On the Structure of Bottlenecks in Processes

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

Process capacity and the associated notions of bottleneck activities and bottleneck resources – which are responsible for limiting process capacity to its present value – are fundamental concepts in the Operations Management literature. However, for processes that involve *collaboration* and *multitasking*, there is little clarity in the literature on what bottlenecks are, how they look like, and how they can be identified. In this paper, we formulate and analyze graph-theoretic optimization problems that determine *bottleneck structures of activities* and the associated *bottleneck sets of resources* in deterministic, single-product processes with possibly multiple copies of one or more resources and possibly multiple sets of resources that can perform each activity. In the presence of both collaboration and multitasking, *sets* of activities that are interconnected in a specific manner via shared resources form bottleneck structures that are responsible for limiting capacity. We use the collaboration graph of the process to either characterize bottleneck structures completely or identify graphical structures that must necessarily be part of any bottleneck structure. Our analysis reveals a natural hierarchy in the algorithmic approach for identifying bottleneck structures as processes become increasingly sophisticated, ranging from the "easy" case where the simple bottleneck formula correctly identifies bottlenecks to more-complex cases where one needs to solve progressively complicated mathematical programs. In turn, this understanding helps us obtain prescriptive answers to several questions of interest to managers; e.g., the budget-constrained procurement of resources to maximize capacity improvement and the design of processes to increase capacity without procuring additional resources.

Key words: Process Capacity, Bottlenecks, Collaboration, Multitasking, Mathematical Programming

1. Introduction

The capacity of a (manufacturing or service) process and the associated bottleneck activities and/or resources – which are responsible for limiting process capacity to its current value – are fundamental concepts in the Operations Management (OM) literature. Enhancing process capacity, which, by definition, requires slackening each of its current bottlenecks, is naturally an important goal for firms operating at near-full capacity. It is, therefore, ironic that there is little clarity in the OM

literature on what the bottlenecks of a process are, how they look like, and how they can be identified. These are basic questions that must be answered first in the quest to effectively improve the capacity of a process. Most textbooks illustrate the calculation of process capacity and the identification of the bottleneck resources using the following simple "bottleneck formula": the capacity of each resource (or resource pool) is first calculated by examining that resource in isolation; then, bottleneck resources are defined as those with the least capacity, which is defined to be the capacity of the process. The simple bottleneck formula correctly identifies bottlenecks (and correctly calculates process capacity) only in certain "simple" settings: for example, in processes where each resource is dedicated to only one activity and in processes where each activity requires only one resource. Gurvich and Van Mieghem (2015, 2017) consider two notions for a process: *collaboration* and *multi-tasking*. Collaboration occurs when at least one activity of the process requires two or more resources simultaneously and multi-tasking occurs when at least one resource is needed for two or more activities of the process. Gurvich and Van Mieghem (2015) show that, in the presence of both collaboration and multitasking, the simple bottleneck formula can significantly overestimate the true capacity of the process. In turn, this brings the potential danger of reaching incorrect conclusions about bottleneck resources and may eventually lead to erroneous decisions with significant financial impact, e.g., investing in procuring an expensive bottleneck resource without being able to obtain the presumed improvement in capacity.

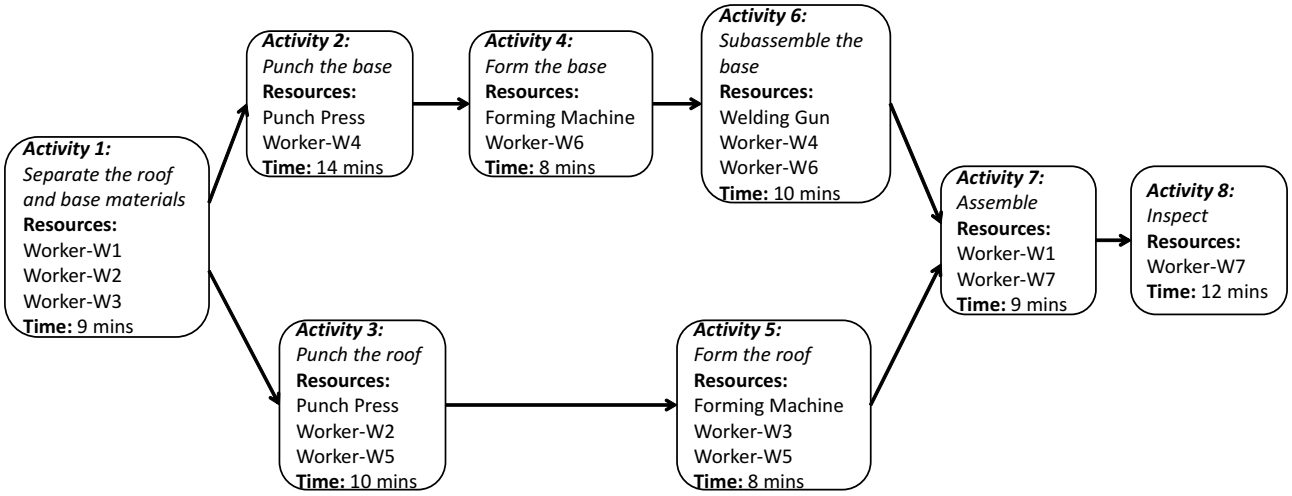
To motivate the research questions we examine and to illustrate our results throughout the paper, we will use a shed manufacturing process as a running example – this process is inspired by an example in Anupindi et al. (2012). The activities of the process are the same as in that textbook; the activity times and resource requirements have been modified to serve our purpose.

Example 1: (Shed Manufacturing) Wonder Shed Inc. uses steel to form both the base and the roof of each shed; these two components are then assembled into the finished product, i.e., a shed. The activities of the manufacturing process and their respective resource requirements are shown in Figure 1. First, three workers (worker-W1, worker-W2 and worker-W3) separate the material needed for the base from that needed for the roof. The base and the roof can then be fabricated in parallel. Base fabrication involves three activities: (i) punching the base by a worker (worker-W4) using a punch press, (ii) forming the base to shape by a worker (worker-W6) using a forming machine, and (iii) sub-assembling the base by two workers (worker-W4 and worker-W6) using a welding gun. Roof fabrication involves two activities: (i) punching the roof by two workers (worker-W2 and worker-W5) using a punch press, and (ii) forming the roof to shape by

two workers (worker-W3 and worker-W5) using a forming machine. Fabricated roofs and bases are then assembled into finished sheds by two workers (worker-W1 and worker-W7). Finally, one worker (worker-W7) inspects the finished products for quality assurance.

Table 1 lists the activities of the process as well as their processing times and required resources. We have one unit of each resource. The capacity of each resource when it is considered in isolation,

Figure 1 The manufacturing process of storage sheds in Example 1.



i.e., the maximum number of sheds that can be processed by that resource per unit of time, is shown in Table 2. Thus, as calculated by the simple bottleneck formula, the process capacity is 5/2 sheds per hour and the "bottleneck resources" are punch press and worker-W4.

Table 1 Activities, resources, and processing times in the shed manufacturing process (Example 1).

Activity	Resources	Time (mins)
1: Separate the roof and base material	Worker-W1, Worker-W2, Worker-W3	9
2: Punch the base	Punch Press, Worker-W4	14
3: Punch the roof	Punch Press, Worker-W2, Worker-W5	10
4: Form the base	Forming Machine, Worker-W6	8
5: Form the roof	Forming Machine, Worker-W3, Worker-W5	8
6: Subassemble the base	Welding Gun, Worker-W4, Worker-W6	10
7: Assemble	Worker-W1, Worker-W7	9
8: Inspect	Worker-W7	12

Notice, however, that any two of activities 1, 3, and 5, share at least one resource in common. Therefore, no two of these three activities can be in progress simultaneously (on different sheds, of course) since they both need a common resource. The total time for activities 1, 3, and 5 is

Table 2 Capacities of the resources in the shed manufacturing process of Example 1.

Resource	Time (mins)	Capacity of resource (sheds/hr)
Worker-W1	18 = 9 + 9	10/3
Worker-W2	19 = 9 + 10	60/19
Worker-W3	17 = 9 + 8	60/17
Punch Press	24 = 14 + 10	5/2
Worker-W4	24 = 14 + 10	5/2
Worker-W5	18 = 10 + 8	10/3
Forming Machine	16 = 8 + 8	15/4
Worker-W6	18 = 8 + 10	10/3
Welding Gun	10	6
Worker-W7	21 = 9 + 12	20/7

9 + 10 + 8 = 27 minutes. Consequently, the process capacity is at most 1/27 sheds/minute or 20/9 sheds/hour. In fact, the process capacity is exactly 20/9 sheds/hour, since we can obtain a feasible schedule whose throughput (i.e., the long-term average rate of completed sheds) achieves this upper bound; one such schedule is shown in Appendix A. The manner of resource-sharing by activities 1, 3, and 5, implies that of the three resources, namely worker-W2, worker-W3, and worker-W5, one is always idle when the other two are being used. Thus, the process capacity is *constrained collectively* by worker-W2, worker-W3, and worker-W5. Consequently, one can view activities 1, 3, and 5, along with the manner in which they share resources, as the "bottleneck structure of activities" for this process. Further, the set of three resources – namely, worker-W2, worker-W3, and worker-W5 – is collectively a "bottleneck set of resources".

Finding the correct bottleneck structure of activities and associated bottleneck set of resources is important for firms interested in increasing capacity. Without careful consideration, a firm may end up investing in procuring expensive resources and subsequently realizing an unsatisfactory improvement in capacity. In the shed manufacturing process, increasing the capacities of the punch press and worker-W4, which are the "bottlenecks" identified by the simple bottleneck formula, will not increase capacity. Indeed, to increase capacity, the manufacturer should necessarily increase the capacity of some of the resources that belong to the bottleneck set of resources.

The main lesson we extract from this example is this: in the presence of both collaboration and multitasking, our understanding of the concepts of bottleneck activities and bottleneck resources needs to be revised. The manner in which resources are shared across activities gives rise to a bottleneck structure of activities that, as a whole, is responsible for limiting the capacity of the

process to its present value. Correspondingly, we have a bottleneck set of resources, which is the collection of resources that generates the bottleneck structure of activities. ■

Our Contribution

Consider a basic prescriptive question that is at the heart of our analysis in this paper: *if the simple bottleneck formula does not always correctly identify a bottleneck set of activities/resources, then how do we correctly identify one?* We show that bottleneck structures can be characterized using linear programs that possess a well-known structure, namely, fractional cliques in graphs. Therefore, we are not only able to answer how to compute bottleneck structures but also say more about their properties using knowledge related to the fractional clique problem. This development is conducted in four steps, as outlined below.

First, for a single-product process in which there is one copy of each resource, we define a bottleneck structure of activities and its associated bottleneck sets of resources using the fractional clique problem on the collaboration graph of the process (Section 5). A bottleneck of the process is then defined by a *pair*: a bottleneck structure of activities which is responsible for limiting the process capacity to its present value and a bottleneck set of resources that generates this structure. Here, there are two important takeaways that significantly refine our understanding relative to the naive (textbook) approach: (1) increasing the capacity of any *one* resource from a bottleneck set of resources can be sufficient for it to cease to be a bottleneck set of resources. However, (2) there can be multiple bottleneck sets of resources corresponding to the *same* bottleneck structure of activities. Therefore, to "eliminate" a given bottleneck structure of activities, one may have to increase the capacity of multiple resources.

Second, the graph-theoretic connection allows us to exploit results in the literature on perfect graphs – e.g., those in Chudnovsky et al. (2006) and Lovász (1972) – to either describe bottleneck structures completely in simple graphical terms or characterize their necessary sub-components (Section 6). In particular, we show that a bottleneck structure of activities of *any process* is either a clique or contains either an odd-hole or an odd-antihole in the collaboration graph. We also show that, if the collaboration graph of a process is perfect, then each of its bottleneck structures is a maximum clique in the collaboration graph.

Third, when there are multiple copies of one or more resources in a process, we provide two characterizations of a bottleneck structure of activities and the corresponding bottleneck sets of resources (Section 7 and Appendix C). In the first, we characterize bottlenecks using collaboration *hypergraphs*. Here, a bottleneck structure of activities for a process and the related bottleneck sets of resources are obtained by solving the fractional clique problem on the collaboration hypergraph.

The second characterization avoids the notion of hypergraphs and, instead, uses an alternate linear program – this approach can be simpler to follow in practice.

Fourth, we characterize the bottleneck structures of activities and the related bottleneck sets of resources for processes with *flexible resource sets* (Section 7 and Appendix D). In such processes, for each activity, there can potentially be multiple sets of resources that can perform the activity. Consequently, there are multiple possible collaboration graphs, each defined by a specific choice of the resource sets employed for the activities of the process. Therefore, the specification of a bottleneck structure of activities for a flexible process is more complex: it consists of a *collection* of fractional cliques (not necessarily maximum fractional cliques), one from each possible collaboration graph. As before, we also provide a linear program for convenient use in practice.

Collectively, the analysis above results in an algorithmic procedure (which we summarize in Section 8) for identifying a bottleneck structure of activities and a corresponding bottleneck set of resources in an arbitrary deterministic, single-product process.

The graph-theoretic connection and the use of math programming also help us delve deeper into the analysis of bottlenecks by enabling the discussion of more advanced questions that practitioners are interested in. Specifically, we discuss three issues:

- **The Resource-Capacity Investment Problem:** Given a budget, find a set of resources to acquire to obtain the best-possible improvement in process capacity (Section 9).

- **The "Design" of Processes:** Operations that can help "break" a bottleneck structure of activities by reducing the need for collaboration (Section 9).

- **Bottleneck Structures in Multi-Product Processes:** Specifically, the use of our analysis to examine bottlenecks in processes that produce two or more products (Section 6).

The remainder of the paper is organized as follows: Section 2 reviews the related literature. Section 3 formally defines an arbitrary, deterministic single-product process and discusses other preliminary notions that are needed in our subsequent analysis. Section 4 reviews a characterization of the capacity of a process using its associated collaboration graph. Section 5 formally defines a bottleneck structure of activities and a corresponding bottleneck set of resources for a process that has one copy of each distinct resource. Section 6 describes bottleneck structures in simple graphical terms for some special cases and identifies the necessary components of a bottleneck structure for an arbitrary process. Section 7 discusses bottleneck structures for two generalizations: (i) processes that have multiple copies of resources, and (ii) "flexible" processes, namely those in which activities

can be performed by any one of multiple sets of resources. Section 8 summarizes the procedure for identifying bottlenecks and Section 9 discusses some advanced questions of practical interest.

2. Literature Review

Our work is related to the literature on cyclic scheduling problems for specific manufacturing systems; e.g., the cyclic flow shop problem (Abadi et al. 2000, McCormick et al. 1989), the cyclic job shop problem (Lee and Posner 1997, Roundy 1992, Matsuo et al. 1991), the cyclic robotic-cell scheduling problem (Dawande et al. 2007, Dawande et al. 2002, Crama and Van De Klundert 1997), and the cyclic project scheduling problem (Alcaide et al. 2007, Levner and Kats 1998). We refer the reader to Levner et al. (2010) for a comprehensive review of the literature on cyclic scheduling problems. Broadly, the focus here is on obtaining the best schedule among a certain class of schedules and the complexity of finding such a schedule. To our knowledge, a general analysis of bottleneck structures is not available in this stream of work.

Another related stream is that of scheduling of a stochastic processing network (SPN), introduced by Harrison (2000, 2002, 2003), with collaboration and multitasking. Here, the goal is to dynamically schedule the activities in an SPN to achieve the capacity of the process/network or to optimize various performance metrics. For instance, Jiang and Walrand (2010) and Dai and Lin (2005) study this problem and propose scheduling policies or algorithms that can achieve a given throughput of the network. Again, this literature does not address bottleneck structures. The notions of collaboration and multitasking are also common in project management; for example, resource-constrained project scheduling (Hartmann and Briskorn 2010, Debels and Vanhoucke 2007, Möhring et al. 2003, Dorndorf et al. 2000), resource-constrained machine scheduling (Wang et al. 2015, Dobson and Karmarkar 1989), and in loss networks (Jung et al. 2019, Bertsimas and Chryssikou 1999, Kelly 1991).

We now discuss the contribution of three papers that are most relevant to our work: Gurvich and Van Mieghem (2015, 2017), and Bo et al. (2018).

