Task Scheduling with Progress Control

Chung-Lun Li

Department of Logistics and Maritime Studies,
The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
Email: chung-lun.li@polyu.edu.hk

Weiya Zhong[1]

Department of Business Administration,
School of Management, Shanghai University,
Shanghai, People's Republic of China
Email: wyzhong@i.shu.edu.cn

February 18, 2017
Revised June 12, 2017
Final Version: September 8, 2017

[1]Corresponding author

Task Scheduling with Progress Control

Abstract

Tasks with long duration often face the requirement of having their progress to be reported periodically to process controllers. Under this requirement, working teams that simultaneously process multiple tasks need to schedule their work carefully in order to demonstrate satisfactory progress of each unfinished task. We present a single-machine scheduling model that reflects this requirement. Our model has multiple milestones at which the tasks are penalized if their progress is below satisfactory levels. We develop polynomial solution methods for the general case with convex nonlinear penalty functions and for the special case with linear penalty functions. Extensions of our model are also discussed.

Keywords: Scheduling; multitasking; polynomial time algorithm; progress milestones

February 18, 2017
Revised June 12, 2017
Final Version: September 8, 2017

1 Introduction

Tasks with long duration often face the requirement of having their progress to be reported periodically to process controllers. For example, office administrators need to report the progress of their ongoing tasks to their supervisors on a daily or weekly basis. Academics need to submit progress reports of their research projects to funding agencies annually. Professional service providers need to report the progress of their services to their clients on certain predetermined dates. To fulfill this requirement, working teams that simultaneously process multiple tasks need to schedule their work carefully so as to demonstrate satisfactory progress of each unfinished task. To do so, a task may need to be split into subtasks. After completing one subtask, the working team needs to switch to a subtask of a different task, and so on. This is a form of "multitasking" which is inevitable under such requirement.

Hall et al. (2015) have identified five principal motivations for multitasking in administrative and business processes: (i) a need to feel or appear productive; (ii) a need to demonstrate progress on different tasks or treat task owners equitably; (iii) anxiety about the processing requirements of waiting tasks; (iv) a need for variety in work; and (v) interruption by routine scheduled activities. This paper specifically deals with motivation (ii). We model this motivation as a single machine scheduling problem with job preemption, and the performance measure is not based on the completion time of the given jobs but depends on the progress of each job at some given milestones.

1.1 Literature review

Different mathematical models for operations scheduling with multitasking have been developed and analyzed by various researchers. Hall et al. (2015) present a scheduling model with multitasking in which all unfinished tasks can interrupt a primary task when the primary task is being processed, and a switching time is incurred when an interruption occurs. They analyze the computational complexity of their model under different scheduling objectives. Under this model, the cost increase and value gained due to multitasking are analyzed. Sum and Ho (2015) apply the model introduced by Hall et al. (2015) and consider the total completion time and total weighted completion time objectives. They derive the expected optimal total cost of the multitasking model

when the job processing times are uniformly distributed, and compare it with the expected optimal total cost when multitasking is absent, so as to measure the effect of multitasking. Hall et al. (2016) develop another scheduling model with multitasking for the "a need for variety in work" motivation and a model for the "interruption by routine scheduled activities" motivation. They analyze the computational complexity of these two models and develop algorithms for them. Zhu et al. (2017a,b,c) study several extensions of Hall et al.'s (2015) multitasking model by introducing deterioration effect and rate-modifying activities. Liu et al. (2017) extend Hall et al.'s (2015) model by introducing a common due date related objective. Our work differs from these works in that we consider specifically the motivation of multitasking where there is a need to demonstrate progress on different tasks or to treat the task owners equitably, and we develop and analyze a mathematical model for this motivation.

Our work is related to job splitting in machine scheduling, since the key decision of our model is to split the given tasks into subtasks and to assign the subtasks to different time intervals. Job splitting has been studied by many researchers with a focus on splitting jobs into sublots so that the sublots can be processed simultaneously on parallel, uniform, or unrelated machines; see, for example, Serafini (1996), Xing and Zhang (2000), Yalaoui and Chu (2003), Logendran and Subur (2004), and Correa et al. (2015). Lot streaming, which refers to splitting production lots into sublots so as to accelerate the flow of the products through a multiple-stage production system, has also been studied extensively; see Chang and Chiu (2005) and Cheng et al. (2013) for comprehensive reviews. Some works have considered job splitting in a single-machine setting. Eynan and Li (1997) consider the splitting of a single job with learning effects, where the net present value collected by the delivery of each sublot is maximized. Serin and Kayaligil (2003) consider two extensions of Eynan and Li's model. The first extension includes a revenue requirement, while the second extension involves multiple item types and the sequencing of shipments for different customers. To the best of our knowledge, none of the works on operations scheduling with job splitting decisions has considered an objective function that measures the progress of the jobs at different milestones.

Models with progress milestones have been studied in job shop scheduling. In a job shop scheduling problem, each job has a technological sequence of machines to be processed, and the processing of a job on a machine is called an operation. The milestones studied in these job shop scheduling models are called "operation due dates" (see Philipoom et al. 1989), where a due date is imposed on every operation of a job. Keskinocak and Tayur (2004) provide an overview of scheduling research with operation due dates. Our model differs from these job shop scheduling models in that the milestones in our model are not associated with different operations of a multistage system. Instead, the milestones in our model are used to monitor the amount of each task that has been processed.

Models with multiple milestones have also appeared in project scheduling, where the activities of a project have predetermined precedence relationships, and progress control is linked to milestones that are defined by completion of certain activities. These include models that use predetermined progress milestones to represent the completion of different phases of a project (see, e.g., Ghoddousi et al. 2017), models that minimize the tardiness of certain milestone activities (see, e.g., Choi and Park 2015), and models that have payment milestones and cash flow considerations (see, e.g., Dayanand and Padman 2001). Unlike these project scheduling models, our model does not consider precedence relationships of tasks. It focuses on task splitting decisions that maximize the tasks' performance at their progress milestones.

1.2 Model description

Our problem can be described mathematically as follows. Using the terminology in traditional scheduling, we refer to the working team as a "machine" and the tasks as "jobs." We are given n jobs $J_1, J_2, \ldots, J_n$ to be scheduled on a single machine. Each job J_i has a processing time $p_i \in \mathbb{Z}^+$ and q_i milestones $\tau_{i1}, \tau_{i2}, \ldots, \tau_{iq_i} \in \mathbb{Z}^+ \cup \{0\}$, where $0 \le \tau_{i1} \le \tau_{i2} \le \cdots \le \tau_{iq_i}$, and $\mathbb{Z}^+$ is the set of all positive integers. We refer to τ_{ik} as the k^{th} milestone of job J_i. All jobs are available for processing at time 0 and can be split into subjobs with integer processing times. For any $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q_i$, let $x_{ik} \in \mathbb{Z}^+ \cup \{0\}$ be the amount of time the machine spends on processing J_i during the time interval $[0, \tau_{ik}]$. Job J_i incurs a penalty of $f_{ik}(x_{ik}) \in \mathbb{Z}^+ \cup \{0\}$ at the milestone τ_{ik},

where f_{ik} is a nonincreasing convex function such that $f_{ik}(x) = 0$ for $x \geq p_i$, and function f_{ik} can be evaluated in constant time. The objective is to determine x_{ik} for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q_i$ such that $0 \leq x_{i1} \leq x_{i2} \leq \cdots \leq x_{iq_i}$ and that the total penalty incurred, $\sum_{i=1}^{n} \sum_{k=1}^{q_i} f_{ik}(x_{ik})$, is minimized. We denote this problem as problem **P**. For notational convenience, we denote $P = \sum_{i=1}^{n} p_i$ and $Q = \sum_{i=1}^{n} q_i$. Table 1 provides a summary of the notation used in this model.

