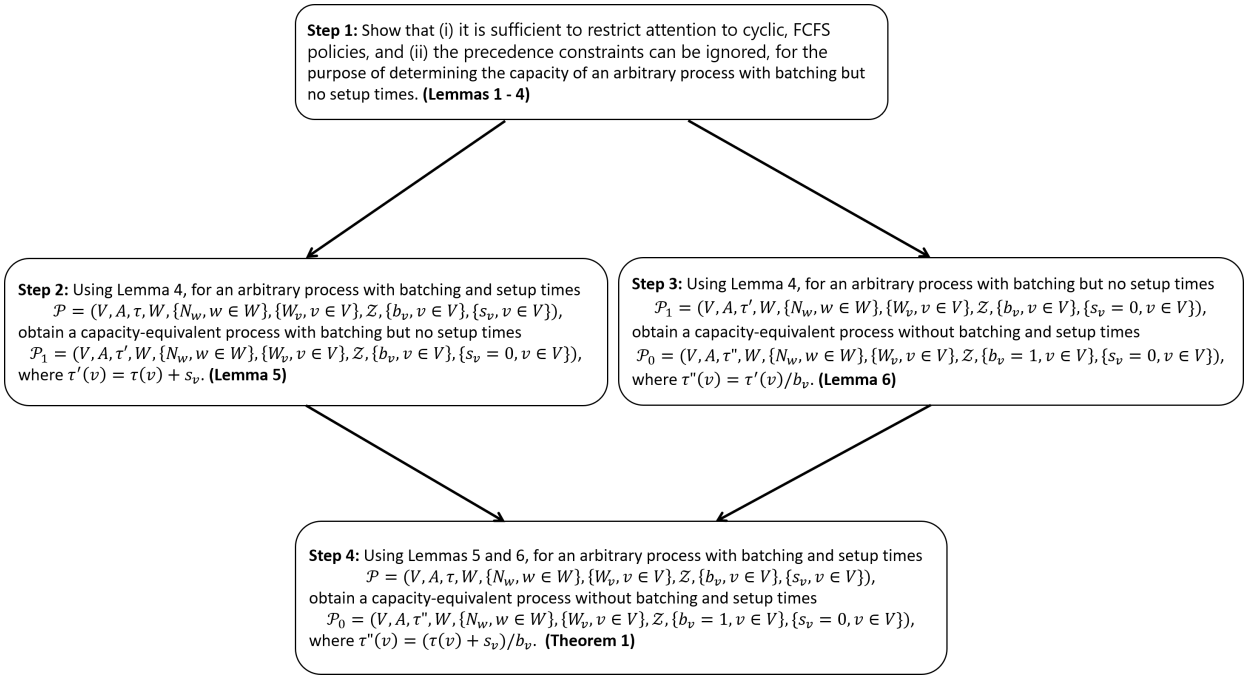
(ii) Similarly, for an arbitrary process with batching, we obtain a capacity-equivalent process without batching.

Given these transformations, we can convert any process with batching and setup times to a capacity-equivalent process that does not have these two features. Then, we can formulate the linear program P_χ to calculate the capacity of the process without batching and setups using recent developments in the literature. The flowchart diagram in Figure 5 illustrates the main steps in the proof of Theorem 1.

Remark 3: Transforming the process $\mathcal{P}$, which involves batching and setup times, to remove *only* the setups is relatively straightforward. Indeed, since a setup for an arbitrary activity $v \in V$ immediately precedes that activity and has the same resource requirements as that activity, it is intuitive that we should be able to merge the setup with the activity itself to create a "modified" activity with processing time $\tau(v) + s_v$. To show this, we only need to show that the precedence constraints can be ignored for the purpose of determining process capacity. Then, with the modified activities, we obtain a capacity-equivalent process that involves batching but no setup times. We formally show this in Lemma 5. Thus, the main contribution of Theorem 1 lies not as much in transforming $\mathcal{P}$ to remove setups but in establishing the capacity equivalence between $\mathcal{P}$ and a process with no batching. ■

The remainder of the paper provides the proof of Theorem 1.

Figure 5 Flowchart of the main steps in the proof of Theorem 1.



3. Proof of Theorem 1

The proof is organized as follows. For the purpose of determining the capacity of a process with batching but no setup times, we establish the following two results: Section 3.1 shows that it is sufficient to restrict our attention to FCFS policies. Section 3.2 shows that we can restrict our attention to cyclic policies, and the precedence constraints can be ignored. In Section 3.3, for an arbitrary process with batching and setup times, we obtain a capacity-equivalent process with batching but no setup times. In Section 3.4, for an arbitrary process with batching and no setup times, we obtain a capacity-equivalent process with no batching and no setup times.

3.1. Determining Process Capacity: Sufficiency of FCFS Policies

Consider a process $\mathcal{P}_1$ with batching but no setup times, i.e., $\mathcal{P}_1 = \big(V, A, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v = 0, v \in V\}\big)$. Since there are no setup times, a generic policy, $\pi : \mathbb{N}^2 \to \{0,1\}^{|V|}$, is defined as follows: For $j, t \in \mathbb{N}$, let

$$\pi(j,t) := (I^v_{j,t}; v \in V),$$

and π satisfies the following constraints:

(a) **Resource availability constraints**. For $t \in \mathbb{N}$ and $w \in W$, we have

$$\sum_{j\in\mathbb{N}} \sum_{v:w\in W_v} \frac{I^v_{j,t}}{b_v} \leqslant N_w. \tag{9}$$

(b) **Activity constraints**. For $j \in \mathbb{N}$ and $v \in V$, we have

$$\sum_{t\in\mathbb{N}} I^v_{j,t} = \tau(v). \tag{10}$$

(c) **Batch constraints.** For $t \in \mathbb{N}$ and $v \in V$, we have

$$\sum_{j\in\mathbb{N}} \frac{I^v_{j,t}}{b_v} \in \mathbb{Z}_+. \tag{11}$$

(d) **Precedence constraints.** For $j \in \mathbb{N}$ and $e_{kl} = (v_k, v_l) \in A$, we have

$$\max\left\{t : t \in \mathbb{N}, I^{v_k}_{j,t} = 1\right\} < \min\left\{t : t \in \mathbb{N}, I^{v_l}_{j,t} = 1\right\}. \tag{12}$$

(e) **Non-preemption constraints.** For $j, T \in \mathbb{N}$ and $v \in V$, we have

$$\mathcal{Z} \cdot I^v_{j,T} \cdot \left[\max\left\{t : t \geqslant T, \prod_{i=T}^{t} I^v_{j,i} = 1\right\} - \min\left\{t : t \leqslant T, \prod_{i=t}^{T} I^v_{j,i} = 1\right\} - \tau(v) \right] = 0. \tag{13}$$

Let Π denote the collection of all feasible policies π for process $\mathcal{P}_1$. Let $\mathcal{P}_1^1$ represent the same process as $\mathcal{P}_1$ except without precedence constraints; i.e., $\mathcal{P}_1^1 = \big(V, \varnothing, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v = 0, v \in V\}\big)$. Let Π_1 denote the collection of all feasible policies π for process $\mathcal{P}_1^1$. Then, $\Pi \subseteq \Pi_1$. Next, we show that it is sufficient to restrict our attention to FCFS policies for determining the capacity of both $\mathcal{P}_1$ and $\mathcal{P}_1^1$.

