Discrete Optimization 17 (2015) 69–79

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Single-machine scheduling with supporting tasks

Alexander V. Kononov[a], Bertrand M.T. Lin[b,*], Kuei-Tang Fang[b]

[a] *Sobolev Institute of Mathematics, Russian Academy of Sciences, Novosibirsk, Russia*
[b] *Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan*

ARTICLE INFO

Article history:
Received 18 November 2014
Received in revised form 24 April 2015
Accepted 4 May 2015

Keywords:
Single-machine scheduling
Supporting precedence
Total weighted completion time
Number of late jobs

ABSTRACT

This paper investigates a single-machine scheduling problem with a set of supporting tasks and a set of jobs. Each job is preceded by a subset of supporting tasks, that is, the job cannot start its processing until all of its supporting tasks are finished. The objective functions are defined solely in job completion times. We discuss the complexities of several special cases for the number of late jobs and the total weighted completion time. This study adds new complexity results to the two standard objective functions under precedence constraints.

1. Introduction

We consider single-machine scheduling problems with supporting tasks. Two disjoint sets of activities $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ are to be processed on a single machine. The elements of set A are called *supporting tasks*, and the elements of set B are called *jobs*. The processing times of task a_i and job b_j are denoted by α_i and β_j, respectively. A supporting relation $\mathcal{R} : A \to B$ is a mapping from set A to set B such that for $a_i \in A$ and $b_j \in B$, if (a_i, b_j) belongs to $\mathcal{R}$ then job b_j cannot start unless task a_i is completed. Each job b_j is associated with a weight w_j and a due date d_j. Denote the completion time of job $b_j \in B$ in a particular schedule by C_j. If job b_j is late, that is $C_j - d_j > 0$, then we set binary variable $U_j = 1$; 0, otherwise. For each task $a_i \in A$, denote $\mu_i = \{b_j \in B | (i, j) \in \mathcal{R}\}$ as the set of jobs supported by a_i. Similarly, for each job $b_j \in B$, denote $\nu_j = \{a_i \in A | (i, j) \in \mathcal{R}\}$ the set of tasks that support b_j. Although all supporting tasks are required to be processed, they do not contribute to the objective functions because their roles are simply preparatory operations for the jobs that they support.

The proposed model is motivated by real-world applications where preparatory operations are required before jobs can be processed. In the context of multi-media scheduling, a playback comprises several media

* Corresponding author. Tel.: +886 3 5131472; fax: +886 3-5723792.
E-mail address: bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

http://dx.doi.org/10.1016/j.disopt.2015.05.001
1572-5286/© 2015 Elsevier B.V. All rights reserved.

objects, including for example, audio, video, icons, and texts. Once a media object is prepared, either downloaded from a digital archive or created at the local site, it can be embedded in several playbacks. Media objects and playbacks are referred to as supporting tasks and jobs, respectively, in the defined scheduling setting. The roaring lion of the MGM films and special animation effects of some TV programs are examples of media objects that are produced once and used many times as appropriate. This unique property introduces a new type of bill of materials and makes the scheduling of on-line media production different from the manufacturing of tangible products, like for example car, cloth, and drink. Another scenario describing the setting is tool or material preparation in a manufacturing environment. A job may require a set of tools installed before its processing. On the other hand, a tool, once installed, may be required by several jobs. In summary, the scheduling setting is applicable to scheduling contexts where preparatory operations occupy a limited resource (the machine) but are not taken into account in the objective functions.

The rest of this paper is organized as follows. In Section 2, we review the related works and summarize the new results conveyed by this paper. Section 3 is dedicated to the objective function of the number of tardy jobs. Minimization of the total weighted completion time is addressed in Section 4. Section 5 discusses special cases where either a task sequence or a job sequence is given *a priori*. We conclude this study and suggest potential research issues in Section 6.

2. Related results and our contributions

This paper discusses two min-sum objective functions: the number of late jobs ($\sum_j U_j$) and the total weighted completion time of jobs ($\sum_j w_j C_j$). Denote the problem by the three-field notation $1|s-prec|\gamma$, where $s-prec$ dictates the supporting precedence and $\gamma \in \{\sum w_j C_j, \sum U_j\}$. We note that the classical min–max criteria in this setting can be easily solved. The makespan of any feasible schedule is fixed. As for minimization of the maximum lateness or tardiness, we can associate the supporting tasks with an infinite due date and then deploy the dynamic programming algorithm of Lawler [1].

In the next section, we consider the minimization of the number of late jobs. Karp [2] shows that the single-machine problem of minimizing the weighted number of late jobs ($1 \parallel \sum w_j U_j$) is NP-hard and proposes a pseudo-polynomial dynamic programming algorithm. The case where all jobs are equally weighted can be solved in polynomial time by Moore–Hodgson algorithm [3]. If all jobs have a unit execution time ($1|p_j = 1|\sum w_j U_j$), then the problem can be solved as follows: For each position starting backward from $\max\{d_j\}$, the non-tardy job with the largest weight is assigned, breaking a tie on weights by assigning the job with the latest due date. The problem with general precedence constraints is strongly NP-hard, even if all jobs are equally weighted and have a unit processing time [4]. The proof is based on a reduction from the Maximum Clique problem to $1|prec, p_j = 1|\sum w_j U_j$. The technique also works in our study. Given a graph G, we treat each vertex as a supporting task and each edge as a job. Let all jobs have a common deadline. Choosing this deadline appropriately we can prove the strong NP-hardness of $1|s-prec|\sum U_j$, even if $\alpha_i = \beta_j = 1$ for all a_i and all b_j or if $\alpha_i = 1$ for all a_i and $\beta_j = 0$ for all b_j. In both cases each job requires exactly two supporting tasks, i.e. $|\nu_j| = 2$. Moreover, the second case is equivalent to the well-known Dense-K-Subgraph problem studied by Feige, Peleg and Kortsarz [5]. In Section 3 we consider several restricted cases where each job is supported by exactly one task, i.e. $|\nu_j| = 1$. More precisely, we prove the strong NP-hardness of $1|s-prec, |\nu_j| = 1, \beta_j = 0|\sum U_j$, $1|s-prec, |\nu_j| = 1, \alpha_i = 1, \beta_j = 1|\sum U_j$, $1|s-prec, |\mu_i| = 3, |\nu_j| = 1, \alpha_i = 1|\sum U_j$ and the NP-hardness of $1|s-prec, |\mu_i| = 2, |\nu_j| = 1|\sum U_j$. Certainly, these results immediately imply the NP-hardness of the corresponding problems of the weighted number of tardy jobs. We also show that $1|s-prec, |\nu_j| = 1, \alpha_i = 1, \beta_j = 0|\sum w_j U_j$ can be solved in $O(n + m^3)$ time. Please see Table 1 for a summary of the results.