In this model, the quantity x_{ik} is the *cumulative* amount of time that job J_i is processed by the k^{th} milestone. The penalty of job J_i at the k^{th} milestone, $f_{ik}(x_{ik})$, is a function of this cumulative processing time. The monotonicity of penalty function f_{ik} implies that getting more work done for J_i by time τ_{ik} is no worse than getting less work done. The convexity of penalty function f_{ik} implies that the drop in penalty, or equivalently the additional benefit gained, by having more work done for J_i by time τ_{ik} diminishes as the percentage of J_i completed becomes higher.

Note that the performance measure of this model is rather general. It allows each job i to have its own penalty functions $f_{i1}, f_{i2}, \ldots, f_{iq_i}$. It also allows each milestone of a job to have its own penalty function. For example, if we wish to set a satisfactory progress level of J_i at the k^{th} milestone and penalize J_i only when this progress level is not met, then we can define function f_{ik} in such a way that $f_{ik}(x) = 0$ for $x \geq \bar{x}_{ik}$, where $\bar{x}_{ik}$ is a given parameter representing the targeted amount of J_i to be completed by τ_{ik}. If we expect J_i to complete all its work by its final milestone,

Table 1: Summary of notation

Notation for problem **P**:	
n	number of jobs
J_i	the i^{th} job
p_i	processing time of J_i
q_i	number of milestones for J_i
τ_{ik}	the k^{th} milestone of J_i
x_{ik}	amount of time the machine spends on J_i during the time interval $[0, \tau_{ik}]$ (decision variable)
$f_{ik}(\cdot)$	penalty function for J_i at milestone τ_{ik}
P	$= \sum_{i=1}^{n} p_i$
Q	$= \sum_{i=1}^{n} q_i$
Notation for special case $\mathbf{P}_1$:	
q	number of milestones
τ_k	the k^{th} milestone for each job

then we can define $\bar{x}_{iq_i} = p_i$. If we wish to put tighter control on jobs at their final milestones, then we may set the function values of f_{iq_i} to be larger than the function values of $f_{i1}, f_{i2}, \ldots, f_{i,q_i-1}$.

The rest of the paper is organized as follows. In Section 2, we first transform problem **P** into a simpler problem and then present efficient solution methods for solving the transformed problem. In Section 3, we consider a special case of problem **P** and discuss how this special case can be solved more efficiently. In Section 4, two extensions of problem **P** are discussed. Section 5 concludes this study and suggests some future research directions.

2 Solution Methods

We let $\mathbf{P}_1$ denote the special case of problem **P** in which all jobs have the same milestones; that is, $q_1 = q_2 = \cdots = q_n$ and $\tau_{1k} = \tau_{2k} = \cdots = \tau_{nk}$ for all k. In problem $\mathbf{P}_1$, we denote $q = q_1 = q_2 = \cdots = q_n$ and $\tau_k = \tau_{1k} = \tau_{2k} = \cdots = \tau_{nk}$ for $k = 1, 2, \ldots, q$ (see Table 1).

Given any instance of problem **P**, we can transform it into an instance of problem $\mathbf{P}_1$ by changing the parameters of the jobs as follows. Let $\hat{p}_i = p_i$ for $i = 1, 2, \ldots, n$. Arrange the milestones $\tau_{11}, \tau_{12}, \ldots, \tau_{1q_1}; \tau_{21}, \tau_{22}, \ldots, \tau_{2q_2}; \ldots; \tau_{n1}, \tau_{n2}, \ldots, \tau_{nq_n}$ in ascending order and remove all duplicated ones. Let $(\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_q)$ denote the sorted list of these distinct milestones. For $k = 1, 2, \ldots, q$ and $i = 1, 2, \ldots, n$, if $\hat{\tau}_k$ is a milestone of J_i in problem **P**, say $\hat{\tau}_k = \tau_{ij}$ for some j, then define $\hat{f}_{ik}(x) = f_{ij}(x)$ for $x \geq 0$. If $\hat{\tau}_k$ is not a milestone of J_i in problem **P**, then define $\hat{f}_{ik}(x) = 0$ for $x \geq 0$. Solving problem $\mathbf{P}_1$ with n jobs, q milestones, and input parameters $\hat{p}_i$, $\hat{\tau}_k$, and $\hat{f}_{ik}$ $(i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, q)$ optimally yields the optimal solution to the original problem **P**. Therefore, in this section, we focus on solving problem $\mathbf{P}_1$.

Denote $\tau_0 = 0$ and $x_{i0} = 0$ for $i = 1, 2, \ldots, n$. Note that for $k = 1, 2, \ldots, q$, given $x_{i,k-1}$, the penalty function f_{ik} depends on how much time is allotted to each job during the time interval $[\tau_{k-1}, \tau_k]$ and is independent of the processing sequence of the subjobs assigned to this time interval.

2.1 Convex integer programming formulation

One straightforward solution approach for problem $\mathbf{P}_1$ is to solve the problem as a nonlinear integer program. The nonlinear integer programming formulation is given as follows:

$$\begin{aligned}
\mathbf{P}_1' : \text{ Minimize } & \quad \textstyle\sum_{i=1}^{n} \sum_{k=1}^{q} f_{ik}(x_{ik}) && && (1)\\
\text{subject to } & \quad \textstyle\sum_{i=1}^{n} x_{ik} \leq \tau_k, && k = 1, 2, \ldots, q && (2)\\
& \quad x_{i,k-1} - x_{ik} \leq 0, && i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, q && (3)\\
& \quad x_{ik} \text{ is integer}, && i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, q && (4)
\end{aligned}$$

Objective function (1) is a separable convex function. Constraint (2) ensures that the amount of time that the machine spends on jobs $J_1, J_2, \ldots, J_n$ during the time interval $[0, \tau_k]$ is no more than τ_k. Constraint (3) ensures that $0 \leq x_{i1} \leq x_{i2} \leq \cdots \leq x_{iq}$ for all i. Note that constraints (2)–(4) do not prohibit the machine from processing J_i for more than p_i time units. However, because $f_{ik}(x) = 0$ for $x \geq p_i$, there exists an optimal solution to $\mathbf{P}_1'$ in which x_{ik} is no greater than p_i for all $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$.