Gurvich and Van Mieghem (2015) show that, in the presence of collaboration and multitasking, the simple textbook bottleneck formula to compute process capacity can be significantly inaccurate. The authors then define the *static planning problem for collaboration* (SPPC) to compute the network utilization for a given flow rate (or throughput) of the process; the network utilization measures the fraction of time the process is busy. The capacity of a process can be obtained indirectly by using SPPC in the sense that the process capacity is that value of the throughput for

which the network is fully utilized (i.e., the network utilization is 1). An important result in the paper is a *necessary and sufficient condition* under which the bottleneck formula correctly computes process capacity. The authors also draw some implications on where to add flexibility; i.e., enabling multiple distinct sets of resources to perform the same activity. Gurvich and Van Mieghem (2017) study the impact of task prioritization (the freedom to prioritize certain tasks over others) on the capacity of processes under what they refer to as a "hierarchical collaboration architecture". They show that misaligned priority levels and collaboration levels incur a significant capacity loss.

Bo et al. (2018) is exclusively devoted to studying the theoretical complexity of computing process capacity. There are two main results in this paper: (1) In the presence of collaboration and multitasking, the problem of computing process capacity is strongly NP-hard. To do this, the authors obtain a novel graph-theoretic characterization of process capacity that relates it to the fractional chromatic number of the associated "collaboration graph" (formally defined in Section 3). (2) It is strongly NP-hard to even approximate process capacity to within a reasonable factor. The authors also discuss special cases where computing process capacity is theoretically efficient.

All the three papers above focus on the capacity of a process in the presence of collaboration and multitasking. Broadly, the discussion in these papers implies that a *set* of resources can collectively determine capacity. However, there is no attempt in these papers to define what the correct notion of a "bottleneck" of a process is. The precise definition of bottleneck structures of activities and the associated bottleneck sets of resources, the ensuing analysis of their properties and computation, and discussions of the associated managerial implications, are the goals of our paper.

3. Preliminaries

Our model and notation of a deterministic, single-product process, and the definition of process capacity, are the same as in Bo et al. (2018)[1]. Such a process is represented using a 7-tuple, whose components are as follows:

1. The set of activities required to produce one flow unit of the product is denoted by $V = \{v_1, v_2, \ldots v_n\}$. Without loss of generality, we can 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 $A = \varnothing$.

[1]This model is a special case of the one considered in Gurvich and Van Mieghem (2015).

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 $v \in V$, 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.

Using these components, a process is defined as $\mathcal{P} = \big(V, A, \tau, W, \{N_w, w \in W\}, \{W_v, v \in V\}, \mathcal{Z}\big)$. The formal definitions of the following three concepts are in Bo et al. (2018); since we do not use their mathematical expressions in our analysis, a non-technical description suffices for our purpose.

- A feasible *operating policy* for $\mathcal{P}$ is a function that specifies the activities that are scheduled on each flow unit during each time unit.
- The *throughput* (or *process rate*) of a given policy for $\mathcal{P}$ is the long-term average rate of completed flow units produced by that policy.
- The *capacity* of $\mathcal{P}$, denoted $\mu(\mathcal{P})$, is the maximum throughput that can be achieved over all feasible policies.

Until Section 7, let us assume that we have one copy of each resource. That is,

$$N_w = 1 \text{ for } w \in W. \tag{1}$$

Later, in Section 7, we will generalize our observations for the setting where $N_w \geqslant 1$ for $w \in W$. Bo et al. (2018) show that, for an arbitrary process $\mathcal{P}$, the precedence and non-preemption constraints can be ignored, without loss of generality, for the purpose of computing process capacity (see Lemma 2 of Bo et al. 2018). Therefore, throughout our discussion, we will ignore precedence constraints and assume preemptive processing of the activities.

For a process in which we have one copy of each resource, the *collaboration graph* (Gurvich and Van Mieghem 2015, Bo et al. 2018) is defined as follows: each node of the graph corresponds to an activity of the process and two activities are connected by an edge if they share at least one common resource.

Before proceeding further, we briefly review the well-known class of perfect graphs; we will use several results for this class of graphs in our analysis. Consider an undirected graph H. A node-induced subgraph of H that is an odd-length cycle of length at least 5 (if one exists) is referred to

as an *odd-hole* in H. A node-induced subgraph of H that is the graph complement of an odd-hole (on the nodes of the odd-hole) is referred to as an *odd-antihole* in H. Figure 2 illustrates an odd-hole and an odd-antihole. The graph H is said to be *perfect* if, for every node-induced subgraph F of H, the chromatic number of F (denoted $\chi^*(F)$) equals its clique number, i.e., the size of the maximum clique in F (denoted $\kappa^*(F)$). An important result related to perfect graphs is the *Strong Perfect Graph Theorem* (SPGT, Chudnovsky et al. 2006), which provides a forbidden-subgraph characterization of perfect graphs: *a graph is perfect* if and only if *it contains no odd-hole and no odd-antihole*. There is a substantial amount of literature on perfect graphs and their properties; see, e.g., Golumbic (2004) and Alfonsin (2001); in Section 5 and beyond, we will introduce several important properties as and when needed.

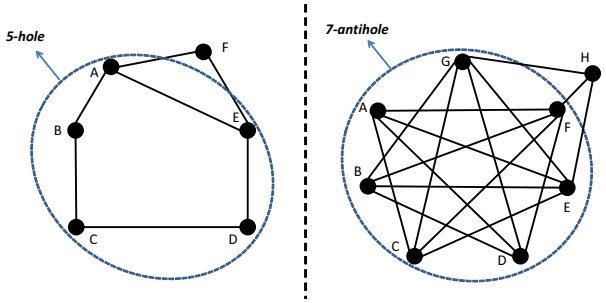


Figure 2 An odd-hole and an odd-antihole: Illustrative examples

In the next section, we review a graph-theoretic characterization of the capacity of a process that is established in Bo et al. (2018).

4. Process Capacity: Characterization Using the Collaboration Graph

Let us first assume that each activity of the process takes one unit of time, i.e., $\tau(v) = 1$ for all $v \in V$. Under this assumption, we will denote the process by $\mathcal{P}^1$. A set of activities $S \subseteq V$ of $\mathcal{P}^1$ is said to be *feasible* if no two activities in S need the same resource. Thus, all the activities that belong to a feasible set can be performed simultaneously. Observe that, in the collaboration graph G^1 of $\mathcal{P}^1$, the nodes that correspond to a feasible set S of activities form an *independent set* (also referred to as a *stable set* or a *node packing*). Let $\mathcal{I}$ denote the collection of all feasible sets of activities in $\mathcal{P}^1$ or, equivalently, all independent sets in G^1. Formally,

$$\mathcal{I} = \left\{ S \subseteq V : \; W_v \cap W_{\tilde{v}} = \varnothing \text{ for all } v, \tilde{v} \in S; v \neq \tilde{v} \right\}.$$

Consider the following LP that computes the *fractional chromatic number* $\chi_f^*(G^1)$ of G^1:

$$\chi_f^*(G^1) = \min \quad \sum_{I \in \mathcal{I}} x_I$$

$$\text{s.t.} \quad \sum_{I: v \in I} x_I \geqslant 1 \quad \text{for} \quad v \in V, \qquad (P_\chi)$$
$$x_I \geqslant 0 \quad \text{for} \quad I \in \mathcal{I}.$$

If we restrict $x_I \in \{0,1\}$, then the resulting linear integer program computes the chromatic number $\chi^*(G^1)$ of G^1. The fractional chromatic number of a graph is an important and well-studied quantity in graph theory; see, e.g., Scheinerman and Ullman (2011) for an extensive discussion. Bo et al. (2018) establish the following fundamental result:

THEOREM 1. (Bo et al. 2018) *Let $\mu(\mathcal{P}^1)$ denote the capacity of process P^1. Then,*

$$\mu(\mathcal{P}^1) = \frac{1}{\chi_f^*(G^1)}.$$

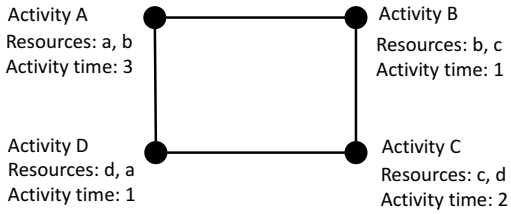
Let $x_I^*, I \in \mathcal{I}$, denote an optimal solution of the LP P_χ; the value x_I^* can be interpreted as the long-term proportion of time allocated to the collection of activities I in the schedule that achieves the process capacity $\mu(\mathcal{P}^1)$. Using Theorem 1, the fact that the problem of finding the fractional chromatic number of a graph is strongly NP-hard (Lund and Yannakakis 1994), and other auxiliary results, Bo et al. (2018) show that the problem of computing $\mu(\mathcal{P}^1)$ is strongly NP-hard. Further, they show that it is also NP-hard to approximate $\mu(\mathcal{P}^1)$ within a reasonable factor.

Now consider a process $\mathcal{P}$ for which the activity times in the process are general positive integers, i.e., $\tau(v) \geqslant 1, v \in V$. Bo et al. (2018) show that the capacity $\mu(\mathcal{P})$ of such a process is the same as that of the *expanded process*, denoted $\mathcal{P}^{exp}$, obtained by dividing each activity $v \in V$ with $\tau(v) > 1$ into $\tau(v)$ activities that each have an activity time of 1 unit and use the same set of resources as the original activity. The *expanded graph*, $G^{exp} = (V^{exp}, E^{exp})$, for process $\mathcal{P}$ is defined as the collaboration graph of the corresponding expanded process $\mathcal{P}^{exp}$. Then,

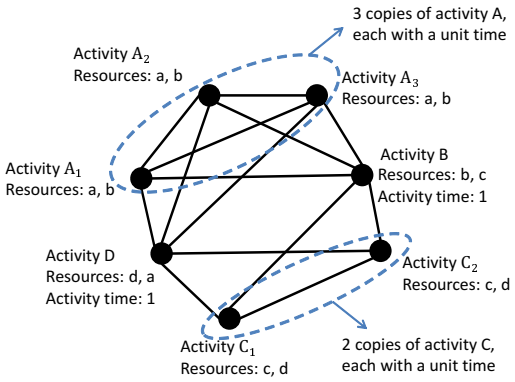
$$\mu(\mathcal{P}) = \mu(\mathcal{P}^{exp}) = \frac{1}{\chi_f^*(G^{exp})}. \qquad (2)$$

The example below illustrates the construction of the expanded graph of a process.

Example 2: Consider the process $\mathcal{P}$ in Figure 3(a). The activities of the process and their processing times, the resources needed for each activity, and the collaboration graph of the process are all shown in the figure. Note that activity A takes three units of time and activity C takes two units of time. In the expanded process $\mathcal{P}^{exp}$, activity A (resp., C) is divided into three (resp., two) activities A_1, A_2 and A_3 (resp., C_1 and C_2), each with a processing time of one unit. Figure 3(b) shows the expanded (collaboration) graph G^{exp} of $\mathcal{P}^{exp}$. ■



(a) A process with $\tau(v) \geqslant 1$, $v \in V$ and its collaboration graph



(b) The expanded process $\mathcal{P}^{exp}$ and its expanded (collaboration) graph G^{exp}

Figure 3 The collaboration graph and the expanded collaboration graph of the process in Example 2.

We now proceed to formally define a bottleneck structure of activities and a corresponding bottleneck set of resources for a process.

5. Fractional Cliques and Bottleneck Structures

Again, let us first consider a process $\mathcal{P}^1$ for which $\tau(v) = 1, v \in V$. The dual of the LP P_χ is commonly referred to as the *fractional clique problem* on G^1 and computes the *fractional clique number* $\kappa_f^*(G^1)$ of G^1 (see, e.g., Scheinerman and Ullman 2011, Godsil and Royle 2001). Let $y_v, v \in V$, denote the dual variables. Then, we have

$$\begin{aligned} \kappa_f^*(G^1) = \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & \sum_{v \in I} y_v \leqslant 1 \quad \text{for} \quad I \in \mathcal{I}, \\ & y_v \geqslant 0 \quad \text{for} \quad v \in V. \end{aligned} \tag{D_κ}$$

This is the LP relaxation of the linear integer program that computes the clique number $\kappa^*(G^1)$ of G^1, i.e., the maximum cardinality (number of nodes) of a clique in G^1.

Let us first assume that the dual problem D_κ has a unique optimal solution. Let $\boldsymbol{y} = (y_v, v \in V)$ denote a feasible solution (referred to as a fractional clique of G^1) of D_κ and $\kappa_f(\boldsymbol{y}, G^1)$ denote its objective value. Let $\boldsymbol{y}^* = (y_v^*, v \in V)$ denote the optimal solution of D_κ (referred to as the maximum fractional clique of G^1); the optimal objective value is, as denoted above, $\kappa_f^*(G^1) = \sum_{v \in V} y_v^*$. Then, from LP duality, we have

$$\kappa_f(\boldsymbol{y}, G^1) \leqslant \kappa_f^*(G^1) = \chi_f^*(G^1).$$

Then, using Theorem 1, we have

$$\mu(\mathcal{P}^1) = \frac{1}{\chi_f^*(G^1)} = \frac{1}{\kappa_f^*(G^1)} \leqslant \frac{1}{\kappa_f(\boldsymbol{y}, G^1)}. \tag{3}$$

Thus, any fractional clique imposes a corresponding upper bound on the capacity of the process. We refer to the value $\frac{1}{\kappa_f(\boldsymbol{y},G^1)}$ as the *processing bound* imposed by the fractional clique $\boldsymbol{y}$. The tightest such bound, which equals process capacity, comes from the maximum fractional clique $\boldsymbol{y}^*$. Let $Q = \{v \in V : y_v^* > 0\}$. The node-induced subgraph S_Q of G^1 corresponding to the activities in Q is responsible for limiting the capacity of the process to its current value, namely $\mu(\mathcal{P}^1)$; we therefore refer to S_Q as the *bottleneck structure of activities* for $\mathcal{P}^1$. The processing bound imposed by the maximum fractional clique $\boldsymbol{y}^*$, namely $\frac{1}{\kappa_f^*(G^1)}$, can also be viewed as the processing bound imposed by the bottleneck structure S_Q. Therefore, we have the following max-min relationship: the capacity $\mu(\mathcal{P}^1)$ of process $\mathcal{P}^1$ (i.e., its maximum long-term processing rate) equals the processing bound imposed by the bottleneck structure S_Q, which is the minimum processing bound over all the fractional cliques of the collaboration graph G^1. Let $R_Q = \bigcup_{v \in Q} W_v$ be the collection of resources that are needed for at least one of the activities in Q. Then, it is immediate that the graph $\hat{G}(Q, E_Q)$, where

$$E_Q = \big\{(v, \tilde{v}) \in Q \times Q : W_v \cap W_{\tilde{v}} \neq \varnothing, v \neq \tilde{v}\big\}, \tag{4}$$

is isomorphic to S_Q. Further, it may not be necessary to consider all the resources in R_Q for $\hat{G}$ to be isomorphic to S_Q. That is, restricting attention to a proper subset of R_Q may also be enough to make $\hat{G}$ isomorphic to S_Q; an illustrative example (Example 3) is provided below. A minimal subset $\hat{R}_Q$ of R_Q that induces a subgraph on the nodes in Q isomorphic to S_Q is defined to be a *bottleneck set of resources* for $\mathcal{P}^1$. Let the set system $\mathfrak{R}_Q$ denote the collection of all such bottleneck sets of resources corresponding to the unique bottleneck structure S_Q. That is, $\mathfrak{R}_Q = \big\{\hat{R}_Q : \hat{R}_Q \text{ is a bottleneck set of resources corresponding to } S_Q\big\}$. It is important to recognize that there can be multiple bottleneck sets of resources $\hat{R}_Q$ corresponding to the S_Q. Finally, we define a *bottleneck* of $\mathcal{P}^1$ as the pair $(S_Q, \hat{R}_Q)$. Thus, each bottleneck of the process is uniquely determined by a bottleneck structure of activities S_Q and a corresponding bottleneck set of resources $\hat{R}_Q$. The set of all the bottlenecks of $\mathcal{P}^1$ is:

$$\Big\{(S_Q, \hat{R}_Q) : \hat{R}_Q \in \mathfrak{R}_Q,\ Q = \{v \in V : y_v^* > 0\}\Big\}.$$

If the dual problem D_κ has multiple optimal solutions, then let Y^* denote the set of all *basic* optimal solutions of D_κ. A basic optimal solution $\boldsymbol{y}^* \in Y^*$ is referred to as a basic maximum

fractional clique of G^1. Given $\boldsymbol{y}^* \in Y^*$, let $Q(\boldsymbol{y}^*) = \{v \in V : y_v^* > 0\}$. We refer to the node-induced subgraph $S_{Q(\boldsymbol{y}^*)}$ as the *bottleneck structure of activities* for $\mathcal{P}^1$ corresponding to $\boldsymbol{y}^*$. Thus, each basic optimal solution $\boldsymbol{y}^* \in Y^*$ corresponds to a bottleneck structure of activities. As before, we let $R_{Q(\boldsymbol{y}^*)}$ denote the set of resources needed for at least one activity in $Q(\boldsymbol{y}^*)$. A minimal subset $\hat{R}_{Q(\boldsymbol{y}^*)}$ of $R_{Q(\boldsymbol{y}^*)}$ that induces a subgraph on the nodes in $Q(\boldsymbol{y}^*)$ isomorphic to $S_{Q(\boldsymbol{y}^*)}$ is defined to be a *bottleneck set of resources* for $\mathcal{P}^1$. The set system $\mathfrak{R}_{Q(\boldsymbol{y}^*)}$ denotes the collection of all such bottleneck sets of resources corresponding to the bottleneck structure $S_{Q(\boldsymbol{y}^*)}$. A *bottleneck* of $\mathcal{P}^1$ is the pair $(S_{Q(\boldsymbol{y}^*)}, \hat{R}_{Q(\boldsymbol{y}^*)})$. Thus, the set of all bottlenecks of $\mathcal{P}^1$ is

$$\left\{(S_{Q(\boldsymbol{y}^*)}, \hat{R}_{Q(\boldsymbol{y}^*)}) : \hat{R}_{Q(\boldsymbol{y}^*)} \in \mathfrak{R}_{Q(\boldsymbol{y}^*)}, Q(\boldsymbol{y}^*) = \{v \in V : y_v^* > 0\}, \boldsymbol{y}^* \in Y^*\right\}.$$

In the presence of collaboration and multitasking, there can be *multiple* bottleneck sets of resources that generate the same bottleneck structure of activities. Therefore, to precisely describe a specific bottleneck, we need to specify a pair, namely a bottleneck structure of activities and an associated bottleneck set of resources that generates it.

