Choi, Byung-Cheon; Briskorn, Dirk

Working Paper

Project scheduling with processing time compression cost and lateness penalties

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 651

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 651

Project Scheduling with Processing Time Compression Cost and Lateness Penalties

Byung-Cheon Choi[1], Dirk Briskorn[2]

January 2010

[1]: Chungnam National University
Department of Business
79 Daehakro, Yuseong-gu Daejeon 305-764, Korea
polytime@cnu.ac.kr

[2]: Christian-Albrechts-Universität zu Kiel
Institut für Betriebswirtschaftslehre
Olshausenstr. 40, 24098 Kiel, Germany
http://www.bwl.uni-kiel.de/Prod/team/briskorn/
briskorn@bwl.uni-kiel.de

Abstract

We consider a project scheduling problem where the precedence constraints graph is an out-tree. The processing times can be compressed by using additional resources, which causes cost and simultaneously reduces the processing times of jobs. The objective is to minimize the sum of total lateness penalties and total compression costs where the cost function for compressing the processing times is linear in the amount of compression. The problem can be decomposed into two types of subproblems. We show that both subproblems can be solved in polynomial time. Using these results, we show that the main problem is solvable in polynomial time if the number of chains in the out-tree is fixed.

Keywords: Project scheduling; controllable processing times; convex programming; shortest path problem.

1 Introduction

Project scheduling is an important task in project management. A project consists of several jobs which may be interrelated by precedence constraints. In order to complete a project, each job has to be completed, while the precedence constraints are satisfied. Additionally, we assume that the processing time of each job can be deceased by using additional resource such as manpower, fuels and so on. Consequently, some costs occur from the compression of a job's processing time.

Quite some pieces of work have been published considering project scheduling with a trade-off between processing times and resource consumption. Elmaghraby [6] introduces the multi-mode resource constrained project scheduling problem (MRCPSP) assuming that each job j can be performed in several modes. Each mode m reflects a feasible combination of processing time $p_{j,m}$ and resource request $r_{j,m,k}$ for resource k such that the corresponding job can be completed in $p_{j,m}$ time units if $r_{j,m,k}$ units of resource k are employed. Renewable as well as non-renewable resources are considered. The objective is to minimize the makespan while availability of each resource is limited. Clearly, MRCPSP is strongly NP-hard as a generalization of the well-known resource-constrained project scheduling problem which has been proven to be strongly NP-hard by Blazewicz et al. [2].

De et al. [4] consider two variants of the discrete time-resource tradeoff problem (DTCTP) where jobs can be performed in different modes and there is a single non-renewable resource. The objective of the first one is to minimize the makespan subject to resource availability, while the objective of the second one is to minimize the total consumption of resource subject to the constraint on the completion time of the project. De et al. [4] show that these problems are NP-hard.

Controllable processing times have mainly been considered in the context of machine scheduling so far. For a survey on machine scheduling with controllable processing times, we refer to Shabtay and Steiner [8].

To the best of our knowledge, few research has been conducted on project scheduling with continuously controllable processing times so far. Deckro et al. [5] consider a continuous version of the DTCTP with a quadratic objective function. Lee and Lei [7] consider two types of controllable processing times in a project scheduling environment without any precedence

constraints: (i) $p_j = p_j^0 + q_j/x_j$ and (ii) $p_j = p_j^0 - q_j x_j$ where, in both cases, p_j^0 and q_j are constants and x_j is the amount of resources employed to carry out job j. They consider the problem to schedule a project with controllable processing times according to (i) or (ii) where total resource employment is restricted as above. No precedence constraints are given. The objective is to minimize total completion time or total weighted tardiness for (i) and minimize of maximum tardiness for (ii). Additionally, for (ii) the problem to minimize total resource employment while providing a schedule without any tardy jobs is considered.

The paper at hand contributes to this field of research by considering a new project scheduling setting. Clearly, an out-tree precedence structure can arise in projects where the start of each job except the first job depends on the completion of exactly one predecessor. The objective of our problem setting clearly reflects the trade-off between compressing processing times and the employment of additional resources.

The remainder of this paper is organized as follows. In Section 2, we define the problem formally. Section 3 shows that the problem can be decomposed into two types of subproblems. In Sections 4 and 5, we prove that both subproblems can be solved in polynomial time by reductions to a shortest path problem and a convex programming problem, respectively. We prove the polynomiality of the problem with the fixed number of chains in Section 6. Finally, we complete the paper with concluding remarks and future works.