Let $I_{q\times q}$ denote the $q \times q$ identity matrix. Let $B_{q\times q}$ denote the following $q \times q$ matrix:

$$B_{q\times q} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & -1 & 1 \end{pmatrix}.$$

Let A denote the following $(q + nq) \times nq$ matrix:

$$A = \begin{pmatrix} -I_{q\times q} & -I_{q\times q} & \cdots & -I_{q\times q} \\ B_{q\times q} & 0 & \cdots & 0 \\ 0 & B_{q\times q} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_{q\times q} \end{pmatrix},$$

which comprises n copies of submatrix $-I_{q\times q}$, n copies of submatrix $B_{q\times q}$, and n^2-n copies of the $q\times q$ zero submatrix. Problem $\mathbf{P}_1'$ can be written as $\min\{\sum_{i=1}^{n}\sum_{k=1}^{q} f_{ik}(\mathbf{x}) \mid A\mathbf{x}\geq \mathbf{b},\ \mathbf{x} \text{ integer}\}$, where $\mathbf{x}$ is the nq-vector of the x_{ik} variables and $\mathbf{b}$ is an integer $(q+nq)$-vector.

Theorem 1 *Matrix A is totally unimodular.*

Proof. Let A_j^i denote the $(q(i-1)+j)^{\text{th}}$ column of A for $i=1,2,\ldots,n$ and $j=1,2,\ldots,q$. A necessary and sufficient condition for a matrix with entries 0, 1, or -1 to be totally unimodular is that each collection of columns of the matrix can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only 0, 1, and -1 (see Schrijver 1986, p. 269). In the following, we show that matrix A satisfies this condition.

Clearly, each entry of matrix A is 0, 1, or -1. Let Γ be any collection of columns of A. We partition the set Γ into two groups Γ_1 and Γ_2 using the following method. We start off with $|\Gamma|$ groups, where each group contains one column from Γ. For $i=1,2,\ldots,n$ and $j=1,2,\ldots,q-1$, if both A_j^i and A_{j+1}^i are in Γ, then merge the group that contains column A_j^i and the group that contains A_{j+1}^i into a single group. Let $G_1,G_2,\ldots,G_s$ denote the column groups obtained.

For $k=1,2,\ldots,s$, let V_k denote the $(q+nq)$-vector obtained by summing all the columns in G_k, and let $\bar{V}_k$ denote the q-vector obtained by truncating the last nq entries of V_k. Note that the above grouping operation only merges adjacent columns of matrix A, and does not merge A_q^i with the A_1^{i+1} for any $i=1,2,\ldots,n-1$. Hence, each entry of $\bar{V}_k$ is either 0 or -1, and the -1 entries are consecutive. Let $\bar{V}=(\bar{V}_1,\bar{V}_2,\ldots,\bar{V}_s)$, which is a $q\times s$ matrix. Then, $-\bar{V}$ is an interval matrix, which is totally unimodular (Schrijver 1986, p. 279). This implies that $\bar{V}$ is also totally unimodular. By the above necessary and sufficient condition, the columns of matrix $\bar{V}$ can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only 0, 1, and -1. Let $\{\bar{V}_{\pi(1)},\bar{V}_{\pi(2)},\ldots,\bar{V}_{\pi(h)}\}$ be the first part and $\{\bar{V}_{\pi(h+1)},\bar{V}_{\pi(h+2)},\ldots,\bar{V}_{\pi(s)}\}$ be the second part, where $(\pi(1),\pi(2),\ldots,\pi(s))$ is a permutation of $(1,2,\ldots,s)$ and $0\leq h\leq s$.

Now, we split the columns in Γ into two parts as follows. If a column belongs to a column

group G_k and $k \in \{\pi(1), \pi(2), \ldots, \pi(h)\}$, then this column belongs to the first part Γ_1; otherwise, it belongs to the second part Γ_2. Then, for any $l = 1, 2, \ldots, q$, the sum of the l^{th} entry of the columns in Γ_1 minus the sum of the l^{th} entry of the columns in Γ_2 is 0, 1, or -1. Next, we consider any $l = q+1, q+2, \ldots, q+nq$. There are two possible cases. Case 1: There is zero or one column in the set Γ where the l^{th} entry is nonzero. In this case, the sum of the l^{th} entry of the columns in Γ_1 minus the sum of the l^{th} entry of the columns in Γ_2 is 0, 1, or -1. Case 2: There are exactly two columns in the set Γ where the l^{th} entry is nonzero. Then, the l^{th} entry of one of these two columns is -1, and the l^{th} entry of the other column is 1. Furthermore, these two columns must be adjacent columns of matrix A and belong to the same group G_k, and hence either both of them belong to Γ_1 or both belong to Γ_2. Thus, in this case, the sum of the l^{th} entry of the columns in Γ_1 minus the sum of the l^{th} entry of the columns in Γ_2 is 0. Therefore, by the above necessary and sufficient condition, matrix A is totally unimodular. ∎

Hochbaum and Shanthikumar (1990) have presented an algorithm for the nonlinear integer program $\min\{F(\mathbf{x}) \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \text{ integer}\}$, where F is a separable convex function. Their algorithm, which solves a series of linear programs iteratively, has a polynomial running time when A is totally unimodular. Specifically, let $T(\nu, \mu, \Delta)$ be the complexity of solving the linear program $\min\{\mathbf{cx} \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$ with a $\mu \times \nu$ coefficient matrix A, where Δ is the bound on the absolute value of a subdeterminant of A. If A is a totally unimodular matrix (thus $\Delta = 1$) and F is a separable convex function, then the nonlinear integer program $\min\{F(\mathbf{x}) \mid A\mathbf{x} \geq \mathbf{b},\ \mathbf{x} \text{ integer}\}$ can be solved in $O\big(\lceil \log(\frac{\mu}{\nu} \|\mathbf{b}\|_\infty) \rceil \cdot T(4\nu^2, \mu, 1)\big)$ time via Hochbaum and Shanthikumar's algorithm. In problem $\mathbf{P}_1'$, $\mu = q + nq$, $\nu = nq$, and $\|\mathbf{b}\|_\infty \leq P$. Therefore, problem $\mathbf{P}_1'$, and hence problem $\mathbf{P}_1$, can be solved in $O(T(4n^2q^2, q+nq, 1) \cdot \log P)$ time.

2.2 Convex cost network flow formulation

Next, we show that problem $\mathbf{P}_1$ can be modeled as a minimum convex cost network flow problem and solved more efficiently. For ease of presentation, we assume that $\tau_q = P$. This assumption is made without loss of generality. This is because if $\tau_q > P$, then we can add a dummy job J_{n+1}

to the problem, where $p_{n+1} = \tau_q - P$ and $f_{n+1,k} \equiv 0$ for $k = 1, 2, \ldots, q$. On the other hand, if $\tau_q < P$, then we can add a dummy milestone $\tau_{q+1} = P$, where the penalty function of each job at this dummy milestone is zero.