We now illustrate the identification of bottlenecks for two simple processes.

Example 3: Figure 4 shows the collaboration graphs of two processes. The activities for each process and the resources needed for each activity are shown in the figure. Assume that, for both processes, each activity takes one unit of time and that we have one unit of each resource.

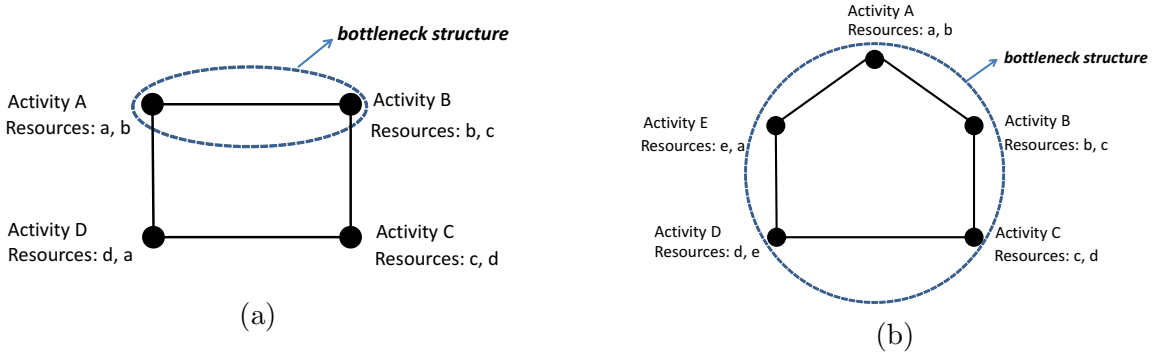


Figure 4 Bottleneck structures in the two processes discussed in Example 3.

- For the 4-activity process shown in Figure 4(a), a basic maximum fractional clique, i.e., a basic optimal solution to problem D_κ, is $y_v^* = 1$ for $v \in \{A, B\}$ and $y_v^* = 0$ for $v \in \{C, D\}$. Thus, $\chi_f^*(G^1) = \kappa_f^*(G^1) = 2$, and $Q = \{A, B\}$. The corresponding node-induced subgraph S_Q is the edge $\{A, B\}$ and is a bottleneck structure of activities for the process. The set of resources needed for the activities in Q is $R_Q = \{a, b, c\}$. However, resource b itself induces a subgraph isomorphic to S_Q. Thus, the singleton set $\{b\}$ is a bottleneck set of resources.

Notice that, for this process, there are multiple basic optimal solutions, which in turn give rise to multiple bottleneck structures of activities and multiple bottleneck sets of resources: edge $\{A, D\}$, edge $\{A, B\}$, edge $\{B, C\}$, and edge $\{C, D\}$, are all bottleneck structures of activities and each of the singleton sets $\{a\}$, $\{b\}$, $\{c\}$, and $\{d\}$ is a bottleneck set of resources. In total, there are four bottlenecks for this process: $(\{A, B\}, \{b\})$, $(\{A, D\}, \{a\})$, $(\{B, C\}, \{c\})$, and $(\{C, D\}, \{d\})$.

- For the 5-activity process shown in Figure 4(b), the *unique* maximum fractional clique is $y_v^* = 0.5$ for $v \in \{A, B, C, D, E\}$. Thus, $\chi_f^*(G^1) = \kappa_f^*(G^1) = 2.5$, and $Q = \{A, B, C, D, E\}$. The subgraph S_Q is the entire collaboration graph (i.e., $S_Q = G^1$) and is the unique bottleneck structure of activities for the process. Finally, $\hat{R}_Q = R_Q = \{a, b, c, d, e\}$ is the unique bottleneck set of resources. ■

Let us now consider a process $\mathcal{P}$ for which $\tau(v) \geqslant 1, v \in V$. Due to the characterization of process capacity in (2), it is convenient to define a bottleneck structure of activities for $\mathcal{P}$ using the expanded graph $G^{exp}(V^{exp}, E^{exp})$, which is the collaboration graph of the expanded process $\mathcal{P}^{exp}$ in which the processing time of each activity is one unit (and, therefore, the discussion above applies). Recall that for every activity $v \in V$ in process $\mathcal{P}$, there are $\tau(v)$ corresponding activities $v^1, v^2, \ldots, v^{\tau(v)}$ in process $\mathcal{P}^{exp}$, each with a processing time of one unit. Thus, $V^{exp} = \left\{v^k : k \in \{1, 2, \ldots, \tau(v)\}, v \in V\right\}$. Let $z_{v^k}^*, k \in \{1, 2, \ldots, \tau(v)\}, v \in V$, denote a basic maximum fractional clique in G^{exp}. This basic optimal solution gives us the bottleneck structure of activities $S_{Q^{exp}}$ in the process $\mathcal{P}^{exp}$, where $Q^{exp} = \{v^k \in V^{exp} : z_{v^k}^* > 0\}$. Since the $\tau(v)$ activities in $\mathcal{P}^{exp}$ that correspond to activity v in $\mathcal{P}$ are identical to each other, both in their processing times (one unit each) and their resource requirements, it is straightforward to see that, for each $v \in V$, we have $z_{v^1}^* = z_{v^2}^* = \ldots = z_{v^{\tau(v)}}^*$. Thus, for each activity v in process $\mathcal{P}$, either all or none of the corresponding $\tau(v)$ activities (namely, $v^1, v^2, \ldots, v^{\tau(v)}$) of $\mathcal{P}^{exp}$ belong to the bottleneck structure of activities in the latter process. Consequently, we can *uniquely* map the bottleneck structure $S_{Q^{exp}}$ in $\mathcal{P}^{exp}$ to the node-induced subgraph S_Q of the collaboration graph G of $\mathcal{P}$ corresponding to the activities in $Q = \{v \in V : z_{v^1}^* = z_{v^2}^* = \ldots = z_{v^{\tau(v)}}^* > 0\}$. Due to this correspondence, for a process $\mathcal{P}$ with $\tau(v) \geqslant 1$, we can refer either to a bottleneck structure of activities in the collaboration graph G^{exp} of the expanded process $\mathcal{P}^{exp}$ or, equivalently, to the corresponding structure of activities in the collaboration graph G of $\mathcal{P}$. In our subsequent discussion, we will interchangeably use these two equivalent structures.

Remark 1: Increasing the capacity of a process requires slackening each of its current bottlenecks, for example, by employing additional resources. In particular, this requires finding a resource set

that contains at least one resource in each bottleneck set of resources. Accordingly, to increase the capacity of a process, one needs to: (i) identify all bottlenecks of the process, (ii) find a minimal resource set containing at least one resource in each bottleneck set of resources, and (iii) employ additional copies of each resource in that minimal resource set. Consider the process in Figure 4(a). There are four bottlenecks for this process: $(\{A, B\}, \{b\})$, $(\{A, D\}, \{a\})$, $(\{B, C\}, \{c\})$, and $(\{C, D\}, \{d\})$. The minimal resource set containing at least one resource in each bottleneck is $\{a, b, c, d\}$. Consequently, enhancing the capacity of the process requires employing additional copies of *each* of the resources a, b, c, and d. In contrast, there is only one bottleneck for the process in Figure 4(b): $(\{A, B, C, D, E\}, \{a, b, c, d, e\})$. Therefore, enhancing the capacity of this process requires employing additional copies of *any* resource in the set $\{a, b, c, d, e\}$. In its general form, the problem of finding a minimal resource set that contains at least one resource in each bottleneck set of resources is the well-known Hitting Set problem (see, e.g., Chvatal 1979 and Slavik 1997). ■

In the next section, we examine how bottleneck structures look like in collaboration graphs.

6. Properties of Bottleneck Structures

Section 6.1 describes bottleneck structures of activities in simple graphical terms for some special cases while Section 6.2 identifies the necessary components of any bottleneck structure for an arbitrary process.

6.1. Bottleneck Structures: Special Cases

Let us consider two types of processes, namely those that do not involve collaboration and those that do not involve multitasking.

6.1.1. No Collaboration or No Multi-Tasking When a process does not involve either collaboration or multitasking, then any maximum clique in the collaboration graph (G^1, if $\tau(v) = 1$ for all $v \in V$, and G^{exp}, otherwise) is a bottleneck structure of activities for the process and a corresponding bottleneck set of resources is a singleton set; see Appendix B. Consequently, the simple bottleneck formula, which considers each resource in isolation, correctly computes the bottleneck set of resources. More generally, whenever there exists a bottleneck set of resources that is a singleton set, the simple bottleneck formula correctly identifies that set. Note that this can occur even in the presence of both collaboration and multitasking, as the example in Figure 4(a) shows.

We now present a more-general condition under which the above properties hold.

6.1.2. A More-General Condition First, let us consider a process $\mathcal{P}^1$ in which $\tau(v) = 1$ for all $v \in V$. Let V_w denote the set of activities that need resource $w \in W$ (and possibly other resources); that is, $V_w = \{v \in V : w \in W_v\}$ and let $\Delta_w = |V_w|$. Let $\Delta^* = \max_{w \in W} \Delta_w$. Thus, Δ^* is the maximum number of activities that need a common resource (with $\arg\max_{w \in W} \Delta_w$ being one such resource). Since, for each $w \in W$, the nodes corresponding to the activities in V_w form a clique in G^1, we have $\kappa^*(G^1) \geqslant \Delta_w$. Therefore,

$$\kappa^*(G^1) \geqslant \Delta^*. \tag{5}$$

Let $W^* = \{w \in W : \Delta_w = \Delta^*\}$. In words, W^* is the collection of resources that are each needed by Δ^* activities of the process. Using (5) and the fact that $\kappa^*(G^1) \leqslant \kappa_f^*(G^1)$, we have

$$\kappa_f^*(G^1) \geqslant \Delta^*. \tag{6}$$

If (6) holds with equality, then, from (3), the capacity of the process is $\mu(\mathcal{P}^1) = 1/\kappa_f^*(G^1) = 1/\Delta^*$. Further, since $\kappa_f^*(G^1) = \kappa^*(G^1) = \Delta^*$, for *any* resource $w^* \in W^*$, the clique in G^1 formed by the nodes in V_{w^*} is a bottleneck structure of activities for $\mathcal{P}^1$. In particular, note that this bottleneck structure is defined by a *single* resource (namely, any resource in W^*); thus, the corresponding bottleneck set of resources is a singleton set. The simple bottleneck formula examines each resource in isolation and designates a resource with the maximum activity time, i.e., a resource $w \in W$ that maximizes $\sum_{v \in V_w} \tau(v)$, as a bottleneck and calculates the capacity of the process as the reciprocal of this maximum value. Since $\tau(v) = 1$ for each $v \in V$, this simple calculation also identifies each resource in W^* as a bottleneck resource and arrives at the correct value of capacity.

In summary, if (6) holds with equality, then a maximum clique – among the cliques in G^1 formed by the activities that share a common resource – is a bottleneck structure of activities for $\mathcal{P}^1$. Further, the singleton set comprising this common resource forms the corresponding bottleneck set of resources and the simple bottleneck formula correctly computes the capacity of the process. It is straightforward to verify that the converse also holds, i.e., if a maximum clique in G^1, among those which are formed by the activities that share a common resource, forms a bottleneck structure of activities for $\mathcal{P}^1$, then (6) holds with equality.

A simple example of a process for which (6) holds is one for which the collaboration graph G^1 is *acyclic*. In this case, $|V_w| \leqslant 2$ for any $w \in W$ (for otherwise, there exists a clique of size 3 or more in G^1 and hence G^1 cannot be acyclic). Therefore, if there exists a resource $w \in W$ with $|V_w| = 2$, then $\kappa_f^*(G^1) = \Delta^* = 2$; otherwise $\kappa_f^*(G^1) = \Delta^* = 1$.

Now, consider a process $\mathcal{P}$ in which $\tau(v) \geqslant 1$ for $v \in V$. We consider the corresponding expanded process $\mathcal{P}^{exp}$ and its collaboration graph $G^{exp}(V^{exp}, E^{exp})$, and recall that $\mu(\mathcal{P}) = \mu(\mathcal{P}^{exp})$. Also, recall that for every "parent" activity $v \in V$ in $\mathcal{P}$, there are $\tau(v)$ corresponding "children" activities $v^1, v^2, \ldots, v^{\tau(v)}$ in $\mathcal{P}^{exp}$, each with a processing time of one unit. Consider any clique in G^{exp} formed by a node set $V_c^{exp} \subseteq V^{exp}$. If $V_c \subseteq V$ denotes the set of parent nodes in G corresponding to the nodes in V_c^{exp}, then it is immediate that the node set V_c forms a clique in G. Therefore, every clique in G^{exp} corresponds to a clique in G. The converse also holds: if a set of nodes $\tilde{V} \subseteq V$ forms a clique in G, then the set of nodes $\bigcup_{v \in \tilde{V}} \{v^1, v^2, \ldots, v^{\tau(v)}\}$ forms a clique in G^{exp}.

Let $\bar{\Delta}_w = \sum_{v \in V_w} \tau(v)$ and let $\bar{\Delta}^* = \max_{w \in W} \bar{\Delta}_w$. For each activity $v \in V_w$, there are $\tau(v)$ corresponding children activities in V_w^{exp}. Thus, there are a total of $\sum_{v \in V_w} \tau(v) = \bar{\Delta}_w$ nodes in V_w^{exp} corresponding to the $|V_w|$ activities in V_w. Further, these nodes form a clique in G^{exp}. Therefore, $\kappa^*(G^{exp}) \geqslant \bar{\Delta}_w$ for each $w \in W$. Consequently, we have

$$\kappa^*(G^{exp}) \geqslant \bar{\Delta}^*. \tag{7}$$

Let $\bar{W}^* = \{w \in W : \bar{\Delta}_w = \bar{\Delta}^*\}$. In words, for each resource in $\bar{W}^*$, its activity time (i.e., the sum of the processing times of the activities that need the resource) is the maximum among all resources. As before, using (7) and the fact that $\kappa^*(G^{exp}) \leqslant \kappa_f^*(G^{exp})$, we have

$$\kappa_f^*(G^{exp}) \geqslant \bar{\Delta}^*. \tag{8}$$

If (8) holds with equality, then the capacity of the process $\mu(\mathcal{P}) = \mu(\mathcal{P}^{exp}) = 1/\kappa_f^*(G^{exp}) = 1/\bar{\Delta}^*$. Further, since $\kappa_f^*(G^{exp}) = \kappa^*(G^{exp}) = \bar{\Delta}^* = \max_{w \in W} \bar{\Delta}_w$, for any resource $w^* \in \bar{W}^*$, the clique in G^{exp} formed by the $\sum_{v \in V_{w^*}} \tau(v)$ nodes in $V_{w^*}^{exp}$ is a bottleneck structure of activities in $\mathcal{P}^{exp}$. Also, as observed above, this clique in G^{exp} corresponds to the clique in G formed by the corresponding parent nodes in G (i.e., the nodes corresponding to the activities in V that use resource w^*). In summary, the bottleneck structure of activities in $\mathcal{P}$ corresponds to a clique in G formed by the activities in V_{w^*} where $w^* \in \bar{W}^*$. Since this is exactly the calculation used in the simple bottleneck formula to identify a bottleneck resource, we conclude that the simple bottleneck formula correctly identifies a bottleneck set of resources and computes the capacity of $\mathcal{P}$ correctly. The converse also holds. We note our conclusion formally below.

