Work Package Sizing and Project Performance

Chung-Lun Li[1]

Department of Logistics and Maritime Studies, Faculty of Business,
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
Email: chung-lun.li@polyu.edu.hk
ORCID: https://orcid.org/0000-0002-4225-0855

Nicholas G. Hall

Fisher College of Business, The Ohio State University,
Columbus, Ohio 43210
Email: hall.33@osu.edu
ORCID: https://orcid.org/0000-0003-4484-9252

September 28, 2016

Revised August 18, 2017

Revised March 5, 2018

Final Version: April 21, 2018

[1]Corresponding author

Abstract

We study how design decisions in project planning affect cost of execution. In organizing a project's tasks into work packages, tradeoffs arise. Defining small work packages increases project complexity and workload, and reduces economies of scale. Whereas, defining large work packages reduces concurrent processing and adversely affects cash flow. Our work is apparently the first to study this tradeoff. We consider the objective of minimizing total project cost, subject to a deadline on project makespan. For serial task networks, we describe an efficient algorithm that finds optimal work package sizes. For acyclic task networks, we develop a heuristic method and a lower bound for the unary *NP*-hard problem. A computational study shows that our heuristic routinely delivers near-optimal solutions that substantially improve on those found by benchmark procedures. Our results demonstrate the value of deliberately varying work package sizes within a project, in contrast to typical project management practice. Related issues including multiple serial paths in parallel, task incompatibility, and generalized precedence constrained work packages, are also discussed. Our work enables more precise planning of work packages to improve performance, documents the value of integrating the planning of work packages and schedules, and provides insights that guide resource allocation decisions.

Keywords: project management; work breakdown structure; work package sizing; project performance.

1 Introduction

One-fifth of the world's economic activity, with an annual value of $12 trillion, is organized as projects (Project Management Institute 2008). The growth of project management as a business process has been fueled in recent years by a widely expanding range of applications. These applications include IT implementations, new product and service development, change management, software development, and research and development projects. As a result, membership in the Project Management Institute has increased from about 50,000 in 1996 to over 500,000 today. Meanwhile, new planning methodologies for project management, including critical chain project management (Goldratt 1997) and agile methods (agilemanifesto.org 2001), have emerged. Overviews of project management planning methodology are provided by Klastorin (2011), Wysocki (2014), and Kerzner (2017). Overviews of important research questions in project management are provided by Hall (2012, 2015, 2016).

Despite these significant developments in project management methodology, the foundation of project planning remains the relationship between the project as a whole and its discrete components, which we study here. The work breakdown structure (WBS) planning tool was developed jointly by the U.S. Department of Defense, NASA, and the aerospace industry (DOD and NASA 1962). It is now used for many widely varied project management applications, and is viewed as a fundamental early step in project planning (Golany and Shtub 2001, Project Management Institute 2013). A WBS is an incremental decomposition of a project using a hierarchical tree structure. The lowest level of the decomposition for planning and scheduling purposes is called a *work package.* Work packages are composed of one or more *elemental tasks.* The responsibility for completion of a work package is assigned to a single person or organizational unit. The contributions of WBS planning include the following (Monnapappa 2013): (i) it provides project stakeholders with a clear understanding of their roles and responsibilities; (ii) it allows for concurrent work on different components of the project; (iii) it supports estimation of cost and time performance, including the use of earned value management metrics; and (iv) it supports project risk management. Demeulemeester and Herroelen (2002) describe the administrative process by which a company typically develops a WBS for its projects.

Our research is motivated by the work of Globerson (1994), who comments that, "The correct use of a WBS contributes significantly to the probability of successful project completion." He illustrates how five distinctly different WBS structures can reasonably be generated for a new restaurant development project, each based on a different sequence of breakdown criteria, level by level, within the WBS. He discusses the relationship between the project WBS and the "organizational structures and management styles" that support it. However, our work assumes that these structures and styles are determined by the culture of the organization. Hence, we take the sequence of breakdown criteria within the WBS as given. Globerson (1994) also comments about work package formation, "Too many levels in the WBS loads the organization with too much information, and complicates the management process of that project. Too few levels generates communication difficulties and poor coordination among organizational units." Indeed, the main decisions in WBS formation are the sizes of work packages and which tasks should be integrated together into a single work package. Focusing on those decisions, we study the use of work package formation to optimize project performance.

Various academic studies consider different issues in WBS design. Deckro et al. (1992) consider a project scheduling problem with a time-cost tradeoff, where the tasks are already partitioned into work packages, each with a given due date and budget. They study the impact of project and work package due dates on cost and budget performance. Luby et al. (1995) discuss the application of a customized component-based WBS to projects in naval shipyards. This design enables continuous accountability for each component, while allowing flexibility for work plans. Jung and Woo (2004) propose a WBS that allows different management approaches to be applied to different work packages, in order to reduce the amount of information required in project planning. Danilovic (2006) notes the benefits of having suppliers involved in new product development projects, but demonstrates how functionally designed WBS and work package structures create barriers that prevent this. Other researchers study WBS design from the perspective of systems development. For example, Golpayegani and Emamizadeh (2007) apply neural networks to WBS planning, and consider project control WBSs, functional WBSs, and relational WBSs. Bai et al. (2009) similarly use domain tree structures, domain WBSs, and relational WBSs. Li and Ren (2013) focus on the relationships between objects in cloud manufacturing, the availability of cloud services resources, and service access issues. Still other researchers use matrix techniques to analyze interdependencies between project tasks that may inform WBS design. Yu et al. (2009) use genetic algorithms and dependency matrix clustering. Lee et al. (2010) use a design structure matrix approach. Ren et al. (2013) use a fuzzy dependency structure matrix to model coupling issues such as two-way dependence between a pair of tasks. Jia et al. (2014) develop mapping rules based on a product structure tree.

Regarding work package formation, we first review information from project management practice. Here, various rules of thumb are suggested. In 1972, the U.S. Department of Defense recommended a work package size of six months of development time or $100,000 of cost, as reported by Brown (1978). Some project managers recommend smaller work package size designs such as using the 4/40 rule, which requires work packages to be at least 4 hours and at most 40 hours of work; see Kennemer (2002). Gardner (2006) provides some practical guidelines for work package sizing in the construction industry. However, there is apparently no research on how to customize work package sizes for a particular project. The challenges of work package sizing are illustrated by the following comments from practicing project managers:

– "Whichever maximum increment of time you choose to breakout your tasks will directly impact your ability to track progress and give accurate project status reports to sr mgmt." (Kastner 2001)
– "The first problem with managing tasks below a 40 hour or 20 hour limit (my personal floor) is the [sheer] task of managing the value and dealing with the ramifications of a task that is not completed in time. The second is that planning at that level of detail more than a week or two in advance of the actual work - and hitting the target - is virtually impossible." (McVey 2001)
– "I have no problem [in] breaking measurable tasks down to 4 or 8 hrs if the deliverable is only estimated at 40 hrs overall. In general, I do think that trying to determine the task progress over 40 hrs can introduce risk unless you are addressing it [in] some other way." (Roche 2010)
– "There are trade-offs. Too high a level, the administrative burden to manage it is less in terms of cost. However, you lose insight into what is going on in the work and to understand where variances are accruing and why. This lack of insight increases your risk of successful ability to mitigate those variances." (Espina 2011)
– "One thing to note, the WBS should be a tool for your teams to decompose their work. Let them get as granular as they want if it helps them figure out what needs to be done." (Wilmot 2011)

The academic literature also addresses the issue of work package formation. Devi and Reddy (2012) suggest that a work package should comprise between 8 and 80 hours of work. Raz and Globerson (1998) identify the several factors and related costs discussed below as being important for project managers to consider in determining the sizes of their work packages. Based on their discussion and basic concepts of risk pooling, we discuss below whether each of these factors is in favor of larger or smaller work packages. We use “M” to denote an indication for more but smaller work packages, and “F” to denote an indication for fewer but larger work packages. However, some of these factors need to be implemented *directly through constraints*, which we denote by “D”.

- *Workload:* Decomposing the project into more but smaller work packages increases the amount of effort spent on planning, measuring, and reporting progress, resulting in an increase in administrative workload. It also incurs higher cost of supervision of work package operators. (F)
- *Cost and Schedule Estimation:* Estimates of cost and duration of the work package are relatively more accurate for work packages with larger content, due to pooling of variance. (F)
- *Monitoring and Control:* Greater precision in measuring performance, and tighter control, can be applied to work packages with smaller content. (M)
- *Cash Flow:* Smaller work packages lead to earlier completion of tasks and improve the project’s cash flow. (M)
- *Network Construction and Concurrent Processing:* Formation of elemental tasks into work packages may reduce opportunities for concurrent processing. (M)
- *Responsibility Assignment:* It is necessary for a specific person or organizational unit to take responsibility for the entire work package. (D)
- *Internal Cohesion:* Some groups of elemental tasks require close coordination without the distraction of managing other tasks. (D)
- *Risk Management:* A high-risk activity should be assigned to a separate work package, allowing more focused risk management attention. (D)

Another factor that the literature identifies as important in making work package sizing decisions is the following:

- *Economies of Scale:* Bairstow (2009) comments on the U.K. aerospace industry, “As the industry becomes more globalised industry there are fewer, larger work packages. Our small companies cannot manage these on their own so we are helping them team up to bid together. They will collaborate to get more resources and economies of scale.” (F)

However, none of the above works considers the issue of how to form work packages in a given project, so as to optimize total project cost. Moreover, practitioners typically assume that the same work package sizing rule should be applied across all work packages. Yet relative costs and network topology may vary within the project, and both of these affect work package sizing choices. Consequently, we question the validity of this assumption, and study its consequences for project cost. We observe that work package formation decisions affect the availability of concurrent processing, and therefore the project makespan. Further, since many practical projects have strict deadlines but flexible cost, we develop and analyze a work package sizing model to minimize cost, subject to a deadline constraint.

Research on work package sizing should not be confused with that on lot streaming. Within the classical machine scheduling literature, lot streaming (Potts and Baker 1989, Trietsch and Baker 1993) is the process of splitting a job into sublots, in order to allows its operations to be overlapped. However, the finite machine capacity constraint in such problems does not allow the concurrent processing that is available in project management. Furthermore, the focus of the lot streaming literature is on classical time-based scheduling objectives, without considering various costs that are significant in project management.

Some planning processes inhibit work package design. In a top-down planning process, for example, work packages are defined based on organizational convenience, but without detailed information that could enable precise planning. Such information includes task durations, precedence relations among tasks, and costs. In such a planning process, this detailed information only becomes available at the task level, after work packages have already been formed (Demeulemeester and Herroelen 2002, p. 9). However, the availability of this information creates the potential for making more precise decisions about work package design to improve project performance. Our work investigates this potential. To model these decisions, we assume that the project has been planned into a given set of tasks that are elemental. That is, no further breakdown of those tasks is possible, for reasons that may be technical, logistical, financial, administrative, or cultural. By considering tasks at the elemental level, we recognize the maximum potential for concurrent processing of work within the project. However, various cost considerations may motivate giving up some concurrency, and thereby allowing an increase in the project makespan, as elemental tasks are formed into work packages.

Our work is apparently the first to model the work package sizing tradeoff mathematically and solve it for optimal project performance. We model all the nine work package sizing factors discussed above. The total project cost we consider has five components: fixed work package cost, cost of inaccurate estimation, monitoring and control cost, economies of scale, and discounted cash flow cost. These costs fall within standard cost accounting categories, which makes them easy to estimate; hence, we model them using deterministic cost parameters. Our objective is to minimize total cost, subject to meeting a deadline on project makespan. This model represents many projects, including a large class of "event" projects, for example preparation for hosting the Olympic Games, with a fixed deadline. For networks of elemental tasks with serial precedence constraints, we develop a computationally efficient solution procedure. For acyclic task networks, we show that the cost minimization problem is unary *NP*-hard (Garey and Johnson 1979); hence, we develop a heuristic method and a lower bound. A computational study shows that our heuristic routinely delivers near-optimal solutions. Sensitivity analysis results show how optimal work package solutions vary with problem parameters. We document the significant project cost savings delivered by our model and algorithm, relative to a benchmark approach that is typically used in practice. We also discuss variations of our model to consider networks with multiple serial paths in parallel, task incompatibility, and generalized precedence constrained work packages. Overall, our work enables more precise project planning, documents the value of integrating the planning of work packages and schedules, and provides insights that guide resource allocation decisions.

The remainder of this paper is organized as follows. In Section 2, we provide our definitions and discuss how we model the nine factors that affect work package sizing decisions. In Section 3, we first describe an efficient algorithm for the special case with tasks in series, and we then describe a heuristic and a lower bound for the unary *NP*-hard problem with an acyclic task network. In Section 4, we test our solution procedures computationally and develop insights from our results. In Section 5, we discuss several extensions and variants of our model. Section 6 contains a conclusion and suggestions for future work. The proofs of all lemmas and theorems, as well as the formulations of several dynamic programs, are provided in online appendices.