Recall that process $\mathcal{P}_1$ is defined as follows: $\mathcal{P}_1 = \big(V, A, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v = 0, v \in V\}\big)$. A FCFS policy $\pi \in \Pi$ for $\mathcal{P}_1$ satisfies the following constraints: For $j \leqslant k$ and $v \in V$,

$$\min\left\{t : t \in \mathbb{N}, I^v_{j,t} = 1\right\} \leqslant \min\left\{t : t \in \mathbb{N}, I^v_{k,t} = 1\right\}. \tag{14}$$

In words, (14) says that if flow unit j arrives before flow unit k, then each activity $v \in V$ is performed on flow unit j no later than flow unit k. Let $\tilde{\Pi}$ denote the collection of all feasible FCFS policies for $\mathcal{P}_1$. Then, we have

LEMMA 1. $\mu(\mathcal{P}_1) = \sup_{\pi\in\Pi} r^\pi = \sup_{\pi\in\tilde{\Pi}} r^\pi$.

Proof of Lemma 1: Since $\tilde{\Pi} \subseteq \Pi$, we have

$$\sup_{\pi \in \tilde{\Pi}} r^{\pi} \leqslant \sup_{\pi \in \Pi} r^{\pi}.$$

Thus, it only remains to show that

$$\sup_{\pi \in \tilde{\Pi}} r^{\pi} \geqslant \sup_{\pi \in \Pi} r^{\pi}.$$

Take an arbitrary $\pi \in \Pi$. We construct a policy $\tilde{\pi} \in \tilde{\Pi}$ that satisfies $r^{\tilde{\pi}} \geqslant r^{\pi}$. For $\pi \in \Pi$, the formal construction of policy $\tilde{\pi}$ is as follows: For $v \in V$, let $i_v^1, i_v^2, \ldots$ be the indices of the flow units, where i_v^j is the j^{th} flow unit that undergoes activity v, under π. That is,

$$\min\left\{t : t \in \mathbb{N}, I^v_{i_v^1,t} = 1\right\} \leqslant \min\left\{t : t \in \mathbb{N}, I^v_{i_v^2,t} = 1\right\} \leqslant \min\left\{t : t \in \mathbb{N}, I^v_{i_v^3,t} = 1\right\} \leqslant \ldots.$$

Under $\tilde{\pi}$, the schedule of activity v on flow unit j copies the schedule of activity v on flow unit i_v^j under π. That is,

$$\tilde{I}^v_{j,t} = I^v_{i_v^j,t} \quad \text{for} \quad t, j \in \mathbb{N}.$$

It is easy to verify that policy $\tilde{\pi}$ satisfies constraints (9)–(13) and (14). Recall that $N^{\pi}[1,T]$ is the number of flow units that enter the process from the start of time unit 1 and are completed by the end of time unit T, under policy π. Consider an arbitrary activity $v \in V$. By the definition of $N^{\pi}[1,T]$, at least $N^{\pi}[1,T]$ flow units have completed activity v by the end of time T, under policy π. Then, by the construction of policy $\tilde{\pi}$, the first $N^{\pi}[1,T]$ flow units have completed all the activities in V by the end of time unit T, under policy $\tilde{\pi}$. Therefore, under policy $\tilde{\pi}$, the number of flow units completed by the end of time unit T, is at least $N^{\pi}[1,T]$, i.e., $N^{\tilde{\pi}}[1,T] \geqslant N^{\pi}[1,T]$. Thus, $r^{\tilde{\pi}} \geqslant r^{\pi}$. ■

Recall that process $\mathcal{P}_1^1$ is defined as follows: $\mathcal{P}_1^1 = \big(V, \varnothing, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v = 0, v \in V\}\big)$. An FCFS policy $\pi \in \Pi_1$ for $\mathcal{P}_1^1$ satisfies constraints (14). Let $\tilde{\Pi}_1$ denote the collection of all feasible FCFS policies for process $\mathcal{P}_1^1$. Following an identical argument to that in the proof of Lemma 1, we can show that it is also sufficient to restrict our attention to FCFS policies for determining the capacity of $\mathcal{P}_1^1$. That is,

LEMMA 2. $\mu(\mathcal{P}_1^1) = \sup_{\pi \in \Pi_1} r^{\pi} = \sup_{\pi \in \tilde{\Pi}_1} r^{\pi}$.

3.2. Determining Process Capacity: Sufficiency of Cyclic Policies and Ignoring Precedence Constraints

This section is devoted to showing the following: In order to determine the capacity of a process $\mathcal{P}_1$ with batching but no setups, we can (a) restrict attention to cyclic policies; and (b) ignore the precedence constraints (12). The main result of this section is Lemma 4. Our analysis here is along the same lines as in Bo et al. (2019), except for some important differences that we briefly now summarize.

In our analysis, we need to construct cyclic policies or policies that satisfy precedence constraints. The manner in which we construct these policies is different from that in Bo et al. (2019). This is because our policies need to satisfy the batch constraints (11) – these constraints are not required for the processes in Bo et al. (2019). For instance, for an arbitrary feasible policy of the process $\mathcal{P}_1$ with batching but no setups, to obtain a feasible cyclic policy with the same long-term throughput, we restrict attention to FCFS policies using Lemma 1; the FCFS structure makes it convenient to construct policies that satisfy the batch constraints. Further, the batch constraints imply that any cyclic FCFS policy in $\mathcal{P}_1^1$ is a kM-unit cyclic policy for $k \in \mathbb{N}$ – this observation also helps us construct policies satisfying the batch constraints.

To the extent possible, we use the same notation as in Bo et al. (2019). First, we provide a formal definition of cyclic policies. For $k \in \mathbb{N}$, a policy π is said to be a *k-unit cyclic policy* if the following is satisfied: There exists $L^\pi \in \mathbb{N}$ such that

$$\pi(nk+j,t) = \begin{cases} \pi(j, t-nL^\pi), & \text{if } t \geqslant nL^\pi + 1, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \quad \text{for } n \in \mathbb{N} \text{ and } j \in \{1,2,\cdots,k\}. \tag{15}$$

That is, the schedule for flow units $(j-1)k+1, (j-1)k+2, \cdots, jk$, $j \in \mathbb{N}$, is exactly the same as the schedule for flow units $1,2,\cdots,k$, except with a time shift of $(j-1)L^\pi$ time units. For $i \in \mathbb{N}$, the finite sequence of activities in time units $(i-1)L^\pi+1, (i-1)L^\pi+2, \cdots, iL^\pi$, is said to be a *cycle* of the process. The integer L^π is said to be the *cycle length* of π. For $k \in \mathbb{N}$, let Π^k denote the collection of all feasible k-unit cyclic policies (i.e., satisfying constraints (9)-(13) and (15)) for process $\mathcal{P}_1$. A policy π is said to be *cyclic* if π is a k-unit cyclic policy for some $k \in \mathbb{N}$. Let Π^{cyclic} denote the collection of all *feasible* cyclic policies for process $\mathcal{P}_1$, i.e., $\Pi^{cyclic} = \cup_{k=1}^{\infty} \Pi^k$; then, $\Pi^{cyclic} \subseteq \Pi$. The result below is useful for the proof of Lemma 4.