2 Problem Definition

We define a chain c to be a inclusion-wise maximum subset of jobs $j_1, \ldots, j_{n_c}$ such that

- j_k is a predecessor of j_{k+1} for each $k = 1, \ldots, n_c - 1$ and
- jobs $j_1, \ldots, j_{n_c-1}$ have exactly one successor each.

Note that j_{n_c} may have an arbitrary number of successors and n_c may equal 1. According to this definition, each job is contained in exactly one chain.

The problem can be specified as follows. We consider a project scheduling problem such that the processing times are compressible and the precedence constraints graph is an out-tree. The out-tree graph consists of l chains. There is one chain whose first job has no predecessor. The first job of remaining chains has exactly one predecessor which is the last job of another chain.

There is a set of n_i jobs $\{(i,1), \ldots, (i,n_i)\}$ in chain i, $i = 1, \ldots, l$. We specify a job by a tuple (i,j) where i gives the chain containing the job and j specifies the position of the job in chain i. We have an initial processing time $p_{i,j}^0$, a due date $d_{i,j}$, a maximal amount for compression $u_{i,j}$, a penalty for lateness $w_{i,j}$, and a compression cost rate $c_{i,j}$ associated with job (i,j), $i = 1, \ldots, l$, $j = 1, \ldots, n_i$. The objective is to find the schedule (T, x) which minimizes

$$z(T,x) = \sum_{(i,j)\in T} w_{i,j} + \sum_{i=1}^{l} \sum_{j=1}^{n_i} c_{i,j} x_{i,j}$$

where T is the set of tardy jobs and x is the vector of compression $x_{i,j}$ for each job (i,j), $i = 1, \ldots, l$, $j = 1, \ldots, n_i$.

Let the unique job without any predecessor be referred to as root job. We refer to jobs with no successor as leaf jobs. Let a chain be called the root (leaf) chain if it contains a root (leaf) job. Furthermore, let chain i be referred to as the chain of level h, if the number of chains is h in the path from the root job to the last job of chain i. Note that since the path between two jobs is unique in a tree graph, the level of each chain is uniquely defined.

For example, consider the project network depicted in Figure 1. We have eight chains. Clearly, each but the last node in a chain and the root node has exactly one outgoing arc and exactly one ingoing arc, respectively. Since job $(1,1)$ is the root job, chain 1 is the root chain, and since jobs $(3,n_3)$, $(4,n_4)$, $(6,n_6)$, $(7,n_7)$, and $(8,n_8)$ are leaf jobs, chains 3, 4, 6, 7, and 8 are the leaf chains. The level of chain 1 is 1, the level of chains 2 and 5 is 2, and the level of the remaining chains is 3.