THEOREM 2. *Consider a process $\mathcal{P}$ in which $\tau(v) \geqslant 1$ for each $v \in V$. Let G (resp., G^{exp}) denote the collaboration graph (resp., expanded collaboration graph) of $\mathcal{P}$. Then* (8) *holds with equality if and only if, for any resource $w^* \in \bar{W}^*$, the clique in G formed by the nodes in V_{w^*} is a bottleneck*

structure of activities for $\mathcal{P}$. In this case, the corresponding bottleneck set of resources is the singleton set $\{w^*\}$ and the simple bottleneck formula correctly identifies a bottleneck set of resources and also correctly computes the capacity of $\mathcal{P}$.

If inequality (6) (or the more-general inequality (8)) does not hold with equality, then a bottleneck structure of activities may still be a clique in the collaboration graph, but the bottleneck formula does not correctly identify a bottleneck set of resources. Example 10 in Appendix B illustrates this.

We now contrast Theorem 2 above with Theorem 2 in Gurvich and Van Mieghem (2015). Define the resource-activity incidence matrix $A = (a_{ij})_{|W|\times|V|}$ as follows:

$$a_{ij} = \begin{cases} 1, & \text{if resource } w_i \text{ is needed by activity } v_j, \\ 0, & \text{otherwise,} \end{cases}$$

for $i \in \{1, 2, \cdots, |W|\}$ and $j \in \{1, 2, \cdots, |V|\}$. Note that A does not specify the processing times of the activities. Gurvich and Van Mieghem (2015) show that if the polytope $P = \{x \in \mathbb{R}_+^V : Ax \leqslant 1\}$ is integral, then the bottleneck formula is valid for all possible processing times, and vice versa. It is important to notice that, for a *specific* process with given activity times, the integrality of P is not a necessary condition for the bottleneck formula to be valid. The integrality of P becomes necessary only if we want the bottleneck formula to be valid for *all* possible processing times. In contrast, Theorem 2 above provides a necessary and sufficient condition under which the bottleneck formula is correct for a given process; i.e., one for which the matrix A and the activity times are given. Clearly, the condition in Gurvich and Van Mieghem (2015) is much more stringent than ours, because their condition ensures the validity of the bottleneck formula for all possible processing times while ours applies only to a given process.

Having identified bottleneck structures for some special processes, we now take a closer look to identify the "minimal" constituents of a bottleneck structure for an arbitrary process.

6.2. A Subgraph Characterization of Bottleneck Structures

Our analysis in this section uses the Strong Perfect Graph Theorem (SPGT; see Section 3) and other results in the literature on perfect graphs – these graphs become relevant due to the following connection: if the collaboration graph of a process is perfect, then each basic optimal solution of problem D_κ, which defines a bottleneck structure of activities in the process, is an integral vector. In turn, this helps us in obtaining a simpler description of a bottleneck structure. Further, if the collaboration graph is not perfect, then too the SPGT helps us identify the necessary components of any bottleneck structure.

It is important to note that, given *any* undirected graph G, one can create a process whose collaboration graph is G (Bo et al. 2018). The class of perfect graphs also gains significance because a wide variety of collaboration graphs are perfect, including complete graphs, bipartite graphs, permutation graphs, interval graphs, split graphs, threshold graphs, comparability graphs, incomparability graphs, and perfectly orderable graphs; see, e.g., Golumbic (2004) and Alfonsin (2001).

As we have done until now, let us first consider a process $\mathcal{P}^1$ for which $\tau(v) = 1, v \in V$; thus, the collaboration graph is denoted by G^1.

(a) If G^1 is perfect[2], then the polytope $P(G^1) = \left\{\boldsymbol{y} \in \mathbb{R}_+^{|V|} : \sum_{v \in I} y_v \leqslant 1 \text{ for } I \in \mathcal{I}\right\}$ is *integral*; i.e., each extreme point of $P(G^1)$ is an integral vector (*Weak Perfect Graph Theorem*; Lovász 1972). Thus, each basic optimal solution of problem D_κ is *integral*; let $y_v^*, v \in V$, denote any such solution. Then, $y_v^* = \{0, 1\}$ for all $v \in V$, and the subgraph S_Q corresponding to the node set $Q = \{v \in V : y_v^* = 1\}$ is a clique in G^1. Thus, *each* bottleneck structure of activities for $\mathcal{P}^1$ is a clique in G^1.

(b) If G^1 is not perfect, then the polytope $P(G^1)$ has at least one fractional extreme point. Consider a fractional basic optimal solution (if one exists) $y_v^*, v \in V$, of problem D_κ; thus, $0 < y_v^* < 1$ for $v \in V^{'} \subseteq V; V^{'} \neq \varnothing$. Let $Q = \{v \in V : y_v^* > 0\}$ and let the subgraph S_Q of G^1 be the corresponding bottleneck structure of activities for $\mathcal{P}^1$. Then, $(y_v^*, v \in Q)$ is a maximum fractional clique in S_Q. It is also easy to verify that this is a *basic* maximum fractional clique in S_Q, i.e., a basic optimal solution of the fractional clique problem on S_Q. Recall that each basic optimal solution of the fractional clique problem on a perfect graph is integral. Thus, the fact that $(y_v^*, v \in Q)$ is fractional implies that the subgraph S_Q is not a perfect graph and, therefore, from the SPGT, contains either an odd-hole or an odd-antihole as a node-induced subgraph.

The argument above shows that any bottleneck structure S_Q is either a clique in the collaboration graph G^1 or *contains* either an odd-hole or an odd-antihole of G^1. The same conclusion holds for a process $\mathcal{P}$ with $\tau(v) \geqslant 1, v \in V$. Recall that, in this case, a bottleneck structure is defined using the expanded graph G^{exp}. Note that the collaboration graph G of $\mathcal{P}$ is perfect if and only if its expanded graph G^{exp} is perfect (Lovász 1972). Thus, the arguments in (a) and (b) again hold, with G^{exp} replacing G^1:

(a) If G is perfect, then so is G^{exp} and, therefore all bottleneck structures of activities for $\mathcal{P}$ are cliques in G^{exp}. Further, as noted earlier (Section 6.1.2), every clique in G^{exp} corresponds to a clique in G. Therefore, all bottleneck structures of activities for $\mathcal{P}$ are cliques in G.

[2]This is equivalent to the system of inequalities that defines problem D_κ, i.e., $\sum_{v \in I} y_v \leqslant 1, I \in \mathcal{I}; y_v \geqslant 0, v \in V$, being Totally Dual Integral (TDI); see Lovász (1972).

(b) If G is not perfect, then G^{exp} too is not perfect. Then, a bottleneck structure of activities for $\mathcal{P}$ is either a clique in G^{exp} or contains either an odd-hole or an odd-antihole of G^{exp}. In the former case, the corresponding structure in G is a clique. We now discuss the latter case. Consider any odd-hole in G^{exp} formed by the node set $V_o^{exp} \subseteq V^{exp}$ and let $V_o \subseteq V$ be the set of the parent nodes in G of the nodes in V_o^{exp}. From the construction of G^{exp}, no odd-hole in G^{exp} includes two or more children nodes of the same parent node in G. Therefore, the nodes in V_o form an odd-hole in G. Similarly, the parent nodes of the nodes of an anti-hole in G^{exp} form an anti-hole in G. Thus, *every odd-hole (resp., odd-antihole) in G^{exp} corresponds to an odd-hole (resp., odd-antihole) in G.* (**Note:** The converse also holds. Suppose the node set $\tilde{V} = \{v_1, v_2, \ldots, v_k\} \subseteq V$ forms an odd-hole (resp., odd-antihole) in G. Recall that node $v_j, j = 1, 2, \ldots, k$, has $\tau(v_j)$ children nodes, say $v_j^1, v_j^2, \ldots v_j^{\tau(v_j)}$, in G^{exp}. Then, any set of nodes in G^{exp} consisting of one child node of each of the nodes in $\tilde{V}$, i.e., any set of nodes $\{v_1^{\ell_1}, v_2^{\ell_2}, \ldots, v_k^{\ell_k}\}$, where $\ell_q \in \{1, 2, \ldots, \tau(v_q)\}; q = 1, 2, \ldots, k$, forms an odd-hole (resp., odd-antihole) in G^{exp}.) Therefore, we again see that a bottleneck structure of $\mathcal{P}$ is either a clique in the collaboration graph G or contains either an odd-hole or an odd-antihole of G.

Combining the discussion above, we have the following result; a formal proof is provided in Appendix B.

THEOREM 3. *Consider a process $\mathcal{P}$ in which $\tau(v) \geqslant 1$ for each $v \in V$. Let G denote the collaboration graph of $\mathcal{P}$. Then, each bottleneck structure of activities in $\mathcal{P}$ is either (i) a clique in G or (ii) contains an odd-hole or an odd-antihole (as a node-induced subgraph) of G.*

Example 4: We illustrate Theorem 3 for the shed manufacturing process in Example 1. The collaboration graph is shown in Figure 5. As discussed in Section 1, the unique bottleneck structure of activities is formed by activities 1, 3, and 5; these three activities form a clique in the collaboration graph. If we decrease the processing time of activity 1 from 9 minutes (as in Example 1) to 7 minutes, then there are two bottleneck structures of activities: one is the clique formed by activities 1, 3, and 5, the other is the odd-hole formed by activities 2, 3, 4, 5, and 6. ■

If the collaboration graph G is perfect, then we know that it does not contain any odd-hole or odd-antihole as an induced subgraph. Therefore, to identify bottleneck structures of activities, it suffices to search for maximum cliques in G; each such clique is a bottleneck structure of activities. If G is not perfect, then Theorem 3 only tells us that any bottleneck structure of activities is either a clique or includes an odd-hole or an odd-antihole of G as an induced subgraph. In other words, if a bottleneck structure of activities is not a clique, then an odd-hole or an odd-antihole is only guaranteed to be a *subgraph* of this structure. In this case, the best way to identify a bottleneck

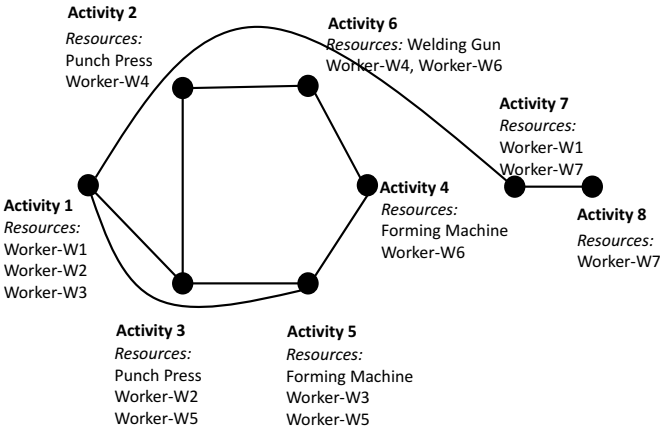


Figure 5 Collaboration graph of the shed manufacturing process in Example 4.

structure of activities is to solve the linear program D_κ (defined in Section 5). Each basic optimal solution of this linear program corresponds to a bottleneck structure of activities.

Remark 2: For a multi-product process, consider a consolidated product that represents a mix of the different products corresponding to their proportions in the mix. For example, consider a two-product process. For simplicity, assume that Product 1 and Product 2 require the same set of activities, denoted by V. For any $v \in V$, let $\tau_1(v)$ (resp., $\tau_2(v)$) denote the processing time of activity v for Product 1 (resp., Product 2). Consider a product mix in which the proportion of Product 1 is $p \in [0,1]$ and that of Product 2 is $1-p$. Then, for the consolidated product representing this product mix, the set of activities is V and the processing time of activity $v \in V$ is $p \cdot \tau_1(v) + (1-p) \cdot \tau_2(v)$. Thus, we now have a single-product process and our analysis can be used to examine bottleneck structures in this process. If μ^* is the capacity of this consolidated process, then the throughput rates of Product 1 and Product 2 are $p \cdot \mu^*$ and $(1-p) \cdot \mu^*$, respectively. ■

7. Extensions: Multiple Copies of Resources and Flexible Resource sets

Our characterization of a bottleneck structure of activities for a process and the corresponding bottleneck set(s) of resources in Section 5 can be generalized for two types of processes: (i) processes that have multiple copies of one or more resources, i.e., $N_w \geqslant 1, w \in W$, and (ii) processes with flexible resource sets; that is, processes in which each activity can potentially be performed by multiple sets of resources (this type of flexibility is discussed in Gurvich and Van Mieghem 2015). For brevity, here we only highlight the basic differences in these characterizations and relegate the entire technical discussion to appendices C and D.

When a process has multiple copies of one or more resources, the characterization of a bottleneck structure of activities and associated bottleneck sets of resources involves the construction of a collaboration *hypergraph*. A hypergraph is a generalization of a (simple) graph in which each edge corresponds to a subset of nodes. Formally, a hypergraph $\mathcal{G}$ is a pair $\mathcal{G} = (X, \mathcal{E})$, where X is the set of nodes, and $\mathcal{E}$ is a set of non-empty subsets of X called *hyperedges*. The following example illustrates the construction of the collaboration hypergraph for a process with multiple copies of one or more resources.

Example 5: Consider a process $\mathcal{P}$ in which each flow unit undergoes six activities, A, B, C, D, E, and F, to complete its processing. The resources needed for each activity are shown in Figure 6. Assume that each of the six activities takes one unit of time and that we have two units of resource a, two units of resource b, one unit of resource c, and one unit of resource d. In the collaboration hypergraph $\mathcal{G} = (X, \mathcal{E})$ of $\mathcal{P}$, each activity in $\mathcal{P}$ is represented by a node in $\mathcal{G}$; thus, we have $X = \{A, B, C, D, E, F\}$. Each hyperedge is a minimal set of activities that cannot be scheduled simultaneously. For instance, resource a is needed by activities A, B, and C, but we have only two units of resource a. Thus, these three activities can not be performed simultaneously and $\{A, B, C\}$ is a hyperedge. It is straightforward to check that the set of hyperedges is $\mathcal{E} = \big\{\{A, B, C\}, \{B, C, D\}, \{D, E\}, \{A, F\}\big\}$. Figure 6 shows the collaboration hypergraph of $\mathcal{P}$. ■

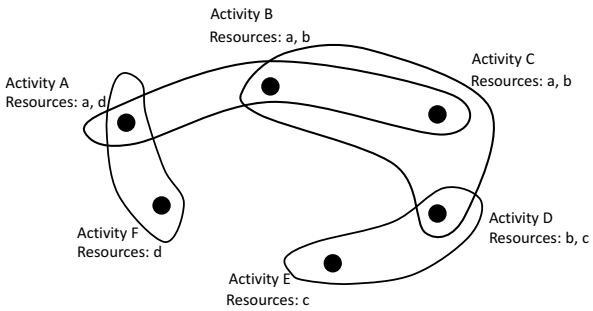


Figure 6 Collaboration hypergraph of the process (with multiple copies of some resources) in Example 5.

For a process with multiple copies of one or more resources, a bottleneck structure of activities and its associated bottleneck sets of resources are obtained by solving the fractional clique problem on its collaboration hypergraph. In Appendix C, we provide the precise definition of bottlenecks for such processes. We also present an alternate mathematical-programming-based formulation that avoids any graphical interpretation and establish the equivalence of these two characterizations.

In a process with flexible resource sets, for each activity, there can potentially be multiple sets of resources that can perform the activity. Consequently, there are multiple possible collaboration

graphs, each defined by a specific choice of the resource sets employed for the activities of the process. The following example illustrates this notion of flexibility.

Example 6: Consider the shed manufacturing process in Example 1. For simplifying the computations, we use the processing times shown in Table 3.

Table 3 Activities and processing times in the shed manufacturing process (Example 6).