LEMMA 3. *(Bo et al. 2019)* [1] *Consider any $k \in \mathbb{N}$ and any $\pi \in \Pi^k$ for process $\mathcal{P}_1$. There exists $n_s^\pi \in \mathbb{N}$ that satisfies the following properties:*

(a) ***Steady-State Property.***

(i) For $n \geqslant n_s^\pi$, we have

$$\sum_{j\in\mathbb{N}} \sum_{t=(n-1)L^\pi+1}^{nL^\pi} I_{j,t}^v = \sum_{j\in\mathbb{N}} \sum_{t=(n_s^\pi-1)L^\pi+1}^{n_s^\pi L^\pi} I_{j,t}^v \quad \text{for } v \in V, \tag{16}$$

where L^π is the cycle length of π. That is, from cycle n_s^π onwards, the amount of time spent on activity v, $v \in V$, is the same in every cycle.

(ii) For $n \geqslant n_s^\pi$, exactly k flow units exit the process (i.e., complete their processing) in the n^{th} cycle. Thus, we have

$$r^\pi = \lim_{T\to\infty} \frac{N^\pi[1,T]}{T} = \frac{k}{L^\pi}. \tag{17}$$

(b) ***Balanced Property.*** *We have*

$$\sum_{j\in\mathbb{N}} \sum_{t=(n-1)L^\pi+1}^{nL^\pi} I_{j,t}^v = k\tau(v) \quad \text{for } v \in V, n \geqslant n_s^\pi. \tag{18}$$

That is, from cycle n_s^π onwards, the amount of time spent on activity v, $v \in V$, equals k times the duration of activity v in every cycle.

For any process $\mathcal{P}_1$, recall that $\mathcal{P}_1^1$ is the same process as $\mathcal{P}_1$ except that it does not have any precedence constraints. Also recall that Π^{cyclic} denotes the collection of all feasible cyclic policies for process $\mathcal{P}_1$. Let Π_1^{cyclic} denote the collection of all feasible, cyclic policies (i.e., satisfying (9)–(11), (13), and (15)) for process $\mathcal{P}_1^1$. Then, $\Pi^{cyclic} \subseteq \Pi_1^{cyclic} \subseteq \Pi_1$. Lemma 4 below establishes that, to find the capacity of process $\mathcal{P}_1$, we can restrict our attention to feasible, cyclic policies and drop the precedence constraints, without loss of generality.

LEMMA 4. $\mu(\mathcal{P}_1) = \mu(\mathcal{P}_1^1) = \sup_{\pi\in\Pi_1^{cyclic}} r^\pi$.

Proof of Lemma 4: Let $\tilde{\Pi}^{cyclic}$ and $\tilde{\Pi}_1^{cyclic}$ denote the collection of all feasible, cyclic, FCFS policies for process $\mathcal{P}_1$ and $\mathcal{P}_1^1$, respectively. Let $\tilde{\Pi}^k$ and $\tilde{\Pi}_1^k$ denote the collection of all feasible, k-unit cyclic, FCFS policies for process $\mathcal{P}_1$ and $\mathcal{P}_1^1$, respectively. We will establish the following claims in the rest of the proof.

[1] The analysis in Bo et al. (2019) does not consider a batch process, as we do here. However, the batch constraints do not play any role in the proof; we therefore credit the result to Bo et al. (2019).

(a) Claim 1: $\mu(\mathcal{P}_1) = \sup_{\pi\in\Pi^{cyclic}} r^\pi = \sup_{\pi\in\tilde{\Pi}^{cyclic}} r^\pi$.

(b) Claim 2: $\sup_{\pi\in\Pi^{cyclic}} r^\pi = \sup_{\pi\in\tilde{\Pi}_1^{cyclic}} r^\pi$.

(c) Claim 3: $\mu(\mathcal{P}_1^1) = \sup_{\pi\in\Pi_1^{cyclic}} r^\pi = \sup_{\pi\in\tilde{\Pi}_1^{cyclic}} r^\pi$.

Consider an arbitrary process $\mathcal{P}_1$ with batching but no setups. Claim 1 implies that, to determine the capacity of $\mathcal{P}_1$, it is sufficient to restrict attention to the class of cyclic policies. Claim 2 implies that, to determine the capacity of $\mathcal{P}_1$, the precedence constraints can be ignored for cyclic policies. Claim 3 implies that, to determine the capacity of a process without precedence constraints, i.e., $\mu(\mathcal{P}_1^1)$, it is also sufficient to restrict attention to the class of cyclic policies. Together, these claims imply Lemma 4:

$$\mu(\mathcal{P}_1) = \sup_{\pi\in\Pi^{cyclic}} r^\pi = \sup_{\pi\in\tilde{\Pi}_1^{cyclic}} r^\pi = \mu(\mathcal{P}_1^1) = \sup_{\pi\in\Pi_1^{cyclic}} r^\pi.$$

The first equality holds by Claim 1. The second equality holds by Claim 2. The last two equalities hold by Claim 3.

Before establishing the four claims, we show the following result, which will be used later in the proof of Claim 2. Recall that M is the least common multiple of all the batch sizes. We have

$$\tilde{\Pi}_1^{cyclic} = \cup_{k=1}^{\infty} \tilde{\Pi}_1^{kM}. \tag{19}$$

Derivation of (19)**:** It is straightforward that $\cup_{k=1}^{\infty} \tilde{\Pi}_1^{kM} \subseteq \tilde{\Pi}_1^{cyclic}$. Then, we only have to show that

$$\tilde{\Pi}_1^{cyclic} \subseteq \cup_{k=1}^{\infty} \tilde{\Pi}_1^{kM}.$$

Suppose not, i.e., there exists $\pi \in \tilde{\Pi}_1^{\hat{k}}$, where $\hat{k} \in \mathbb{N}$ and $\frac{\hat{k}}{M} \notin \mathbb{N}$. Then, there exists $v \in V$ such that $\frac{\hat{k}}{b_v} \notin \mathbb{N}$. Therefore, $\hat{k} = xb_v + y$ for $x \in \mathbb{N}$ and $1 \leqslant y < b_v$. Then, activity v is executed on flow units $lb_v + 1, \cdots, (l+1)b_v$ simultaneously for $l = 0, 1, \cdots, x-1$ under π. Next, we show that the batch constraints (11) do not hold. For $j, t \in \mathbb{N}$, let $\pi(j,t) = (I_{j,t}^v; v \in V)$. Note that

$$\sum_{j\in\mathbb{N}} \frac{I_{j,t}^v}{b_v} = \sum_{n\in\mathbb{N}} \sum_{j=(n-1)\hat{k}+1}^{(n-1)\hat{k}+xb_v} \frac{I_{j,t}^v}{b_v} + \sum_{n\in\mathbb{N}} \sum_{j=(n-1)\hat{k}+xb_v+1}^{n\hat{k}} \frac{I_{j,t}^v}{b_v}. \tag{20}$$