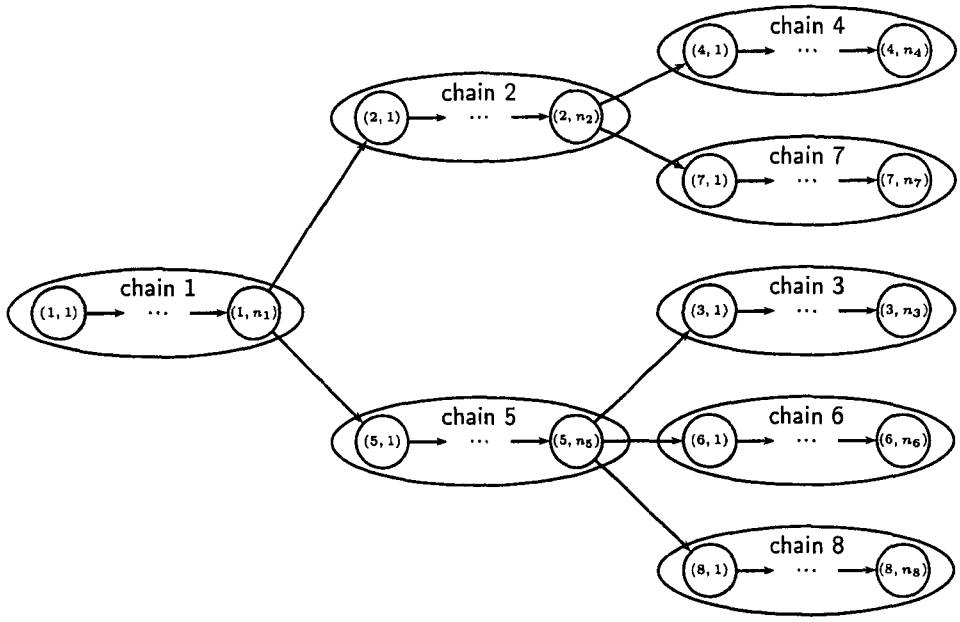

Figure 1: Example for project network

We refer to a job (i,j) completed exactly at its due date, that is $C_{i,j} = d_{i,j}$, as a JIT job.

3 Decomposition

Consider an optimal schedule. We identify

- the first and the last JIT jobs a_i and b_i, respectively, in chain i, $i = 1,\ldots,l$, if there are at least two JIT jobs in chain i,
- the unique JIT job b_i in chain i, $i = 1,\ldots,l$, if there is exactly one JIT job in chain i, and
- those chains having no JIT job at all.

Figure 2 provides an exemplary project network. In each chain the first and the last JIT job is drawn boldly if they exist. For example, chain 1 has exactly one JIT job, chain 4 has two JIT jobs, and chain 5 has none.

We decompose the project network as follows. We disconnect each of the JIT jobs described above from its immediate successors by dropping arc $i \longrightarrow j$ from the network where i and j are a JIT job and one of its immediate successors, respectively. For the original problem provided in Figure 2 we outline the decomposed network in Figure 3.

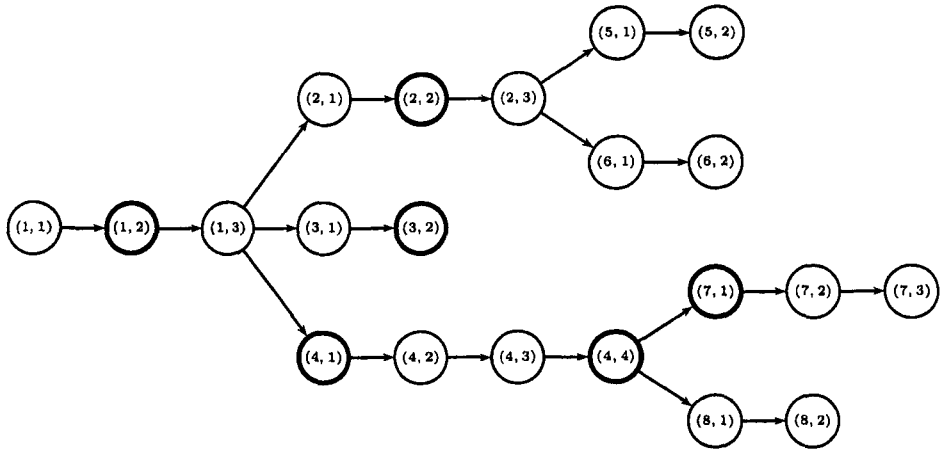

Figure 2: Original project network

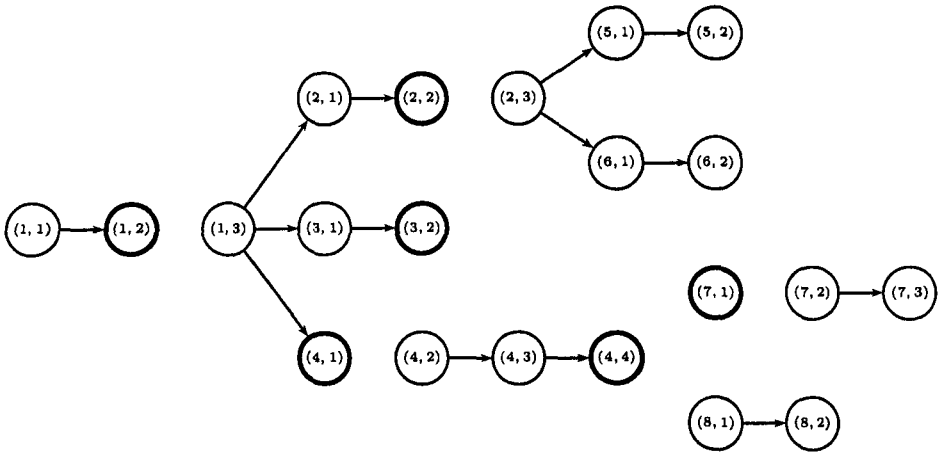

Figure 3: Decomposed Subproblems for the original project network

Then, we obtain two types of subproblems.