Lenstra and Rinnooy Kan [6] prove the strong NP-hardness of the $1|chains, p_j = 1|\sum U_j$ problems. But this approach is not applicable to our problem. In their proof, the precedence graph consists of several long

Table 1
Complexity results of $\sum w_j U_j$.

Conditions	Complexity
$\lvert\nu_j\rvert = 2, \alpha_i = 1, \beta_j = 0, w_j = 1, d_j = D$	snp-h (Section 2)
$\lvert\nu_j\rvert = 2, \alpha_i = 1, \beta_j = 1, w_j = 1, d_j = D$	snp-h (Section 2)
$\lvert\nu_j\rvert = 1, \alpha_i = 1, \beta_j = 0$	$O(n + m^3)$ (Theorem 1)
$\lvert\nu_j\rvert = 1, \alpha_i = 1, \beta_j = 1, w_j = 1$	snp-h (Theorem 2)
$\lvert\nu_j\rvert = 1, \beta_j = 0, w_j = 1$	snp-h (Theorem 3)
$\lvert\nu_j\rvert = 1, \lvert\mu_i\rvert = 3, \alpha_i = 1, w_j = 1$	snp-h (Theorem 4)
$\lvert\nu_j\rvert = 1, \lvert\mu_i\rvert = 2, w_j = 1$	np-h (Theorem 5)

snp-h: Strongly NP-hard; np-h: NP-hard.

chains. The precedences considered in this paper are represented by a bi-level uni-directed graph studied in Erdos and Moser [7]. There is no implication of complexity status between the two corresponding problems.

Section 4 discusses the minimization of the total weighted completion time. Single-machine scheduling to minimize the total weighted completion time ($1 \parallel \sum w_j C_j$) can be tackled using the weighted shortest processing time (WSPT) first rule [8]. Baker [9] and Horn [10] propose algorithms for the case with precedence constraints in trees. Adolphson and Hu [11] present an $O(n \log n)$ implementation of the algorithms. The case with generalized series–parallel graphs are also polynomial solvable [1]. When general precedence graphs are considered, Lawler [1] proves the strong NP-hardness of the cases with $p_j = 1$ and $w_j \in \{k, k+1, k+2\}$ for any integer k. The proof was adapted for the case with $w_j = 1$ and $p_j \in \{1, 2, 3\}$. Lawler [1] and Lenstra and Rinnooy Kan [12] also show the case with $w_j = 1$ and $p_j \in \{0, 1\}$ to be strongly NP-hard.

The supporting model can be considered as a scheduling problem with bi-level, uni-directed precedence constraints where all supporting tasks have a zero weight. Another key feature lies in the structure of precedence graphs. In the proofs presented by Lawler [1] and Lenstra and Rinnooy Kan [12], the constructed precedence graphs have long chains of unit-execution-time (or unit-weight) jobs. The precedence graph defined by the supporting relation is however in the form of bi-level, uni-directed graphs, where the arcs all emit from the task nodes into the job nodes. Therefore, not all known complexity results can be applied to the supporting precedence.

The minimum latency set cover problem investigated by Hassin and Levin [13] is relevant to the $1|s-prec|\sum w_j C_j$ problem. The minimum latency set cover problem involves several subsets of unit-time tasks. A subset is complete when all of its tasks are finished. The objective function is the total weighted completion time of subsets. They give a reduction from $1|prec, p_j = 1|\sum w_j C_j$ to show the strong NP-hardness. The result is then extended to the unweighted case through a pseudo-polynomial time reduction. The minimum latency set cover problem corresponds to the $1|s-prec|\sum w_j C_j$ problem when $\alpha_i = 1$ for all $a_i \in A$ and $\beta_j = 0$ for all $b_j \in B$. Therefore, the $1|s-prec, \alpha_i = 1, \beta_j = 0|\sum w_j C_j$ problem is strongly NP-hard. In Section 4, we give the complexity result of the case with $\alpha_i = 1$ for all tasks a_i and $\beta_j = 1$ for all jobs b_j. We give a pseudo-polynomial reduction from a more restricted version of $1|prec, p_j = 1|\sum w_j C_j$, where $w_j \in \{1, 2, 3\}$.

In Section 5, we discuss the complexity of the problems where the order of tasks or the order of jobs is known and fixed *a priori*. Almost all known metaheuristics use the fixed order of jobs to describe a solution in scheduling problems. Direct implementation of this rule for the single-machine scheduling problems with supporting tasks gives us $(n + m)!$ different solutions. More justifications of the assumption of a fixed task/job sequence can be found in Shafransky and Strusevich [14] and Hwang, Kovalyov and Lin [15]. We show that the problems with both considered criteria are polynomially solvable if either the order of tasks or the order of jobs is fixed. With these results the solution space is reduced from $O((n + m)!)$ to $O(k!)$, where $k = \min\{n, m\}$. Moreover, it follows that both problems are polynomially solvable if either the number of tasks or the number of jobs is bounded by $\log L$, where L is the problem input length in binary encoding.