We show that there exists a specific time unit t' for which the first part on the right-hand-side of (20) is an integer whereas the second part is not an integer. Let L^π be the cycle

length of π. Let us analyze the first part on the right-hand-side of (20). For an arbitrary $t \in \mathbb{N}$, we have

$$\sum_{j=(n-1)\hat{k}+1}^{(n-1)\hat{k}+xb_v} \frac{I^v_{j,t}}{b_v} = \sum_{j=1}^{xb_v} \frac{I^v_{j,t-(n-1)L^\pi}}{b_v} = \sum_{l=0}^{x-1} \sum_{j=lb_v+1}^{(l+1)b_v} \frac{I^v_{j,t-(n-1)L^\pi}}{b_v}.$$

Since activity v is executed on flow units $lb_v+1, \cdots, (l+1)b_v$ simultaneously for $l = 0, 1, \cdots, x-1$, $\sum_{j=lb_v+1}^{(l+1)b_v} \frac{I^v_{j,t-(n-1)L^\pi}}{b_v}$ equals 0 or 1. Therefore, the first part is an integer for all $t \in \mathbb{N}$.

We now analyze the second part on the right-hand-side of (20). Let $t' = \min\{t : \exists\ j \in \{xb_v+1, \cdots, xb_v+y\},\ \text{s.t. } I^v_{j,t} = 1\}$, i.e. t' is the first time unit such that $I^v_{j,t} = 1$ for some $j \in \{xb_v+1, \cdots, xb_v+y\}$. Then, we have

$$I^v_{j,t} = 0 \text{ for } j \in \{xb_v+1, \cdots, xb_v+y\} \text{ and } t < t'. \tag{21}$$

Consequently, for $t = t'$, the second part satisfies

$$\sum_{n\in\mathbb{N}} \sum_{j=(n-1)\hat{k}+xb_v+1}^{n\hat{k}} \frac{I^v_{j,t'}}{b_v} = \sum_{j=xb_v+1}^{xb_v+y} \frac{I^v_{j,t'}}{b_v} + \sum_{n\in\mathbb{N}:n\geqslant 2} \sum_{j=xb_v+1}^{xb_v+y} \frac{I^v_{j,t'-(n-1)L^\pi}}{b_v} = \sum_{j=xb_v+1}^{xb_v+y} \frac{I^v_{j,t'}}{b_v}.$$

The second equality holds by (21). It is clear that $0 < \sum_{j=xb_v+1}^{xb_v+y} \frac{I^v_{j,t'}}{b_v} < 1$, since $I^v_{j,t'} = 1$ for some $j \in \{xb_v+1, \cdots, xb_v+y\}$ and $1 \leqslant y < b_v$. Therefore, for $t = t'$, $\sum_{j\in\mathbb{N}} \frac{I^v_{j,t'}}{b_v}$ is not an integer, which contradicts the batch constraints. ■

- **Proof of claim 1:** Since $\tilde{\Pi}^{cyclic} \subseteq \Pi^{cyclic} \subseteq \Pi$, we have

$$\sup_{\pi\in\tilde{\Pi}^{cyclic}} r^\pi \leqslant \sup_{\pi\in\Pi^{cyclic}} r^\pi \leqslant \mu(\mathcal{P}_1).$$

Then, we only have to show that

$$\sup_{\pi\in\tilde{\Pi}^{cycle}} r^\pi > \mu(\mathcal{P}_1) - \epsilon \quad \text{for all } \epsilon > 0. \tag{22}$$

Take an arbitrary constant $\epsilon > 0$. By Lemma 1 and (7), there exists $\pi^* \in \tilde{\Pi}$ and $T^* \in \mathbb{N}$ such that

$$\frac{\hat{N}^{\pi^*}[1, T^*]}{T^*} \cdot M > \mu(\mathcal{P}_1) - \epsilon. \tag{23}$$

Let $k^* = \hat{N}^{\pi^*}[1, T^*]$. Note that $1, 2, \cdots, Mk^*$, are the indices of the first Mk^* flow units that exit the process by the end of time unit T^*, under π^*. We construct a policy $\breve{\pi}$ as follows:

$$\breve{\pi}(nMk^* + j, t) := \begin{cases} \pi^*(j, t - nT^*), & \text{if } t \geqslant nT^* + 1, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad \text{for } n \in \mathbb{N} \cup \{0\} \text{ and } j \in \{1, 2, \cdots, Mk^*\}.$$

It is easy to see that $\breve{\pi} \in \tilde{\Pi}^{cyclic}$. By applying statement (a) of Lemma 3 and using (23), we have

$$r^{\breve{\pi}} = \frac{Mk^*}{T^*} > \mu(\mathcal{P}_1) - \epsilon.$$

which implies (22). Thus, we have established claim 1. ■

- **Proof of claim 2:** See Appendix A. ■
- **Proof of claim 3:** Since $\tilde{\Pi}_1^{cyclic} \subseteq \Pi_1^{cyclic} \subseteq \Pi_1$, we have

$$\sup_{\pi \in \tilde{\Pi}_1^{cyclic}} r^\pi \leqslant \sup_{\pi \in \Pi_1^{cyclic}} r^\pi \leqslant \mu(\mathcal{P}_1^1),$$

Then, we only have to show that

$$\sup_{\pi \in \tilde{\Pi}_1^{cycle}} r^\pi > \mu(\mathcal{P}_1^1) - \epsilon \quad \text{for all } \epsilon > 0. \tag{24}$$

Take an arbitrary constant $\epsilon > 0$. By Lemma 2, there exists $\pi_1^* \in \tilde{\Pi}_1$ and $T_1^* \in \mathbb{N}$ such that

$$\frac{\hat{N}^{\pi_1^*}[1, T_1^*]}{T_1^*} \cdot M > \mu(\mathcal{P}_1^1) - \epsilon. \tag{25}$$

Let $k_1^* = \hat{N}^{\pi_1^*}[1, T_1^*]$. Note that $1, 2, \cdots, Mk_1^*$, are the indices of the first Mk_1^* flow units that exit the process by the end of time unit T_1^*, under π_1^*. We construct a policy $\bar{\pi}$ as follows:

$$\bar{\pi}(nMk_1^* + j, t) := \begin{cases} \pi_1^*(j, t - nT_1^*), & \text{if } t \geqslant nT_1^* + 1, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad \text{for } n \in \mathbb{N} \cup \{0\} \text{ and } j \in \{1, 2, \cdots, Mk_1^*\}.$$

It is easy to see that $\bar{\pi} \in \tilde{\Pi}_1^{cyclic}$. By applying statement (a) of Lemma 3 and using (25), we have

$$r^{\bar{\pi}} = \frac{Mk_1^*}{T_1^*} > \mu(\mathcal{P}_1^1) - \epsilon.$$

which implies (24). Thus, we have established claim 3. ■

3.3. Capacity-Equivalent Process with Batching but No Setup Times

In this section, for an arbitrary process with batching and setup times, we obtain a capacity-equivalent process with batching but no setup times. Recall that an arbitrary process with batching and setup times is defined as $\mathcal{P} = \big(V, A, \tau, W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v, v \in V\}\big)$. Let $\mathcal{P}_1(\mathcal{P}) = \big(V, A, \tau', W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v = 0, v \in V\}\big)$, where $\tau'(v) = \tau(v) + s_v$, denote a corresponding process with batching but no setup times. Next, we show that the capacities of these two processes are equal, i.e., $\mu(\mathcal{P}) = \mu(\mathcal{P}_1(\mathcal{P}))$.