2 Model

We consider a project consisting of a set $N = \{1, 2, \ldots, n\}$ of elemental tasks (hereafter referred to as "tasks") and precedence constraints between the tasks. Each task $a \in N$ has a given *duration* $t_a \geq 0$ and a given *work content* $x_a \geq 0$. Work content x_a, which can be measured in worker-hours or in monetary units, represents the size of the task. The project is required to complete by a given deadline d. Using the standard activity-on-node (AoN) representation, we define a node set N and an arc set $A \subseteq \{a \to a' \mid 1 \leq a < a' \leq n\}$, where $a \to a' \in A$ if task a' is an immediate successor of task a. We refer to a task with no predecessor as a *beginning task* and a task with no successor as an *ending task*. We assume that two dummy tasks with zero work contents and zero durations have been added at the beginning and the end of the task network, respectively. Hence, there is exactly one beginning task, exactly one ending task, and $n-2$ other tasks. Following the academic literature, we also assume that all input parameters are integers, i.e., $d \in \mathbb{Z}^+$ and $x_a, t_a \in \mathbb{Z}^+$ for each non-dummy task a, where $\mathbb{Z}^+$ is the set of all positive integers.

We need to group the tasks into work packages to minimize the total project cost. Consider a work package $W = \{a_1, a_2, \ldots, a_k\}$ formed by grouping tasks $a_1, a_2, \ldots, a_k \in N$. Once the processing of a task in W begins, the other tasks in W are processed as early as possible, subject to the precedence constraints between tasks. Thus, tasks within work package W may be processed in parallel if there is no precedence constraint between them. The work content of this work package is $x_W = \sum_{a \in W} x_a$, which is the sum of the work content of the tasks in the work package. The duration of this work package, denoted by t_W, is the length of the critical path within W, which equals the difference between the finish time of the last task in W and the start time of the first task in W. Observe that $t_W \leq \sum_{a \in W} t_a$. We let C_W denote the completion time of work package W in a given schedule.

Project management practice and the related academic literature discuss two alternative perspectives about work packages. Under the first perspective, a work package is the lowest level of the WBS for scheduling purposes. Then project scheduling, for example with the use of the critical path method (Kelley and Walker 1959), is performed using work packages as activities; see Popescu and Charaenngam (1995, p. 200), who explain, "The detailed schedule development is based on ... work packages, to schedule activities. To control the project in detail, one work package will represent an activity in the network." This perspective is also adopted by, for example, Baxendale (1991), Eldin (1991), Jaskowski and Sabotka (2006), Harpum (2007), Powell (2010), and Nicholas and Steyn (2017). Using this perspective, if task a in work package W is a predecessor of task a' in work package W', then all tasks in W precede all tasks in W'. We refer to this first perspective about work packages as *strict precedence constrained work packages.* Under the second perspective, a work package comprises one or more activities, and strict precedence relations exist between activities instead of work packages. This perspective is adopted by, for example, Deckro et al. (1992), Demeulemeester and Herroelen (2002), and Mubarak (2015). Under this perspective, the processing of work packages W and W' may be partially overlapped even though one of the tasks in W is a predecessor of one of the tasks in W'. As a result, the precedence relations between work packages are generalized precedence relations. We refer to this second perspective about work packages as *generalized precedence constrained work packages.*

Our work mainly considers the first perspective in the previous paragraph, strict precedence constrained work packages. Our reasons for this choice are as follows. First, a major motivation for companies to use work packages is simplification of the project, which is achieved by planning and scheduling at the work package level and suppressing task information for those purposes. Second, our review of the business literature suggests that few practicing project managers make use of generalized precedence relations. Since we intend that our work should be used in practice, our focus is on improving planning methods with which most project managers are already comfortable, rather than imposing a new one on them. Nonetheless, as a guide to work package formation for project managers who would accept a more complicated planning model, an analysis of the generalized precedence constrained work packages model is provided in Section 5.3.

Two issues that affect work package sizing are feasibility and concurrency. Regarding feasibility, strict precedence constrained work packages can only be formed if the resulting network is acyclic. For example, consider the network with five tasks shown in Figure 1(a). Suppose we group tasks

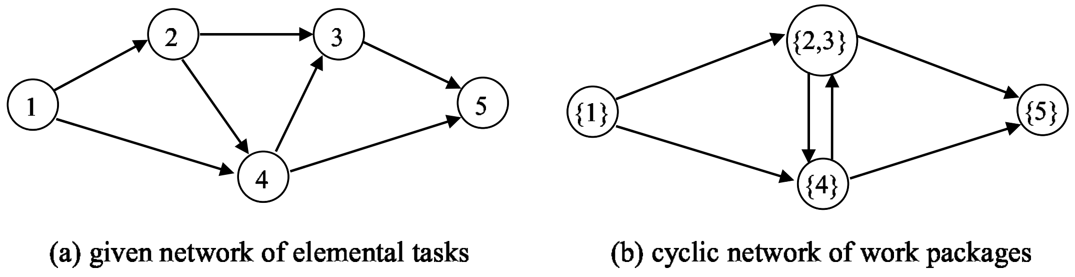

1
2
3
4
5
(a) given network of elemental tasks
{1}
{2,3}
{4}
{5}
(b) cyclic network of work packages

Figure 1: Feasibility of work package formation.

2 and 3 into a work package, and let tasks 1, 4, and 5 each form a separate work package. The resulting network, shown in Figure 1(b), contains a cycle. Hence, this grouping of tasks into work packages is infeasible.

Regarding concurrency, when a work package $\{a_1, a_2, \ldots, a_k\} \subseteq N$ is formed, the completion time of all tasks in the work package is equal to the completion time of the whole work package. Hence, the formation of this work package may reduce opportunities for concurrent processing and delay the completion of some tasks and the overall project. For example, consider the network with seven tasks shown in Figure 2(a). If we let each task form a separate work package, then the makespan of this project is 9. However, suppose we group tasks 2, 3, 4, and 6 into a work package, and let tasks 1, 5, and 7 each form a separate work package. The resulting network, which has

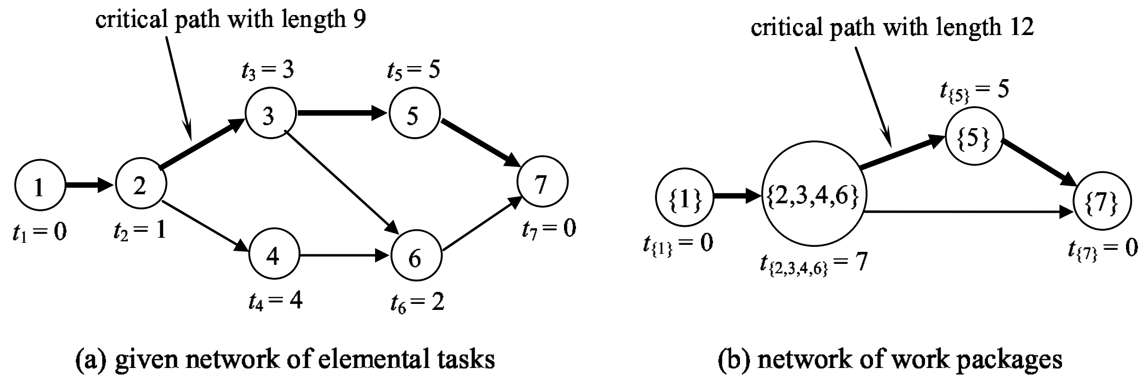


Figure 2: Network construction and concurrent processing.

strict precedence constraints between work packages, is shown in Figure 2(b). Now, tasks 5 and 6 cannot be processed concurrently and, as a result, the start time of task 5 is delayed from time 4 to time 7, and the makespan of the project is increased from 9 to 12.

We now model each of the work package sizing factors identified in Section 1:

- *Workload:* Each work package has a fixed cost ω, which represents the cost of administration and maintenance, and is independent of the contents of the work package. Raz and Globerson (1998) motivate this cost component by describing potentially disproportionate time spent measuring and reporting progress.
- *Cost and Schedule Estimation:* Inaccuracy in time estimation of a work package can be hedged by allocating extra resources such as manpower, while inaccuracy in cost estimation can be hedged by allocating a larger budget in advance. The amount of hedging required can be estimated from experience with similar projects, or from industry standards. The cost of extra resources and budget for hedging is dependent on the work content of the work package. We model this cost as a function $f(\cdot)$ of the work content. This function is concave nondecreasing, since this factor encourages having fewer but larger work packages.
- *Monitoring and Control:* We model the cost of monitoring and controlling the progress of the work package as a function $g(\cdot)$ of its work content. This function is convex nondecreasing, since this factor encourages having more but smaller work packages.
- *Cash Flow:* Consider tasks $a_1, a_2, \ldots, a_k$ that are formed into a single work package W. Milestone payments (Dayanand and Padman 2001) that are receivable in respect of tasks $a_1, a_2, \ldots, a_k$ have an opportunity cost that extends from the start of the project through completion of the work package, that is, over the period $[0, C_W]$. We discount this opportunity cost at a rate $\alpha \geq 0$

in continuous time, following the cash flow discounting method of Yang et al. (1992). At a cost of ξ per unit of work content the total discounted cash flow cost of the work package is $\xi x_W(1 - e^{-\alpha C_W})$.

- *Network Construction and Concurrent Processing:* If a task $a_i \in N \setminus \{a_1, a_2, \ldots, a_k\}$ is a successor of one of $a_1, a_2, \ldots, a_k$, then the formation of the work package $\{a_1, a_2, \ldots, a_k\}$ makes a_i a successor of all the tasks $a_1, a_2, \ldots, a_k$. As in Figure 2, this may reduce opportunities for concurrent processing and thus increase the project makespan. The formation of a work package is allowed only if the deadline constraint is not violated.
- *Responsibility Assignment:* Where the responsibility for a group of tasks $R = \{a_1, a_2, \ldots, a_k\}$ needs to be assigned to a specific person or organizational unit, we require R to be a single work package without other tasks.
- *Internal Cohesion:* Where a group of tasks $R = \{a_1, a_2, \ldots, a_k\}$ requires internal cohesion, we require R to be a single work package without other tasks.
- *Risk Management:* Where a specific group of high-risk tasks $R = \{a_1, a_2, \ldots, a_k\}$ needs to be kept separate to allow focused risk management, we require R to be a single work package without other tasks.
- *Economies of Scale:* We model economies of scale from repetition and similarity of tasks within a work package as a function $h(\cdot)$ of its work content. This function is concave nondecreasing, since this factor encourages having fewer but larger work packages.

We assume that $f(0) = g(0) = h(0) = 0$, and that the functions $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ can be evaluated in constant time. Let

$$F(x) = f(x) + g(x) + h(x),$$

for $x \geq 0$. Clearly, $F(\cdot)$ is a nondecreasing function. Then, the cost associated with a work package W is

$$\omega + F(x_W) + \xi x_W(1 - e^{-\alpha C_W}).$$

Let $R_1, R_2, \ldots, R_m$ denote all the subsets of tasks that are required to form their own work packages due to responsibility assignment, internal cohesion, or risk management, where $R_k \cap R_l = \emptyset$ for any $1 \leq k < l \leq m$. Let $\bar{R} = N \setminus (R_1 \cup R_2 \cup \cdots \cup R_m)$ and $\bar{n} = |\bar{R}| > 0$. We refer to the tasks in $R_1 \cup R_2 \cup \cdots \cup R_m$ as *inactive tasks* and the tasks in $\bar{R}$ as *active tasks.* The two dummy tasks in the problem are inactive tasks, and each of them is required to form a work package by itself. We let $\bar{X} = \sum_{i \in \bar{R}} x_i$ denote the sum of the work contents of all the active tasks.

Denote the number of work packages of active tasks as p, which is a decision variable. Thus, the number of non-dummy tasks is $p + m - 2$. The objective of our problem is to partition $\bar{R}$ into work packages $W_1, W_2, \ldots, W_p$ to minimize the total cost:

$$TC = \omega(p+m-2) + \sum_{k=1}^{p} F(x_{W_k}) + \sum_{k=1}^{m} F(x_{R_k}) + \xi \sum_{k=1}^{p} x_{W_k}(1 - e^{-\alpha C_{W_k}}) + \xi \sum_{k=1}^{m} x_{R_k}(1 - e^{-\alpha C_{R_k}}), \quad (1)$$

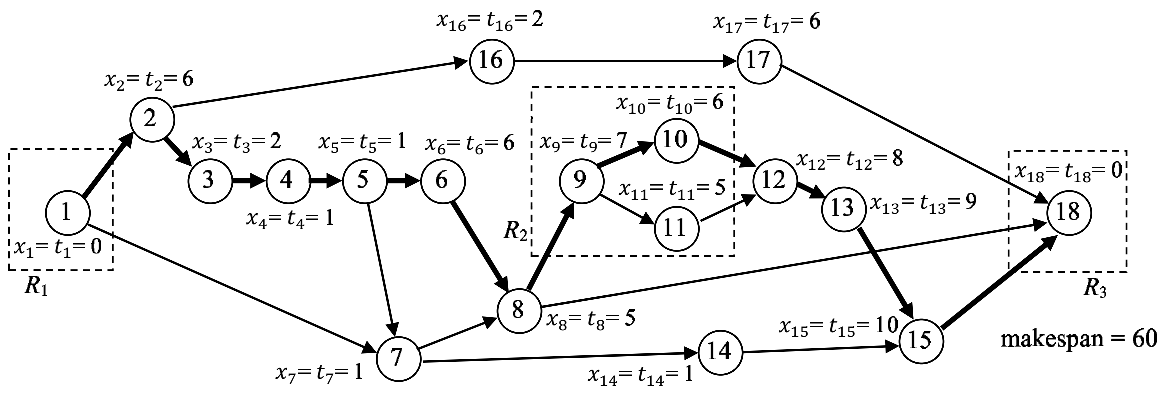
subject to the constraint that the precedence network of the work packages $W_1, W_2, \ldots, W_p$, $R_1, R_2, \ldots, R_m$ is acyclic, and that the project makespan is no greater than d.