Consider the network diagram depicted in Figure 1. In this network, each arc is associated with an ordered pair $(f(x), u)$, where $f(x)$ is the cost incurred when the arc has x units of flow, and u is the capacity of the arc. Note that the objective value of problem $\mathbf{P}_1$ is independent of the processing sequence of the jobs after time τ_q. Clearly, there exists an optimal solution to $\mathbf{P}_1$ in

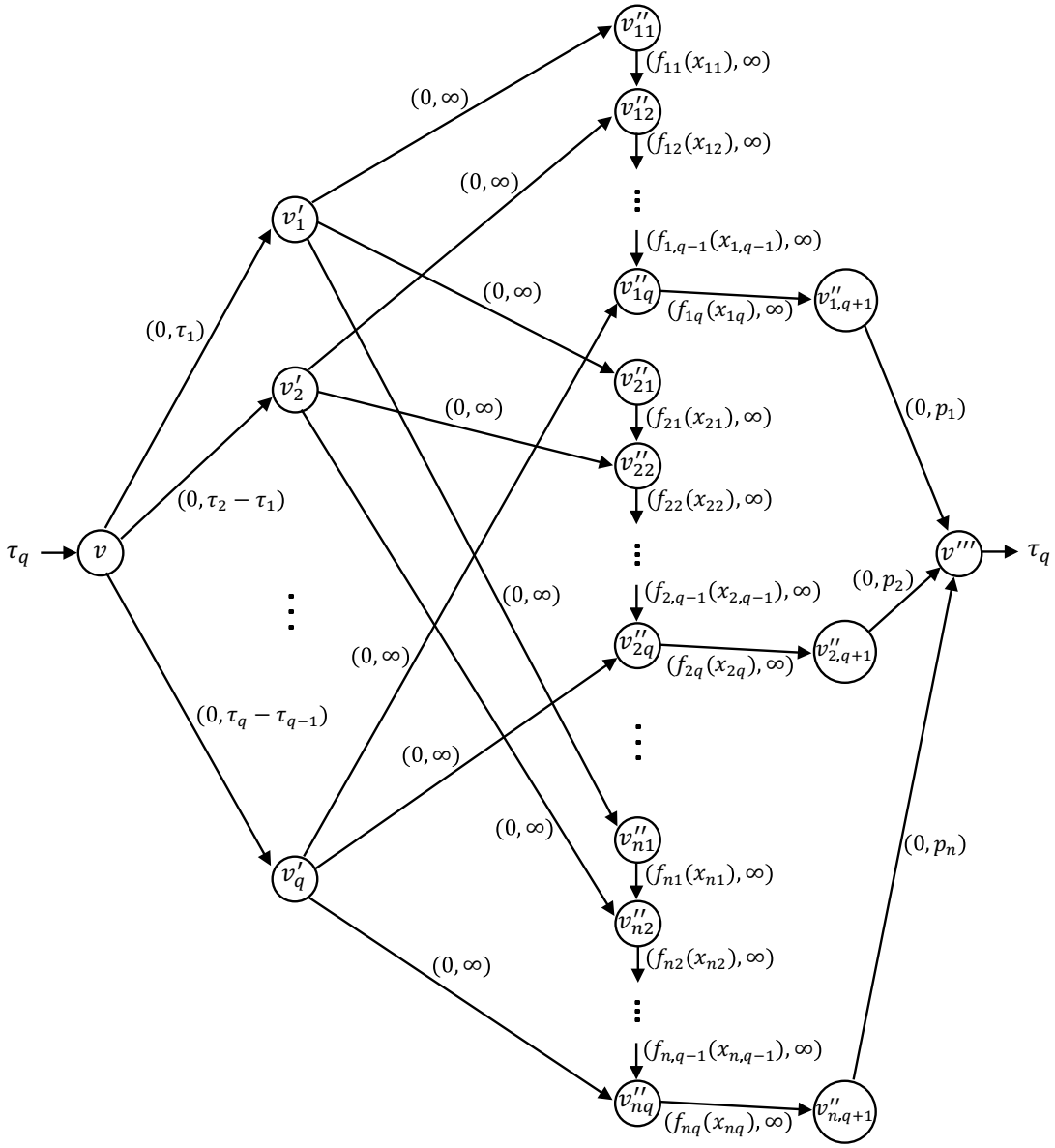

Figure 1: Network diagram for problem $\mathbf{P}_1$.

which there is no machine idle time during the time period $[0, \tau_q]$. Thus, it suffices to consider the job schedule in the time interval $[0, \tau_q]$. Hence, in Figure 1, the source vertex v has an inflow of τ_q, and the sink vertex v''' has an outflow of τ_q. For $k = 1, 2, \ldots, q$, there is an arc $v \rightarrow v'_k$ with capacity $\tau_k - \tau_{k-1}$. The flow along this arc represents the total amount of job processing time assigned to the time interval $[\tau_{k-1}, \tau_k]$. For $k = 1, 2, \ldots, q$ and $i = 1, 2, \ldots, n$, there is an arc $v'_k \rightarrow v''_{ik}$. The flow along this arc represents the amount of time allotted to J_i during the time interval $[\tau_{k-1}, \tau_k]$. For $k = 1, 2, \ldots, q$ and $i = 1, 2, \ldots, n$, there is an arc $v''_{ik} \rightarrow v''_{i,k+1}$. The flow along this arc represents the amount of time allotted to J_i during the time interval $[0, \tau_k]$. Thus, the flow along this arc is x_{ik}, and the cost incurred by the flow is $f_{ik}(x_{ik})$. For $i = 1, 2, \ldots, n$, there is an arc $v''_{i,q+1} \rightarrow v'''$ with capacity p_i, which ensures that the total processing time of J_i during time interval $[0, \tau_q]$ is no more than p_i.

The convex cost network flow problem with integral flow can be solved in $O((\mu \log U)S(\nu, \mu, C))$ time, where ν is the number of vertices, μ is the number of arcs, U is the largest supply/demand or finite arc capacity, C is the largest arc cost, and $S(\nu, \mu, C)$ is the time required for solving a shortest path problem on the network (Ahuja et al. 1993, p. 560). In the network depicted in Figure 1, $\nu = O(nq)$, $\mu = O(nq)$, and $U = O(P)$. The best known strongly polynomial time bound for $S(\nu, \mu, C)$ is $O(\mu + \nu \log \nu)$ (Ahuja et al. 1993, p. 123). Hence, problem $\mathbf{P}_1$ can be solved in $O(n^2 q^2 \log nq \log P)$ time. This implies that problem $\mathbf{P}$ can be solved in $O(n^2 Q^2 \log Q \log P)$ time.

3 The Case with Linear Penalty Functions

In this section, we consider the special case of problem $\mathbf{P}$ where the penalty functions are linear. Specifically, we consider the case where function f_{ik} is of the form

$$f_{ik}(x) = \lambda_{ik}(p_i - x)$$

for $0 \leq x \leq p_i$, where $\lambda_{ik} \geq 0$, for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$. Same as Section 2.2, we first transform the given problem into problem $\mathbf{P}_1$ in which all jobs have the same milestones and assume that $\tau_q = P$.