3. Number of late jobs ($\sum_j U_j$)

In this section we discuss the minimization of the number of late jobs and study the case where the precedence is given by an out-forest, i.e. $|\nu_j| = 1$ for all jobs b_j. Under the specified conditions, each job is preceded by exactly one task and the out-trees are disjoint. A job is called *early* if it is completed before or on its due date. With this definition we have the following simple result.

Lemma 1. *There is an optimal schedule to $1|s-prec|\sum U_j$ in which the early jobs are sequenced by the EDD rule.*

Proof. Validity of the lemma follows from the adjacent job interchange argument. □

The first case we consider is $1|s-prec, |\nu_j| = 1, \alpha_i = 1, \beta_j = 0|\sum w_j U_j$. The problem can be solved by a transformation to the assignment problem. Since all jobs has a zero processing time, an optimal schedule will not complete any job later than time m, the number of tasks. Therefore, any job due-dates larger than m can be redefined as m. This requires $O(n)$ time. For each task a_i and each time point $t \leq m$, the jobs that are supported by task a_i and have due-date t can be consolidated into a composite job with an aggregate weight. As a consequence, the number of jobs is now bounded by $O(m^2)$ after the above consolidation. The time required is also $O(n)$. An instance of the assignment problem is defined as follows. Let a set X correspond to the task set A, and a set of Y contain unit-time intervals $\{[0,1), [1,2), \ldots, [0,m)\}$. For task $a_i \in X$ and interval $[t-1,t)$, define weight $\bar{w}_{i,t} = \sum_{j\in\mu_i: d_j<t} w_j$. The weight $\bar{w}_{i,t}$ is equal to the weighted number of late jobs in μ_i based upon the scenario where task a_i occupies interval $[j-1,j)$ and its jobs, requiring 0 time units, follow immediately. The assignment problem is to assign the m tasks to the m intervals so that the total weight is minimum. The transformation takes $O(m^2)$ time to calculate the weights $\bar{w}_{i,t}$. Since the assignment problem can be solved in $O(m^3)$ time [16–18], the following theorem immediately follows.

Theorem 1. *Problem $1|s-prec, |\nu_j| = 1, \alpha_i = 1, \beta_j = 0|\sum U_j$ is solvable in $O(n+m^3)$.*

In contrast with the condition $\beta = 0$ in Theorem 1, we show that the $1|s-prec|\sum U_j$ problem remains strongly NP-hard even if $|\nu_j| = 1$ and $\alpha_i = \beta_j = 1$ for all tasks $a_i \in A$ and all jobs $b_j \in B$. Our proof is based on a reduction from Exact Cover by 3-Sets [4].
Exact Cover by 3-Sets: Given a set $T = \{1, \ldots, 3t\}$ and a family $\mathbf{T} = \{T_1, T_2, \ldots, T_r\}$ of three-element subsets of T. Does there exist a subfamily $\mathbf{T}' \subset \mathbf{T}$ such that $|\mathbf{T}'| = t$ and each element of T is involved in exactly one of the subsets of $\mathbf{T}'$?

Theorem 2. *Problem $1|s-prec|\sum U_j$ is NP-hard in the strong sense even if $|\nu_j| = 1$ and $\alpha_i = \beta_j = 1$ for all tasks $a_i \in A$ and all jobs $b_j \in B$.*

Proof. Given instance I of Exact Cover by 3-Sets, we construct instance I' of $1|s-prec|\sum U_j$ with r supporting tasks and $N = \sum_{i=1}^{r}\sum_{j\in T_i} t(3t-j+1)$ jobs as follows. For each subset $T_i = \{j_1, j_2, j_3\} \in \mathbf{T}$ with $1 \leq j_1 < j_2 < j_3 \leq 3t$, define a task a_i that supports (1) $t(3t-j_1+1)$ jobs, denoted by subset $B_{i,j_1} = \{b_{i,j_1,1}, b_{i,j_1,2}, \ldots, b_{i,j_1,t(3t-j_1+1)}\}$, (2) $t(3t-j_2+1)$ jobs, denoted by subset $B_{i,j_2} = \{b_{i,j_2,1}, b_{i,j_2,2}, \ldots, b_{i,j_2,t(3t-j_2+1)}\}$ and (3) $t(3t-j_3+1)$ jobs, denoted by subset $B_{i,j_3} = \{b_{i,j_3,1}, b_{i,j_3,2}, \ldots, b_{i,j_3,t(3t-j_3+1)}\}$. All tasks and all jobs require a unit time for processing. Define due dates $d_j = t + \sum_{l=1}^{j} t(3t-l+1)$ for all $1 \leq j \leq 3t$. For subset T_i and element $j \in T_i$, the jobs of $B_{i,j} = \{b_{i,j,1}, b_{i,j,2}, \ldots, b_{i,j,t(3t-j+1)}\}$ are associated with due date d_j. To facilitate the following discussion, define an auxiliary due date $d_0 = t$. For different subsets T_i, $T_{i'}$, and $j \in T_i \cap T_{i'}$, we have $|B_{i,j}| = |B_{i',j}|$. In the following discussion, if no confusion would arise, we call subset $B_{i,j}$ a B_j-type job set.

With the two instances I and I', we claim that the answer to instance I of Exact Cover by 3-Sets is affirmative if and only if there is a feasible schedule of instance I' which has exactly $N - \sum_{j=1}^{3t} t(3t - j + 1) = N - (d_{3t} - t)$ late jobs.

Assume that sets $T_1, T_2, \dots, T_t$ constitute a partition of Exact Cover by 3-Sets. We obtain a feasible schedule with exactly $N - \sum_{j=1}^{3t} t(3t - j + 1)$ late jobs as follows. Tasks $a_1, \dots, a_t$ are arranged in the interval $[0, t)$. After the execution of the t supporting tasks, the available jobs defined by element $j \in T$ are scheduled in the interval $[d_{j-1}, d_j)$. The remaining $r - t$ tasks start at time d_{3t} and the remaining jobs follow. The constructed schedule adheres to the supporting precedence relation, and the jobs that start before d_{3t} are early.