Example 1

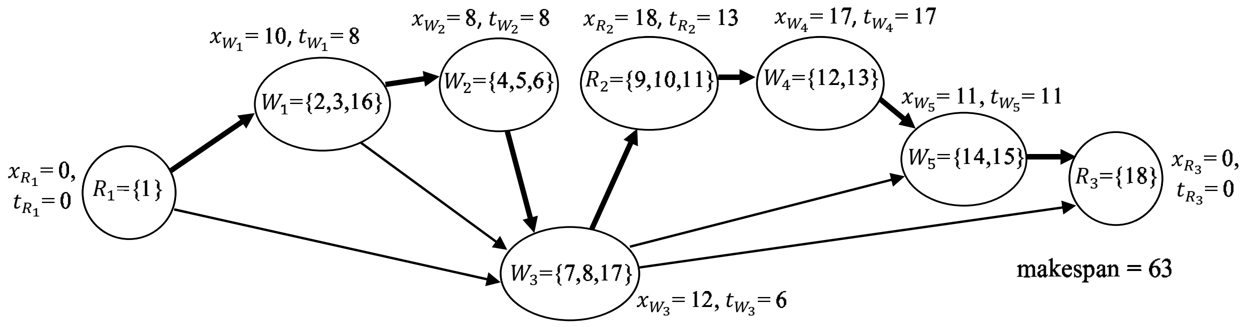
Consider an example with $n = 18$, $N = \{1, 2, \ldots, 18\}$, $m = 3$, $R_1 = \{1\}$, $R_2 = \{9, 10, 11\}$, $R_3 = \{18\}$, $\omega = 50$, $\xi = 100$, $\alpha = 0.001$, $f(x) = h(x) = 3x^{0.8}$, $g(x) = x^{1.2}$, $d = 64$, $(x_1, x_2, \ldots, x_{18}) = (t_1, t_2, \ldots, t_{18}) = (0, 6, 2, 1, 1, 6, 1, 5, 7, 6, 5, 8, 9, 1, 10, 2, 6, 0)$, and the task network shown in Figure 3(a). Tasks 1 and 18 are dummy beginning and ending tasks, respectively, of the project. They are represented by inactive tasks with zero work contents and durations. A feasible solution of this instance is shown in Figure 3(b). In this solution, five work packages of active tasks are formed: $W_1 = \{2, 3, 16\}$, $W_2 = \{4, 5, 6\}$, $W_3 = \{7, 8, 17\}$, $W_4 = \{12, 13\}$, and $W_5 = \{14, 15\}$. Clearly, the precedence network of the work packages is acyclic. This solution has a project makespan of $63 \leq 64 = d$. Hence, this solution is feasible. In this solution, $p = 5$, $(x_{W_1}, x_{W_2}, x_{W_3}, x_{W_4}, x_{W_5}, x_{R_2}) = (10, 8, 12, 17, 11, 18)$, $(t_{W_1}, t_{W_2}, t_{W_3}, t_{W_4}, t_{W_5}, t_{R_2}) = (8, 8, 6, 17, 11, 13)$, and $(C_{W_1}, C_{W_2}, C_{W_3}, C_{W_4}, C_{W_5}, C_{R_2}) = (8, 16, 22, 52, 63, 35)$. Thus, $F(x_{W_1}) = F(10) = 53.7$, $F(x_{W_2}) = F(8) = 43.8$, $F(x_{W_3}) = F(12) = 63.5$, $F(x_{W_4}) = F(17) = 87.8$, $F(x_{W_5}) = F(11) = 58.6$, $F(x_{R_2}) = F(18) = 92.7$, $1 - e^{-\alpha C_{W_1}} = 0.0080$, $1 - e^{-\alpha C_{W_2}} = 0.0159$, $1 - e^{-\alpha C_{W_3}} = 0.0218$, $1 - e^{-\alpha C_{W_4}} = 0.0507$, $1 - e^{-\alpha C_{W_5}} = 0.0611$, and $1 - e^{-\alpha C_{R_2}} = 0.0344$. The total cost of this solution is

$$\begin{aligned} TC &= \omega(p + m - 2) + \sum_{k=1}^{5} F(x_{W_k}) + F(x_{R_2}) + \xi \sum_{k=1}^{5} x_{W_k}(1 - e^{-\alpha C_{W_k}}) + \xi x_{R_2}(1 - e^{-\alpha C_{R_2}}) \\ &= (50)(6) + (53.7 + 43.8 + 63.5 + 87.8 + 58.6) + (92.7) + (100)(10 \times 0.0080 \\ &\quad + 8 \times 0.0159 + 12 \times 0.0218 + 17 \times 0.0507 + 11 \times 0.0611) + (100)(18 \times 0.0344) = 962.2. \end{aligned}$$

We describe tasks $a_1, a_2, \ldots, a_k$ as *active serial tasks* if (i) $a_1, a_2, \ldots, a_k \in \bar{R}$; and (ii) a_i is the only immediate predecessor of a_{i+1}, and a_{i+1} is the only immediate successor of a_i, for $i =$



(a) given task network with groups of inactive tasks



(b) network of work packages

Figure 3: Task network and network of work packages in Example 1.

$1, 2, \dots, k-1$. For example, tasks $3, 4, 5$ in Example 1 are active serial tasks (see Figure 3(a)). If, besides the dummy beginning and ending tasks, the given task network contains only active serial tasks, then the problem is denoted by $\mathbf{P}_S$. If the given task network is a directed acyclic network, then the problem is denoted by $\mathbf{P}_A$.

3 Solution Methods

Section 3.1 provides an efficient algorithm for problem $\mathbf{P}_S$. Section 3.2 discusses issues of feasibility and computational complexity, and presents a heuristic solution procedure and a lower bounding procedure, for problem $\mathbf{P}_A$.

3.1 Networks with active serial tasks

In this section, we focus on the formation of work packages for a group of active serial tasks $a_1, a_2, \dots, a_k$ with the precedence relations $a_1 \to a_2 \to \cdots \to a_k$. Project management applications which have precedence subnetworks with similar features include job shop scheduling (Beck et al. 2003) and assembly of mechanical products (Wang et al. 2005).

Any work package constructed from tasks $a_1, a_2, \ldots, a_k$ must be of the form $\{a_{i+1}, a_{i+2}, \ldots, a_j\}$, where $0 \leq i < j \leq k$. Thus, the work package sizing decision for these tasks is to partition $\{a_1, a_2, \ldots, a_k\}$ into consecutive groups of tasks. Since the active tasks form a serial network, the grouping of these active serial tasks does not affect the completion times of the tasks, and it does not affect the feasibility or makespan of the project. Hence, the situations shown in Figures 1 and 2 do not arise. The work content of work package $\{a_{i+1}, a_{i+2}, \ldots, a_j\}$ is $\sum_{l=i+1}^{j} x_{a_l}$. Thus, the discounted cash flow cost of this work package is $\xi \sum_{l=i+1}^{j} x_{a_l}\big(1 - e^{-\alpha C_{a_j}}\big)$, where C_{a_j} is the completion time of task a_j, i.e., the completion time of the work package. For $0 \leq i < j \leq k$, let

$$c(i,j) = \omega + F\bigg(\sum_{l=i+1}^{j} x_{a_l}\bigg) + \xi \sum_{l=i+1}^{j} x_{a_l}\big(1 - e^{-\alpha C_{a_j}}\big), \tag{2}$$

which is the total cost of work package $\{a_{i+1}, a_{i+2}, \ldots, a_j\}$ if this work package is formed. In Appendix B, we describe a dynamic programming algorithm that groups the active tasks $a_1, a_2, \ldots, a_k$ into work packages with minimum total cost. This algorithm is the same as the shortest path algorithm for acyclic networks, see Dreyfus and Law (1977, pp. 51–53), and shows that problem $\mathbf{P}_S$ with active tasks $a_1, a_2, \ldots, a_k$ can be solved optimally in $O(k^2)$ time. A generalization of problem $\mathbf{P}_S$ with multiple serial paths in parallel is discussed in Section 5.1.

3.2 Acyclic task networks

In Section 3.2.1, we first discuss the feasibility and complexity of problem $\mathbf{P}_A$, and then present a heuristic procedure for solving it. In Section 3.2.2, we present a procedure for generating a lower bound on the optimal solution value.

3.2.1 Feasibility, complexity, and a heuristic

Let Δ denote the project makespan when each active task forms its own work package. The following result provides a condition for the feasibility of the problem.

Lemma 1 *Problem $\mathbf{P}_A$ is feasible if and only if $\Delta \leq d$.*

From Lemma 1, the feasibility of a given instance of problem $\mathbf{P}_A$ is easily determined by giving each active task its own work package. In our analysis presented below, we assume that the given instance under consideration is feasible. The next result establishes the *NP*-hardness of the work package sizing problem $\mathbf{P}_A$.

Theorem 1 *The recognition version of problem $\mathbf{P}_A$ is unary* NP*-complete, even if $x_a = t_a$ for all $a \in N$.*

Theorem 1 implies that it is unlikely that a polynomial-time optimal algorithm exists, even for a special case of problem $\mathbf{P}_A$. Hence, we propose the following Heuristic MergeTasks to solve this problem. The heuristic first forms work packages in subnetworks of serial tasks. It then applies a greedy procedure to combine elemental tasks into work packages, where doing so does not create any cycle in the precedence network, and does not violate the deadline constraint.

Heuristic MergeTasks

Step 1. Identify all subsets of active serial tasks that are maximal by inclusion. For each such subset $\{a_1, a_2, \dots, a_k\}$, apply the algorithm described in Section 3.1 to form work packages. For the other (non-serial) tasks, initialize each of them as a separate work package.

Step 2. Consider all possible pairs of work packages such that

(i) both work packages contain only active tasks;

(ii) merging those two work packages does not create any cycle in the network;

(iii) merging those two work packages decreases the total cost; and

(iv) merging those two work packages does not increase the project makespan.

If such a pair of work packages exists, then select a pair of work packages for which a merger provides the largest decrease in total cost (with ties broken arbitrarily), merge them, and return to Step 2.

Step 3. Consider all possible pairs of work packages such that

(i) both work packages contain only active tasks;

(ii) merging those two work packages does not create any cycle in the network;

(iii) merging those two work packages decreases the total cost; and

(iv) merging those two work packages does not increase the project makespan to a value greater than d.

If such a pair of work packages exists, then select a pair of work packages for which a merger provides the largest ratio of decrease in total cost to increase in project makespan (with ties broken arbitrarily), merge them, and return to Step 2; otherwise, stop.

Step 1 of Heuristic MergeTasks forms work packages for the serial tasks only. Since doing so does not create any cycle in the network and does not increase the project makespan, there is no need to perform a cyclicity check or deadline violation check in this step. Step 2 performs pairwise merging of work packages, where two work packages are merged only if (i) the resulting network is acyclic, (ii) there is a decrease in total cost, and (iii) there is no increase in project makespan. Step 3 similarly performs pairwise merging of work packages to reduce cost, but allows an increase in project makespan, provided that the deadline constraint is not violated. Steps 2 and 3 are executed repeatedly until no further cost reduction is possible.

Theorem 2 *Heuristic MergeTasks delivers a feasible solution for problem* $\mathbf{P}_A$ *in* $O(\bar{n}^3|A|)$ *time.*

3.2.2 Lower bound

Our overall strategy for finding a strong lower bound on the optimal solution value is to condition on the number of work packages used. To achieve this, we define $\mathbf{P}_A(p)$ as a restricted version of problem $\mathbf{P}_A$ with the additional requirement that the total number of work packages containing active tasks is exactly p, for $1 \leq p \leq \bar{n}$. Moreover, the following procedure reduces the range of p values that need to be examined, by determining a lower bound $\tilde{p}$ on the number of work packages with active tasks.

Procedure pMin

Step 1. Given the task network (N, A) with work packages $R_1, R_2, \ldots, R_m$ of inactive tasks, construct a new directed acyclic network (N', A') with m nodes, where node $k \in N'$ corresponds to R_k, and there exists an arc $k \rightarrow l \in A'$ if and only if R_k is a predecessor of R_l.

Step 2. Arc $k \rightarrow l$ has length 1 if there exists an active task $a \in \bar{R}$ such that R_k is a predecessor of a and R_l is a successor of a, and has length 0 otherwise.

Step 3. Determine the length, $\tilde{p}$, of the longest path in (N', A').