LEMMA 5. $\mu(\mathcal{P}) = \mu(\mathcal{P}_1(\mathcal{P}))$.

Proof of Lemma 5: Let $\mathcal{P}_1^1(\mathcal{P})$ represent the same process as $\mathcal{P}_1(\mathcal{P})$ except without precedence constraints. By Lemma 4, we have $\mu(\mathcal{P}_1(\mathcal{P})) = \mu(\mathcal{P}_1^1(\mathcal{P}))$. Let Π (resp., Π_1) denote the set of all feasible policies for process $\mathcal{P}_1(\mathcal{P})$ (resp., $\mathcal{P}_1^1(\mathcal{P})$). Let Ψ denote the set of all feasible policies for process $\mathcal{P}$.

First, we show that

$$\sup_{\psi \in \Psi} r^{\psi} \geqslant \sup_{\pi \in \Pi} r^{\pi}.$$

Take an arbitrary $\pi \in \Pi$. We construct a policy $\psi(\pi) \in \Psi$ that satisfies

$$r^{\psi(\pi)} \geqslant r^{\pi}.$$

For $j, t \in \mathbb{N}$, let $\pi(j,t) = (I_{j,t}^v; v \in V)$. The activity constraints and non-preemption constraints together imply that for $j \in \mathbb{N}$ and $v \in V$, there exists $T(j,v) \in \mathbb{N}$, the first time unit for which activity v is on flow unit j under policy π, such that

$$I_{j,t}^v = \begin{cases} 1 & \text{if } t = T(j,v), T(j,v)+1, \cdots, T(j,v)+\tau(v)+s_v-1; \\ 0 & \text{otherwise.} \end{cases}$$

We construct a policy $\psi(\pi)$: for $j \in \mathbb{N}$ and $v \in V$, let

$$\hat{I}_{j,t}^v = \begin{cases} 1 & \text{if } t = T(j,v)+s_v, T(j,v)+s_v+1, \cdots, T(j,v)+\tau(v)+s_v-1; \\ 0 & \text{otherwise.} \end{cases} \quad \text{and}$$

$$\hat{S}_{j,t}^v = \begin{cases} 1 & \text{if } t = T(j,v), T(j,v)+1, \cdots, T(j,v)+s_v-1; \\ 0 & \text{otherwise.} \end{cases}$$

For $j,t\in\mathbb{N}$, let $\psi(\pi)(j,t)=(\hat{I}^v_{j,t},\hat{S}^v_{j,t};v\in V)$. It is easy to see that $\psi(\pi)\in\Psi$ and $r^{\psi(\pi)}=r^\pi$.

Next, we show that

$$\sup_{\pi\in\Pi_1} r^\pi=\sup_{\pi\in\Pi} r^\pi\geqslant \sup_{\psi(\pi)\in\Psi} r^\psi.$$

Take an arbitrary $\psi\in\Psi$. We construct a policy $\pi(\psi)\in\Pi_1$ that satisfies

$$r^{\pi(\psi)}\geqslant r^\psi.$$

For $j,t\in\mathbb{N}$, let $\psi(j,t)=(I^v_{j,t},S^v_{j,t};v\in V)$. We construct a policy $\pi(\psi)$: for $j,t\in\mathbb{N}$ and $v\in V$, let

$$\check{I}^v_{j,t}=\begin{cases}1 & \text{if } I^v_{j,t}=1 \text{ or } S^v_{j,t}=1;\\ 0 & \text{otherwise.}\end{cases}$$

For $j,t\in\mathbb{N}$, let $\pi(\psi)(j,t)=(\check{I}^v_{j,t};v\in V)$. It is easy to see that $\pi(\psi)\in\Pi_1$ and $r^{\pi(\psi)}=r^\psi$. ■

3.4. Capacity-Equivalent Process with No Batching and No Setup Times

In this section, for an arbitrary process with batching but no setup times, we obtain a capacity-equivalent process with no batching and no setup times. Recall that an arbitrary process with batching but no setup times is defined as $\mathcal{P}_1=\big(V,A,\tau,W,\{N_w,w\in W\},\{W_v,v\in V\},\mathcal{Z},\{b_v,v\in V\},\{s_v=0,v\in V\}\big)$. Let $\mathcal{P}_0(\mathcal{P}_1)=\big(V,A,\tau',W,\{N_w,w\in W\},\{W_v,v\in V\},\mathcal{Z},\{b_v=1,v\in V\},\{s_v=0,v\in V\}\big)$, where $\tau'(v)=\tau(v)/b_v$, denote a corresponding process with no batching and no setup times. We now show that the capacities of these two processes are equal, i.e., $\mu(\mathcal{P}_1)=\mu(\mathcal{P}_0(\mathcal{P}_1))$.

LEMMA 6. $\mu(\mathcal{P}_1)=\mu(\mathcal{P}_0(\mathcal{P}_1))$.

Proof of Lemma 6: Let $\mathcal{P}^1_1$ (resp., $\mathcal{P}^1_0(\mathcal{P}_1)$) represent the same process as $\mathcal{P}_1$ (resp., $\mathcal{P}_0(\mathcal{P}_1)$) except without precedence constraints. By Lemma 4, we have $\mu(\mathcal{P}_1)=\mu(\mathcal{P}^1_1)$ and $\mu(\mathcal{P}_0(\mathcal{P}_1))=\mu(\mathcal{P}^1_0(\mathcal{P}_1))$. Let Π (resp., Π_1) denote all feasible policies for process $\mathcal{P}_1$ (resp., $\mathcal{P}^1_1$). Let $\tilde{\Pi}$ denote all feasible FCFS policies for process $\mathcal{P}_1$. Let Φ (resp., Φ_1) denote all feasible policies for process $\mathcal{P}_0(\mathcal{P}_1)$ (resp., $\mathcal{P}^1_0(\mathcal{P}_1)$). Let Φ^{cyclic} denote all feasible cyclic policies for process $\mathcal{P}_0(\mathcal{P}_1)$. By Lemma 4, $\mu(\mathcal{P}_0(\mathcal{P}_1))=\sup_{\phi\in\Phi^{cyclic}} r^\phi$.