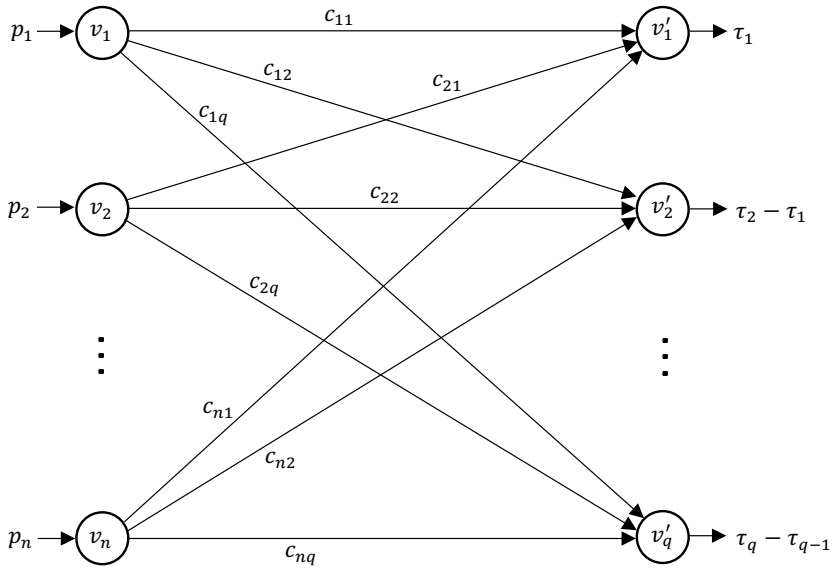

Figure 2: Network diagram for the case with linear penalty functions.

Note that for this special case, the network diagram in Figure 1 becomes a linear cost network, where the unit cost of arc $v''_{ik} \to v''_{i,k+1}$ is $-\lambda_{ik}$ for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$, and the objective value of this linear minimum cost flow problem differs from that of the original problem by a constant $\sum_{i=1}^{n} \sum_{k=1}^{q} \lambda_{ik} p_i$. The linear minimum cost flow model is solvable in $O((\mu \log \nu)(\mu + \nu \log \nu))$ time, where ν is the number of vertices and μ is the number of arcs (Ahuja et al. 1993, p. 395). Thus, this special case can be solved via this approach in $O(n^2 q^2 \log^2 nq)$ time. In the following, we propose a more efficient solution method for this special case.

Let

$$c_{ik} = -\sum_{j=k}^{q} \lambda_{ij}.$$

Consider the $n \times q$ transportation problem with a network diagram depicted in Figure 2. For $i = 1, 2, \ldots, n$, the supply at node v_i is p_i, which represents the total number of units of J_i to be processed in the time interval $[0, \tau_q]$. For $k = 1, 2, \ldots, q$, the demand at node v'_k is $\tau_k - \tau_{k-1}$, which represents the number of units available for processing during the time interval $[\tau_{k-1}, \tau_k]$. A flow of y_{ik} units along the arc $v_i \to v'_k$ represents the allocation of y_{ik} time units in the time interval $[\tau_{k-1}, \tau_k]$ to job J_i. Since all supply and demand parameters of this transportation problem are integers, there exists an optimal solution with integer flows.

Consider any feasible solution to this transportation problem with integer flows $\{y_{ik} \mid i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, q\}$. For $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$, we let $x_{ik} = \sum_{j=1}^{k} y_{ij}$. Clearly, x_{ik} satisfies constraints (2)–(4). Thus, $\{x_{ik} \mid i = 1, 2, \ldots, n;\ k = 1, 2 \ldots, q\}$ is a feasible solution to problem $\mathbf{P}_1$. The total cost of this transportation problem solution is

$$\sum_{i=1}^{n}\sum_{k=1}^{q} c_{ik}y_{ik} = -\sum_{i=1}^{n}\sum_{k=1}^{q}\sum_{j=k}^{q} \lambda_{ij}y_{ik} = -\sum_{i=1}^{n}\sum_{k=1}^{q} \lambda_{ik}\sum_{j=1}^{k} y_{ij} = -\sum_{i=1}^{n}\sum_{k=1}^{q} \lambda_{ik}x_{ik} = \sum_{i=1}^{n}\sum_{k=1}^{q} f_{ik}(x_{ik}) - K,$$

where $K = \sum_{i=1}^{n}\sum_{k=1}^{q} \lambda_{ik}p_i$. Hence, minimizing the total cost of this transportation problem yields a feasible solution of the given problem $\mathbf{P}_1$ with a minimum total penalty.

The $\nu_1 \times \nu_2$ transportation problem with κ feasible arcs can be solved in $O((\nu_1 \log \nu_1)(\kappa + \nu_2 \log \nu_2))$ time using Kleinschmidt and Schannath's (1995) algorithm when $\nu_1 \geq \nu_2$. Thus, $\mathbf{P}_1$ can be solved in $O(\max\{(n \log n)(nq + q \log q), (q \log q)(nq + n \log n)\})$ time. Since $n \leq Q$ and $q \leq Q$, problem $\mathbf{P}$ with linear penalty functions can be solved in $O(nQ^2 \log Q)$ time.

4 Extensions

In this section, we consider two generalizations of problem $\mathbf{P}$.

4.1 Jobs release dates

In problem $\mathbf{P}$, it is assumed that all tasks are available for processing at time 0. A more general model would allow positive job release dates. For $i = 1, 2, \ldots, n$, let r_i be the release date of job J_i before which J_i cannot be processed, where $r_i \in \mathbb{Z}^+ \cup \{0\}$ and $r_i < \tau_{i1}$. We denote this extended problem as $\mathbf{P}_2$.

It is not difficult to modify the solution methods presented in Sections 2 and 3 for solving problem $\mathbf{P}_2$. First, the convex integer programming formulation $\mathbf{P}'_1$ can be modified for problem $\mathbf{P}_2$ as follows. For each $i = 1, 2, \ldots, n$, we add a milestone r_i to the problem. Then, we transform the problem into problem $\mathbf{P}_1$ and set up the convex integer program $\mathbf{P}'_1$. Let $(\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_q)$ be the sorted list of distinct milestones in problem $\mathbf{P}_1$. In the convex integer programming formulation $\mathbf{P}'_1$, we add a new constraint "$x_{ik} = 0$" if $\hat{\tau}_k = r_i$, for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$. This new

constraint prohibits x_{ik} from having a positive value if the k^{th} milestone is the release date of J_i. It is easy to show that Theorem 1 remains valid after adding these constraints to $\mathbf{P}'_1$. Hence, problem $\mathbf{P}_2$ can be solved optimally in polynomial time using the modified convex integer program.