Lemma 2 *Any feasible solution to problem* $\mathbf{P}_A$ *uses at least* $\tilde{p}$ *work packages for active tasks.*

We now describe a method for generating a lower bound $LB(p)$ for problem $\mathbf{P}_A(p)$. The lower bound $LB(p)$ is obtained by solving a relaxation of $\mathbf{P}_A(p)$, which determines the values of $2p$ variables $q_1, q_2, \ldots, q_p; \chi_1, \chi_2, \ldots, \chi_p$. The values of $q_1, q_2, \ldots, q_p$ satisfy some conditions that the number of tasks in p work packages must satisfy, but they do not necessarily equal the number of tasks of the work packages in a feasible solution of $\mathbf{P}_A(p)$ for the given instance. Similarly, the values of $\chi_1, \chi_2, \ldots, \chi_p$ satisfy some conditions that the work contents in p work packages must satisfy, but they do not necessarily equal the work contents of the work packages in a feasible solution of $\mathbf{P}_A(p)$ for the given instance. Once $LB(p)$ is determined for $p = \tilde{p}, \tilde{p}+1, \ldots, \bar{n}$, a lower bound for problem $\mathbf{P}_A$ is given by $LB = \min_{p=\tilde{p},\tilde{p}+1,\ldots,\bar{n}}\{LB(p)\}$.

Since each group of inactive tasks R_i is required to form its own work package, for ease of presentation, we replace each group of inactive tasks R_i by a single inactive task a_i. For notational convenience, we assume that the tasks are indexed in such a way that tasks $1, 2, \ldots, \bar{n}$ are active and tasks $\bar{n}+1, \bar{n}+2, \ldots, \bar{n}+m$ are inactive. Denote $N_A = \{1, 2, \ldots, \bar{n}\}$ and $N_I = \{\bar{n}+1, \bar{n}+2, \ldots, \bar{n}+m\}$. Thus, $N = N_A \cup N_I$. Denote $\tilde{C}_i$ as the completion time of task i for $i = 1, 2, \ldots, \bar{n}+m$ if each active task forms its own work package. We also assume that the active tasks are indexed such that $\tilde{C}_1 \leq \tilde{C}_2 \leq \cdots \leq \tilde{C}_{\bar{n}}$, where the values of $\tilde{C}_1, \tilde{C}_2, , \ldots, \tilde{C}_{\bar{n}}$ are determined by the critical path method. For each $a \in N$, we let $Pred(a)$ denote the set of all predecessors of a, and let $Succ(a)$ denote the set of all successors of a.

To determine $LB(p)$, we first derive two necessary conditions under which a work package can be formed. Both conditions make use of the structure of the given task network and rely on the requirement that the network of work packages cannot contain cycles. They restrict the feasible combinations of $q_1, q_2, \ldots, q_p; \chi_1, \chi_2, \ldots, \chi_p$, based on the following two principles.

Network Inclusion Principle: Consider any active tasks $s_1, s_2, \ldots, s_\nu$ and any work package W that contains these tasks. If an inactive task a is a predecessor of some task s_i, $1 \leq i \leq \nu$, then no predecessor of a is in W. Symmetrically, if an inactive task a' is a successor of some task s_i, $1 \leq i \leq \nu$, then no successor of a' is in W.

Network Exclusion Principle: Consider any inactive tasks $r_1, r_2, \ldots, r_\kappa$ and any work package W that contains active tasks. For $i = 1, 2, \ldots, \kappa$, either no predecessor of r_i is in work package W or no successor of r_i is in work package W.

We now provide a formal description of the necessary condition derived from each principle above, as well as a result that establishes its validity.

For the Network Inclusion Principle, let $s_1, s_2, \ldots, s_\nu$ be any ν active tasks, where ν is a prespecified value. Define

$$V_{s_1,\ldots,s_\nu} = \bigcap_{r \in N_I \cap \bigcup_{i=1}^{\nu} Pred(s_i)} (N_A \setminus Pred(r)) \cap \bigcap_{r \in N_I \cap \bigcup_{i=1}^{\nu} Succ(s_i)} (N_A \setminus Succ(r)),$$

which is the set of active tasks that neither contains any predecessor of an inactive predecessor of some task in $\{s_1, s_2, \ldots, s_\nu\}$ nor contains any successor of an inactive successor of some task in

$\{s_1, s_2, \ldots, s_\nu\}$. For $q = 1, 2, \ldots, \bar{n}$, define

$$\bar{\Theta}'_{s_1,\ldots,s_\nu}(q) = \begin{cases} \emptyset, & \text{if there exist } r \in N_I \text{ and } i, i' \in \{1, 2, \ldots, \nu\} \\ & \text{such that } s_i \in \mathit{Pred}(r) \text{ and } s_{i'} \in \mathit{Succ}(r); \\ \{\chi \mid \sum_{j \in V_{s_1,\ldots,s_\nu}} y_j = q;\ \sum_{j \in V_{s_1,\ldots,s_\nu}} x_j y_j = \chi; & \\ \quad y_{s_1} = \cdots = y_{s_\nu} = 1;\ \text{for some } y_j \in \{0,1\},\ j \in V_{s_1,\ldots,s_\nu}\}, & \text{otherwise.} \end{cases}$$

If there exists $r \in N_I$ such that $s_i \in \mathit{Pred}(r)$ and $s_{i'} \in \mathit{Succ}(r)$ for some $i, i' \in \{1, 2, \ldots, \nu\}$, then including both s_i and $s_{i'}$ in a work package creates a cycle in the network of work packages. In such a case, $\bar{\Theta}'_{s_1,\ldots,s_\nu}(q)$ is an empty set. Otherwise, any feasible work package with exactly q tasks that contains $s_1, s_2, \ldots, s_\nu$ must have the sum of its task contents equal to some $\chi \in \bar{\Theta}'_{s_1,\ldots,s_\nu}(q)$; see the proof of Lemma 3 below for details. We determine $\bar{\Theta}'_{s_1,\ldots,s_\nu}(q)$ using dynamic programming, where the formulation is provided in Appendix B. For $q = 1, 2, \ldots, \bar{n}$, define

$$\Theta'(q) = \bigcup_{s_1,\ldots,s_\nu \in N_A \text{ s.t. } s_1 \le s_2 \le \cdots \le s_\nu} \bar{\Theta}'_{s_1,\ldots,s_\nu}(q).$$

The following lemma states the necessary condition defined by $\Theta'(q)$ under which a work package can be formed.

Lemma 3 *Any feasible work package with exactly q active tasks must have work content equal to some $\chi \in \Theta'(q)$.*

For the Network Exclusion Principle, let $r_1, r_2, \ldots, r_\kappa$ be any κ inactive tasks, where κ is a prespecified value. Define

$$\mathcal{U}_{r_1,\ldots,r_\kappa} = \Big\{ N_A \setminus (U_1 \cup U_2 \cup \cdots \cup U_\kappa) \ \Big|\ U_i \in \{\mathit{Pred}(r_i), \mathit{Succ}(r_i)\}, \text{ for } i = 1, \ldots, \kappa \Big\}.$$

For $U \in \mathcal{U}_{r_1,\ldots,r_\kappa}$ and $q = 1, 2, \ldots, \bar{n}$, define

$$\Psi_U(q) = \left\{ \chi \ \middle|\ \sum_{j \in U} y_j = q \text{ and } \sum_{j \in U} x_j y_j = \chi \text{ for some } y_j \in \{0,1\},\ j \in U \right\}.$$

We determine $\Psi_U(q)$ using dynamic programming, where the formulation is provided in Appendix B. For $q = 1, 2, \ldots, \bar{n}$, define

$$\bar{\Theta}''_{r_1,\ldots,r_\kappa}(q) = \bigcup_{U \in \mathcal{U}_{r_1,\ldots,r_\kappa}} \Psi_U(q).$$

The set $\bar{\Theta}''_{r_1,\ldots,r_\kappa}(q)$ defines a necessary condition under which a single work package of q active tasks can be formed; see the proof of Lemma 4 below for details. For $q = 1, 2, \ldots, \bar{n}$, define

$$\Theta''(q) = \bigcap_{r_1,\ldots,r_\kappa \in N_I \text{ s.t. } r_1 \le r_2 \le \cdots \le r_\kappa} \bar{\Theta}''_{r_1,\ldots,r_\kappa}(q).$$

The following lemma states the necessary condition defined by $\Theta''(q)$ under which a work package can be formed.

Lemma 4 *Any feasible work package with exactly q active tasks must have work content equal to some $\chi \in \Theta''(q)$.*

Next, we consider a collection of work packages with given total work content. Note that for $q = 1, 2, \ldots, \bar{n}$,

$$\Psi_{N_A}(q) = \left\{ \chi \;\middle|\; \sum_{j=1}^{\bar{n}} y_j = q \text{ and } \sum_{j=1}^{\bar{n}} x_j y_j = \chi \text{ for some } y_1, \ldots, y_{\bar{n}} \in \{0,1\} \right\}.$$

Thus, there exist q active tasks such that the sum of their work contents is equal to χ if and only if $\chi \in \Psi_{N_A}(q)$. Hence, the set $\Psi_{N_A}(q)$ enables us to derive a necessary condition under which one or more work packages containing a total of q active tasks from N_A can be formed.

Next, we study how grouping tasks to form a work package increases the discounted cash flow of those tasks. For $i = 1, 2, \ldots, \bar{n}$, define

$$\Xi_i = \{ j \mid j \notin \mathit{Pred}(r) \text{ for all } r \in N_I \cap \mathit{Pred}(i);\ j = 1, \ldots, i \},$$

which contains all of tasks $1, 2, \ldots, i$, except those tasks that are predecessors of some inactive predecessor of task i. For $i = 1, 2, \ldots, \bar{n}$ and $\chi \in \cup_{q=1}^{\bar{n}} [\Theta'(q) \cap \Theta''(q)]$ such that $\sum_{l \in \Xi_i} x_l \ge \chi$, define

$$\theta_i(\chi) = \min \left\{ j \;\middle|\; \sum_{l \in \{i-j, i-j+1, \ldots, i\} \cap \Xi_i} x_l \ge \chi;\ j = 0, 1, \ldots, i-1 \right\}.$$

For $\chi \in \cup_{q=1}^{\bar{n}} [\Theta'(q) \cap \Theta''(q)]$, define

$$\begin{aligned}\phi(\chi) = \min_{i=1,2,\ldots,\bar{n} \text{ s.t. } \sum_{l \in \Xi_i} x_l \ge \chi} \Bigg\{ & \sum_{j \in \{i-\theta_i(\chi)+1, i-\theta_i(\chi)+2, \ldots, i\} \cap \Xi_i} x_j e^{-\alpha(d - \tilde{C}_i + \tilde{C}_j)} \\ & + \left(\chi - \sum_{j \in \{i-\theta_i(\chi)+1, i-\theta_i(\chi)+2, \ldots, i\} \cap \Xi_i} x_j \right) e^{-\alpha(d - \tilde{C}_i + \tilde{C}_{i-\theta_i(\chi)})} \Bigg\} - \chi e^{-\alpha d}.\end{aligned}$$

The quantity $\phi(\chi)$ is used as a lower bound on the increase in total discounted cash flow when we group some active tasks to form a work package with work content of at least χ; see the proof of

Lemma 5 below for details. The index i on the right side of this equation specifies the last task in the work package that is potentially being formed.

Let

$$\Pi = \Bigg\{ (q_1, \ldots, q_p; \chi_1, \ldots, \chi_p) \;\Bigg|\; \sum_{k=1}^{p} q_k = \bar{n};\ \sum_{k=1}^{p} \chi_k = \bar{X};$$

$$q_k, \chi_k \in \mathbb{Z}^+,\ \chi_k \in \Theta'(q_k) \cap \Theta''(q_k),\ \text{and} \sum_{j=1}^{k} \chi_j \in \Psi_{N_A}\Bigg(\sum_{j=1}^{k} q_j\Bigg) \text{ for } k = 1, \ldots, p \Bigg\}$$

and

$$LB(p) = \omega(p+m-2) + \min_{(q_1,\ldots,q_p;\chi_1,\ldots,\chi_p)\in\Pi} \sum_{k=1}^{p} \big[F(\chi_k) + \xi\phi(\chi_k)\big] + \sum_{i=\bar{n}+1}^{\bar{n}+m} F(x_i) + \xi \sum_{i=1}^{\bar{n}+m} x_i(1 - e^{-\alpha\tilde{C}_i}).$$

Lemma 5 *For $p = 1, 2, \ldots, \bar{n}$, a lower bound on the total cost of problem $\mathbf{P}_A(p)$ is given by $LB(p)$.*

For notational convenience, we define

$$\Lambda(p) = \min_{(q_1,\ldots,q_p;\chi_1,\ldots,\chi_p)\in\Pi} \sum_{k=1}^{p} \big[F(\chi_k) + \xi\phi(\chi_k)\big].$$

Thus, $LB(p) = \omega(p+m-2) + \Lambda(p) + \sum_{i=\bar{n}+1}^{\bar{n}+m} F(x_i) + \xi \sum_{i=1}^{\bar{n}+m} x_i(1 - e^{-\alpha\tilde{C}_i})$. The value of $\Lambda(p)$ is found using dynamic programming, where the formulation is provided in Appendix B.