The second part of the proof assumes that there exists a feasible schedule having exactly $N - \sum_{j=1}^{3t} t(3t - j + 1) = N - (d_{3t} - t)$ late jobs. Two basic properties can be easily established: (1) All early jobs precede all late jobs. (2) The early jobs are sequenced in the EDD order.

Suppose that τ tasks are executed within the interval $[0, d_{3t})$. If $\tau > t$, then it is clear that at most $d_{3t} - t - 1$ can be successfully processed in the interval. On the other hand, we consider the case $\tau < t$. By the definition of the instance, each task supports at most $9t^2$ jobs. Consider the following inequalities:

$$9t^2\tau \le 9t^2(t-1) < 9t^3 - 9t^2/2 + t/2 = d_{3t} - t.$$

Since both cases lead to a contradiction, we have that $\tau = t$ must hold. So, every time slot in $[0, d_{3t})$ is occupied by either a task or an early job. The t tasks permit the processing of exactly $3t$ subsets of jobs, among which some may be defined by the same element $j \in T$. Let x_j (≥ 0) denote the number of B_j-type subsets scheduled within $[0, d_{3t})$. Equality $\sum_{j=1}^{3t} x_j = 3t$ holds. If $x_1 = x_2 = \dots = x_{3t} = 1$, then a partition is found.

Assume that the condition $x_1 = x_2 = \dots = x_{3t} = 1$ is not satisfied in the schedule. We want to show that under this assumption, some slots before d_{3t} will not be occupied, implying that the number of early jobs is less than $d_{3t} - t$. Let $\ell, 1 \le \ell \le 3t$ be the largest integer such that $x_\ell = 0$. It is clear that $\sum_{j=\ell+1}^{3t} x_j \ge 3t - \ell$, which further implies $\sum_{j=1}^{\ell-1} x_j = \sum_{j=1}^{\ell} x_j \le \ell$.

Note that $|B_1| > |B_2| > \dots > |B_{3t}|$ by the definition of instance I'. Assume $\sum_{j=1}^{\ell-1} x_j = \ell - 1 - k$ for $k > 1$. Within interval $[0, d_\ell)$, the number of slots not occupied by the t tasks and the jobs of $B_1 \cap B_2 \cap \dots \cap B_{\ell-1}$ is not less than $k \times |B_\ell|$. The fact that $\sum_{j=\ell+1}^{3t} x_j = 3t - \ell + k$ implies that no more than $k \times |B_{\ell+1}|$ jobs of $B_{\ell+1} \cap B_{\ell+2} \cap \dots \cap B_{3t}$ can be allocated to fill up the empty slots before d_ℓ. Since $|B_\ell| > |B_{\ell+1}|$, some slots must be empty. Therefore, $x_1 = x_2 = \dots = x_{3t} = 1$ must hold and a partition is found. □

Next, we show that for arbitrary processing times of tasks $a_i \in A$ the $1|s-prec|\sum U_j$ problem is also strongly NP-hard even if $|\nu_j| = 1$ and $\beta_j = 0$ for all jobs $b_j \in B$. Our proof is based on a pseudo-polynomial reduction from 3-Partition [4].
3-Partition: Given an integer M a set of $3t$ integers $X = \{1, 2, \dots, 3t\}$. Each $i \in X$ has a size of x_i satisfying $M/4 < x_i < 3M/4$ and $\sum_{i \in X} x_i = tM$. Is there a partition of set X into t disjoint subsets $X_1, X_2, \dots, X_t$ such that $\sum_{i \in X_j} x_i = M$ for all $j = 1, \dots, t$?

Theorem 3. *Problem $1|s-prec|\sum U_j$ is NP-hard in the strong sense even if $|\nu_j| = 1$ and $\beta_j = 0$ for all jobs $b_j \in B$.*

Proof. Given an instance I of 3-Partition, we create an instance I' of problem $1|s-prec|\sum U_j$ that consists of $3t$ tasks and t^2M jobs. Task $i, 1 \le i \le 3t$ has a processing time $tM + x_i$ and supports tx_i jobs, which are categorized into t groups $B_{i1}, B_{i2}, \dots, B_{it}$, each having exactly x_i jobs. All jobs in group $B_{ik}, 1 \le k \le t$, have the same deadline $d_k = k(3t+1)M$. Thus, for every k exactly tM jobs have deadline d_k. Processing times of all jobs are 0.

With the two instances I and I', we claim that the answer to instance I of 3-Partition is affirmative if and only if there is a feasible schedule of instance I' which has exactly $\frac{t(t-1)}{2}M$ late jobs.

First, let subsets $X_1, X_2, \ldots, X_t$ constitute a partition of set X. We then construct a task sequence in which the tasks defined by the three elements of X_1 come first, followed by the jobs supported by the three tasks. The same dispatching policy applies to $X_2, \ldots, X_t$. It is easy to see that the number of late jobs is $(t-1)M + (t-2)M + \cdots + (t-1)M = \frac{t(t-1)}{2}M$.

Next, let σ be an arbitrary feasible schedule. We note that at most $3k$ tasks are completed before or at d_k. It is true for $k = t$ because instance I' has exactly $3t$ tasks. Suppose that for $k < t$, there are $3k+1$ or more tasks completed before or at d_k in σ. The total processing length of the $3k+1$ is at least $(3k+1)tM$, which is larger than $(3t+1)Mk$ for any $k < t$. A contradiction arises.