Next, we show that the convex cost network flow formulation in Section 2.2 can also be modified for problem $\mathbf{P}_2$. Similar to the method above, we add a milestone r_i to each job J_i and transform the problem into problem $\mathbf{P}_1$. Let $(\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_q)$ be the sorted list of distinct milestones. We set up the nonlinear cost network as shown in Figure 1. For $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$, we change the capacity of the arc $v''_{ik} \rightarrow v''_{i,k+1}$ to zero if $\hat{\tau}_k = r_i$, so that $x_{ik} = 0$ if the k^{th} milestone is the release date of J_i. Since the optimal solution to problem $\mathbf{P}_2$ no longer possesses the "no machine idle time" property, we add an arc $v \rightarrow v'''$ with zero cost and infinite capacity so that the total amount of flow through the network via other arcs is allowed to be less than $\min\{P, \tau_q\}$. Note that for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$, because f_{ik} is a nonincreasing function, the cost of arc $v''_{ik} \rightarrow v''_{i,k+1}$ is nonincreasing in the flow along this arc. Thus, more cost saving can be obtained by assigning flows to arc $v''_{ik} \rightarrow v''_{i,k+1}$ than assigning flows to arc $v \rightarrow v'''$, unless the flow along arc $v''_{ik} \rightarrow v''_{i,k+1}$ has reached a level x such that $f_{ik}(x) = 0$. Hence, there exists an optimal solution in which flows are assigned to arc $v \rightarrow v'''$ only when they cannot be assigned to the other paths connecting v and v'''. Solving this modified minimum cost network flow problem yields an optimal solution to problem $\mathbf{P}_2$.

A similar approach may be used to modify the transportation problem formulation in Section 3 for problem $\mathbf{P}_2$ when the penalty functions are linear. We add a milestone r_i to each job J_i, transform the problem into problem $\mathbf{P}_1$, and then set up the transportation problem as shown in Figure 2. For each arc $v_i \rightarrow v'_k$, we change the unit transportation cost to $+\infty$ if $\hat{\tau}_k \leq r_i$, so that no time unit in the time interval $[0, r_i]$ can be assigned to job J_i. Since the optimal solution to problem $\mathbf{P}_2$ no longer possesses the "no machine idle time" property, we add a dummy supply node v_0 and a dummy demand node v'_0 to the transportation network, where the supply at node v_0 and the demand at node v'_0 are both equal to some large number M. The arcs $v_0 \rightarrow v'_0$, $v_0 \rightarrow v'_k$ (for $k = 1, 2, \ldots, q$), and $v_i \rightarrow v'_0$ (for $i = 1, 2, \ldots, n$) have zero costs. Note that $c_{ik} \leq 0$ for

$i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, q$. Thus, more cost saving can be obtained by assigning flows to arc $v_i \rightarrow v'_k$ $(i, k \neq 0)$ than assigning flows to arc $v_0 \rightarrow v'_k$ or $v_0 \rightarrow v'_0$. Hence, there exists an optimal solution in which flows are assigned to arcs $v_0 \rightarrow v'_k$ and $v_0 \rightarrow v'_0$ only when they cannot be assigned to other arcs. Solving this modified transportation problem yields an optimal solution to problem $\mathbf{P}_2$ when the penalty functions are linear.

4.2 Job rejection

Another interesting extension of problem $\mathbf{P}$ is to allow the rejection of jobs, where a penalty is incurred when a job is rejected. Scheduling problems with job rejection have been studied extensively; see Slotnick (2011) and Shabtay et al. (2013) for recent reviews. For $i = 1, 2, \ldots, n$, let $\gamma_i \in \mathbb{Z}^+$ denote the penalty incurred when J_i is rejected. Let y_i be a binary decision variable such that $y_i = 1$ if J_i is rejected, and that $y_i = 0$ if J_i is accepted. Then, the extended model has an objective of minimizing $\sum_{i=1}^{n} \sum_{k=1}^{q_i} f_{ik}(x_{ik})(1 - y_i) + \sum_{i=1}^{n} \gamma_i y_i$. We denote this extended problem as $\mathbf{P}_3$. Unfortunately, the following theorem implies that the existence of a polynomial time or pseudo-polynomial time algorithm for problem $\mathbf{P}_3$ is unlikely.

Theorem 2 *Problem $\mathbf{P}_3$ is NP-hard in the strong sense.*

Proof. We transform Exact Cover by 3-Sets (X3C) to the decision version of problem $\mathbf{P}_3$. Given a set X with $3u$ elements, where $u \in \mathbb{Z}^+$, and a collection Γ of 3-element subsets of X, X3C asks if there exists a subset Γ' of Γ where every element of X occurs in exactly one member of Γ'? X3C is known to be NP-complete in the strong sense (Garey and Johnson 1979). For notational convenience, we denote $X = \{1, 2, \ldots, 3u\}$ and $m = |\Gamma|$, and for $i = 1, 2, \ldots, m$, we denote the i^{th} element of Γ as $\Gamma_i = \{a_{i1}, a_{i2}, a_{i3}\}$, where $a_{i1} < a_{i2} < a_{i3}$. Note that in any "yes" instance of X3C, $|\Gamma'| = u$.

Given an arbitrary instance of X3C, we construct a corresponding instance of problem $\mathbf{P}_3$ as follows:

- Number of jobs: $n = m$
- Processing time of J_i: $p_i = a_{i1} + a_{i2} + a_{i3}$

- Number of milestones of J_i: $q_i = 3$
- The k^{th} milestone of J_i: $\tau_{ik} = \frac{1}{2}a_{ik}(a_{ik}+1)$
- Penalty function of J_i at milestone τ_{ik}: $f_{ik}(x) = \max\{m(\sum_{l=1}^{k} a_{il} - x),\, 0\}$
- Rejection penalty of J_i: $\gamma_i = 1$

Let $L = m - u$ be a threshold value on the total penalty. It is easy to see that this instance of $\mathbf{P}_3$ can be constructed in polynomial time, and all the parameter values are bounded by a polynomial in m and u. We will show that this constructed instance has a feasible solution with a total penalty no greater than L if and only if there exists $\Gamma' \subseteq \Gamma$ such that every element of X occurs in exactly one member of Γ'.

("If" part) Suppose that there exists $\Gamma' \subseteq \Gamma$ such that every element of X occurs in exactly one member of Γ'. Then, $u = |\Gamma'|$. We consider the following solution of the constructed instance of $\mathbf{P}_3$: For $i = 1, 2, \ldots, n$, accept J_i if $\Gamma_i \in \Gamma'$, and reject J_i otherwise. For each accepted job J_i and for $k = 1, 2, 3$, process a_{ik} units of J_i in the time interval $[\frac{1}{2}(a_{ik}-1)a_{ik}, \frac{1}{2}a_{ik}(a_{ik}+1)]$.