Theorem 3 *A lower bound on the total cost of problem $\mathbf{P}_A$ is given by $LB = \min_{p=\tilde{p},\tilde{p}+1,\ldots,\bar{n}}\{LB(p)\}$, and found in $O(\bar{n}^{\nu+2}\bar{X} + 2^{\kappa}m^{\kappa}\bar{n}^2\bar{X} + \bar{n}^3\bar{X}^2)$ time.*

While the conditions "$\chi \in \Theta'(q)$" and "$\chi \in \Theta''(q)$" in Lemmas 3 and 4 provide strong restrictions on the search space of $q_1, q_2, \ldots, q_p; \chi_1, \chi_2, \ldots, \chi_p$, it is computationally difficult to implement the sets $\Theta'(q)$ and $\Theta''(q)$ for large values of ν and κ. Hence, our implementation of $\Theta'(q)$ and $\Theta''(q)$ is limited to $\nu = \kappa = 2$, so that the overall time requirement for determining our lower bound is $O(\bar{n}^4\bar{X} + m^2\bar{n}^2\bar{X} + \bar{n}^3\bar{X}^2) \leq O(\bar{n}^2(\bar{n}+m)\bar{X}^2) = O(\bar{n}^2 n\bar{X}^2)$. Nonetheless, these two conditions enable us to find strong lower bounds, as reported in Section 4.1.

4 Computational Study and Insights

We conduct a computational study to evaluate the performance of Heuristic MergeTasks. In Section 4.1, we test the performance of the heuristic against the lower bound described in Section 3.2.2, and discuss how various parameter settings influence work package sizes. In Section 4.2, we first describe a benchmark approach from the business literature, for comparison with our heuristic.

We then test the performance of Heuristic MergeTasks against this benchmark approach. In Section 4.3, we compare the effect on project makespan of our model of Section 2, relative to an approach that considers work package costs and schedule issues separately.

4.1 Analysis of MergeTasks solutions

We test the performance of Heuristic MergeTasks using randomly generated data. To generate a random problem, we first generate a task network using the network generator RanGen2 (Vanhoucke et al. 2008). When using RanGen2, we specify the following network measures:

I_1 (network size indicator): The number of non-dummy tasks, $n - 2$.

I_2 (serial or parallel indicator): $I_2 = (\eta - 1)/(I_1 - 1)$, where η is the depth of the network, and $0 \leq I_2 \leq 1$.

The network generated is close to a serial network if I_2 is close to 1, and is close to a network with all tasks in parallel if I_2 is close to 0. For the other network measures of RanGen2, we use their default settings. We generate the other parameter values of each random problem as follows. Among the I_1 non-dummy tasks in the network generated by RanGen2, we randomly select $\bar{n}$ tasks to be active, where each task is equally likely to be selected. We let each of the remaining $I_1 - \bar{n} + 2$ tasks in the network, including the two dummy tasks, represent a group of inactive tasks that is required to form a single work package. Thus, $m = I_1 - \bar{n} + 2$. The work content x_a and duration t_a of each active task a are both uniformly distributed integers from $\{1, 2, \ldots, 10\}$, where x_a and t_a are correlated with a correlation coefficient ρ. The work content x_a and duration t_a of each work package of non-dummy inactive tasks are uniformly distributed integers from $\{1, 2, \ldots, 40\}$ and $\{1, 2, \ldots, 20\}$, respectively, where x_a and t_a are correlated with a correlation coefficient ρ.

In our first experiment, we set $I_1 \in \{40, 80, 120, 160\}$ and $I_2 \in \{0.2, 0.4, 0.6, 0.8, 1\}$, giving 20 combinations of I_1 and I_2. For each of these combinations, we generate 10 random test instances, for a total of 200 instances. For each test instance, we set $\bar{n} = 0.9 \times I_1$ and $\rho = 0.5$. We set the project deadline to $d = 1.1 \times \Delta$, where Δ is the minimum project makespan that is achieved when each active task forms its own work package. We set the discount rate as $\alpha = 0.00025$, which represents an annual discount rate of approximately 8.7% if each time unit represents a calendar day. We set $\omega = 50$, $\xi = 50$, $f(x) = h(x) = 3x^{0.8}$, and $g(x) = x^{1.2}$. These cost parameters are selected in such a way that the five cost components, namely fixed cost, cost of inaccurate estimation, monitoring/control cost, economies of scale, and discounted cash flow cost, have the same order of magnitude in most test instances. We choose power functions of the form kx^{λ} for

Table 1: Accuracy, efficiency, and solution characteristics of Heuristic MergeTasks.

I_1 ($\bar{n}$)	I_2	e (%)	Running time of MergeTasks (sec.)	Running time of lower bound procedure (sec.)	Mean number of tasks in a work package	Standard deviation of number of tasks in a work package
40 (36)	0.2	7.09	6.6	13.3	9.1	4.3
	0.4	3.90	4.2	10.9	10.0	5.1
	0.6	3.20	1.7	6.1	9.4	5.7
	0.8	2.35	0.2	5.4	9.8	5.9
	1.0	1.76	0.0	3.0	8.3	4.7
80 (72)	0.2	4.87	89.0	253.0	13.1	6.5
	0.4	4.20	52.8	140.5	11.9	6.5
	0.6	3.10	18.1	109.2	10.8	6.0
	0.8	2.54	2.4	90.2	9.9	6.4
	1.0	1.88	0.0	43.1	8.5	5.4
120 (108)	0.2	6.19	645.8	843.3	10.8	7.1
	0.4	4.77	362.4	585.4	10.7	7.2
	0.6	3.51	144.2	500.4	10.6	6.6
	0.8	2.77	18.9	487.7	9.6	6.6
	1.0	2.03	0.0	253.8	8.4	5.8
160 (144)	0.2	5.03	2222.2	4523.1	12.9	9.2
	0.4	4.02	1257.6	2261.3	11.2	7.5
	0.6	3.98	260.2	1424.4	9.3	7.4
	0.8	2.81	37.6	1203.5	8.9	6.2
	1.0	1.79	0.0	679.2	8.7	5.8

$f(\cdot)$, $g(\cdot)$, and $h(\cdot)$, in order to allow a simple robustness check in the second experiment below.

For each test instance, we run Heuristic MergeTasks, and let z^H denote the total cost, as defined by (1), of the heuristic solution. For each test instance, we also compute a lower bound LB on the cost function (1), as described in Section 3.2.2. The performance of the heuristic, relative to a lower bound on total cost for any test instance, is given by:

$$e = \frac{z^H - LB}{LB} \times 100\%.$$

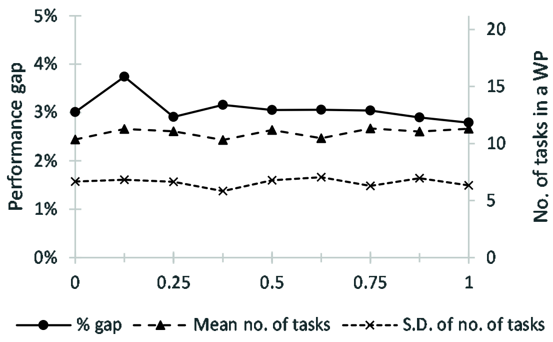
Heuristic MergeTasks and the lower bound procedure are coded in Excel VBA, and the experiments are run on a PC with a 3.40 GHz Intel Core i7-6700 processor.

Table 1 summarizes our results, where each row represents the mean result of the 10 random test instances. We observe that the performance gap of the heuristic is less sensitive to problem size than it is to network shape as defined by I_2. The performance gap tends to be higher when I_2 is smaller. The results in Table 1 also demonstrate that Heuristic MergeTasks finds accurate solutions to problem $\mathbf{P}_A$ in an amount of time that is reasonable within the context of project planning. The last two columns of Table 1 show the mean and standard deviation of the number of tasks in the work packages of active tasks in the heuristic solutions. The results indicate that, especially for large size problems, the solutions tend to form work packages with more tasks and less consistent number of tasks when the task network is closer to a parallel network. This is because when there are more parallel active tasks, the discounted cash flow costs of the work packages tend to be relatively smaller and less important, and this effect is particularly strong when the problem size is large. Thus, for large size problems, there is less incentive to form more but smaller work packages, or to form work packages with consistent sizes. The solutions also typically form work packages with less consistent number of tasks when the problem size is larger.

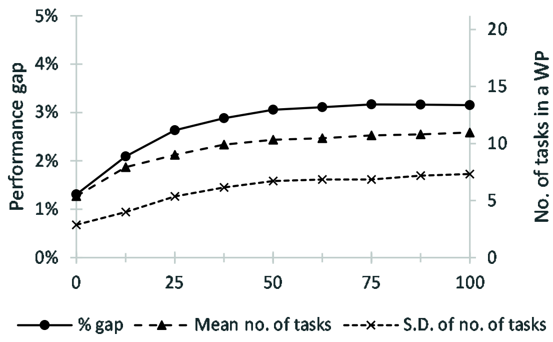
A discussion of the serial network case provides useful insights about our results. When $I_2 = 1$, the task network contains only serial tasks, and thus Heuristic MergeTasks obtains the optimal solution in Step 1. As a result, we know that the remaining gap e, with average ranging between 1.76% and 2.03%, arises entirely from approximation of the lower bound. Hence, the performance gaps shown in Table 1 typically overestimate the relative errors of the heuristic solutions found.

In our second experiment, we fix $I_1 = 100$ and $I_2 = 0.6$. We let $f(x) = h(x) = 3x^{\lambda}$, $g(x) = x^{\mu}$, $d = \zeta \times \Delta$, and $\bar{n} = I_1 - m + 2$. We initially set $\rho = 0.5$, $\omega = 50$, $\xi = 50$, $\alpha = 0.00025$, $\lambda = 0.8$, $\mu = 1.2$, $\zeta = 1.1$, $m = 12$, and test the heuristic by varying the values of ρ, ω, ξ, α, λ, μ, ζ, and m one at a time. Figure 4 shows our results, where each point on a graph represents the average result of the 10 test instances. We observe that the performance gap increases modestly as ω, ξ, and α increase and as λ and μ decrease. The performance gap increases sharply as ζ drops to 1. This is the special situation where the project deadline is equal to the minimum possible project makespan. Overall, the results in Figures 4(a)–(h) demonstrate that Heuristic MergeTasks finds accurate solutions for most parameter settings.

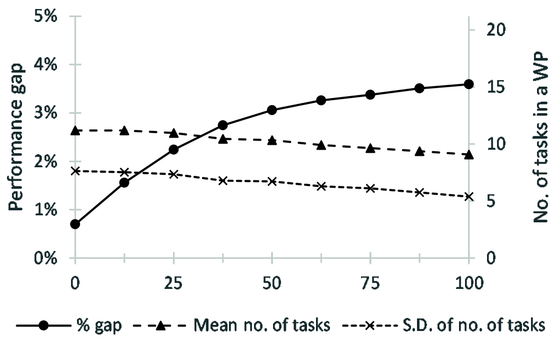
In our third experiment, we study the work package sizes found in the instances of the previous experiment. We compute the mean and standard deviation of the number of tasks in the work packages of active tasks for each test instance. Figure 4 also shows these results, where each point on a graph represents the average result of the 10 test instances. We observe that the mean number of tasks per work package tends to increase as ω and ζ increase, and as ξ, α, λ, μ, and m decrease. The reasons are as follows. As ω increases, the fixed cost becomes relatively more important, hence the solution tends to form fewer but larger work packages. As ζ increases, the deadline constraint becomes less restrictive to the formation of work packages, and thus the solution tends to form larger work packages. As λ decreases, the concavity of functions $f(\cdot)$ and $h(\cdot)$ becomes stronger. Thus, accurate cost/schedule estimates and economies of scale become relatively more significant, and the solution tends to form fewer but larger work packages. As ξ, α, and μ decrease, the