First, we show that

$$\mu(\mathcal{P}_0(\mathcal{P}_1))=\mu(\mathcal{P}^1_0(\mathcal{P}_1))=\sup_{\phi\in\Phi_1} r^\phi\geqslant\sup_{\pi\in\tilde{\Pi}} r^\pi=\mu(\mathcal{P}_1).$$

Take an arbitrary $\pi \in \tilde{\Pi}$. We construct a policy $\phi(\pi) \in \Phi_1$ that satisfies

$$r^{\phi(\pi)} \geqslant r^{\pi}.$$

For $j, t \in \mathbb{N}$, let $\pi(j,t) = (I^v_{j,t}; v \in V)$. The activity constraints and non-preemption constraints together imply that for $j \in \mathbb{N}$ and $v \in V$, there exists $T(j,v) \in \mathbb{N}$, the first time unit for which activity v is on flow unit j under policy π, such that

$$I^v_{j,t} = \begin{cases} 1 & \text{if } t = T(j,v), T(j,v)+1, \cdots, T(j,v)+\tau(v)-1; \\ 0 & \text{otherwise.} \end{cases}$$

For a FSFC policy π, the batch constraints imply that $T((k-1)b_v+1, v) = \cdots = T(kb_v, v)$ for all $k \in \mathbb{N}$.

We construct a policy $\phi(\pi)$: for $k \in \mathbb{N}$, $l \in \{1, \cdots, b_v\}$, and $v \in V$, let

$$\hat{I}^v_{(k-1)b_v+l,t} = \begin{cases} 1 & \text{if } t = T((k-1)b_v+1, v) + (l-1)\frac{\tau(v)}{b_v}, \cdots, T((k-1)b_v+1, v) + l\frac{\tau(v)}{b_v} - 1; \\ 0 & \text{otherwise.} \end{cases}$$

For $j, t \in \mathbb{N}$, let $\phi(\pi)(j,t) = (\hat{I}^v_{j,t}\, v \in V)$. It is easy to see that $\phi(\pi) \in \Phi_1$ and $r^{\phi(\pi)} = r^{\pi}$.

Next, we show that

$$\mu(\mathcal{P}_1) = \mu(\mathcal{P}_1^1) = \sup_{\pi \in \Pi_1} r^{\pi} \geqslant \sup_{\phi \in \Phi^{cyclic}} r^{\phi} = \mu(\mathcal{P}_0(\mathcal{P}_1)).$$

Take an arbitrary $\phi \in \Phi^{cyclic}$. There exists $k \in \mathbb{N}$ such that ϕ is a k-unit cyclic policy. Let L denote the length of the corresponding cycle. We construct a cyclic policy $\pi(\phi) \in \Pi_1$ that satisfies

$$r^{\pi(\phi)} \geqslant r^{\phi}.$$

The policy $\pi(\phi)$ will be constructed in the following manner: Let H be the least common multiple of the processing times $\tau(v)$ of all activities $v \in V$. In every time unit t, we repeat the activities under ϕ H times under $\pi(\phi)$. This is to make sure that $\pi(\phi)$ satisfies the non-preemption constraints. Then, for each activity v on a flow unit under ϕ, we repeat that activity on b_v flow units under $\pi(\phi)$. This ensures that $\pi(\phi)$ satisfies the batch constraints. The formal construction of a policy $\pi(\phi)$ is as follows:

(a) Let H denote the least common multiple of $\tau(v)$, $v \in V$. That is,

$$H = \min\{H' : H' \in \mathbb{N},\ \frac{H'}{\tau(v)} \in \mathbb{N} \text{ for } v \in V\}.$$

For $j, t \in \mathbb{N}$, let $\phi(j,t) = (I^v_{j,t}; v \in V)$. The activity constraints and non-preemption constraints together imply that for $j \in \mathbb{N}$ and $v \in V$, there exists $\hat{T}(j,v) \in \mathbb{N}$, the first time unit for which activity v is on flow unit j under policy ϕ, such that

$$I^v_{j,t} = \begin{cases} 1 & \text{if } t = \hat{T}(j,v), \hat{T}(j,v)+1, \cdots, \hat{T}(j,v) + \frac{\tau(v)}{b_v} - 1; \\ 0 & \text{otherwise.} \end{cases}$$

(b) We now define the schedule for flow units $1, 2, \cdots, Hk$. For $v \in V$, let

$$h_v = \frac{H}{\tau(v)}.$$

For $j \in \{1, 2, \cdots, k\}$, $v \in V$, $\beta \in \{0, 1, \cdots, \frac{\tau(v)}{b_v} - 1\}$, $\theta \in \{0, 1, \cdots, h_v - 1\}$, and $l \in \{1, 2, \cdots, b_v\}$, define $p(j,v,\beta,\theta,l) = H \cdot (j-1) + \beta h_v b_v + \theta b_v + l$ and $q(j,v,\beta,\theta) = H \cdot (\hat{T}(j,v) + \beta - 1) + \theta\tau(v)$, and

$$\check{I}^v_{p(j,v,\beta,\theta,l), q(j,v,\beta,\theta)+\xi} = \begin{cases} 1, & \text{if } \xi \in \{1, 2, \cdots, \tau(v)\}, \\ 0, & \text{otherwise.} \end{cases}$$

That is, activity v is scheduled on flow units $H \cdot (j-1) + \beta h_v b_v + \theta b_v + 1$ through $H \cdot (j-1) + \beta h_v b_v + (\theta+1) b_v$ in time units $H \cdot (\hat{T}(j,v) + \beta - 1) + \theta\tau(v) + 1$ through $H \cdot (\hat{T}(j,v) + \beta - 1) + (\theta+1)\tau(v)$.

(c) We now completely define $\pi(\phi)$:

$$\pi(\phi)(nHk + j, t) := \begin{cases} \pi(\phi)(j, t - nHL), & \text{if } t \geqslant nHL + 1, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad \text{for } n \in \mathbb{N} \text{ and } j \in \{1, 2, \cdots, Hk\}.$$

It is easy to see that $\pi(\phi) \in \Pi_1$. By statement (a) of Lemma 3, the process rate is

$$r^{\pi(\phi)} = \frac{Hk}{HL} = \frac{k}{L} = r^{\phi}.$$

■

Recall that an arbitrary process with batch processing and setup times is defined as $\mathcal{P} = \big(V, A, \tau, W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v, v \in V\}\big)$. Let $\mathcal{P}_0(\mathcal{P}) = \big(V, A, \tau', W, \{N_w; w \in W\}, \{W_v; v \in V\}, \mathcal{Z}, \{b_v = 1, v \in V\}, \{s_v = 0, v \in V\}\big)$, where $\tau'(v) =$

$\frac{\tau(v)+s_v}{b_v}$, be a corresponding process with no batching and no setup times. Combining Lemmas 5 and 6, we have $\mu(\mathcal{P}) = \mu(\mathcal{P}_0(\mathcal{P}))$.