For completeness we define $d_0 = 0$. Denote by X_i the set of tasks completed within $(d_{k-1}, d_k]$ under schedule σ after time d_{k-1} and before or at time d_k. It follows that tasks from a set $\mathcal{X}_k = X_1 \bigcup X_2 \bigcup \cdots \bigcup X_k$ are completed before or at time d_k. Therefore, we have $\sum_{i\in\mathcal{X}_k}(tM + x_i) \le d_k$. Moreover, $tM - \sum_{i\in\mathcal{X}_k} x_i$ jobs are late subject to the deadline d_k in schedule σ. The first inequality implies

$$\sum_{i\in\bar{X}_k} x_i \le kM. \tag{1}$$

Summing the numbers of late jobs with deadlines d_k over all k's, we can see that schedule σ has $t^2M - \sum_{k=1}^{t}\sum_{i\in\mathcal{X}_k} x_i$ late jobs. Using (1), we obtain

$$t^2M - \sum_{k=1}^{t}\sum_{i\in\mathcal{X}_k} x_i \ge t^2M - \sum_{k=1}^{t} kM = \frac{t(t-1)}{2}M. \tag{2}$$

Equality in Eq. (2) is attained if and only if $\sum_{i\in X_k} x_i = M$ for all $k = 1, \ldots, t$. We can therefore conclude that the answer to instance of 3-Partition is affirmative if and only if there is a feasible schedule of instance I' which has exactly $\frac{t(t-1)}{2}M$ late jobs. □

In the next case to investigate, every task takes a unit execution time and supports exactly three jobs, every job is supported by exactly one task, and the processing times of jobs are arbitrary.

Theorem 4. *Problem $1|s-prec|\sum U_j$ is NP-hard in the strong sense even if $|\mu_i| = 3$, $\alpha_i = 1$ for all tasks $a_i \in A$ and $|\nu_j| = 1$ for all jobs $b_j \in B$.*

Proof. The reduction is also carried out from Exact Cover by 3-Sets. An instance of $1|s-prec|\sum U_j$ with r supporting tasks and $3r$ jobs as follows. For each subset $T_i = \{j_1, j_2, j_3\} \in \mathbf{T}$ with $1 \le j_1 < j_2 < j_3 \le 3t$, define a unit execution time task a_i that supports three jobs $b_{i,j_1}, b_{i,j_2}, b_{i,j_3}$ with processing times $t(3t-j_1+1), t(3t-j_2+1), t(3t-j_3+1)$. Define due dates $d_j = t + \sum_{l=1}^{j} t(3t-l+1)$ for all $1 \le j \le 3t$. Job $b_{i,j}$ of any subset T_i is associated with due date d_j. We define an auxiliary due date $d_0 = t$. Using the proof techniques of Theorem 2, we can prove that an exists cover exists Exact Cover by 3-Sets is affirmative if and only if there is a feasible schedule of $1|s-prec|\sum U_j$ with exactly $3r - 3t$ late jobs. □

The following theorem gives the complexity result of another case where each task supports exactly two jobs.

Theorem 5. *The $1|s-prec|\sum U_j$ remains NP-hard even when $|\mu_i| = 2$ for all tasks a_i and $|\nu_j| = 1$ for all jobs b_j.*

Proof. We give the proof by a polynomial-time reduction from Equal-Size-Partition, which is defined as follows:

Equal-Size-Partition: Given a set of $2t$ integers $X = \{1, 2, \dots, 2t\}$ and, for each i a size of x_i such that $\sum_{i\in X} x_i = 2M$, does there exist a partition X_1 and X_2 of X such that $\sum_{i\in X_1} x_i = \sum_{i\in X_2} x_i$ and $|X_1| = |X_2| = t$?

Given an instance I of Equal-Size-Partition, we create an instance I' of problem $1|s-prec|\sum U_j$ that consists of $2t$ tasks and $4t$ jobs. For each element $i \in X$, define two jobs $b_{i,1}$ and $b_{i,2}$ and their common supporting task a_i. The associated parameters are given as processing time $\alpha_i = \theta^3$; processing time $\beta_{i,1} = \theta + x_i$ and due date $d_{i,1} = t\theta^3 + t\theta + M$; processing time $\beta_{i,2} = \theta^2 - 2x_i$ and due date $d_{i,2} = t\theta^3 + t\theta^2 + t\theta - M$, where θ is an appropriate large positive number.

Denote $B_1 = \{b_{i,1}|1 \le i \le 2t\}$ and $B_2 = \{b_{i,2}|1 \le i \le 2t\}$. To prove the theorem, it suffices to show that if the answer to instance I is affirmative, then there is a feasible solution to instance I' with at most $2t$ late jobs, and vice versa.

Let X_1 and X_2 be a partition of instance I. We schedule the tasks and jobs of instance I' as follows. The sequence starts with t tasks a_i defined by the elements of X_1. The t jobs $b_{i,1}$ released by the t scheduled tasks follow. Then, the corresponding t jobs $b_{i,2}$ follow. The remaining tasks and jobs are scheduled late. The t jobs $b_{i,1}$ are completed at time $t\theta^3 + t\theta + \sum_{i\in X_1} x_i = t\theta^3 + t\theta + M = d_{i,1}$. The t jobs $b_{i,2}$ complete at $(t\theta^3 + t\theta + M) + t\theta^2 - 2\sum_{i\in X_1}^{t} x_i = t\theta^3 + t\theta^2 + t\theta - M = d_{i,2}$. Therefore, we obtain a feasible schedule with exactly $2t$ early jobs.

We now consider a schedule of instance I' with no more than $2t$ late jobs. Because of the large number θ^3 in the processing times of tasks $\{a_i\}$ and the due date $d_{i,2}$, no more than t tasks a_i can be completed before $d_{i,2}$. This implies that at most t jobs of B_1 and at most t jobs of B_2 are eligible to be processed as early. Combining the observations with the fact that at least $2t$ jobs are scheduled early in schedule $\sigma(I')$, we conclude that exactly t jobs of B_1 and t jobs of B_2 are early.