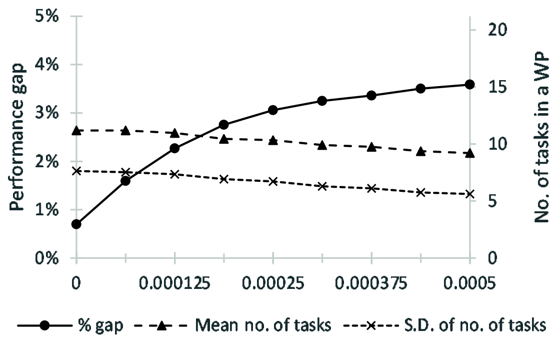
(a) Impact of ρ



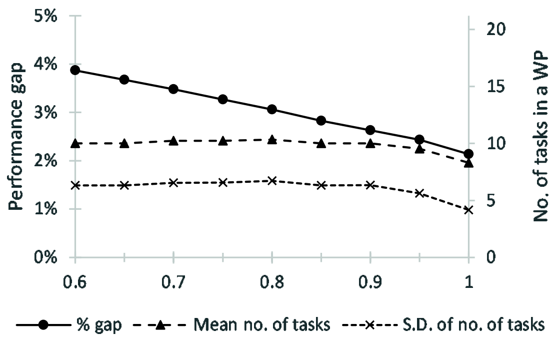
(b) Impact of ω



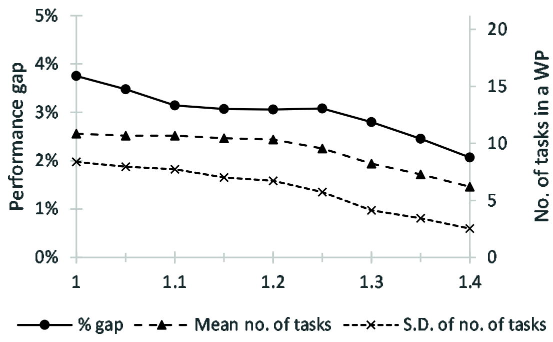
(c) Impact of ξ



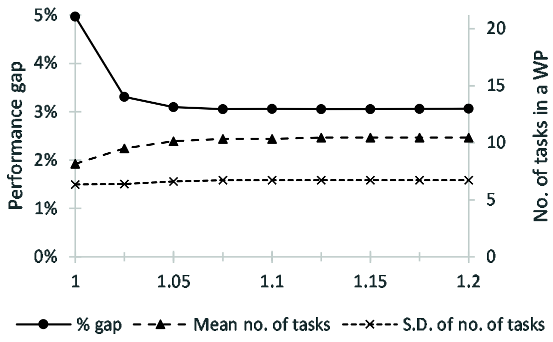
(d) Impact of α



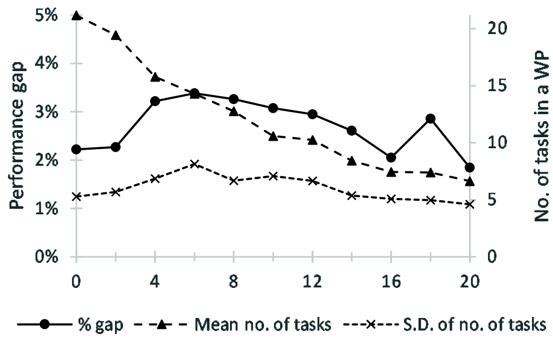
(e) Impact of λ



(f) Impact of μ



(g) Impact of ζ



(h) Impact of m

Figure 4: Impact of parameters on the performance gap and number of tasks per work package.

discounted cash flow cost and monitoring/control cost become relatively less important, hence the solution tends to form fewer but larger work packages. As m decreases, the acyclicity constraint becomes less restrictive to the formation of work packages, and thus the solution tends to form larger work packages. Figure 4(a) shows that parameter ρ has little impact on work package sizes.

From Figure 4, we also observe that the standard deviation of the number of tasks in a work package tends to increase as ω increases and as ξ, α, λ, and μ decrease. The ratio of the standard deviation to the mean of the number of tasks per work package, which reflects the relative variability of work package size, remains stable as ρ varies, and tends to increase modestly as ω increases and as ξ, α, λ, and ζ decrease. However, this ratio increases significantly as m increases. This is because having more groups of inactive tasks results in fewer choices for forming work packages, hence the feasible work packages tend to have less consistent sizes as m increases. This ratio also decreases significantly as μ increases. This is because the monitoring and control cost is a convex nondecreasing function of the work package size. When the convexity of the cost function is strong, using large work packages is costly; therefore, using more consistent work package sizes provides a lower cost solution.

4.2 MergeTasks vs. constant target work package size

As discussed in Section 1, a common practice among project managers is to use rules of thumb to determine work package sizes. These rules of thumb typically have the characteristic that the same rule applies to all work packages throughout the project. Such rules include, for example, "six months of development time or \$100,000 of cost" (Brown 1978), "4/40 rule" (Kennemer 2002), etc. In this section, we computationally compare the performance of Heuristic MergeTasks with a benchmark procedure designed to implement this characteristic. This benchmark procedure applies a constant target work package size across all work packages.

Let X denote the target work package size. The procedure keeps at most one work package open at any time, and constructs it by adding active tasks one at a time. We say that an active task a is *available* if a is not included in a work package and each of the predecessors of a is either included in some work package or is inactive. Let $\hat{x}$ denote the work content in the currently open work package. Let $\mathcal{N}'$ denote the precedence network that contains the constructed work packages and the work packages of inactive tasks. In designing the procedure, we ignore the effect of the deadline constraint on work package decisions. We do so, first for simplicity, and second for comparability with the work package sizing rules used in practice and discussed in Section 1. A formal statement of the procedure follows.

Procedure Benchmark

Step 1. If all active tasks are included in work packages, then stop. Otherwise, open a new work package; put the smallest available active task a in this work package; set $\hat{x} := x_a$; and update the set of available active tasks.

Step 2. Search for the next smallest available active task a such that adding task a to the current work package does not create a cycle in network $\mathcal{N}'$. If no such task a exists or if $x_a \geq 2(X - \hat{x})$, then close the current work package and go to Step 1.

Step 3. Add task a to the current work package. Set $\hat{x} := \hat{x} + x_a$. Update the set of available active tasks. Go to Step 2.

In Step 2 of Procedure Benchmark, we add the next available task a to the current work package only when $x_a < 2(X - \hat{x})$. This ensures that the content of the work package formed is close to the target X.

Let $z^B(X)$ denote the total cost of the solution delivered by Procedure Benchmark using a constant target work package size X. Let z^H denote the solution delivered by Heuristic MergeTasks with deadline d set equal to infinity. Let $\lfloor y \rceil$ denote the closest integer to y, for any real number y. We define three performance measures: $\varepsilon = [z^B(X^*) - z^H]/z^B(X^*) \times 100\%$, $\varepsilon^- = [z^B(X^-) - z^H]/z^B(X^-) \times 100\%$, and $\varepsilon^+ = [z^B(X^+) - z^H]/z^B(X^+) \times 100\%$, where $X^* = \text{argmin}_{X=1,2,\ldots}\{z^B(X)\}$, $X^- = \lfloor (0.8)X^* \rceil$, and $X^+ = \lfloor (1.2)X^* \rceil$. Thus, X^- and X^+ represent underestimation and overestimation of the optimized target X^* by 20%, respectively.

The results in Table 2 estimate the value delivered by our model and algorithm, relative to traditional project management practice. The test instances used are the same as in Section 4.1, and each cell in Table 2 reports the average result of the test instances. For the largest instances studied, with 144 active tasks, the fourth column shows that our model delivers up to a 3.40% cost reduction, compared to an optimized choice of work package size target. In reality however, most companies would find it difficult to optimize this choice. Hence we investigate, in the third and fifth columns, the relative cost reduction from our model, compared to suboptimal choices that are too low and too high, respectively, by 20%. These results show an increase of several percentage points in the relative cost reduction achieved by our model, for most ranges of data. While the relative cost reductions are smaller at large values of I_2, i.e., as networks become closer to serial, there are many practical projects for which $I_2 \leq 0.4$ (Vanhoucke et al. 2016). Furthermore, there is

Table 2: Comparison with benchmark solutions.

I_1 ($\bar{n}$)	I_2	ε^- (%)	ε (%)	ε^+ (%)
40 (36)	0.2	3.62	0.47	1.26
	0.4	2.79	0.36	0.85
	0.6	2.96	0.20	0.67
	0.8	2.61	0.07	0.20
	1.0	2.82	0.00	0.08
80 (72)	0.2	5.06	2.65	3.60
	0.4	3.17	1.48	1.74
	0.6	3.15	0.78	1.09
	0.8	2.64	0.69	0.86
	1.0	1.57	0.03	0.10
120 (108)	0.2	4.79	2.95	3.69
	0.4	4.30	2.21	2.72
	0.6	2.78	1.42	1.93
	0.8	2.72	1.10	1.33
	1.0	0.92	0.02	0.16
160 (144)	0.2	5.02	3.40	4.29
	0.4	4.77	3.19	4.08
	0.6	3.25	1.83	2.38
	0.8	1.84	0.81	1.06
	1.0	0.73	0.07	0.21

improvement in the cost reductions achieved by our model as projects become larger. Based on this evidence, we believe that the cost reductions achieved would still be substantial in larger projects. As an additional comment, our work is apparently the first to provide estimates of the effectiveness of the long-established project management practice of applying the same work package sizing rule to all work packages. Those estimates suggest that, especially for projects with less close to serial network structures, the traditional rule of thumb approach incurs avoidable costs that most project companies and their clients would find unacceptable.

4.3 Integrated planning

Traditionally, work breakdown structure planning is performed without consideration of the schedulability of the resulting work packages (see, e.g., Nicholas and Steyn 2017). This often results in delays in project completion time. Our model, however, takes into account the precedence constraints between tasks when making work package sizing decision. Hence, it can be viewed as an *integration of work package sizing and scheduling decisions*. In this section, we study the value of this integrated planning approach. Specifically, we use our model to study the typical effect on project makespan of two different approaches. The first approach is similar to solving problem $\mathbf{P}_A$, but with the removal of the elements that are related to the schedule or makespan. We use this simplified model to represent the optimization of work breakdown structure costs without regard to

Table 3: Comparison of decomposed and integrated approaches.

I_1 $(\bar{n})$	I_2	Δ^D	Δ^I	ε^I (%)
40 (36)	0.2	73.6	54.6	25.8
	0.4	107.4	101.3	5.5
	0.6	154.4	150.0	2.8
	0.8	191.1	189.6	0.8
	1.0	237.8	237.8	0.0
80 (72)	0.2	131.6	111.2	15.3
	0.4	219.6	207.8	5.3
	0.6	306.2	304.4	0.6
	0.8	388.6	386.9	0.4
	1.0	481.5	481.5	0.0
120 (108)	0.2	233.6	186.9	20.1
	0.4	364.8	336.0	7.7
	0.6	471.0	457.0	3.0
	0.8	602.1	595.8	1.0
	1.0	735.0	735.0	0.0
160 (144)	0.2	335.4	291.1	12.9
	0.4	496.8	463.8	6.6
	0.6	678.5	654.6	3.5
	0.8	830.4	820.7	1.2
	1.0	968.7	968.7	0.0

scheduling considerations. Having designed the work packages using this approach, they are then scheduled to evaluate the makespan. We refer to this approach as a *decomposed approach.* The second approach, which we refer to as an *integrated approach*, is to solve problem $\mathbf{P}_A$, where the scheduling considerations are integrated into the work package formation decisions.

We computationally compare the project makespans of the solutions generated by the decomposed approach with those generated by the integrated approach. To implement the integrated approach, we execute Heuristic MergeTasks. To implement the decomposed approach, we relax the deadline constraint, remove the discounted cash flow terms "$\xi \sum_{k=1}^{p} x_{W_k}(1 - e^{-\alpha C_{W_k}}) + \xi \sum_{k=1}^{m} x_{R_k}(1 - e^{-\alpha C_{R_k}})$" from the total cost function (1), and then execute Heuristic MergeTasks. Let Δ^D denote the makespan of the solution generated by the decomposed approach, and Δ^I denote the makespan of the solution generated by the integrated approach. Then, $\varepsilon^I = (\Delta^D - \Delta^I)/\Delta^D \times 100\%$ represents the percentage reduction in project makespan accomplished through integrated planning.

The test instances used are the same as in Section 4.1, and each cell in Table 3 reports the average result of the test instances. For instances with $I_2 = 1$, both approaches generate solutions with the same makespan, which is independent of work package formation decisions, for a serial network. From Table 3, we observe that the decomposed approach delays project completion more significantly when the task network is closer to a parallel network. On the other hand, the problem size does not have consistent impact on the increase in project makespan caused by the decomposed approach.

5 Extensions

In Section 5.1, we analyze a special case of $\mathbf{P}_A$ in which the given task network is a group of serial paths in parallel. Section 5.2 considers the issue of incompatibility between tasks. Section 5.3 discusses the model with generalized precedence constrained work packages.

5.1 Multiple serial paths in parallel

In this section, we analyze the solvability of a special case of problem $\mathbf{P}_A$ in which the given task network is a group of u serial paths in parallel. In this case, there is a set $\{a_{ij} \mid i = 1, \ldots, u; \, j = 1, \ldots, k_i\}$ of active tasks, where (i) $a_{i1}, a_{i2}, \ldots, a_{ik_i}$ are serial tasks for $i = 1, 2, \ldots, u$, and (ii) a_{ij} is neither a predecessor nor a successor of $a_{i'j'}$ whenever $i \neq i'$. We denote this problem by $\mathbf{P}_{SP}$.

Theorem 4 *The recognition version of problem $\mathbf{P}_{SP}$ is binary* NP*-complete for any fixed $u \geq 2$, even if $x_a = t_a$ for all tasks a.*

Theorem 4 implies that it is unlikely that a polynomial-time optimal algorithm exists for problem $\mathbf{P}_{SP}$, even when there are only two serial paths in parallel.