In this solution, the u accepted jobs are divided into $3u$ portions, and these portions are processed in the time intervals $[0,1], [1,3], [3,6], \ldots, [\frac{1}{2}(3u-1)3u, \frac{1}{2}3u(3u+1)]$. Because every element of X occurs in exactly one member of Γ', these $3u$ time intervals are occupied by the $3u$ job portions with exactly one job portion per interval. Thus, for each accepted job J_i and for $k = 1, 2, 3$, a total of $\sum_{l=1}^{k} a_{il}$ time units of J_i are processed by time τ_{ik}. Since $f_{ik}(\sum_{l=1}^{k} a_{il}) = 0$, the penalty of accepted job J_i at the k^{th} milestone is 0. Hence, the accepted jobs do not incur any penalty. Therefore, the total penalty of this solution is $\sum_{\Gamma_i \notin \Gamma'} \gamma_i = m - u = L$.

("Only if" part) Suppose the constructed instance of $\mathbf{P}_3$ has a feasible solution σ with a total penalty no greater than L. Let r be the number of accepted jobs in this solution. Note that $r \geq u$ since otherwise the total rejection penalty is greater than L. Solution σ has the following property:

Property 1: For each accepted job J_i and for $k = 1, 2, 3$, at least $\sum_{l=1}^{k} a_{il}$ time units of J_i are processed by time τ_{ik}.

Property 1 holds because $f_{ik}(x) \geq m > L$ for any nonnegative integer $x < \sum_{l=1}^{k} a_{il}$.

Let

$$S = \{a_{ik} \mid J_i \text{ is accepted in } \sigma;\ i = 1, 2, \ldots, n;\ k = 1, 2, 3\},$$

which is a multiset (i.e., a set of possibly duplicated elements) with $3r$ elements. Arrange the elements of S in ascending order (with ties broken arbitrarily), and let the sorted elements be $a_{i_1k_1}, a_{i_2k_2}, \ldots, a_{i_{3r}k_{3r}}$. Thus, $a_{i_1k_1} \leq a_{i_2k_2} \leq \cdots \leq a_{i_{3r}k_{3r}}$. We have the following property:

Property 2: If $h < j$ and $i_h = i_j$, then $k_h < k_j$.

Property 2 follows from the fact that $a_{i1} < a_{i2} < a_{i3}$ for all i. We will show that $a_{i_jk_j} = j$ for $j = 1, 2, \ldots, 3r$. Suppose, to the contrary, that $a_{i_jk_j} \neq j$ for some $j = 1, 2, \ldots, 3r$. Then, let

$$\xi = \min\{j \mid a_{i_jk_j} \neq j;\ j = 1, 2, \ldots, 3r\}.$$

We divide the analysis into two cases.

Case 1: $a_{i_\xi k_\xi} < \xi$. In this case, $a_{i_jk_j} = j$ for $j = 1, 2, \ldots, \xi - 1$. In addition, $a_{i_\xi k_\xi} = \xi - 1$ (because $\xi - 1 = a_{i_{\xi-1}k_{\xi-1}} \leq a_{i_\xi k_\xi} < \xi$). By Property 1, at least $\sum_{l=1}^{k_j} a_{i_jl}$ time units of J_{i_j} are processed by time $a_{i_jk_j}(a_{i_jk_j} + 1)/2$ for $j = 1, 2, \ldots, \xi$. Thus, at least $\sum_{l=1}^{k_j} a_{i_jl}$ time units of J_{i_j} are processed by time $(\xi - 1)\xi/2$ for $j = 1, 2, \ldots, \xi$. By Property 2, this implies that at least $\sum_{h \leq j \text{ s.t. } i_h = i_j} a_{i_hk_h}$ time units of J_{i_j} are processed by time $(\xi - 1)\xi/2$ for $j = 1, 2, \ldots, \xi$. Let

$$Y_\xi = \{j \mid h \leq j \text{ for all } h \text{ such that } i_h = i_j;\ j = 1, 2, \ldots, \xi\}.$$

Note that $i_j \neq i_{j'}$ (i.e., J_{i_j} and $J_{i_{j'}}$ are different jobs) for any $j, j' \in Y_\xi$ such that $j \neq j'$. Hence, at least $\sum_{j \in Y_\xi} \sum_{h \leq j \text{ s.t. } i_h = i_j} a_{i_hk_h}$ time units of accepted jobs are processed by time $(\xi - 1)\xi/2$. Note that $\sum_{j \in Y_\xi} \sum_{h \leq j \text{ s.t. } i_h = i_j} a_{i_hk_h} = a_{i_1k_1} + a_{i_2k_2} + \cdots + a_{i_\xi k_\xi}$. Therefore, at least $a_{i_1k_1} + a_{i_2k_2} + \cdots + a_{i_\xi k_\xi}$ time units of accepted jobs are processed by time $(\xi - 1)\xi/2$. This is impossible, as $a_{i_1k_1} + a_{i_2k_2} + \cdots + a_{i_\xi k_\xi} = [1 + 2 + \cdots + (\xi - 1)] + (\xi - 1) > (\xi - 1)\xi/2$.

Case 2: $a_{i_\xi k_\xi} > \xi$. In this case, $a_{i_j k_j} = j$ for $j = 1, 2, \ldots, \xi - 1$, and $a_{i_\xi k_\xi} > \xi$. Let

$$\zeta = \min\{j \mid a_{i_j k_j} \leq j;\ j = \xi + 1, \xi + 2, \ldots, 3r\}$$

(note that ζ exists since otherwise $a_{i_{3r} k_{3r}} > 3r \geq 3u$, which is impossible). By Property 1, at least $\sum_{l=1}^{k_j} a_{i_j l}$ time units of J_{i_j} are processed by time $a_{i_j k_j}(a_{i_j k_j} + 1)/2$ for $j = 1, 2, \ldots, \zeta$. Thus, at least $\sum_{l=1}^{k_j} a_{i_j l}$ time units of J_{i_j} are processed by time $\zeta(\zeta+1)/2$ for $j = 1, 2, \ldots, \zeta$. By Property 2, this implies that at least $\sum_{h \leq j \text{ s.t. } i_h = i_j} a_{i_h k_h}$ time units of J_{i_j} are processed by time $\zeta(\zeta + 1)/2$ for $j = 1, 2, \ldots, \zeta$. Let

$$Y_\zeta = \{j \mid h \leq j \text{ for all } h \text{ such that } i_h = i_j;\ j = 1, 2, \ldots, \zeta\}.$$

Following the same argument as in Case 1, at least $\sum_{j \in Y_\zeta} \sum_{h \leq j \text{ s.t. } i_h = i_j} a_{i_h k_h}$ time units of accepted jobs are processed by time $\zeta(\zeta + 1)/2$. Therefore, at least $a_{i_1 k_1} + a_{i_2 k_2} + \cdots + a_{i_\zeta k_\zeta}$ time units of accepted jobs are processed by time $\zeta(\zeta + 1)/2$. This is impossible, as $a_{i_1 k_1} + a_{i_2 k_2} + \cdots + a_{i_\zeta k_\zeta} \geq [1 + 2 + \cdots + (\xi - 1)] + [(\xi + 1) + (\xi + 2) + \cdots + \zeta] + \zeta > \zeta(\zeta + 1)/2$.