By Lemma 1, we assume that the jobs of B_1 precede the jobs of B_2. In other words, the schedule starts with t tasks a_i, followed by a block of t jobs of B_1 and a block of t jobs of B_2. We examine the completion times of the $2t$ jobs. The t jobs of B_1 are completed at time $t\theta^3 + t\theta + \sum_{i\in X_1} x_i$, where $X_1 \subset X$ contains the elements defining the t tasks. To ensure the early status of the first t jobs, $t\theta^3 + t\theta + \sum_{i\in X_1} x_i$ must be smaller than or equal to the due date $d_{i,1}$. Thus, we have $\sum_{i\in X_1} x_i \le M$. Similarly, the completion time of the t early $\{b_{i,2}\}$ jobs is $(t\theta^3 + t\theta + \sum_{i\in X_1} x_i) + t\theta^2 - 2\sum_{i\in X_1} x_i = t\theta^3 + t\theta^2 + t\theta - \sum_{i\in X_1} x_i$, which must be smaller than or equal to the due date $d_{i,2}$. Inequality $\sum_{i\in X_1} x_i \ge M$ follows. The two inequalities lead to $\sum_{i\in X_1} x_i = M$. A desired partition is found. □

4. Total weighted completion time ($\sum_j w_j C_j$)

This section discusses the minimization of the total weighted completion time. We present the following result concerning the UET case, i.e. all tasks and all jobs take a unit execution time.

Theorem 6. *The $1|s-prec, \alpha_j = \beta_j = 1|\sum C_j$ problem is strongly NP-hard.*

Proof. The proof is based on a reduction from $1|prec, p_j = 1|\sum w_j C_j$ with $w_j \in \{1, 2, 3\}$, which is strongly NP-hard [1]. We are given an instance I with n jobs such that job j has weight w_j. An acyclic directed graph defines the precedence constraints. For each job j, let P_j denote the set of all predecessors of j. Define an instance I' of $1|s-prec, \alpha_j = \beta_j = 1|\sum C_j$ as follows: Create w_j type-j

jobs $b_{j_1},\dots,b_{j_{w_j}}$ and $4-w_j$ type-j supporting tasks $a_{j_1},\dots,a_{j_{4-w_j}}$. For job b_{j_i}, define the set of supporting tasks $\nu_{j_i} = \{a_{k_l} \in A | k \in P_j, l = 1,\dots,4-w_k\} \bigcup_{l=1,\dots,4-w_j}\{a_{j_l}\}$. Therefore, the ordered pair (a_{k_l}, b_{j_i}) belongs to relation $\mathcal{R}$ if either $k = j$ or $k \in P_j$. Since $1 \le w_j \le 3$ for each job j, the reduction from I to I' is polynomial.

Let $\sigma(I')$ be a feasible schedule of instance I'. We transform $\sigma(I')$ without increasing its objective value into a feasible schedule $\sigma^*(I')$ that satisfies the following properties. For each j, supporting tasks $a_{j_1},\dots,a_{j_{4-w_j}}$ and jobs $b_{j_1},\dots,b_{j_{w_j}}$ are executed consecutively without interruption. Moreover, if job i is in P_j, then tasks $a_{i_1},\dots,a_{i_{4-w_j}}$ and jobs $b_{i_1},\dots,b_{i_{w_j}}$ precede tasks $a_{j_1},\dots,a_{j_{4-w_j}}$ and jobs $b_{j_1},\dots,b_{j_{w_j}}$ in schedule $\sigma^*(I')$.

The transformation procedure consists of four parts. (1) For all tasks a_{i_k} and $a_{j_l}, i \in P_j$, if task a_{j_l} precedes task a_{i_k}, then we swap the their positions. Since $\mu_{j_l} \subset \mu_{i_k}$, the derived schedule remains feasible. (2) For all jobs b_{i_k} and $b_{j_l}, \in P_j$, if job b_{j_l} precedes job b_{i_k}, then we swap their positions. Since $\nu_{i_k} \subset \nu_{j_l}$, the derived schedule is still feasible and the objective function value remains unchanged. (3) Assume that some type-j jobs are not executed in consecutive positions. Let b_{j_l} be the first such job in $\sigma(I')$. Let $b_{j'_l}$ be the next type-j job in $\sigma(I')$ and $C_{j'_l} > C_{j_l} + 1$. Assume there are r_1 jobs and r_2 tasks scheduled between job b_{j_l} and job $b_{j'_l}$. It is evident that all the r_1 jobs and r_2 tasks are independent of b_{j_l} and $b_{j'_l}$. We schedule $b_{j'_l}$ directly after b_{j_l} and shift the r_1 jobs and r_2 tasks one position to the right. The derived schedule is feasible and the objective value decreases by $r_1 + r_2$ and increases by r_1. (4) Finally, if some type-j task is not executed directly before the first type-j job, then we move this task to the position immediately preceding the first type-j job and move all the corresponding tasks and jobs one position to the left. Feasibility and the objective value remains the same. Moreover, the derived schedule is exactly $\sigma^*(I')$.

The schedule obtained through the above transformation process is called a regular schedule. Assume that $\hat{\sigma}(I)$ is a feasible schedule of instance I with cost K. Let t_j be the starting time of job j in $\hat{\sigma}(I)$. We schedule tasks $a_{j_1},\dots,a_{j_{4-w_j}}$ and jobs $b_{j_1},\dots,b_{j_{w_j}}$ within an interval $[4t_j, 4t_j + 4)$ in $\sigma^*(I')$. It is easy to check that each feasible schedule of I corresponds to a regular schedule of I', and vice versa. Assign coefficient $\tilde{w_j}$ to each job of instance I' by letting $\tilde{w_j} = 0$ if $w_j = 1$, $\tilde{w_j} = 1$ if $w_j = 2$, and $\tilde{w_j} = 3$ if $w_j = 3$. The cost of $\sigma^*(I')$ is $4K - \sum_j \tilde{w_j}$. For any regular schedule, the value of $\sum_j \tilde{w_j}$ is a constant. Thus for a given solution to I with a cost K, there is a solution to I' with a cost no greater than $4K - \sum_j \tilde{w_j}$, and vice versa. It follows that $1|s-prec, \alpha_j = \beta_j = 1|\sum C_j$ is NP-hard in the strong sense. □

5. Fixed task or job sequence

In this section we study the problems where the order of tasks or the order of jobs is known and fixed *a priori*.