1. The first type of subproblems consists of a single chain such that the last job is a JIT job or a leaf job in the original problem and the first job has a release date. We refer to this type of subproblem as SubChain in the following. Let chain i' be the chain such that job $(i', n_{i'})$ is the immediate predecessor of job $(i, 1)$.

   (a) If chain i is neither the root chain nor a leaf chain, job $(i', n_{i'})$ is no JIT job, and chain i has at least two JIT jobs, then we obtain an instance of SubChain consisting of the set of jobs $\{(i, a_i + 1), \ldots, (i, b_i)\}$.

(b) If chain i is neither the root chain nor a leaf chain, job $(i', n_{i'})$ is a JIT job, and chain i has at least one JIT job, then we obtain an instance of SubChain consisting of the set of jobs $\{(i,1),\ldots,(i,b_i)\}$.

(c) If the root chain 1 has at least one JIT job, then we obtain an instance of SubChain consisting of the set of jobs $\{(1,1),\ldots,(1,b_1)\}$.

(d) If chain i is a leaf chain, job $(i', n_{i'})$ is no JIT job, and i has at least one JIT job, then we obtain an instance of SubChain consisting of the set of jobs $\{(i,a_i+1),\ldots,(i,n_i)\}$ and $\{(i,b_i+1),\ldots,(i,n_i)\}$ if i has at least two JIT jobs and i has exactly one JIT job.

(e) If chain i is a leaf chain and job $(i', n_{i'})$ is a JIT job, then we obtain an instance of SubChain consisting of the set of jobs $\{(i,1),\ldots,(i,n_i)\}$.

(f) If none of conditions (a) to (e) holds for chain i, then we do not obtain an instance of SubChain from chain i.

The release time of the first job of the instance of each SubChain is derived from the fixed completion time of its immediate predecessor. The completion time of the last job is fixed if it is chosen to be a JIT job. Note that the root job is released at time zero. In the example provided by Figures 2 and 3 we have an instance of SubChain according to (1a) which consists of jobs $(4,2)$, $(4,3)$, and $(4,4)$. Furthermore, we have an instance of SubChain according to (1b) consisting of job $(7,1)$ only. Jobs $(1,1)$ and $(1,2)$ form an instance of SubChain according to (1c). Additionally, we have one subproblem of type (1d) which consists of jobs $(7,2)$ and $(7,3)$ and one subproblem of type (1e) consisting of jobs $(8,1)$ and $(8,2)$.

2. Nodes not contained in an instance of SubChain derived according to (1a) to (1e) are contained in an instance of the second type of subproblems. We refer to this type of subproblem as SubOutTree in the following. SubOutTree is a special case of the underlying problem. Note that since arc $i \rightarrow j$ in the original problem is dropped if and only if i is one of the chosen JIT jobs, each of the chosen JIT jobs must be a leaf job in one of the subproblems. In an instance of SubOutTree the root node has a release time which is derived as in (1). Each leaf node is either a JIT job or a leaf job in the underlying original problem. Note that there may be subproblems of type (2) having no JIT job at all by choice of the first and the last JIT jobs for each chain in the underlying original problem. In the example provided by Figures 2 and 3 we have one subproblem of type (2) without any JIT job which consists of jobs $(2,3)$, $(5,1)$, $(5,2)$, $(6,1)$, and $(6,2)$. Furthermore, we have one subproblem with JIT jobs consisting of jobs $(1,3)$, $(2,1)$, $(2,2)$, $(3,1)$, $(3,2)$, and $(4,1)$.

By the decomposition scheme outlined above we derive two types of subproblems having a special structure. Of course, we do not know the set of JIT jobs in advance. However, there are $O\left(n^{2l}\right)$ feasible choices for the set of the first and the last JIT jobs in each chain, i.e., $\{a_i, b_i | i = 1,2,...,l\}$. Hence, the overall procedure to solve the problem at hand is to enumerate all feasible choices. Based on each feasible choice we decompose the original problem to no more than $O(l)$ subproblems which belong to SubChain or SubOutTree. After solving each subproblem, we can easily obtain an explicit optimal schedule for the original problem and the choice at hand. It remains to show that both, SubOutTree and SubChain, can be solved in polynomial time. Consequently, in Sections 4 and 5 we present polynomial time algorithms to solve SubOutTree and SubChain, respectively.