Activity	Time (mins)
1: Separate the roof and base material	3
2: Punch the base	1
3: Punch the roof	2
4: Form the base	1
5: Form the roof	2
6: Subassemble the base	1
7: Assemble	1
8: Inspect	3

It is straightforward to verify that the bottleneck structure for this process is the clique formed by activities 1, 3, and 5, the bottleneck set of resources is {worker-W2, worker-W3, worker-W5}, and the process capacity is $\frac{1}{7}$ sheds/minute.

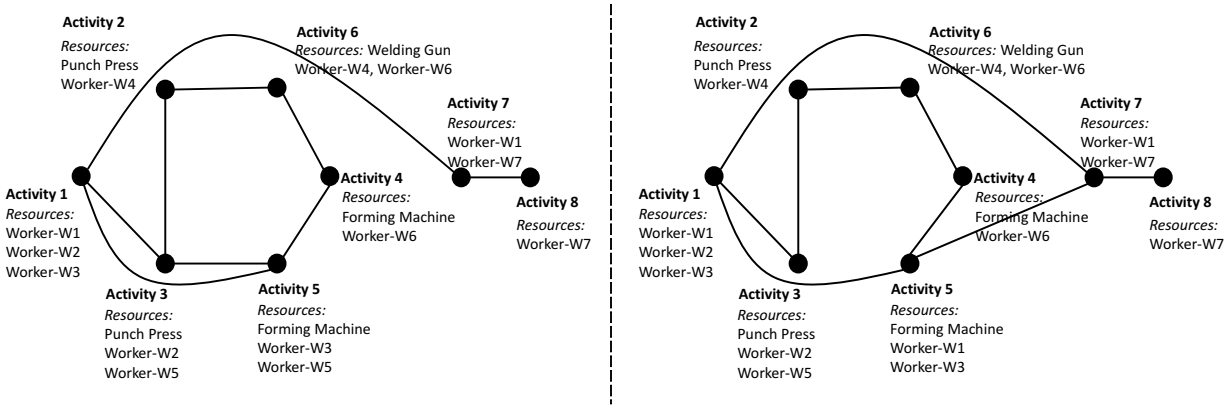


Figure 7 Collaboration graphs of the process (with flexible resource sets) in Example 6.

Next, we introduce flexibility by training worker-W1 to replace worker-W5 in activity 5. This leads to two optional resource sets {forming machine, worker-W3, worker-W5} and {forming machine, worker-W1, worker-W3} that can perform activity 5, and we can choose how to distribute the activity's load between them. This added flexibility results in two optional collaboration graphs shown in Figure 7. Observe that the introduction of flexibility allows us to shift some

work from worker-W5 to worker-W1; this increases the capacity of the process from $\frac{1}{7}$ sheds/minute to $\frac{2}{11}$ sheds/minute (see Appendix D). ■

With multiple possible collaboration graphs, the specification of a bottleneck structure of activities for a flexible process is more complex: it consists of a *collection* of fractional cliques (not necessarily maximum fractional cliques), one from each possible collaboration graph. In Appendix D, we provide the precise definition of bottlenecks for a flexible process.

The technical developments in sections 5, 6, and 7 can be summarized to obtain an algorithmic procedure for identifying bottlenecks in an arbitrary deterministic, single-product process. Next, we provide this summary.

8. Identification of Bottlenecks: A Summary

Not surprisingly, there is a natural hierarchy in the description below, ranging from the "easy" case at the one extreme, where the simple bottleneck formula correctly identifies bottlenecks, to more-sophisticated cases, where one needs to solve math programs.

1. We first summarize the identification of bottlenecks for a process $\mathcal{P}$ that has one copy of each resource, i.e., $N_w = 1$ for all $w \in W$ (Section 4, Section 5, and Section 6).

   - Define the collaboration graph G of process $\mathcal{P}$.

     ◦ If each activity takes one unit of time (i.e., $\tau(v) = 1$, for all $v \in V$), then each node of G corresponds to an activity of the process and two nodes are connected by an edge if the corresponding activities share at least one common resource.

     ◦ If the activity times are general positive integers (i.e., $\tau(v) \geqslant 1$, for all $v \in V$), then each node of G corresponds to an activity of the expanded process $\mathcal{P}^{exp}$, which is obtained by splitting each activity $v \in V$ with $\tau(v) > 1$ into $\tau(v)$ activities that each take unit time. Two nodes of G are connected by an edge if the corresponding activities share at least one common resource.

   - If $\mathcal{P}$ does not involve either collaboration or multitasking, then any maximum clique in G formed by a single resource is a bottleneck structure of activities for $\mathcal{P}$, and a corresponding bottleneck set of resources is a singleton set. In this case, the simple bottleneck formula correctly identifies each bottleneck resource.

   - If $\mathcal{P}$ involves both collaboration and multitasking, then we check whether or not G is a perfect graph. This can be determined in polynomial time (Chudnovsky et al. 2005). Recall

that for each bottleneck structure of activities, a corresponding bottleneck set of resources is identified by finding a minimal set of resources which generates that structure.

- If G is perfect, then each maximum clique in G is a bottleneck structure of activities for $\mathcal{P}$. A maximum clique in a perfect graph can be obtained in polynomial time (see, e.g., Grötschel et al. 2012).

- If G is not perfect, then each basic optimal solution of the fractional clique problem D_κ (defined in Section 5) on G corresponds to a bottleneck structure of activities for $\mathcal{P}$. Further, from Theorem 3, each bottleneck structure of activities in $\mathcal{P}$ is either a clique in G or contains an odd-hole or an odd-antihole of G.

2. Next, we summarize two ways to identify the bottlenecks for a process $\mathcal{P}$ that has multiple copies of one or more resources, i.e., $N_w \geqslant 1, w \in W$ (Section 7 and Appendix C). As before, we define the collaboration graph G of $\mathcal{P}$.

   - One approach, that uses collaboration hypergraphs, is as follows: (i) for $k \in \mathbb{N}$ satisfying $k \cdot \tau(v) \geqslant N_w$ for $w \in W_v, v \in V$, use G to construct the collaboration hypergraph $\mathcal{G}^k$ of $\mathcal{P}$ (defined in Appendix C.1). (ii) each basic optimal solution of the fractional clique problem D_κ^k (defined in Appendix C.1) on $\mathcal{G}^k$ corresponds to a bottleneck structure of activities for $\mathcal{P}$.

   - The other approach is purely mathematical-programming-based: each basic optimal solution of the linear program $D_{\hat{\kappa}}$ (Appendix C.2) corresponds to a bottleneck structure of activities for $\mathcal{P}$.

   Given a bottleneck structure of activities, a corresponding bottleneck set of resources is obtained by obtaining a minimal set of resources that generates it.

3. Finally, we summarize the identification of bottlenecks for a process $\mathcal{P}^F$ with flexible resource sets (Section 7 and Appendix D). In such processes, for each activity, there can potentially be multiple sets of resources that can perform the activity. Consequently, there are multiple possible collaboration graphs, each defined by a specific choice of resource sets employed for the activities of the process. If $N_w = 1, w \in W$, then the bottlenecks for $\mathcal{P}^F$ can be identified as follows: (i) construct all possible collaboration graphs of the process; (ii) using these collaboration graphs, define the linear program D_κ^F (defined in Appendix D); (iii) each basic optimal solution of D_κ^F corresponds to a bottleneck structure of activities for $\mathcal{P}^F$; (iv) for each bottleneck structure of activities, a corresponding bottleneck set of resources is obtained by finding a minimal set of resources that generates it.

The extension for the case where a process with flexible resource sets has $N_w \geqslant 1, w \in W$, is similar, but uses collaboration hypergraphs.

9. Discussion and Concluding Remarks

Apart from the study of bottleneck structures of activities (and the associated bottleneck sets of resources) in processes, our analysis can also enable the examination of several sophisticated questions of practical interest. We discuss two such issues below:

- **The Resource-Capacity Investment Problem:** Given a budget B, find a set of resources to acquire to maximize the improvement in process capacity. We now formulate this problem.

Consider a generic process $\mathcal{P}$ that currently has $N_w \geqslant 1$ copies of resource w, $w \in W$. Recall that V denotes the set of activities in the process and W_v is the set of resources needed for activity v; $v \in V$. Let c_w denote the cost of purchasing a unit of resource $w \in W$. Let $a_w \in \mathbb{N}$ denote the number of additional units of each resource $w \in W$ to acquire, given the budget B. Clearly, a *feasible* acquisition $\boldsymbol{a} = (a_w, w \in W)$ satisfies $\sum_{w \in W} a_w c_w \leqslant B$. Following such an acquisition, the number of units of resource $w \in W$ increases to $N_w + a_w$. For this process, an *independent tuple* $I = (n_v^I)_{v \in V}$ is a $|V|$-tuple satisfying

$$\sum_{v: w \in W_v} n_v^I \leqslant N_w + a_w \text{ for } w \in W, \quad \text{and} \quad n_v^I \geqslant 0 \text{ is an integer for } v \in V \text{ (see Appendix C.2).}$$

For an independent tuple $I = (n_v^I)_{v \in V}$, it is immediate that n_v^I "copies" of activity v, $v \in V$, can be scheduled simultaneously (i.e., in the same time unit). Let $\mathcal{I}^{\boldsymbol{a}}$ be the collection of all the independent tuples. Then, process capacity is the *reciprocal* of the optimal objective value of problem $D_{\hat{\kappa}}$ defined in Appendix C.2. Thus, the acquisition $(a_w, w \in W)$ that provides the maximum increase in process capacity is obtained by solving the following problem:

$$\begin{aligned} \min_{a_w, w \in W} \max_{y_v, v \in V} \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & \sum_{v \in V} n_v^I y_v \leqslant 1 \quad \text{for} \quad I \in \mathcal{I}^{\boldsymbol{a}}, \\ & \sum_{w \in W} a_w c_w \leqslant B, \\ & y_v \geqslant 0 \quad \text{for} \quad v \in V, \\ & a_w \in \mathbb{N} \quad \text{for} \quad w \in W. \end{aligned} \tag{9}$$

Example 7: Consider the shed manufacturing process (with no flexibility) in Example 6, and assume the following unit costs of the resources: (i) hiring a worker: 5, (ii) acquiring a punch

press: 7, (iii) acquiring a forming machine: 7, (iv) acquiring a welding gun: 4. Consider a budget of $B = 10$. By solving problem (9) above, we find that the maximum improvement in process capacity (from 1/7 sheds/minute to 1/4 sheds/minute) is achieved by acquiring an additional worker with skills identical to worker-W2 and an additional worker with skills identical to worker-W3. ■

- **The "Design" of Processes:** For obvious reasons, operations that can potentially increase capacity *without* procuring additional resources are attractive for practitioners. We discuss two basic operations, namely *splitting an activity* and *enhancing a resource*.

Splitting an activity: Consider an activity, denoted A, that requires a set $R \subseteq W, |R| \geqslant 2$, of resources simultaneously. We define the basic operation of "splitting" activity A as one in which this activity is replaced by two sub-activities, say A_1 and A_2, such that activity A_1 (resp., A_2) requires the resources in set $R_1 \neq \varnothing$ (resp., $R_2 \neq \varnothing$), where $R_1 \cup R_2 = R$ and $R_1 \cap R_2 = \varnothing$.

The splitting operation reduces the need for collaboration among the resources that are required for activity A in the original process. However, splitting an activity may slow down either or both the sub-activities. The following example illustrates this tradeoff in the splitting operation.

Example 8: Consider the shed manufacturing process (with no flexibility) in Example 6; the collaboration graph is shown in the left panel of Figure 8 below. The bottleneck structure is the clique formed by activities 1, 3, and 5, and the capacity is $\frac{1}{7}$ sheds/minute. Suppose that Activity 5, which simultaneously requires the forming machine, worker-W3, and worker-W5, is split into two activities: 9 and 10, with the former requiring worker-W5 and the latter requiring the forming machine and worker-W3. The right panel of Figure 8 shows the collaboration graph of the modified process. Recall that $\tau(v)$ denotes the processing time of activity $v \in V$ and note that $\tau(\text{Activity 5}) = 2$ minutes. Depending on the processing times of sub-activities 9 and 10, the capacity of the modified process may either increase or decrease. For example:

- If $\tau(\text{Activity 9}) = \tau(\text{Activity 10}) = 2$ minutes, then capacity increases to $\frac{1}{5}$ sheds/minute.
- If $\tau(\text{Activity 9}) = 2$ mins and $\tau(\text{Activity 10}) = 5$ minutes, then capacity decreases to $\frac{1}{8}$ sheds/minute. ■

Enhancing a resource: The basic operation of enhancing a resource endows that resource with the ability of one or more other resources. Here is a simple example: consider a resource, say a, that is needed simultaneously with another resource, say b, (and possibly other resources as well) for performing one or more activities of the process. Then, one possible enhancement of resource a is to make it capable of performing the role of resource b as well. That is, the enhanced resource,

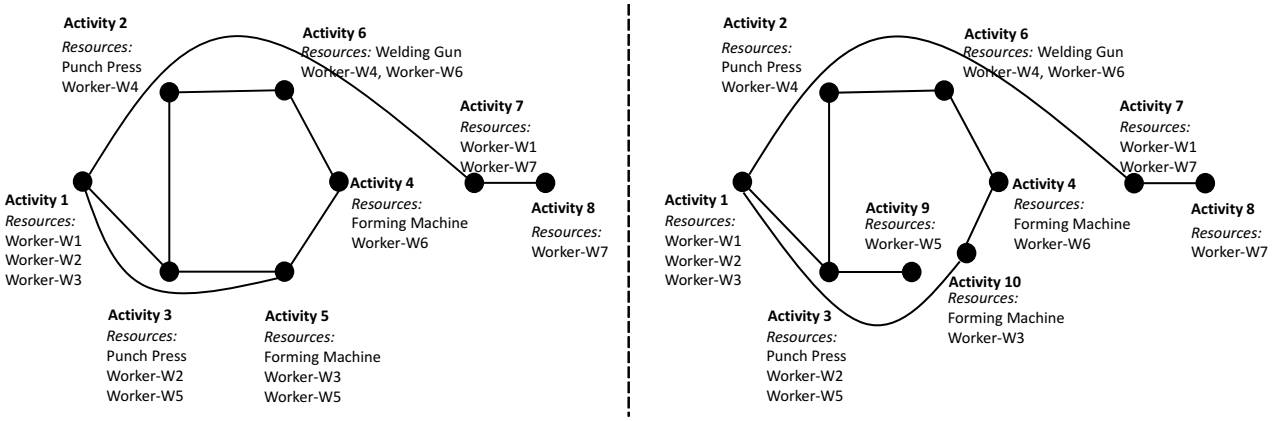


Figure 8 Splitting an activity: collaboration graphs in Example 8.

denoted $\hat{a}$, can be used for activities that require either resource a or b or both these resources in the original process. Note that resource b stays unaffected by this enhancement operation on resource a, and may be used, if needed, as it was prior to the enhancement.