To complete the proof of Theorem 1, recall from Remark 1 that the capacity of process $\mathcal{P}_0(\mathcal{P})$ can be obtained by solving the LP P_χ (Section 2.3), and $\mu(\mathcal{P}_0(\mathcal{P})) = \mu(\mathcal{P}_0^{exp}(\mathcal{P})) = \frac{1}{\chi^*(\mathcal{P}_0^{exp}(\mathcal{P}))}$, where $\chi^*(\mathcal{P}_0^{exp}(\mathcal{P}))$ is the optimal objective value of P_χ. Thus, $\mu(\mathcal{P}) = \mu(\mathcal{P}_0(\mathcal{P})) = \frac{1}{\chi^*(\mathcal{P}_0^{exp}(\mathcal{P}))}$.

Remark 4 (Generalization on the frequency of setups): For $v \in V$, suppose that a setup time s_v needs to be incurred after every $k_v \in \mathbb{N}$ batches of flow units complete activity v using the same set of resources. Define such a process as a 10-tuple $\mathcal{Q} = \big(V, A, \tau, W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}, \{b_v, v \in V\}, \{s_v, v \in V\}, \{k_v, v \in V\}\big)$. Consider a corresponding process with no batching and no setups $\mathcal{P}_0(\mathcal{Q}) = \big(V, A, \tau', W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}, \{b_v = 1, v \in V\}, \{s_v = 0, v \in V\}\big)$, where $\tau'(v) = \frac{\tau(v)}{b_v} + \frac{s_v}{k_v b_v}$. Then, using a proof technique similar to the one used to establish Theorem 1, it can be shown that the capacities of these two processes are equal, i.e., $\mu(\mathcal{Q}) = \mu(\mathcal{P}_0(\mathcal{Q}))$. The capacity of the latter process, $\mu(\mathcal{P}_0(\mathcal{Q}))$, can be computed by solving a linear program, as described in Section 2.3.

References

Abadi, I. K., N. G. Hall, C. Sriskandarajah. 2000. Minimizing Cycle Time in a Blocking Flowshop. *Oper. Res.* **48**(1) 177–180.

Anupindi, R., S. Deshmukh, J. A. Van Mieghem, E. Zemel. 2012. *Managing Business Process Flows,* 3rd Edition. Pearson.

Bo, Y., M. Dawande, T. Huh, G. Janakiraman, M. Nagarajan. 2019. Determining Process Capacity: Intractability and Efficient Special Cases. *Manufacturing & Service Oper. Management* **21**(1) 139–153.

Cachon, G., C. Terwiesch. 2013. *Matching Supply with Demand, 3rd Edition*. McGraw-Hill.

Crama, Y., J. Van De Klundert. 1997. Cyclic Scheduling of Identical Parts in a Robotic Cell. *Oper. Res.* **45**(6) 952–965.

Dawande, M., Z. Feng, G. Janakiraman. 2020. On the Structure of Bottlenecks in Processes. *Management Science* (to appear). *Available at* `https://doi.org/10.1287/mnsc.2020.3704` .

Dawande, M., C. Sriskandarajah, S. Sethi. 2002. On Throughput Maximization in Constant Travel-Time Robotic Cells. *Manufacturing & Service Oper. Management* **4**(4) 296–312.

Geismar, N., L. M. A. Chan, M. Dawande, C. Sriskandarajah. 2008. Approximation Algorithms for Optimal k-Unit Cycles in Dual-Gripper Robotic Cells. *Production and Operations Management* **17**(5) 551–563.

Geismar, N., M. Dawande, C. Sriskandarajah. 2006. Throughput Optimization in Constant Travel-time Dual Gripper Robotic Cells with Parallel Machines. *Production and Operations Management* **15**(2) 311–328.

Gurvich, I., J. A. Van Mieghem. 2015. Collaboration and Multitasking in Networks: Architectures, Bottlenecks, and Capacity. *Manufacturing & Service Oper. Management* **17**(1) 16–33.

Gurvich, I., J. A. Van Mieghem. 2017. Collaboration and Multitasking in Networks: Prioritization and Achievable Capacity. *Management Science* **64**(5) 2390–2406.

Hopp, W., M. Spearman. 2011. *Factory Physics*. Waveland Pr Inc.

Kreipl, S., M. Pinedo. 2004. Planning and Scheduling in Supply Chains: An Overview of Issues in Practice. *Production and Operations Management* **13**(1) 77–92.

Lee, T. E., M. E. Posner. 1997. Performance Measures and Schedules in Periodic Job Shops. *Operations Research* **45**(1) 72–91.

Matsuo, H., J. S. Shang, R. S. Sullivan. 1991. A Crane Scheduling Problem in a Computer-Integrated Manufacturing Environment. *Management Science* **37**(5) 587–606.

McCormick, S. T., M. L. Pinedo, S. Shenker, B. Wolf. 1989. Sequencing in an Assembly Line with Blocking to Minimize Cycle Time. *Operations Research* **37**(6) 925–935.

Pinedo, M. L. 2009. *Planning and Scheduling in Manufacturing and Services*. Springer.

Pound, E., J. Bell, M. Spearman. 2014. *Factory Physics for Managers*. McGraw-Hill.

Schroeder, B. 2016. *Ordered Sets: An Introduction with Connections from Combinatorics to Topology*. Birkhauser.

Stevenson, W. J. 2017. *Operations Management: Theory and Practice, 13th Edition*. McGraw-Hill Irwin.

Online Appendix

Appendix A Proof of Claim 2 of Lemma 4

Proof of claim 2: Since $\tilde{\Pi}^{cyclic} \subseteq \tilde{\Pi}_1^{cyclic}$, we have

$$\sup_{\pi \in \Pi^{cyclic}} r^{\pi} = \sup_{\pi \in \tilde{\Pi}^{cyclic}} r^{\pi} \leqslant \sup_{\pi \in \tilde{\Pi}_1^{cyclic}} r^{\pi}.$$

Then, we only have to show that

$$\sup_{\pi \in \Pi^{cyclic}} r^{\pi} \geqslant \sup_{\pi \in \tilde{\Pi}_1^{cyclic}} r^{\pi}.$$

For the set of precedence constraints, A, we note that there exists another set of precedence constraints, A_{total}, that satisfies (i) $A_{total} = \{(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \cdots, (v_{i_{|V|-1}}, v_{i_{|V|}})\}$ for some permutation $\{i_1, i_2, \cdots, i_{|V|}\}$ of $\{1, 2, \cdots, |V|\}$, and (ii) A_{total} respects all the precedence constraints in A. This is because A represents a *partial order*, and for every partial order, there exists a *total order* that respects all the ordering relationships of this partial order; for details, see, e.g., Schroeder (2016). We assume $A_{total} = \{(v_1, v_2), (v_2, v_3), \cdots, (v_{|V|-1}, v_{|V|})\}$ without loss of generality. Let Π_{total}^{cyclic} be the collection of all $\pi \in \Pi^{cyclic}$ such that π satisfies the precedence constraints A_{total}. Then, $\Pi_{total}^{cyclic} \subseteq \Pi^{cyclic}$, and we have

$$\sup_{\pi \in \Pi_{total}^{cyclic}} r^{\pi} \leqslant \sup_{\pi \in \Pi^{cyclic}} r^{\pi}.$$