Theorem 5 *If u is fixed, then problem $\mathbf{P}_{SP}$ with active task set $\{a_{ij} \mid i = 1, \ldots, u; \, j = 1, \ldots, k_i\}$ can be solved optimally in $O(k_1^2 k_2^2 \cdots k_u^2 d^u)$ time.*

Theorem 5 implies that when the number of serial paths is fixed, problem $\mathbf{P}_{SP}$ is solvable in pseudopolynomial time. However, the complexity of the pseudopolynomial time algorithm is high, even when there are few serial paths.

5.2 Task incompatibility

In various project management applications, some tasks are incompatible with others. As a result, they should not be placed in the same work package under close proximity. Examples include high and low temperature operations, processes involving mutually antagonistic chemicals, or tasks that require work by two or more aggressively competitive subcontractors. In this section, we consider an extension of our model in which some pairs of active tasks are mutually incompatible. Pairwise incompatibility between the active tasks of a project can be represented by an $\bar{n} \times \bar{n}$ symmetric binary matrix, where a "1" in row i and column j means that tasks i and j can be placed within the same work package, and a "0" means that they are incompatible. We denote these generalizations of problems $\mathbf{P}_S$ and $\mathbf{P}_A$ as problems $\mathbf{P}'_S$ and $\mathbf{P}'_A$, respectively.

The algorithm for $\mathbf{P}_S$ can be extended to solve problem $\mathbf{P}'_S$ by setting $c(i,j) = +\infty$ if there exists a pair of incompatible tasks among tasks $\{a_i, a_{i+1}, \ldots, a_j\}$. For problem $\mathbf{P}'_A$, the methodology described in Section 3 for problem $\mathbf{P}_A$ can also be extended. For finding feasible solutions, we propose the use of Heuristic MergeTasks with the following changes:

(1) In Step 1, apply the extended algorithm for problem $\mathbf{P}'_S$ to active serial task subsets.

(2) In Steps 2(ii) and 3(ii), merge two work packages that do not create any cycle in the network, or put a second incompatible task into the same work package.

The existing lower bound method presented in Section 3.2.2 is valid for problem $\mathbf{P}'_A$. Furthermore, we strengthen the lower bound in the presence of pairwise incompatible tasks by limiting the range considered for the number of work packages, using the following strengthened version of Procedure pMin.

Procedure pMin2

Step 1. Given the task network (N, A) with work packages $R_1, R_2, \ldots, R_m$ of inactive tasks, construct a new directed acyclic network (N', A') with m nodes, where node $k \in N'$ corresponds to R_k, and there exists an arc $k \to l \in A'$ if and only if R_k is a predecessor of R_l.

Step 2. For each arc $k \to l \in A'$:

(a) If there is no active task $a \in \bar{R}$ such that R_k is a predecessor of a and R_l is a successor of a, then arc $k \to l$ has length 0.

(b) Otherwise, determine the length of arc $k \to l$ as follows: Construct a directed acyclic network (N''_{kl}, A''_{kl}), where each node i in N''_{kl} corresponds to an active task i such that R_k is a predecessor of i and R_l is a successor of i, and an arc $i \to j$ exists in A''_{kl} if and only if i is a predecessor of j in the task network. In network (N''_{kl}, A''_{kl}), an arc $i \to j$ has length 1 if tasks i and j are incompatible, and has length 0 otherwise. The length of arc $k \to l \in A'$ is set equal to the length of the longest path in network (N''_{kl}, A''_{kl}) plus one.

Step 3. Determine the length, $\bar{p}$, of the longest path in (N', A').

Lemma 6 *Any feasible solution to problem $\mathbf{P}'_A$ uses at least $\bar{p}$ work packages for active tasks.*

Because of the complexity involved in the detailed development of an accurate lower bound for problem $\mathbf{P}'_A$, such work belongs within a study of specific applications for which task incompatibility is documented as a major issue. Since our focus is on generic project management applications, we leave this study as a topic for future research.

5.3 Generalized precedence constrained work packages

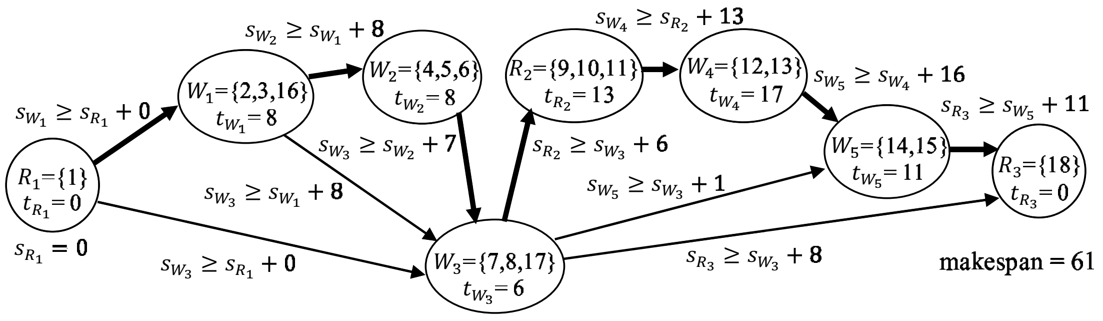
In this section, we consider a variant of problem $\mathbf{P}_A$ with generalized precedence constrained work packages, as briefly described in Section 1. In this problem, if a task $a_i \in N \setminus \{a_1, a_2, \ldots, a_k\}$ is a successor of one of tasks $a_1, a_2, \ldots, a_k$, then the formation of the work package $\{a_1, a_2, \ldots, a_k\}$ does not necessarily make a_i a successor of all the tasks $a_1, a_2, \ldots, a_k$. However, this problem has the following requirements: (i) If task a_i is a successor of task a_j in the given task network, then the start time of task a_i must be no earlier than the start time of task a_j after the work packages are formed. (ii) For each work package W, once any task in W has started, the other tasks in W are required to be processed as early as possible, subject to the precedence constraints between tasks within W. Requirement (ii) is motivated by the fact that a work package is conventionally managed by a single person or organizational unit, and thus its operations should be scheduled independently of the detailed schedules of other work packages. Note that this requirement implies that the duration of a work package is the length of the critical path of the tasks within the work package. We denote this problem by $\mathbf{P}_G$.

Consider any pair of work packages $W = \{a_1, a_2, \ldots, a_k\}$ and $W' = \{a'_1, a'_2, \ldots, a'_{k'}\}$ where at least one task in W' is a successor of a task in W. For $i = 1, 2, \ldots, k$, let σ_i denote the difference between the start time of task a_i and the start time of work package W when the tasks in W are processed as early as possible within W. For $j = 1, 2, \ldots, k'$, let σ'_j denote the difference between the start time of task a'_j and the start time of work package W' when the tasks in W' are processed as early as possible within W'. The "Start-to-Start" or "SS" relation between these two work packages is given by $s_{W'} \geq s_W + \tau_{W,W'}$, where s_W and $s_{W'}$ are the start times of work packages W and W', respectively, and

$$\tau_{W,W'} = \max_{a_i \in W;\, a'_j \in W';\, a_i \to a'_j \in A} \{\sigma_i + t_{a_i} - \sigma'_j\}.$$

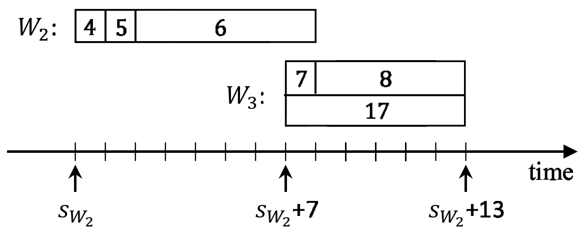
Thus, the network of work packages for problem $\mathbf{P}_G$ is a network with generalized precedence relations (see, e.g., Elmaghraby and Kamburowski 1992 for a discussion of generalized precedence relations). To check whether a solution is feasible, we compute the start times of all work packages using these SS relations between work packages, and check whether the start time of the work package that contains the dummy ending task is d or less. This is equivalent to solving a longest path problem in the network of work packages in which the length of each arc $W \rightarrow W'$ is given by $\tau_{W,W'}$, where the existence of a positive length cycle in this network implies that the solution is infeasible.

Consider problem $\mathbf{P}_G$ with Example 1, where the task network is shown in Figure 3(a). Consider the solution where the active tasks are grouped into work packages $W_1 = \{2, 3, 16\}$, $W_2 = \{4, 5, 6\}$, $W_3 = \{7, 8, 17\}$, $W_4 = \{12, 13\}$, and $W_5 = \{14, 15\}$. The network of work packages and the SS relations between work packages are shown in Figure 5(a). The SS relation between W_2 and W_3, for example, can be explained as follows. The precedence relations between the tasks in W_2 and the tasks in W_3 include $5 \rightarrow 7$ and $6 \rightarrow 8$. The precedence relation $5 \rightarrow 7$ requires task 7 to start no earlier than the completion time of task 5, which implies that task 7 cannot start earlier than $s_{W_2} + 2$. The precedence relation $6 \rightarrow 8$ requires task 8 to start no earlier than the completion time of task 6, which implies that task 8 cannot start earlier than $s_{W_2} + 8$. Since task 7 must start at s_{W_3} and task 8 must start at $s_{W_3} + 1$, we have $s_{W_3} \geq s_{W_2} + 2$ and $s_{W_3} + 1 \geq s_{W_2} + 8$. Hence, $s_{W_3} \geq s_{W_2} + 7$; see Figure 5(b) which shows a schedule with the earliest possible start time of W_3 relative to that of W_2.



R_1={1} t_{R_1}=0
$s_{R_1} = 0$
$s_{W_1} \geq s_{R_1} + 0$
W_1={2,3,16} t_{W_1}=8
$s_{W_2} \geq s_{W_1} + 8$
W_2={4,5,6} t_{W_2}=8
$s_{W_3} \geq s_{W_2} + 7$
$s_{W_3} \geq s_{W_1} + 8$
$s_{W_3} \geq s_{R_1} + 0$
W_3={7,8,17} t_{W_3}=6
R_2={9,10,11} t_{R_2}=13
$s_{R_2} \geq s_{W_3} + 6$
$s_{W_4} \geq s_{R_2} + 13$
W_4={12,13} t_{W_4}=17
$s_{W_5} \geq s_{W_4} + 16$
$s_{W_5} \geq s_{W_3} + 1$
W_5={14,15} t_{W_5}=11
$s_{R_3} \geq s_{W_5} + 11$
R_3={18} t_{R_3}=0
$s_{R_3} \geq s_{W_3} + 8$
makespan = 61

(a) network of work packages



(b) earliest possible start time of W_3

Figure 5: Network of work packages and SS relations.

We discuss several similarities and differences between problem $\mathbf{P}_A$ and problem $\mathbf{P}_G$. First, it is easy to check that Lemma 1 and Theorem 1 remain valid for problem $\mathbf{P}_G$. The condition "$\sum_{j=1}^{k} \chi_j \in \Psi_{N_A}(\sum_{j=1}^{k} q_j)$" in set Π remains valid for the lower bound on the total cost of problem $\mathbf{P}_G$. However, the condition "$\chi_k \in \Theta'(q_k) \cap \Theta''(q_k)$" is no longer valid for problem $\mathbf{P}_G$, since negative length cycles are allowed in the work package network. The value of $\tilde{p}$ generated by Procedure pMin is also invalid for problem $\mathbf{P}_G$.

Assuming that the given instance is feasible, Heuristic MergeTasks can be adapted to find feasible solutions for problem $\mathbf{P}_G$, with the elimination of any steps which create positive length cycles in the network of work packages. Hence, in Steps 2(ii) and 3(ii) of MergeTasks, the detection of "cycle" is replaced by the detection of "positive cycle." The implementation of Heuristic MergeTasks for problem $\mathbf{P}_G$ requires more computation time than it does for problem $\mathbf{P}_A$, since the network of work packages for $\mathbf{P}_G$ is not necessarily acyclic, as summarized in the following result.

Theorem 6 *The modified Heuristic MergeTasks delivers a feasible solution for problem* $\mathbf{P}_G$ *in* $O(\bar{n}^3 n|A|)$ *time.*

We note that when Heuristic MergeTasks is applied to problem $\mathbf{P}_A$, Steps 2(ii) and 3(ii) disallow a merger due to the formation of a cycle, and the logic behind the formation of cycles is straightforward. However, in problem $\mathbf{P}_G$, when two work packages are merged, the formation of positive cycles may be due to the creation of slack time within another work package. Consider an example with $\omega = 1$, $\xi = \alpha = 0$, $f(x) = h(x) = 0$ for $x \geq 0$,

$$g(x) = \begin{cases} 0, & \text{if } 0 \leq x \leq 6; \\ +\infty, & \text{if } x > 6, \end{cases}$$

$d = 10$, and a task network shown in Figure 6. Tasks 2, 3, 4, 5, and 6 are active tasks, while tasks 1 and 7 are dummy tasks. Suppose work packages $\{1\}$, $\{3\}$, $\{5\}$, $\{7\}$, and $\{2, 4, 6\}$ have been formed. Merging work packages $\{3\}$ and $\{5\}$ can reduce the total cost. However, Steps 2 and 3

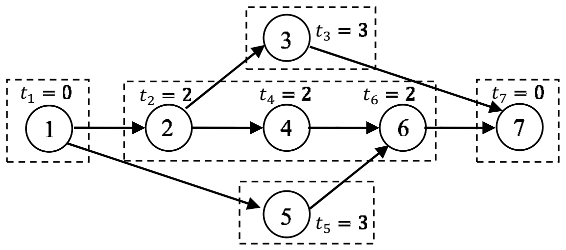


Figure 6: Work package formation in problem $\mathbf{P}_G$.

of MergeTasks disallow this merger, because merging these two work packages implies that tasks 3 and 5 must start processing simultaneously, which makes work package $\{2, 4, 6\}$ unable to process its tasks without inserting slack time between them. Hence, this merger creates a positive cycle in the generalized precedence network of work packages. A more complex heuristic that takes this characteristic of problem $\mathbf{P}_G$ into account should perform better than MergeTasks for this problem.