Combining Cases 1 and 2, we conclude that $a_{i_j k_j} = j$ for $j = 1, 2, \ldots, 3r$, and that $r = u$. Consider the solution of the given instance of X3C where $\Gamma' = \{\Gamma_i \mid J_i \text{ is accepted in } \sigma\}$. In this solution, every element of X occurs in exactly one member of Γ'. ∎

5 Conclusions and Future Work

This paper presents a multitasking model for the situation where a working team needs to demonstrate progress on different tasks or to treat the task owners equitably. Polynomial time solution methods are presented for the general model. A more efficient solution method is provided for the special case with linear penalties. Two generalizations of the problem are also discussed.

Some future directions on this research are of interest. In Section 4.2, we have shown that problem $\mathbf{P}_3$ is strongly NP-hard. One interesting future research topic would be to develop good performing heuristics for this problem. Another interesting research topic is to study more com-

putationally tractable special cases of $\mathbf{P}_3$, for example, the case where each task has only two milestones, namely a "mid-term milestone" and a "final milestone."

The following extensions of problem $\mathbf{P}$ are also worth investigating. First, in our model we have assumed that each task can be divided into subtasks with any integer duration. However, in some applications there are constraints on how a task can be broken down into subtasks. For example, there may be a minimum requirement on the duration of each subtask to avoid excessive switching of tasks. A more general model would allow such constraints. Second, in our model there is no overhead cost when the working team switches the processing from one task to another task. A more general model would allow either a "switching cost" or a "switching time" between consecutive subtasks. Third, in some applications a lump-sum penalty will incur if a concern percentage of a task cannot be completed by a certain milestone. However, in our model the penalty function f_{ik} is convex and cannot capture this kind of lump-sum penalties. A more general model would allow a lump-sum penalty to be imposed if a certain level of progress is not met by a milestone. Fourth, in Section 1 we have discussed how to impose tighter control on jobs at their final milestones. However, in some applications the actual completion date of a task is one of the performance measures of the task. Thus, a more general model would have a bicriteria objective, where one criterion is to minimize the total penalty at the milestones, and another criterion is to minimize a cost which depends on the completion times of the tasks. Fifth, processing times of tasks performed by human workers can often be reduced by injecting additional financial resources through, for example, hiring additional workers or having workers work overtime. Hence, a more general model would allow controllable processing times, where the duration of a subtask can be compressed through incurring a cost. Finally, our model is a single-machine model, which implicitly assumes that only one working team processes one task at a time. A more general model would have a more general machine structure such as parallel machines, uniform machines, or parallel machines with eligibility constraints. Developing efficient and effective solution methods for these extensions would be of future research interest.

Acknowledgments

The authors thank two anonymous referees for their helpful comments. This research was supported in part by the Research Grants Council of Hong Kong under grant PolyU5195/13E.

References

Ahuja, R.K., Magnanti T.L., Orlin J.B. (1993). *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ.

Chang, J.H., Chiu, H.N. (2005). A comprehensive review of lot streaming. *International Journal of Production Research* **43**(8), 1515–1536.

Cheng, M., Mukherjee, N.J., Sarin, S.C. (2013). A review of lot streaming. *International Journal of Production Research* **51**(23–24), 7023–7046.

Choi, B.-C., Park, M.-J. (2015). A continuous time-cost tradeoff problem with multiple milestones and completely ordered jobs. *European Journal of Operational Research* **244**(3), 748–752.

Correa, J., Marchetti-Spaccamela, A., Matuschke, J., Stougie, L., Svensson, O., Verdugo, V., Verschae, J. (2015). Strong LP formulations for scheduling splittable jobs on unrelated machines. *Mathematical Programming* **154**(1–2), 305–328.

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

Eynan, A., Li, C.-L. (1997). Lot-splitting decisions and learning effects. *IIE Transactions* **29**(2), 139–146.

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

Ghoddousi, P., Ansari, R., Makui, A. (2017). An improved robust buffer allocation method for the project scheduling problem. *Engineering Optimization* **49**(4), 718–731.

Hall, N.G., Leung, J.Y.-T., Li, C.-L. (2015). The effects of multitasking on operations scheduling. *Production and Operations Management* **24**(8), 1248–1265.

Hall, N.G., Leung, J.Y.-T., Li, C.-L. (2016). Multitasking via alternate and shared processing:

Algorithms and complexity. *Discrete Applied Mathematics* **208**, 41–58.

Hochbaum, D.S., Shanthikumar, J.G. (1990). Convex separable optimization is not much harder than linear optimization. *Journal of the Association for Computing Machinery* **37**(4), 843–862.

Keskinocak, P., Tayur, S. (2004). Due date management policies. In: D. Simchi-Levi, S.D. Wu, Z.M. Shen (eds.), *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*, Kluwer, Boston, pp. 485–554.

Kleinschmidt, P., Schannath, H. (1995). A strongly polynomial algorithm for the transportation problem. *Mathematical Programming* **68**(1–3), 1–13.

Liu, M., Wang, S., Zheng, F., Chu, C. (2017). Algorithms for the joint multitasking scheduling and common due date assignment problem. *International Journal of Production Research* **55**(20), 6052–6066.

Logendran, R., Subur, F. (2004). Unrelated parallel machine scheduling with job splitting. *IIE Transactions* **36**(4), 359–372.

Philipoom, P.R., Markland, R.E., Fry, T.D. (1989). Sequencing rules, progress milestones and product structure in a multistage job shop. *Journal of Operations Management* **8**(3), 209–229.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*, Wiley, Chichester.

Serafini, P. (1996). Scheduling jobs on several machines with the job splitting property. *Operations Research* **44**(4), 617–628.

Serin, Y., Kayaligil, S. (2003). Lot splitting under learning effects with minimal revenue requirements and multiple lot types. *IIE Transactions* **35**(8), 689–698.

Shabtay, D., Gaspar, N., Kaspi, M. (2013). A survey on offline scheduling with rejection. *Journal of Scheduling* **16**(1), 3–28.

Slotnick, S.A. (2011). Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research* **212**(1), 1–11.

Sum, J., Ho, K. (2015). Analysis on the effect of multitasking. In: *Proceedings of 2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 205–209.

Xing, W., Zhang, J. (2000). Parallel machine scheduling with splitting jobs. *Discrete Applied*

Mathematics **103**(1–3), 259–269.

Yalaoui, F., Chu, C. (2003). An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times. *IIE Transactions* **35**(2), 183–190.

Zhu, Z., Li, J., Chu, C. (2017a). Multitasking scheduling problems with deterioration effect. *Mathematical Problems in Engineering* **2017**, Article ID 4750791 (10 pages).

Zhu, Z., Liu, M., Chu, C., Li, L. (2017b). Multitasking scheduling with multiple rate-modifying activities. *International Transactions in Operational Research*, DOI:10.1111/itor.12393.

Zhu, Z., Zheng, F., Chu, C. (2017c). Multitasking scheduling problems with a rate-modifying activity. *International Journal of Production Research* **55**(1), 296–312.