The operation of enhancing resource a reduces the collaboration between resources a and b. However, there could be an increase in the processing times of activities that earlier required a and b simultaneously but now use only $\hat{a}$. The following example illustrates this tradeoff:

Example 9: Consider the same process as in the previous example; the collaboration graph is shown in the left panel of Figure 9. Recall that the process capacity is $\frac{1}{7}$ sheds/minute. Suppose that worker-W3 is enhanced and endowed with the additional capability of worker-W5. Activity 5, which earlier simultaneously needed the forming machine, worker-W3, and worker-W5, now only uses the forming machine and the "enhanced" worker-W3. The right panel of Figure 9 shows the collaboration graph of the modified process. Depending on the revised processing time $\hat{\tau}(\text{Activity } 5)$ of Activity 5, the process capacity may either increase or decrease. For example:

- If $\hat{\tau}(\text{Activity } 5) = 2$ minutes, then capacity increases to $\frac{1}{5}$ sheds/minute.
- If $\hat{\tau}(\text{Activity } 5) = 5$ minutes, then capacity decreases to $\frac{1}{8}$ sheds/minute. ■

It is reasonable to assume that, driven by the desire to increase capacity, practitioners would be interested in both identifying and altering bottlenecks in their processes. For a given process, there are multiple options to influence a bottleneck structure of activities, including (a) procuring more resources, (b) reducing the extent of collaboration by splitting an activity or enhancing a resource, and (c) increasing flexibility by cross-training resources so that activities can be performed by any one of multiple resource sets. For a budget-constrained firm interested in increasing capacity,

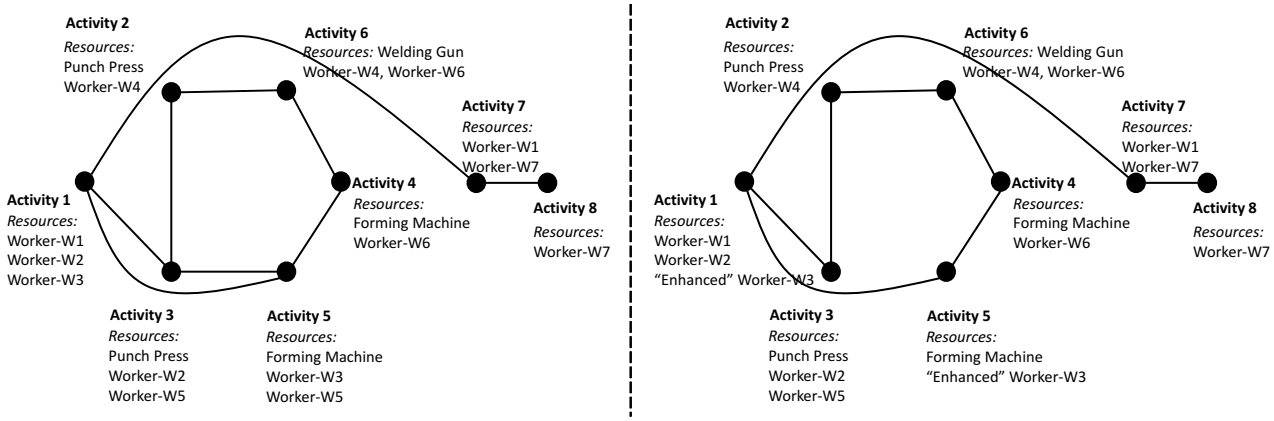


Figure 9 Enhancing a resource: collaboration graphs in Example 9.

determining a mix of these options that yields the best results is a challenging problem. We believe that our analysis can serve as a foundation for future work on this topic.

References

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

Alcaide, D., C. Chu, V. Kats, E. Levner, G. Sierksma. 2007. Cyclic Multiple-Robot Scheduling with Time-Window Constraints Using a Critical Path Approach. *European Journal of Oper. Res.* **177**(1) 147–162.

Alfonsin, J. L. R. 2001. *Perfect Graphs*. Wiley.

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

Bertsimas, D., T. Chryssikou. 1999. Bounds and Policies for Dynamic Routing in Loss Networks. *Oper. Res.* **47**(3) 379–394.

Bo, Y., M. Dawande, T. Huh, G. Janakiraman, M. Nagarajan. 2018. Determining Process Capacity: Intractability and Efficient Special Cases. *Manufacturing & Service Oper. Management* **21**(1) 16–33.

Chudnovsky, M., G. Cornuéjols, X. Liu, P. Seymour, K. Vusković. 2005. Recognizing Berge Graphs. *Combinatorica* **25**(2) 143–186.

Chudnovsky, M., N. Roberson, P. D. Seymour, R. Thomas. 2006. The Strong Perfect Graph Theorem. *Annals of Mathematics* **164**(1) 51–229.

Chvatal, V. 1979. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research* **4**(3) 233–235.

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

Dai, J. G., W. Lin. 2005. Maximum Pressure Policies in Stochastic Processing Networks. *Oper. Res.* **53**(2) 197–218.

Dawande, M., H. N. Geismar, S. P. Sethi, C. Sriskandarajah. 2007. *Throughput Optimization in Robotic Cells*. Springer.

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

Debels, D., M. Vanhoucke. 2007. A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. *Oper. Res.* **55**(3) 457–469.

Dobson, G., U. S. Karmarkar. 1989. Simultaneous Resource Scheduling to Minimize Weighted Flow Times. *Oper. Res.* **37**(4) 592–600.

Dorndorf, U., E. Pesch, T. Phan-Huy. 2000. A Time-Oriented Branch-and-Bound Algorithm for Resource-Constrained Project Scheduling with Generalised Precedence Constraints. *Management Science* **46**(10) 1365–1384.

Edmonds, J., R. Giles. 1984. Total Dual Integrality of Linear Inequality Systems. In *Progress in Combinatorial Optimization* 117–129.

Godsil, C., G. Royle. 2001. *Algebraic Graph Theory*. Springer-Verlag, New York.

Golumbic, M. C. 2004. *Algorithmic Graph Theory and Perfect Graphs*. Elsevier.

Grötschel, M., L. Lovász, A. Schrijver. 2012. *Geometric Algorithms and Combinatorial Optimization*. Springer Science & Business Media, Berlin.

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.

Harrison, J. M. 2000. Brownian Models of Open Processing Networks: Canonical Representation of Workload. *The Annals of Applied Probability* **10**(1) 75–103.

Harrison, J. M. 2002. Stochastic Networks and Activity Analysis. *American Mathematical Society Translations* 53–76.

Harrison, J. M. 2003. A Broader View of Brownian Networks. *The Annals of Applied Probability* **13**(3) 1119–1150.

Hartmann, S., D. Briskorn. 2010. A Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem. *European Journal of Oper. Res.* **207**(1) 1–14.

Jiang, L., J. Walrand. 2010. Scheduling and Congestion Control for Wireless and Processing Networks. *Synthesis Lectures on Communication Networks* **3**(1) 1–156.

Jung, K., Y. Lu, D. Shah, M. Sharma, M. S. Squillante. 2019. Revisiting Stochastic Loss Networks: Structures and Approximations. *Mathematics of Operations Research* **44**(3) 890–918.

Kelly, F. P. 1991. Loss Networks. *The Annals of Applied Probability* **1**(3) 319–378.

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

Levner, E., V. Kats. 1998. A Parametric Critical Path Problem and an Application for Cyclic Scheduling. *Discrete Applied Mathematics* **87**(1) 149–158.

Levner, E., V. Kats, D. Alcaide López de Pablo, T.C.E. Cheng. 2010. Complexity of Cyclic Scheduling Problems: A State-of-the-Art Survey. *Computers & Industrial Engineering* **59**(2) 352–361.

Lovász, L. 1972. Normal Hypergraphs and the Perfect Graph Conjecture. *Discrete Mathematics* **2**(3) 253–267.

Lund, C., M. Yannakakis. 1994. On the Hardness of Approximating Minimization Problems. *Journal of the ACM* **41**(5) 960–981.

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. *Oper. Res.* **37**(6) 925–935.

Möhring, R. H., A. S. Schulz, F. Stork, M. Uetz. 2003. Solving Project Scheduling Problems by Minimum Cut Computations. *Management Science* **49**(3) 330–350.

Roundy, R. 1992. Cyclic Schedules for Job Shops with Identical Jobs. *Mathematics of Operations Research* **17**(4) 842–865.

Scheinerman, E., D. Ullman. 2011. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs*. Courier Corporation.

Schrijver, A. 1998. *Theory of Linear and Integer Programming*. Wiley Series in Discrete Mathematics and Optimization, Wiley.

Slavik, P. 1997. A Tight Analysis of the Greedy Algorithm for Set Cover. *Journal of Algorithms* **25**(2) 237–254.

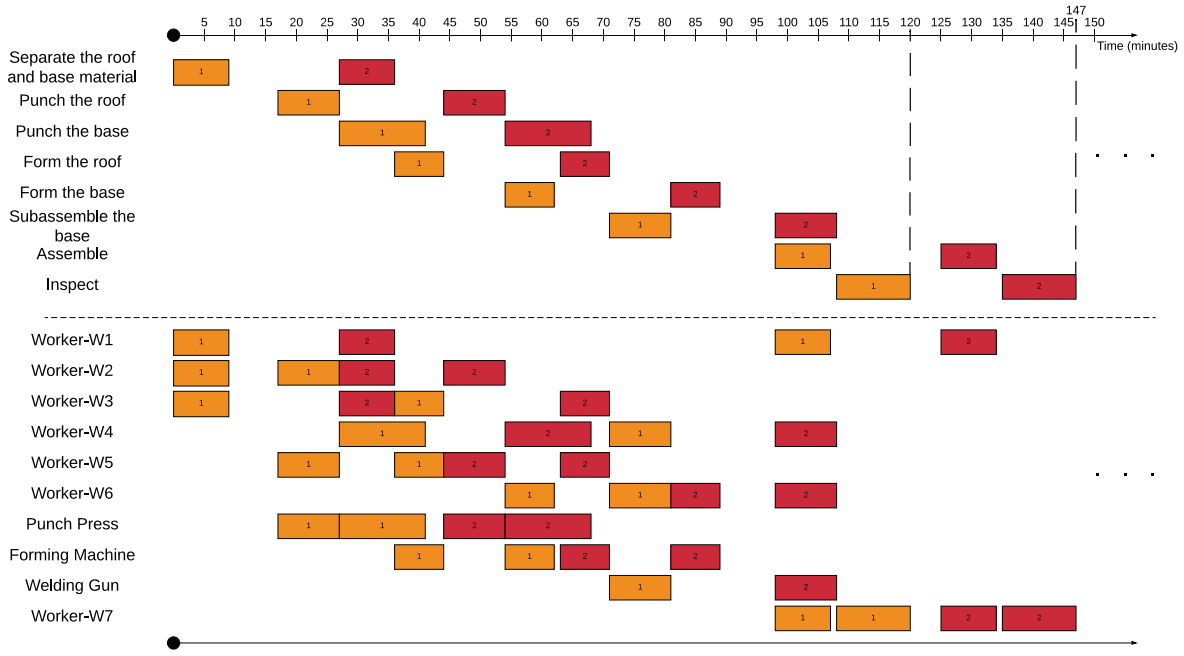
Wang, S., H. Su, G. Wan. 2015. Resource-Constrained Machine Scheduling with Machine Eligibility Restriction and Its Applications to Surgical Operations Scheduling. *Journal of Combinatorial Optimization* **30**(4) 982–995.

Online Appendix

Appendix A Feasible Schedule for the Shed Manufacturing Process in Example 1

In this section, we present a feasible schedule that achieves the capacity of the shed manufacturing process in Example 1 (Section 1). Recall that in this process, each flow unit completes the following activities: separate the roof and base material, punch the base, punch the roof, form the base, form the roof, subassemble the base, assemble, and inspect. The processing time and the required resources for each of these activities are listed in Table 1 (Section 1). Figure 10 shows the Gantt chart of a cyclic schedule that achieves the capacity of the process (which equals 1/27 sheds/minute or 20/9 sheds/hour). Here, different colors represent different flow units. Notice that the time between the completion of two consecutive units is 27 minutes; thus, the throughput under this schedule is 1/27 sheds/minute or 20/9 sheds/hour.

Figure 10 Gantt chart of a feasible schedule for the shed manufacturing process in Example 1.



Appendix B Technical Details in Section 6

No Collaboration or No Multitasking: First, let us assume that each activity takes one unit of time; i.e., $\tau(v) = 1$, for all $v \in V$.

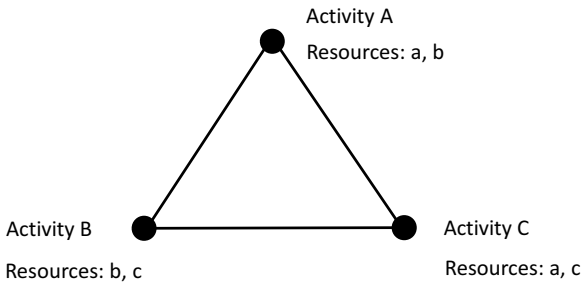
- If there is no activity that needs the collaboration of two or more resources, then the collaboration graph G^1 of the process is a collection of *disjoint* cliques – each corresponding to the set of activities that require the same resource. It is then straightforward to verify that a basic maximum fractional clique in G^1 is a clique with the highest cardinality (i.e., one containing the largest number of nodes) among these disjoint cliques. Each such maximum clique in G^1 is a bottleneck structure of activities for the process; the corresponding bottleneck set of resources is the *singleton set* consisting of the resource that is used for each activity in that clique.

- If there is no multitasking; i.e., no resource is needed for two or more activities, then G^1 is an "empty" graph (i.e., it has no edges). Thus, each node of G^1 is a basic maximum fractional clique. Consequently, the description of the bottlenecks for the process is trivial: each activity, by itself, defines a bottleneck structure and each resource is a bottleneck set of resources.

If the activity times are general positive integers (i.e., $\tau(v) \geqslant 1, v \in V$), then, as discussed earlier, we consider the expanded process $\mathcal{P}^{exp}$ – recall that this process is obtained by splitting each activity $v \in V$ with $\tau(v) > 1$ into $\tau(v)$ activities that each take unit time – and obtain the corresponding collaboration graph G^{exp}. In the absence of either collaboration or multitasking, G^{exp} is a collection of disjoint cliques. In both cases, a clique of maximum size in G^{exp} corresponds to a bottleneck structure of activities for the original process $\mathcal{P}$. In the absence of collaboration, for each resource $w \in W$, let V_w denote the set of activities in $\mathcal{P}$ that need resource w; that is, $V_w = \big\{v \in V : W_v = \{w\}\big\}$ and let $L(w) = \sum_{v \in V_w} \tau(v)$. Then, each resource $w^* = \arg\max_{w \in W} L(w)$ is a bottleneck set of resources. In the absence of multitasking, each resource that is needed for activity $v^* = \arg\max_{v \in V} \tau(v)$ is a bottleneck set of resources.

Example 10: Consider the process whose collaboration graph is shown in Figure 11. Assume that each of the three activities, namely, A, B, and C, takes one unit of time and that we have one unit of each of the three resources, namely, a, b, and c.

Figure 11 The collaboration graph of the process in Example 10.



For this process, the bottleneck structure of activities is the entire collaboration graph, which is a clique of size 3. Since $\Delta_a = \Delta_b = \Delta_c = 2$, we have $\Delta^* < \kappa_f^* = 3$; thus, (6) does not hold with equality. The bottleneck formula identifies each individual resource (i.e., $\{a\}$, $\{b\}$, or $\{c\}$) as a bottleneck resource and claims the process capacity to be $1/\Delta^* = 1/2$, while the unique bottleneck set of resources is the set $\{a, b, c\}$ and the correct value of the process capacity is $1/\kappa_f^* = 1/3$. ■

Proof of Theorem 3: Let $G^{exp} = (V^{exp}, E^{exp})$ be the expanded collaboration graph for the process. Consider any basic optimal solution $\boldsymbol{y}^* = (y_v^*, v \in V^{exp})$ of problem D_κ. Let $Q = \{v \in V^{exp} : y_v^* > 0\}$ and let the subgraph S_Q of G^{exp} be the corresponding bottleneck structure of activities for $\mathcal{P}$. Then, $(y_v^*, v \in Q)$ is a fractional clique of S_Q, i.e., a feasible solution to the fractional clique problem on S_Q. The polytope $P(S_Q) = \left\{\boldsymbol{y} \in \mathbb{R}_+^{|Q|} : \sum_{v \in I_Q} y_v \leqslant 1 \text{ for } I_Q \in \mathcal{I}_Q\right\}$ is defined by the set of constraints for the fractional clique problem on S_Q, where $\mathcal{I}_Q$ denotes the collection of all the independent sets of S_Q. Now, observe that $(y_v^*, v \in Q)$ is an extreme point of $P(S_Q)$. Suppose not, then $(y_v^*, v \in Q)$ can be expressed as a convex combination of two distinct points in $P(S_Q)$, say $(x_v, v \in Q)$ and $(z_v, v \in Q)$. By defining $x_v = z_v = 0$ for $v \in V^{exp} \backslash Q$, we obtain two distinct points $(x_v, v \in V^{exp})$ and $(z_v, v \in V^{exp})$ that are both feasible for D_κ. Then, by construction, $\boldsymbol{y}^* = (y_v^*, v \in V^{exp})$ is a convex combination of the points $(x_v, v \in V^{exp})$ and $(z_v, v \in V^{exp})$, contradicting the assumption that $\boldsymbol{y}^*$ is a basic optimal solution of D_κ.

Since $(y_v^*, v \in Q)$ is an extreme point of $P(S_Q)$, if S_Q is perfect, then $(y_v^*, v \in Q)$ is integral. This, together with the fact that $y_v^* > 0$ for $v \in Q$ implies that $y_v^* = 1$ for each $v \in Q$ and, hence, S_Q is a clique in G^{exp}. Otherwise, S_Q is not perfect and, by the SPGT, contains an odd-hole or an odd-antihole of G^{exp} as a node-induced subgraph.