Thus, to establish claim 2, we only have to show that

$$\sup_{\pi \in \Pi_{total}^{cyclic}} r^{\pi} \geqslant \sup_{\pi \in \tilde{\Pi}_1^{cyclic}} r^{\pi}. \tag{26}$$

Take an arbitrary $\pi_1 \in \tilde{\Pi}_1^{cyclic}$. Let L_1 denote the length of the cycle. To show (26), we only have to construct a policy $\hat{\pi} \in \Pi_{total}^{cyclic}$ that satisfies

$$r^{\hat{\pi}} \geqslant r^{\pi_1}.$$

For $\pi_1 \in \tilde{\Pi}_1^{cyclic}$, by (19), there exists $k_1 \in \mathbb{N}$ such that $\pi_1 \in \tilde{\Pi}_1^{k_1 M}$. The construction of the policy $\hat{\pi}$ is discussed in Steps 1 and 2 below. Step 1 defines the schedule for flow units $1, 2, \cdots, k_1 M$. Step 2 defines the schedule for flow units $(\eta - 1)k_1 M + 1$ through $\eta k_1 M$, $\eta \geqslant 2$. The formal construction of a $k_1 M$-unit cyclic policy $\hat{\pi}$ is as follows:

1. We first define the schedule for flow units $1, 2, \cdots, k_1 M$. We keep the same steady-state schedule of activities as in π_1 and perform these activities on carefully chosen flow units such that the precedence constraints are satisfied. Let $V = \{v_1, v_2, \cdots, v_{|V|}\}$ without loss of generality. For $j, t \in \mathbb{N}$, let $\pi_1(j,t) = (I_{j,t}^{v_x}; x \in \{1, 2, \cdots, |V|\})$.

   The schedule for flow units $1, 2, \cdots, k_1 M$, under $\hat{\pi}$ is as follows:

   (a) We first define the schedule of activity v_1. Let $n_j^1 = 0$. The schedule of activity v_1 on flow unit j under $\hat{\pi}$ copies the schedule of activity v_1 on flow unit $n_j^1 k_1 M + j$ under π_1. That is,

   $$\hat{I}_{j,t}^{v_1} = I_{n_j^1 k_1 M + j, t}^{v_1} \quad \text{for} \quad t \in \mathbb{N}.$$

   (b) We then define the schedule of activity v_2. Under $\hat{\pi}$, activity v_2 should be on flow units $lb_{v_2} + 1, \cdots, (l+1)b_{v_2}$ simultaneously for $l \in \{0, 1, \cdots, \frac{k_1 M}{b_{v_2}} - 1\}$. Therefore, we let activity v_2 on flow units $lb_{v_2} + 1, \cdots, (l+1)b_{v_2}$ begin after activity v_1 on flow unit $(l+1)b_{v_2}$ ends, under $\hat{\pi}$. Below is the formal construction:

   For $j \in \{lb_{v_2} + 1 : l = 0, 1, \cdots, \frac{k_1 M}{b_{v_2}} - 1\}$, let n_j^2 be any positive integer such that activity v_2 on flow unit $n_j^2 k_1 M + j$ under π_1 begins after activity v_1 on flow unit $j + b_{v_2} - 1$ under $\hat{\pi}$ ends. That is, n_j^2 satisfies

   $$\min\{t : I_{n_j^2 k_1 M + j, t}^{v_2} = 1\} > \max\{t : \hat{I}_{j + b_{v_2} - 1, t}^{v_1} = 1\}.$$

   Let $n_{j+1}^2 = \cdots = n_{j + b_{v_2} - 1}^2 = n_j^2$. Then, let the schedule of activity v_2 on flow unit j under $\hat{\pi}$ copy the schedule of activity v_2 on flow unit $n_j^2 k_1 M + j$ under π_1. That is,

   $$\hat{I}_{j,t}^{v_2} = I_{n_j^2 k_1 M + j, t}^{v_2} \quad \text{for} \quad t \in \mathbb{N} \text{ and } j = 1, 2, \cdots, k_1 M.$$

   (c) In general, once we have defined n_j^x and $\hat{I}_{j,t}^{v_x}$ for $j = 1, 2, \cdots, k_1 M$, we define the schedule of activity v_{x+1}. Under $\hat{\pi}$, activity v_{x+1} should be on flow units $lb_{v_{x+1}} + 1, \cdots, (l+1)b_{v_{x+1}}$ simultaneously for $l \in \{0, 1, \cdots, \frac{k_1 M}{b_{v_{x+1}}} - 1\}$. Therefore, we let activity v_{x+1} on flow units $lb_{v_{x+1}} + 1, \cdots, (l+1)b_{v_{x+1}}$ begin after activity v_x on flow unit $(l+1)b_{v_{x+1}}$ ends, under $\hat{\pi}$. Below is the formal construction:

   For $j \in \{lb_{v_{x+1}} + 1 : l = 0, 1, \cdots, \frac{k_1 M}{b_{v_{x+1}}} - 1\}$, let n_j^{x+1} be any positive integer such that activity v_{x+1} on flow unit $n_j^{x+1} k_1 M + j$ under π_1 begins after activity v_x on flow unit $j + b_{v_{x+1}} - 1$ under $\hat{\pi}$ ends. That is, n_j^{x+1} satisfies

   $$\min\{t : I_{n_j^{x+1} k_1 M + j, t}^{v_{x+1}} = 1\} > \max\{t : \hat{I}_{j + b_{v_{x+1}} - 1, t}^{v_x} = 1\}.$$

Let $n_{j+1}^{x+1} = \cdots = n_{j+b_{v_{x+1}}-1}^{x+1} = n_j^{x+1}$. Then, let the schedule of activity v_{x+1} on flow unit j under $\hat{\pi}$ copy the schedule of activity v_{x+1} on flow unit $n_j^{x+1}k_1M+j$ under π_1. That is,

$$\hat{I}_{j,t}^{v_{x+1}} = I_{n_j^{x+1}k_1M+j,t}^{v_{x+1}} \quad \text{for} \quad t \in \mathbb{N} \text{ and } j = 1, 2, \cdots, k_1M.$$

(d) The above procedure ends when we have defined n_j^x and $\hat{I}_{j,t}^{v_x}$ for $x \in \{1, 2, \cdots, |V|\}$.

2. We now completely define $\hat{\pi}$:

$$\hat{\pi}(nk_1M+j,t) := \begin{cases} \hat{\pi}(j, t-nL_1), & \text{if } t \geqslant nL_1+1, \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad \text{for } n \in \mathbb{N} \text{ and } j \in \{1, 2, \cdots, k_1M\}.$$

That is, the schedule for flow units $(\eta-1)k_1M+1$ through ηk_1M, $\eta \geqslant 2$, is exactly the same as the schedule for flow unit 1 through k_1M, except with a time shift of $(\eta-1)L_1$ time units.

It is easy to see that $\hat{\pi} \in \Pi_{total}^{k_1M} \subseteq \Pi_{total}^{cyclic}$. By statement (a) of Lemma 3, we have

$$r^{\hat{\pi}} = \frac{k_1M}{L_1} = r^{\pi_1},$$

which implies (26). Claim 2 follows. ■