First, we investigate a special case where the job sequence is given. We present a simple algorithm which produces an optimal solution for any regular criterion. An objective function is called regular if it is non-decreasing in the job completion times. Therefore, no idle time is inserted in optimal schedules. Since no idle time is assumed, a schedule is uniquely implied by a given sequence. Note that the both objective functions considered in this paper are regular.

Lemma 2. *For any regular objective, there exists an optimal schedule such that if there are supporting tasks scheduled between two consecutive jobs b_{j_1} and b_{j_2}, then these supporting tasks all belong to ν_{j_2}.*

Proof. Assume it is not the case for some optimal sequence. Consider the supporting tasks that are scheduled between jobs b_{j_1} and b_{j_2} but do not belong to ν_{j_2}. We move these tasks to the positions immediately following

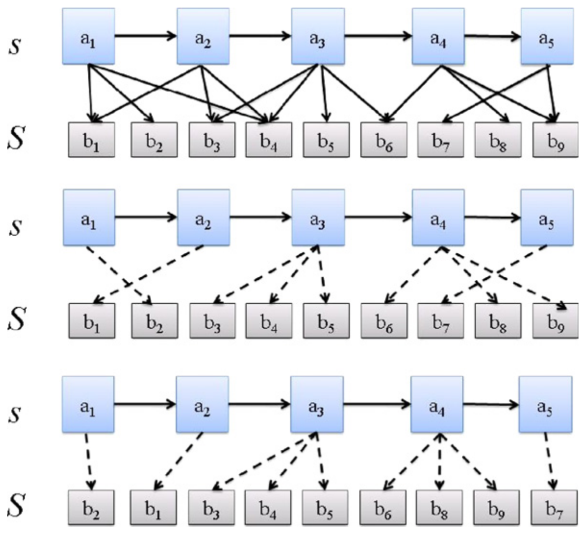

Fig. 1. Example of a fixed task sequence.

job b_{j_2}. Since all jobs supported by these tasks are scheduled after job b_{j_2}, the derived schedule remains feasible. Moreover, the move will not increase the completion time of any job. □

Let $S = (b_1, b_2, \dots, b_n)$ be a given job sequence. The following algorithm produces an optimal sequence $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_{m+n})$ for $1|s{-}prec|\gamma$ subject to job sequence S.

Algorithm 1: Fixed_Job_Sequence(S)

```
$t \leftarrow 1$;
for $j = 1$ to $n$ do
    Arrange the supporting tasks of $\nu_j$ in arbitrary order as $\sigma_t, \dots, \sigma_{t+|\nu_j|-1}$;
    $\sigma_{t+|\nu_j|} \leftarrow b_j$;
    $t \leftarrow t + |\nu_j| + 1$;
    forall the jobs $\ell \in \{j+1, \dots, n\}$ do
        $\nu_\ell \leftarrow \nu_\ell - \nu_\ell \cap \nu_j$;
```

The correctness of Algorithm Fixed_Job_Sequence (S) directly follows from Lemma 1. As for the running time, we note that each task is scheduled once. When task a_i is scheduled, it is removed from each ν_j for $j \in \mu_i$. Deletion of an element from a set can be carried out in constant time. Therefore, after scheduling task a_i, $O(\mu_i) = O(n)$ time is required to adjust the sets ν_j for $j \in \mu_i$. The overall running time of Algorithm Fixed_Job_Sequence (S) is thus $O(mn)$. In fact, the upper bound of the running time is $|\mathcal{R}| \le mn$ since each ordered pair is removed only once.

Second, we address another special case in which the sequence of all supporting tasks is given and fixed, in contrast to the results of Algorithm Fixed_Job_Sequence (S). Let $s = (a_1, a_2, \dots, a_m)$ be a task sequence. We suppose that no task can be rejected or skipped to yield to its successors. For any job b_j, let $a_{i_1}, a_{i_2}, \dots, a_{i_{|\nu_j|}}$ be its supporting tasks as they appear in sequence s. We can eliminate the ordered pairs $(a_{i_1}, b_j), \dots, (a_{i_{|\nu_j|-1}}, b_j)$ from the instance without altering the precedence constraints. We thus come up with a new instance, in which $|\nu_j| = 1$ for all b_j. Please refer to Fig. 1 for the above reduction process.

It follows that the $1|s - prec|\sum w_jC_j$ problem is reduced to $1 \parallel \sum w_jC_j$ with precedences given by an out-tree. The above reduction takes $O(mn)$ time due to the relation size $|\mathcal{R}|$, and solving the instance of n

jobs in $1|out\text{-}tree|\sum w_jC_j$ takes $O(n\log n)$ time. Therefore, the $1|s-prec|\sum w_jC_j$ problem is solvable in $O(n\max\{m,\log n\})$ time.

Now, let us consider an instance $(A,B,\mathcal{R})$ of $1|s-prec|\sum U_j$ with a fixed task sequence $s=(a_1,a_2,\ldots,a_m)$. We construct an instance $\tilde{B}=\{\tilde{b}_1,\tilde{b}_2,\ldots,\tilde{b}_n\}$ of $1\parallel\sum U_j$. Recall that each job is supported by exactly one task. If job $\tilde{b}_j$ is supported by task a_i, then define due date $\tilde{d}_j=d_j-\sum_{h=1}^{i}a_h$.

Lemma 3. *There is a one-to-one correspondence between the solution space of instance $(A,B,\mathcal{R})$ in $1|s-prec|\sum U_j$ subject to task sequence $(a_1,\ldots,a_m)$ and the solution space of instance $\tilde{B}$ in $1\parallel\sum U_j$.*

Proof. Let $\sigma=(\sigma_1,\sigma_2,\ldots,\sigma_{m+k})$ be a sequence of m tasks and k early jobs of instance $(A,B,\mathcal{R})$ in which $\sigma_{i_1}=a_1,\sigma_{i_2}=a_2,\ldots,\sigma_{i_m}=a_m$. We retract the m tasks from σ. The resulting sequence is denoted by $\tilde{\sigma}=(\tilde{\sigma}_1,\tilde{\sigma}_2,\ldots,\tilde{\sigma}_k)$. Any job $\tilde{\sigma}_j$ in $\tilde{\sigma}$ are completed by their due dates in the context of instance $\tilde{B}$ in problem $1\parallel\sum U_j$.