As explained in Sections 6.1.2 and 6.2, (i) every clique in G^{exp} corresponds to a clique in G, and (ii) every odd-hole (resp., odd-antihole) in G^{exp} corresponds to an odd-hole (resp., odd-antihole) in G. Therefore, each bottleneck structure of activities in $\mathcal{P}$ is either a clique in G or contains an odd-hole or an odd-antihole (as a node-induced subgraph) of G. ■

Appendix C Extension: Multiple Copies of Resources

In this section, we characterize a bottleneck structure of activities and the corresponding bottleneck set(s) of resources for processes that have multiple copies of one or more resources, i.e., $N_w \geqslant 1, w \in W$; thus, this analysis generalizes our discussion in Section 5. We present two different, but equivalent, generalizations: Section C.1 characterizes bottleneck structures using collaboration *hypergraphs* while Section C.2 presents an LP-based definition and avoids any graphical interpretation.

Throughout this section, we consider a generic process $\mathcal{P} = \big(V, \varnothing, \tau, W, \{N_w; w \in W\}, \{W_v; v \in V\}, 0\big)$, with $N_w \geqslant 1, w \in W$.

C.1 Characterization of Bottleneck Structures Using Hypergraphs

For expositional simplicity, we assume that each activity takes one unit of time; i.e., $\tau(v) = 1, v \in V$. If the activity times are general positive integers (i.e., $\tau(v) \geqslant 1, v \in V$), then, as discussed in Section 4, we consider the expanded process $\mathcal{P}^{exp}$, obtained by splitting each activity v with $\tau(v) > 1$ into $\tau(v)$ activities that each take unit time, and work with the corresponding collaboration graph G^{exp}. We first review a characterization, established in Bo et al. (2018), of process capacity using hypergraphs when there are multiple copies of some or all of the resources. A hypergraph $\mathcal{G}$ is a pair $\mathcal{G} = (X, \mathcal{E})$, where X is the set of nodes, and $\mathcal{E}$ is a set of non-empty subsets of X called *hyperedges*. For $k \in \mathbb{N}$, we define the *collaboration hypergraph* $\mathcal{G}^k(V^k, \mathcal{E}^k)$ of a process $\mathcal{P}$ as follows:

(a) The set of nodes V^k: For each activity $v \in V$, we make k copies and denote these as $v^1, v^2, \cdots, v^k$. For each copy v^j, the duration of v^j and the set of resources needed to perform v^j are the same as those for activity v. The set of nodes V^k of $\mathcal{G}^k$ is defined as follows:

$$V^k := \Big\{ v^j : j \in \{1, 2, \cdots, k\}, v \in V \Big\}.$$

(b) The set of hyperedges $\mathcal{E}^k$: For $v \in V^k$, let W_v denote the set of resources needed to perform v. For $S \subseteq V^k$, we have $S \in \mathcal{E}^k$ if and only if the following two conditions are satisfied:

(i) All the activities in S cannot be scheduled simultaneously, i.e.,

$$\big|\{v : v \in S, w \in W_v\}\big| > N_w \text{ for some } w \in W. \tag{10}$$

(ii) There exists no $S' \subseteq V^k$ satisfying (10) such that $S' \subsetneq S$. In other words, S is a "minimal set" that satisfies (10).

An *independent set* $\tilde{I}$ of the hypergraph $\mathcal{G}^k$ is a subset of V^k such that all the activities in $\tilde{I}$ can be performed simultaneously. That is,

$$\big|\{v : v \in \tilde{I}, w \in W_v\}\big| \leqslant N_w \text{ for each } w \in W.$$

Let $\tilde{\mathcal{I}}$ denote the collection of all the independent sets of $\mathcal{G}^k$. The following LP computes the fractional chromatic number $\chi_f^*(\mathcal{G}^k)$ of $\mathcal{G}^k$:

$$\begin{aligned} \chi_f^*(\mathcal{G}^k) = \min \quad & \sum_{\tilde{I} \in \tilde{\mathcal{I}}} x_{\tilde{I}} \\ \text{s.t.} \quad & \sum_{\tilde{I} : v \in \tilde{I}} x_{\tilde{I}} \geqslant 1 \quad \text{for } v \in V^k, \\ & x_{\tilde{I}} \geqslant 0 \quad \text{for } \tilde{I} \in \tilde{\mathcal{I}}. \end{aligned} \tag{P_χ^k}$$

Bo et al. (2018) establish the following result:

THEOREM 4. (Bo et al. 2018) *Let $\mathcal{P}$ be an arbitrary deterministic, single-product process, as defined in Section 3. For every k, $k \in \mathbb{N}$, satisfying*

$$k \cdot \tau(v) \geqslant N_w \text{ for } w \in W_v, v \in V, \tag{11}$$

we have,

$$\mu(\mathcal{P}) = \frac{k}{\chi_f^*(\mathcal{G}^k)}.$$

The dual of the LP P_χ^k above computes the fractional clique number $\kappa_f^*(\mathcal{G}^k)$ of $\mathcal{G}^k$. Letting y_v, $v \in V^k$, denote the dual variables, the formulation of the dual is:

$$\begin{aligned} \kappa_f^*(\mathcal{G}^k) = \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & \sum_{v \in \tilde{I}} y_v \leqslant 1 \quad \text{for} \quad \tilde{I} \in \tilde{\mathcal{I}}, \\ & y_v \geqslant 0 \quad \text{for} \quad v \in V^k, \end{aligned} \tag{D_κ^k}$$

Let $\boldsymbol{y} = (y_v, v \in V^k)$ denote a feasible solution of D_κ^k (referred to as a fractional clique in $\mathcal{G}^k$) and $\kappa_f(\boldsymbol{y}, \mathcal{G}^k)$ denote its objective value. Let $\boldsymbol{y}^* = (y_v^*, v \in V^k)$ denote a basic optimal solution of D_κ^k (i.e., a basic maximum fractional clique in $\mathcal{G}^k$). From LP duality, we have

$$\kappa_f(\boldsymbol{y}, \mathcal{G}^k) \leqslant \kappa_f^*(\mathcal{G}^k) = \chi_f^*(\mathcal{G}^k).$$

Then, using Theorem 4, we have

$$\mu(\mathcal{P}) = \frac{k}{\chi_f^*(\mathcal{G}^k)} = \frac{k}{\kappa_f^*(\mathcal{G}^k)} \leqslant \frac{k}{\kappa_f(\boldsymbol{y}, \mathcal{G}^k)}, \text{ for } k \text{ satisfying (11)}.$$

Let $Q = \{v \in V^k : y_v^* > 0\}$. The node-induced sub-hypergraph $\mathcal{S}_Q$ of $\mathcal{G}^k$ corresponding to the activities in Q limits the capacity of the process to $\mu(\mathcal{P})$; thus, $\mathcal{S}_Q$ is a *bottleneck structure of activities* for $\mathcal{P}$. Let $R_Q = \bigcup_{v \in Q} W_v$ be the collection of resources that are needed for at least one of the activities in Q. Any minimal subset $\hat{R}_Q$ of R_Q that induces a sub-hypergraph isomorphic to $\mathcal{S}_Q$ is a *bottleneck set of resources* for $\mathcal{P}$.

C.2 An Alternate Characterization of Bottleneck Structures

One can avoid the notion of hypergraphs and, instead, present an alternate mathematical-programming-based definition of a bottleneck structure of activities. We now present this alternate

characterization. Again, for expositional simplicity, $\tau(v) = 1, v \in V$. We refer to a $|V|$-tuple $I = (n_v^I)_{v \in V}$ as an *independent tuple* of $\mathcal{P}$ if it satisfies the following:

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

Let $\mathcal{I}$ be the collection of all the independent tuples of $\mathcal{P}$. It is easy to see that if $I \in \mathcal{I}$, then n_v^I "copies" of activity v, $v \in V$, can be scheduled simultaneously (i.e., in the same time unit). Consider the following LP:

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

As expected, LP P_χ in Section 4 is a special case of this LP corresponding to $N_w = 1$ for $w \in W$. The following result establishes a new characterization of process capacity.

THEOREM 5. *Let $\mathcal{P}$ be an arbitrary deterministic, single-product process, as defined in Section 3. Then,*

$$\mu(\mathcal{P}) = \frac{1}{\hat{\chi}^*(\mathcal{P})}.$$

Proof of Theorem 5 We first present the LP formulation in Gurvich and Van Mieghem (2015) of the Static Planning Problem for Collaboration (SPPC), which computes the capacity of $\mathcal{P}$:

$$\begin{aligned} \rho^{net}(\lambda) = \min \quad & \rho \\ \text{s.t.} \quad & \sum_{I: v \in I} n_v^I x_I = \lambda \quad \text{for} \quad v \in V, \\ & \sum_{I \in \mathcal{I}} x_I \leqslant \rho, \\ & x_I \geqslant 0 \quad \text{for} \quad I \in \mathcal{I}, \end{aligned} \tag{12}$$

for $\lambda > 0$. The *network capacity*, λ^*, is the unique solution to $\rho^{\text{net}}(\lambda^*) = 1$. Gurvich and Van Mieghem (2015) show that

$$\mu(\mathcal{P}) = \lambda^*. \tag{13}$$

For $\lambda > 0$, we show that $\frac{\rho^{net}(\lambda)}{\lambda} = \rho^{net}(1)$:

$$\begin{aligned} \frac{\rho^{net}(\lambda)}{\lambda} = \min \quad & \frac{\rho}{\lambda} \\ \text{s.t.} \quad & \sum_{I: v \in I} \frac{n_v^I x_I}{\lambda} = 1 \text{ for } v \in V, \quad \sum_{I \in \mathcal{I}} \frac{x_I}{\lambda} \leqslant \frac{\rho}{\lambda}, \quad \frac{x_I}{\lambda} \geqslant 0 \text{ for } I \in \mathcal{I} \end{aligned}$$

$$
\begin{aligned}
&= \min \quad \rho' \\
&\quad \text{s.t.} \sum_{I: v \in I} n_v^I x'_I = 1 \text{ for } v \in V, \quad \sum_{I \in \mathcal{I}} x'_I \leqslant \rho', \quad x'_I \geqslant 0 \text{ for } I \in \mathcal{I} \\
&= \rho^{net}(1).
\end{aligned}
$$

Combining the fact that $\rho^{\text{net}}(\lambda^*) = 1$ and $\frac{\rho^{net}(\lambda)}{\lambda} = \rho^{net}(1)$, we have

$$\frac{1}{\lambda^*} = \frac{\rho^{net}(\lambda^*)}{\lambda^*} = \rho^{net}(1). \tag{14}$$

Next we show that

$$\rho^{net}(1) = \hat{\chi}^*(\mathcal{P}). \tag{15}$$

Comparing the linear program $(P_{\hat{\chi}})$ and (12), we have

$$
\begin{aligned}
\rho^{\text{net}}(1) &= \min \quad \rho \\
&\quad \text{s.t.} \sum_{I: v \in I} n_v^I x_I = 1 \text{ for } v \in V, \quad \sum_{I \in \mathcal{I}} x_I \leqslant \rho, \quad x_I \geqslant 0 \text{ for } I \in \mathcal{I} \\
&= \min \quad \sum_{I \in \mathcal{I}} x_I \\
&\quad \text{s.t.} \sum_{I: v \in I} n_v^I x_I = 1 \text{ for } v \in V, \quad x_I \geqslant 0 \text{ for } I \in \mathcal{I} \\
&= \hat{\chi}^*(\mathcal{P}).
\end{aligned}
$$

Combining (13), (14), and (15), we have

$$\mu(\mathcal{P}) = \lambda^* = \frac{1}{\rho^{net}(1)} = \frac{1}{\hat{\chi}^*(\mathcal{P})}.$$

■

Combining Theorems 4 and 5, we have

$$\mu(\mathcal{P}) = \frac{k}{\chi_f^*(\mathcal{G}^k)} = \frac{1}{\hat{\chi}^*(\mathcal{P})}, \text{ for } k \text{ satisfying (11)},$$

which gives us the precise connection between the optimal objective value of $P_{\hat{\chi}}$ and the fractional chromatic number of the hypergraph $\mathcal{G}^k$, namely, $\hat{\chi}^*(\mathcal{P}) = \frac{\chi_f^*(\mathcal{G}^k)}{k}$. The dual of the LP $P_{\hat{\chi}}$ is as follows, where $y_v, v \in V$, denote the dual variables:

$$
\begin{aligned}
\hat{\kappa}^*(\mathcal{P}) = \max \quad & \sum_{v \in V} y_v \\
\text{s.t.} \quad & \sum_{v \in V} n_v^I y_v \leqslant 1 \quad \text{for} \quad I \in \mathcal{I}, \\
& y_v \geqslant 0 \quad \text{for} \quad v \in V.
\end{aligned} \tag{$D_{\hat{\kappa}}$}
$$

Let $\boldsymbol{y} = (y_v, v \in V)$ denote a feasible solution of $D_{\hat{\kappa}}$ and $\hat{\kappa}(\boldsymbol{y}, \mathcal{P})$ denote its objective value. Let $\boldsymbol{y}^* = (y_v^*, v \in V)$ denote a basic optimal solution of $D_{\hat{\kappa}}$; its objective value is, as denoted above, $\hat{\kappa}^*(\mathcal{P})$. We refer to the solution $\boldsymbol{y}^*$ as the bottleneck structure of activities for $\mathcal{P}$. From LP duality, we have

$$\hat{\kappa}(\boldsymbol{y}, \mathcal{P}) \leqslant \hat{\kappa}^*(\mathcal{P}) = \hat{\chi}^*(\mathcal{P}).$$

Using Theorem 5, we have

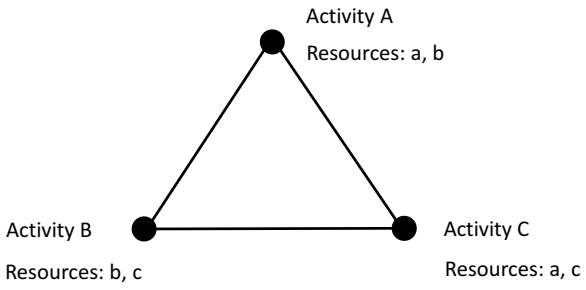
$$\mu(\mathcal{P}) = \frac{1}{\hat{\chi}^*(\mathcal{P})} = \frac{1}{\hat{\kappa}^*(\mathcal{P})} \leqslant \frac{1}{\hat{\kappa}(\boldsymbol{y}, \mathcal{P})}.$$

The *processing bound* (see Section 5) imposed by the bottleneck structure $\boldsymbol{y}^*$ is $B^* = \frac{1}{\sum_{v \in V} y_v^*}$. Let $Q = \{v \in V : y_v^* > 0\}$. Let $R_Q = \bigcup_{v \in Q} W_v$ be the collection of resources that are needed for at least one of the activities in Q. For an arbitrary set $\hat{R}_Q \subseteq R_Q$, consider the following "truncated" process: we restrict attention only to the activities $\hat{Q} \subseteq Q$ that require at least one of the resources in $\hat{R}_Q$. Further, we ignore all the resources not in $\hat{R}_Q$ needed for the activities in $\hat{Q}$. Thus, the new process includes only the activities in $\hat{Q}$ and, for each of these activities, ignores all required resources (as defined in the original process) except those in $\hat{R}_Q$. A minimal set $\hat{R}_Q \subseteq R_Q$ for which the processing bound of the bottleneck structure in the truncated process remains B^* is defined as a bottleneck set of resources for the original process $\mathcal{P}$.

We now illustrate the characterizations in Section C.1 and Section C.2.

Example 11: Consider the process, which we denote by $\mathcal{P}$, whose collaboration graph is shown in Figure 12. Assume that each of the three activities, namely, A, B, and C, takes one unit of time and that we have two units of resource a, one unit of resource b, and one unit of resource c.

Figure 12 The collaboration graph of the process in Example 11.