As a result of the above differences between $\mathbf{P}_A$ and $\mathbf{P}_G$, problem $\mathbf{P}_G$ requires substantially different solution and lower bound techniques that would form part of a separate paper.

6 Concluding Remarks

In this paper, we study how tradeoffs in work package sizing decisions affect project performance. Our work is apparently the first to evaluate these tradeoffs in order to optimize project performance. We study the objective of minimizing the total of several costs identified in the project management literature, subject to a project makespan deadline. For serial task networks, we develop a computationally efficient algorithm. For acyclic task networks, the cost minimization problem is unary *NP*-hard, hence we develop a heuristic method and a lower bound. A computational study shows that this heuristic routinely delivers near-optimal solutions that improve substantially on benchmarks from project management practice. Moreover, our computational results identify problem characteristics where the benefits from our work package sizing methodology are typically greatest. We demonstrate the improvement in project makespan that results from integrating scheduling considerations into decisions about work breakdown structure design. We also discuss several extensions and variants of the work package sizing problem.

Project companies can benefit from our work in several ways. First, they can achieve a better understanding of the mechanisms by which various factors related to work package sizing affect project performance. Second, they can estimate the cost increase from using equal target size work packages throughout the project. Third, they can use our solution procedures to develop work packages that significantly reduce project cost. Fourth, work package sizes identified by our model can be used to allocate resources more efficiently. Fifth, project companies can use our results to identify for which projects the cost reduction from using our methodology is likely to be most significant. A further benefit of our work is that companies will be motivated to obtain more accurate data about their work package sizing costs.

Besides the extensions and variants discussed in Section 5, several other topics remain open for future research. First, for particular project management applications, the factors that affect work package sizing may vary from the generic ones described here. It would be useful to identify such factors for those applications. Second, it would be valuable to conduct an empirical study to estimate the relative influence of the various factors we consider on project performance. Third, the crashing of tasks using a budget or other resources shared by tasks within the same work package is relevant. Fourth, it would be interesting to generalize our model to allow stochastic task times and study how task uncertainty affects work package sizing decisions. Extensions of our model in such directions would be valuable. Finally, we note that our work focuses on work package sizing, rather than on the sequence of criteria that are used to break the project down. It may be possible to develop a more complicated model that simultaneously optimizes the sequence of criteria used and the work package sizes. The difficulty of such an approach is that, as Globerson (1994) notes, the sequence of criteria interacts strongly with organizational and managerial culture. In conclusion, we hope that our work will encourage further research on this topic, with the potential to enhance the performance of many projects.

Acknowledgments

The authors thank the Area Editor Professor Chung-Piaw Teo, the associate editor, and the referees for their helpful comments and suggestions. The first author was supported in part by the Research Grants Council of Hong Kong under grant PolyU152163/15E. The second author was supported in part by the Summer Fellowship Program of the Fisher College of Business, The Ohio State University, and in part by the Faculty of Business of The Hong Kong Polytechnic University.

References

agilemanifesto.org. 2001. Manifesto for Agile Software Development. Available at www.agilemanifesto.org. Retrieved March 4, 2018.

Bai, Y., Y. Zhao, Y. Chen, L. Chen. 2009. Designing domain work breakdown structure (DWBS) using neural networks. *Lecture Notes in Computer Science* **5533** 1146–1153.

Bairstow, S. 2009. Quoted in “North-west: Aerospace,” *Financial Times*, October 27, 2009.

Baxendale, A.T. 1991. Management information systems: the use of work breakdown structure in the integration of time and cost. A. Bezelga, P. Brandon, eds. *Management, Quality and Economics in Building.* E & FN Spon, London, 14–23.

Beck, J.C., P. Prosser, E. Selensky. 2003. Vehicle routing and job shop scheduling: What's the difference? *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003)*, June 9–13, 2003, Trento, Italy, 267–276.

Brown, R.A. 1978. Probabilistic models of project management with design implications. *IEEE Transactions on Engineering Management* **EM-25**(2) 43–49.

Danilovic, M. 2006. Bring your suppliers into your projects—Managing the design of work packages in product development. *Journal of Purchasing and Supply Management* **12**(5) 246–257.

Dayanand, N., R. Padman. 2001. Project contracts and payment schedules: The client's problem. *Management Science* **47**(12) 1654–1667.

De, P., E.J. Dunne, J.B. Ghosh, C.E. Wells. 1997. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research* **45**(2) 302–306.

Deckro, R.F., J.E. Hebert, W.A. Verdini. 1992. Project scheduling with work packages. *Omega* **20**(2) 169–182.

Demeulemeester, E.L., W.S. Herroelen. 2002. *Project Scheduling: A Research Handbook*. Kluwer, Boston, MA.

Devi, T.R., V.S. Reddy. 2012. Work breakdown structure of the project. *International Journal of Engineering Research and Applications* **2**(2) 683–686.

DOD and NASA. 1962. *DOD and NASA Guide: PERT COST Systems Design*. United States Department of Defense, Washington, DC.

Dreyfus, S.E., A.M. Law. 1977. *The Art and Theory of Dynamic Programming*. Academic Press, Orlando, FL.

Eldin, N.N. 1991. Management of engineering/design phase. *Journal of Construction Engineering and Management* **117**(1) 163–175.

Elmaghraby, S.E., J. Kamburowski. 1992. The analysis of activity networks under generalized precedence relations (GPRs). *Management Science* **38**(9) 1245–1263.

Espina, D. 2011. Post to *Project Management Stack Exchange*, May 3. Retrieved March 4, 2018, http://pm.stackexchange.com/questions/2003/what-is-the-right-approach-to-work-package-size.

Gardner, G.R. 2006. Effective construction work packages. *AACE International Transactions* 13.1–13.10.

Garey, M.R., D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York.

Globerson, S. 1994. Impact of various work-breakdown structures on project conceptualization. *International Journal of Project Management* **12**(3) 165–171.

Golany, B., A. Shtub. 2001. Work breakdown structure. G. Salvendy, ed. *Handbook of Industrial Engineering: Technology and Operations Management*, 3rd edition. Wiley, New York, 1263–1280.

Goldratt, E.M. 1997. *The Critical Chain.* North River Press, Great Barrington, MA.

Golpayegani, S.A.H., B. Emamizadeh. 2007. Designing work breakdown structures using modular neural networks. *Decision Support Systems* **44**(1) 202–222.

Hall, N.G. 2012. Project management: Recent developments and research opportunities. *Journal of Systems Science and Systems Engineering* **21**(2) 129–143.

Hall, N.G. 2015. Further research opportunities in project management. C. Schwindt, J. Zimmermann, eds. *Handbook on Project Management and Scheduling.* Springer, Cham, Switzerland, 945–970.

Hall, N.G. 2016. Research and teaching opportunities in project management. A. Gupta, A. Capponi, eds. *Optimization Challenges in Complex, Networked and Risky Systems*, INFORMS, Catonsville, MD, 329–388.

Harpum, P. 2007. Project control. P.W.G. Morris, J.K. Pinto, eds. *The Wiley Guide to Project Control.* Wiley, Hoboken, NJ, 1–25.

Jaskowski, P., A. Sabotka. 2006. Multicriteria construction project scheduling method using evolutionary algorithm. *Operational Research: An International Journal* **6**(3) 283–297.

Jia, P., Q. Gao, X. Ji, T. Xu. 2014. Task decomposition method of R&D project based on product structure tree. *Journal of Software* **9**(7) 1894–1902.

Jung, Y., S. Woo. 2004. Flexible work breakdown structure for integrated cost and schedule control. *Journal of Construction Engineering and Management* **130**(5) 616–625.

Kastner, R. 2001. Post to *Project Management Central*, July 17. Retrieved March 4, 2018, http://www.projectmanagement.com/discussion-topic/2031/The-standard-for-work-packages-?sort=desc&pageNum=2.

Kelley, J.E., M.R. Walker. 1959. Critical-path planning and scheduling. *Proceedings of the Eastern Joint Computer Conference*, December 1–3, 1959, Boston, MA, 160–173.

Kennemer, B. 2002. Define and schedule tasks in Microsoft Project, September 18. Retrieved March 4, 2018, http://www.techrepublic.com/article/define-and-schedule-tasks-in-microsoft-project/.

Kerzner, H. 2017. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 12th edition. Wiley, Hoboken, NJ.

Klastorin, T. 2011. *Project Management: Tools and Trade-offs.* Pearson, Boston, MA.

Lee, J., W.-Y. Deng, W.-T. Lee, S.-J. Lee, K.-H. Hsu, S.-P. Ma. 2010. Integrating process and work breakdown structure with design structure matrix. *Journal of Advanced Computational Intelligence and Intelligent Informatics* **14**(5) 512–522.

Li, Y., N. Ren. 2013. Work breakdown and service access in cloud manufacturing environment based on knowledge sharing. *Journal of Convergence Information Technology* **8**(6) 771–778.

Luby, R.E., D. Peel, W. Swahl. 1995. Component-based work breakdown structure. *Project Management Journal* **26**(4) 38–43.

McVey, J. 2001. Post to *Project Management Central*, May 18. Retrieved March 4, 2018, http://www.projectmanagement.com/discussion-topic/2031/The-standard-for-work-packages-?sort=desc&pageNum=2.

Monnapappa, A. 2013. Project scope management—work package, January 25. Retrieved March 4, 2018, http://www.simplilearn.com/project-scope-management-work-package-article.

Mubarak, S. 2015. *Construction Project Scheduling and Control*, 3rd edition. Wiley, Hoboken, NJ.

Nicholas, J.M., H. Steyn. 2017. *Project Management for Engineering, Business and Technology*, 5th edition. Routledge, London.

Popescu, C.M., C. Charoenngam. 1995. *Project Planning, Scheduling, and Control in Construction: An Encyclopedia of Terms and Applications.* Wiley, New York.

Potts, C.N., K.R. Baker. 1989. Flow shop scheduling with lot streaming. *Operations Research Letters* **8**(6) 297–303.

Powell, M. 2010. Project control. P. Harpum, ed. *Portfolio, Program, and Project Management in the Pharmaceutical and Biotechnology Industries.* Wiley, Hoboken, NJ, 101–134.

Project Management Institute. 2008. *Should You Be Teaching Project Management?* Project Management Institute, Newtown Square, PA. Retrieved March 4, 2018, http://www.mosaicprojects.com.au/PDF/Why_teach_PM.pdf.

Project Management Institute. 2013. *Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 5th edition. Project Management Institute, Newtown Square, PA.

Raz, T., S. Globerson. 1998. Effective sizing and content definition of work packages. *Project Management Journal* **29**(4) 17–23.

Ren, N., J. Xue, Y. Wang. 2013. The task decoupling study on work breakdown structure of complex product. *Applied Mechanics and Materials* **411–414** 2486–2491.

Roche, K. 2010. Post to *Project Management Central*, August 26. Retrieved March 4, 2018, http://www.projectmanagement.com/discussion-topic/2031/The-standard-for-work-packages-?sort=desc&pageNum=1.

Trietsch, D., K.R. Baker. 1993. Basic techniques for lot streaming. *Operations Research* **41**(6) 1065–1076.

Vanhoucke, M., J. Coelho, J. Batselier. 2016. An overview of project data for integrated project management and control. *The Journal of Modern Project Management* **3**(3) 6–21.

Vanhoucke, M., J. Coelho, D. Debels, B. Maenhout, L.V. Tavares. 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research* **187**(2) 511–524.

Wang, J.F., J.H. Liu, Y.F. Zhong. 2005. A novel ant colony algorithm for assembly sequence planning. *International Journal of Advanced Manufacturing Technology* **25**(11–12) 1137–1143.

Wilmot, J. 2011. Post to *Project Management Stack Exchange*, May 3. Retrieved March 4, 2018, http://pm.stackexchange.com/questions/2003/what-is-the-right-approach-to-work-package-size.

Wysocki, R.K. 2014. *Effective Project Management: Traditional, Agile, Extreme*, 7th edition. Wiley, Indianapolis, IN.

Yang, K.K., F.B. Talbot, J.H. Patterson. 1992. Scheduling a project to maximize its net present value: An integer programming approach. *European Journal of Operational Research* **64**(2) 188–198.

Yu, T.-L., D.E. Goldberg, K. Sastry, C.F. Lima, M. Pelikan. 2009. Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary Computation* **17**(4) 595–626.