To establish the other side of the proof, we let $\tilde{\sigma}=(\tilde{\sigma}_1,\tilde{\sigma}_2,\ldots,\tilde{\sigma}_k)$ be an arbitrary solution having exactly k early jobs for instance $\tilde{B}$. The following procedure iteratively inserts m tasks into the sequence to make a solution of exactly k early jobs for instance (A,B,R). First, find the task a_i that supports job $\tilde{\sigma}_1$. Insert subsequence $(a_1,\ldots,a_i)$ in front of job $\tilde{\sigma}_1$. For $j=2$ to k, do the following two steps for insertion operations.

1. Find the task a_i that supports job $\tilde{\sigma}_j$.
2. If task a_i is already inserted, then it is done for job $\tilde{\sigma}_j$. Otherwise, insert task a_i and all its predecessors that are not yet inserted into the sequence as the immediate predecessors of job $\tilde{\sigma}_j$.

When the procedure stops, a sequence of m tasks and k jobs are derived. It is easy to verify that the supporting precedence is observed and the k jobs remain early in the derived sequence. □

The instance $\tilde{B}$ for $1\parallel\sum U_j$ has n jobs and thus can be solved in $O(n\log n)$ time. Reducing the $1|s-prec|\sum U_j$ problem with a fixed task sequence to $1\parallel\sum U_j$ takes $O(mn)$ time.

Summing up the results obtained in this section we have the following theorem.

Theorem 7. 1. *Given a job sequence, the $1|s-prec|\gamma$ problem can be solved in $O(|\mathcal{R}|)$ time for any regular criteria γ.*
2. *The $1|s-prec|\sum w_jC_j$ problem with a fixed task sequence can be solved in $O(n\max\{m,\log n\})$ time.*
3. *The $1|s-prec|\sum U_j$ problem with a fixed task sequence can be solved in $O(n\max\{m,\log n\})$ time.* □

A corollary immediately follows.

Corollary 1. *If either the number of tasks or the number of jobs is bounded by the input length in binary encoding $\log L$, then $1|s-prec|\sum w_jC_j$ and $1|s-prec|\sum U_j$ are solvable in polynomial time.* □

6. Conclusions

This paper proposes a new scheduling problem by introducing supporting tasks, whose completion times are not considered in the objective functions. Two standard min-sum objectives, namely, the total weighted completion time and the number of late jobs, are investigated. We analyzed the complexity status of these objectives under various restrictions. The study extends the relevant results in the literature.

For future research, we can design and analyze approximation algorithms for the identified hard cases to tighten the existing performance ratios. It would also be interesting to investigate the complexities of the cases where the out-degrees of tasks ($|\mu_i|$) and in-degrees of jobs ($|\nu_j|$) are bounded from above by a constant. Another specific problem to address is the minimization of the number of late jobs when $|\nu_j| = 1, |\mu_j| = 2$ (or any fixed number), and $|\beta_j| = 0$. The complexity status is still open.

Acknowledgments

The authors are grateful for the anonymous reviewers for their constructive comments. This research was supported in part by project grants MOST102-2923-H-009-001-MY3 and RFH-NSC 13-22-10002.

References

[1] E.L. Lawler, Sequencing jobs to minimize total weighted completion time subject to precedence constraints, Ann. Discrete Math. 2 (1978) 75–90.
[2] R.M. Karp, Reducibility among combinatorial problems, Complex. Comput. Comput. 4 (1972) 85–103.
[3] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs, Manag. Sci. 15 (1968) 102–109.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman, San Francisco, CA, 1979.
[5] U. Feige, D. Peleg, G. Kortsarz, The dense k-subgraph problem, Algorithmica 29 (2001) 410–421.
[6] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity results for scheduling chains on a single machine, European J. Oper. Res. 4 (1980) 270–275.
[7] P. Erdos, L. Moser, On the representation of directed graphs as unions of orderings, Math. Inst. Hung. Acad. Sci. 9 (1964) 125–132.
[8] W.E. Smith, Various optimizers for single-stage production, Nav. Res. Logist. Q. 3 (1956) 59–66.
[9] K.E. Baker, Single machine sequencing with weighting factors and precedence constraints, Unpublished Papers, 1971.
[10] W.A. Horn, Single-machine job sequencing with treelike precedence ordering and linear delay penalties, SIAM J. Appl. Math. 23 (1972) 189–202.
[11] D. Adolphson, T.C. Hu, Optimal linear ordering, SIAM J. Appl. Math. 25 (1973) 403–423.
[12] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, Oper. Res. 26 (1978) 22–35.
[13] R. Hassin, A. Levin, An approximation algorithm for the minimum latency set cover problem, in: Lecture Notes in Computer Science, vol. 3669, 2005, pp. 726–733.
[14] Y.M. Shafransky, V.A. Strusevich, The open shop scheduling problem with a given sequence of jobs on one machine, Nav. Res. Logist. 41 (1998) 705–731.
[15] F.J. Hwang, M.Y. Kovalyov, B.M.T. Lin, Scheduling for fabrication and assembly in a two-machine flowshop with a fixed job sequence, Ann. Oper. Res. 27 (2014) 263–279.
[16] G.L. Nemhauser, A.W. Laurence, Integer and Combinatorial Optimization, Wiley, New York, 1988.
[17] H.W. Kuhn, The Hungarian method for the assignment and transportation problems, Nav. Res. Logist. Q. 2 (1955) 83–97.
[18] S. Martello, P. Toth, Linear assignment problems, in: S. Martello, G. Laporte, M. Minoux, C. Ribeiro (Eds.), Surveys in Combinatorial Optimization, in: Annals of Discrete Mathematics, vol. 31, North-Holland, Amsterdam, 1987, pp. 259–282.