- Characterization using hypergraphs: Let $k = 2$, so that Condition (11) is satisfied. The collaboration hypergraph $\mathcal{G}^k$ of $\mathcal{P}$ is shown in Figure 13. There are two basic maximum fractional cliques in $\mathcal{G}^k$, or equivalently, two basic optimal solutions to problem D^k_κ:

$$\boldsymbol{y^*_1}: \quad y^*_{A^1} = y^*_{A^2} = y^*_{B^1} = y^*_{B^2} = 1, y^*_{C^1} = y^*_{C^2} = 0;$$
$$\boldsymbol{y^*_2}: \quad y^*_{B^1} = y^*_{B^2} = y^*_{C^1} = y^*_{C^2} = 1, y^*_{A^1} = y^*_{A^2} = 0.$$

Thus, the capacity of the process $\mathcal{P}$ is $\mu(\mathcal{P}) = \frac{k}{\kappa^*_f(\mathcal{G}^k)} = \frac{2}{4} = \frac{1}{2}$. Corresponding to $\boldsymbol{y^*_1}$ (resp., $\boldsymbol{y^*_2}$), we have $Q_1 = \{A^1, A^2, B^1, B^2\}$ (resp., $Q_2 = \{B^1, B^2, C^1, C^2\}$). So, the two bottleneck structures of activities for this process are the sub-hypergraphs S_{Q_1} (induced by Q_1) and S_{Q_2} (induced by Q_2), both of which are cliques in $\mathcal{G}^k$. The bottleneck sets of resources corresponding to S_{Q_1} and S_{Q_2} are, respectively, $\hat{R}_{Q_1} = \{b\}$ and $\hat{R}_{Q_2} = \{c\}$.

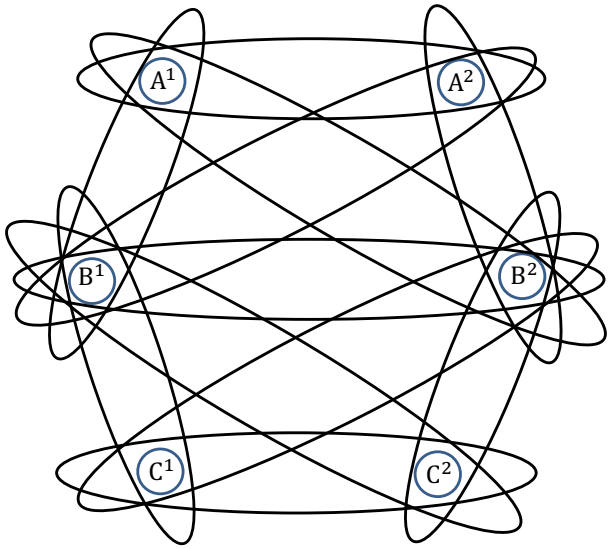


Figure 13 The collaboration hypergraph of the process in Example 11. Here, each hyperedge consists of two nodes; thus, this hypergraph is a simple graph.

- The alternate characterization using the LP $D_{\hat{\kappa}}$: The collection of the independent tuples (n^I_A, n^I_B, n^I_C) of $\mathcal{P}$ is:

$$\mathcal{I} = \big\{(1,0,0), (0,1,0), (0,0,1), (1,0,1)\big\}.$$

There are two basic optimal solutions to problem $D_{\hat{\kappa}}$:

$$\boldsymbol{y^*_1}: \ y_A = y_B = 1, y_C = 0;$$
$$\boldsymbol{y^*_2}: \ y_B = y_C = 1, y_A = 0.$$

Thus, the capacity of the process $\mathcal{P}$ is $\mu(\mathcal{P}) = \frac{1}{\hat{\kappa}^*(\mathcal{P})} = \frac{1}{2}$. Corresponding to the bottleneck structure $\boldsymbol{y_1^*}$ (resp., $\boldsymbol{y_2^*}$), we have $Q_1 = \{A, B\}$ (resp., $Q_2 = \{B, C\}$) and the bottleneck set of resources $\hat{R}_{Q_1} = \{b\}$ (resp., $\hat{R}_{Q_2} = \{c\}$). ■

Appendix D Extension: Flexible Resource Sets

In this section, we characterize a bottleneck structure of activities and the associated bottleneck set(s) of resources for a process with flexible resource sets, in the sense that any one of multiple sets of resources is capable of performing the same activity. Recall from our model of a process in Section 3 that activity $v \in V$ of the process needs the simultaneous availability of all the resources in the set $W_v \subseteq W$. Here, activity v can be performed by any one of *multiple* sets of resources. For expositional simplicity, we consider a process where $\tau(v) = 1, v \in V$, and $N_w = 1, w \in W$, i.e., each activity takes one unit of time and we have one copy of each resource. Example 6 in Section 7 illustrates the notion of flexibility.

A natural question arises: how do we characterize bottleneck structures of activities and bottleneck sets of resources for a process with flexible resource sets? Notice that adding flexibility enlarges the collection of independent sets (which correspond to sets of activities that can be performed simultaneously), relative to that in our original primal LP P_χ (Section 4). In Example 6, for instance, training worker-W1 to replace worker-W5 for Activity 5 results in the set {Activity 3, Activity 5} becoming an independent set, since Activity 3 and Activity 5 can now be performed simultaneously.

Consider an arbitrary process $\mathcal{P}^F$ with flexible resource sets and let $\mathcal{W}_v$ be the collection of resource sets that can perform activity $v \in V$. Note that in the no-flexibility setting we have discussed thus far, the collection $\mathcal{W}_v$ consists of a single set of resources – namely, the set W_v. Let $\boldsymbol{W} = (W_v \in \mathcal{W}_v, v \in V)$ be a $|V|$-tuple of resource sets, one for each activity $v \in V$, and let $\boldsymbol{\mathcal{W}}$ be the collection of all such tuples. For each $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$, let $G_{\boldsymbol{W}}$ be the corresponding collaboration graph. A set of activities $S \subseteq V$ is defined to be an independent set of $\mathcal{P}^F$ (in the sense that the activities in this set can be performed simultaneously) if there exists $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$ such that for any two distinct activities $v_i, v_j \in S$, we have $W_{v_i} \cap W_{v_j} = \varnothing$. In other words, S is an independent set of activities if there exists $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$ such that the nodes in S form an independent set in the collaboration graph $G_{\boldsymbol{W}}$. Let $\mathcal{I}^F$ denote the collection of all the independent sets of $\mathcal{P}^F$. That is,

$$\mathcal{I}^{\mathcal{F}} = \big\{S \subseteq V : \exists\ \boldsymbol{W} \in \boldsymbol{\mathcal{W}} \text{ s.t. } W_{v_i} \cap W_{v_j} = \varnothing \text{ for } v_i, v_j \in S; v_i \neq v_j\big\}.$$

Consider the following LP:

$$\begin{aligned} \chi_f^*(\mathcal{P}^F) = \min \quad & \sum_{I \in \mathcal{I}^{\mathcal{F}}} x_I \\ \text{s.t.} \quad & \sum_{I: v \in I} x_I \geqslant 1 \quad \text{for} \quad v \in V, \\ & x_I \geqslant 0 \quad \text{for} \quad I \in \mathcal{I}^F. \end{aligned} \tag{$\mathcal{P}_\chi^F$}$$

Note that LP P_χ in Section 4 is a special case of this LP corresponding to the no-flexibility setting. We have

THEOREM 6. *Let $\mu(\mathcal{P}^F)$ denote the capacity of process $\mathcal{P}^F$. Then,*

$$\mu(\mathcal{P}^F) = \frac{1}{\chi_f^*(\mathcal{P}^F)}.$$

The proof of Theorem 6 is similar to that of Theorem 1 (Bo et al. 2018) and Theorem 5 (Appendix C.2), and is, therefore, omitted for brevity. The dual of the LP $\mathcal{P}_\chi^F$ is as follows, where $y_v, v \in V$, denote the dual variables:

$$\begin{aligned} \kappa_f^*(\mathcal{P}^F) = \max \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & \sum_{v \in I} y_v \leqslant 1 \quad \text{for} \quad I \in \mathcal{I}^F, \\ & y_v \geqslant 0 \quad \text{for} \quad v \in V. \end{aligned} \tag{D_κ^F}$$

Let $\boldsymbol{y}^* = (y_v^*, v \in V)$ denote a basic optimal solution of D_κ^F. Note that $\boldsymbol{y}^*$ is a fractional clique, but not necessarily a maximum fractional clique in each collaboration graph $G_{\boldsymbol{W}}$, $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$. Let $Q = \{v \in V : y_v^* > 0\}$. In each collaboration graph $G_{\boldsymbol{W}}$, $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$, let $S_Q(\boldsymbol{W})$ be the node-induced subgraph of $G_{\boldsymbol{W}}$ corresponding to the activities in Q and let $R_Q(\boldsymbol{W}) = \bigcup_{v \in Q, W_v \in \boldsymbol{W}} W_v$ be the set of resources that are needed for at least one of the activities in Q for the choice $\boldsymbol{W}$ of resource sets. We refer to the tuple $\big(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\big)$ as a *bottleneck structure of activities* for $\mathcal{P}^F$. A minimal subset $\hat{R}_Q$ of $\bigcup_{\boldsymbol{W} \in \boldsymbol{\mathcal{W}}} R_Q(\boldsymbol{W})$ that induces a subgraph of $G_{\boldsymbol{W}}$ isomorphic to $S_Q(\boldsymbol{W})$ for each $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$ is defined to be a *bottleneck set of resources* for $\mathcal{P}^F$.

Remark 3: For a flexible process $\mathcal{P}^F$, while the processing bound (see Section 5) imposed by the *entire* bottleneck structure $\big(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\big)$ equals the process capacity $\mu(\mathcal{P}^F)$, the processing bound imposed by each *individual component* $S_Q(\boldsymbol{W})$ can be *strictly smaller* than $\mu(\mathcal{P}^F)$. For instance, in Example 6 (see Section 7), the processing bounds imposed by the two components of the bottleneck structure are $\frac{1}{7}$ sheds/min and $\frac{1}{6}$ sheds/min, while the capacity of the process is $\frac{2}{11}$ sheds/min. Thus, by exploiting flexible resource sets, the process can achieve a higher processing rate than the maximum rate allowed by each individual configuration of resource sets $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$.

The bottleneck set of resources generates each component of $\left(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\right)$ and, therefore, correctly characterizes the capacity $\mu(\mathcal{P}^F)$. Note that these features do not arise in processes without flexibility: for such processes, we have a single collaboration graph and, therefore, a bottleneck structure of activities has a single component. Finally, to increase capacity, if there is a unique bottleneck set of resources, then we must necessarily acquire additional copies of at least one resource from this set. If there are multiple bottleneck sets of resources, then one approach is to obtain a minimal set of resources that contains at least one resource in each bottleneck set of resources, and employ additional copies of *each* resource in that minimal resource set. ■

Example 6 (Continued): We identify the bottleneck structures for the process shown in Figure 7. Since the activity times in the process are general positive integers (i.e., $\tau(v) \geqslant 1$, for all $v \in V$), we consider the expanded process, which is obtained by splitting each activity $v \in V$ with $\tau(v) > 1$ into $\tau(v)$ activities that each take unit time. Let (i, j) denote the j^{th} copy of Activity i in the expanded process. Then one optimal solution to problem D_κ^F is:

$$y_{(1,1)} = y_{(1,2)} = y_{(1,3)} = 1;\ y_{(3,1)} = y_{(3,2)} = y_{(5,1)} = y_{(5,2)} = y_7 = \frac{1}{2};$$
$$y_2 = y_4 = y_6 = y_{(8,1)} = y_{(8,2)} = y_{(8,3)} = 0.$$

Thus, $\kappa_f^*(\mathcal{P}^F) = y_{(1,1)} + y_{(1,2)} + y_{(1,3)} + y_{(3,1)} + y_{(3,2)} + y_{(5,1)} + y_{(5,2)} + y_7 = \frac{11}{2}$ and the capacity of this process is $\mu(\mathcal{P}^F) = \frac{1}{\kappa_f^*(\mathcal{P}^F)} = \frac{2}{11}$ sheds/minute. Further, $Q = \{v \in V : y_v^* > 0\} = \{$Activity 1, Activity 3, Activity 5, Activity 7$\}$. The bottleneck structure of activities for the process consists of the two subgraphs (of the two collaboration graphs in Figure 7) formed by activities in Q, and the bottleneck set of resources is $\hat{R}_Q = \{$worker-W1, worker-W2, worker-W3, worker-W5$\}$. Observe that each of the two components of the bottleneck structure is neither a clique nor contains an odd-hole or an odd-antihole. ■

Recall that in the no-flexibility setting we discussed earlier, a bottleneck structure of activities for a process is either a clique or contains an odd-hole or an odd-antihole (Theorem 3). Example 6 illustrates that, for a flexible process, it is possible that none of the components of the bottleneck structure $S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}$, is a clique or contains an odd-hole or an odd-antihole. We now discuss two special cases in which this property continues to hold:

Consider any basic optimal solution $\boldsymbol{y}^* = (y_v^*, v \in V)$ of D_κ^F.

- If $\boldsymbol{y}^*$ is integral, then it is immediate that the node set $Q = \{v \in V : y_v^* > 0\} = \{v \in V : y_v^* = 1\}$ induces a clique in each collaboration graph $G_{\boldsymbol{W}}$, $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$. That is, each component of the bottleneck structure of activities $\left(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\right)$ is a clique. Now, if the system $\sum_{v \in I} y_v \leqslant 1, I \in \mathcal{I}^F; y_v \geqslant 0, v \in V$ (i.e., the set of constraints of D_κ^F), is *Totally Dual Integral (TDI)*, then the polytope

$P^F = \left\{ \boldsymbol{y} \in \mathbb{R}_+^{|V|} : \sum_{v \in I} y_v \leqslant 1, I \in \mathcal{I}^F \right\}$ is integral and, therefore, each basic feasible solution of D_κ^F is integral (Edmonds and Giles 1984, Schrijver, A. 1998). Consequently, all the components of *each* bottleneck structure of activities are cliques. Thus, we have

THEOREM 7. *Consider a flexible process $\mathcal{P}^F$. If the system $\sum_{v \in I} y_v \leqslant 1, I \in \mathcal{I}^F; y_v \geqslant 0, v \in V$ is TDI, then each component of any bottleneck structure of activities $\left(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\right)$ is a clique.*

- If all the components of the bottleneck structure $\left(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\right)$ are identical, then $(y_v^*, v \in Q)$ is a basic optimal solution to the fractional clique problem defined on the graph $S_Q(\boldsymbol{W})$ for each $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$. This can be seen as follows: for a given $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$, recall that $\boldsymbol{y}^*$ is a fractional clique in the corresponding collaboration graph $G_{\boldsymbol{W}}$. Thus, $(y_v^*, v \in Q)$ is a fractional clique in $S_Q(\boldsymbol{W})$. Further, if there exists a fractional clique $(z_v^*, v \in Q)$ in $S_Q(\boldsymbol{W})$ such that $\sum_{v \in Q} z_v^* > \sum_{v \in Q} y_v^*$, then by defining $z_v^* = 0$ for $v \in V \backslash Q$, we obtain a feasible solution $(z_v^*, v \in V)$ of problem D_κ^F with $\sum_{v \in V} z_v^* > \sum_{v \in V} y_v^*$, contradicting the fact that $\boldsymbol{y}^*$ is optimal for that problem. Thus, $(y_v^*, v \in Q)$ is a maximum fractional clique of $S_Q(\boldsymbol{W})$. It is also easy to see that this is a basic solution to the fractional clique problem on $S_Q(\boldsymbol{W})$. Therefore, for a given $\boldsymbol{W}$, if $S_Q(\boldsymbol{W})$ is perfect, then $(y_v^*, v \in Q)$ is integral. This, together with the fact that $y_v^* > 0$ for $v \in Q$ implies that $y_v^* = 1$ for each $v \in Q$ and, hence, $S_Q(\boldsymbol{W})$ is a clique in $G_{\boldsymbol{W}}$. Otherwise, $S_Q(\boldsymbol{W})$ is not perfect and, by the SPGT, contains an odd-hole or an odd-antihole of $G_{\boldsymbol{W}}$ as an induced subgraph. Therefore, we have

THEOREM 8. *Consider a flexible process $\mathcal{P}^F$. Let $\{G_{\boldsymbol{W}}, \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\}$ denote the collection of all the collaboration graphs of $\mathcal{P}^F$ and let $\left(S_Q(\boldsymbol{W}), \boldsymbol{W} \in \boldsymbol{\mathcal{W}}\right)$ denote any bottleneck structure of $\mathcal{P}^F$. If all the components of the bottleneck structure are identical, i.e., $S_Q(\boldsymbol{W}) = S_Q(\boldsymbol{W}')$, for any $\boldsymbol{W}, \boldsymbol{W}' \in \boldsymbol{\mathcal{W}}$, then for each $\boldsymbol{W} \in \boldsymbol{\mathcal{W}}$, $S_Q(\boldsymbol{W})$ is either (i) a clique in $G_{\boldsymbol{W}}$ or (ii) contains an odd-hole or an odd-antihole of $G_{\boldsymbol{W}}$ as an induced subgraph.*

Theorem 8 can be proved in a manner similar to Theorem 3; we therefore avoid providing a detailed proof for brevity.