4 Polynomiality of SubOutTree

In this section we consider an instance of SubOutTree. As outlined in Section 3 only leaf jobs may be JIT jobs here. Note that, however, a leaf job may not be a JIT job. Let $S_{i,j}$ be the set of job (i,j) and its successors.

Lemma 1. *If there does not exist any JIT job in $S_{i,j}$, then jobs belonging to $S_{i,j}$ are uncompressed in each optimal schedule.*

Proof Consider an optimal schedule (T^*, x^*) such that job a is compressed, where $a \in S_{i,j}$. Then, let (T, x) be a new schedule constructed by letting $x_a = x_a^* - \epsilon$ where $\epsilon > 0$ is sufficiently small and $x_j = x_j^*$ for $j \neq a$. Clearly, since there exists no JIT job in $\{(i,j)\} \cup S_{i,j}$, the set of tardy jobs does not change, i.e., $T = T^*$. Thus,

$$z(T,x) = \sum_{j \in T} w_j + \sum_{j=1}^{n} c_j x_j = z(T^*, x^*) - c_a \epsilon < z(T^*, x^*).$$

This is a contradiction. ■

By Lemma 1 we can drop the jobs in $S_{i,j}$ from the problem instance for each job (i,j) such that $S_{i,j}$ does not contain a JIT job. Henceforth, we assume that each leaf job is a JIT job. For simplicity of notations, let chain 1 be the root chain and L be the set of leaf chains in the instance of SubOutTree.

4.1 Optimality Properties

Furthermore, we say that job (i,j) is partially compressed if $0 < x_{i,j} < u_{i,j}$ and introduce

- $\bar{d}_i$ representing the due date of the leaf job in chain i, $i \in L$,
- A_i and B_i representing the sets of compressed jobs and uncompressed jobs in chain i, that is, $x_{i,j} > 0$ if and only if $(i,j) \in A_i$, $i = 1, \ldots, l$, and $x_{i,j} = 0$ if and only if $(i,j) \in B_i$, $i = 1, \ldots, l$,
- $c_{max}^{A_i} = \max\{c_{i,j} | j \in A_i\}$ and $c_{min}^{B_i} = \min\{c_{i,j} | j \in B_i\}$, $i = 1, \ldots, l$.

Theorem 1. *There exists an optimal schedule (T^*, x^*) such that for $i = 1, 2, ..., l$,*

1. $c_{max}^{A_i} \leq c_{min}^{B_i}$,
2. $\max\{j | (i,j) \in A_i, c_{i,j} = c_{min}^{B_i}\} < \min\{j | (i,j) \in B_i, c_{i,j} = c_{max}^{A_i}\}$, *and*
3. *There is no more than one partially compressed jobs in A_i.*

Proof (1) Consider an optimal schedule (T^*, x^*) such that $c_{min}^{B_k} < c_{max}^{A_k}$ for $1 \leq k \leq l$. This implies that there exist jobs $(k,a) \in A_k$ and $(k,b) \in B_k$ such that $c_{k,a} > c_{k,b}$. We can construct a new schedule (T, x) by setting

- $x_{k,a} = x_{k,a}^* - \epsilon$,

- $x_{k,b} = \epsilon$, and
- $x_{k,j} = x^*_{k,j}$ for $\forall j \notin \{a, b\}$,

where $\epsilon > 0$ is sufficiently small. Since there is no JIT job among jobs between (k, a) and (k, b), the set of tardy jobs does not change, i.e., $T = T^*$. Since $c_{k,a} > c_{k,b}$,

$$z(T, x) = \sum_{(i,j)\in T} w_{i,j} + \sum_{i=1}^{l}\sum_{j=1}^{n_i} c_{i,j}x_{i,j} = z(T^*, x^*) - (c_{k,a} - c_{k,b})\epsilon < z(T^*, x^*).$$

This is a contradiction, which completes the proof of (1).

(2) Consider an optimal schedule (T^*, x^*) such that $a > b$ where $a = \max\{j|(k, j) \in A_k, c_{k,j} = c^{B_k}_{min}\}$ and $b = \min\{j|(k, j) \in B_k, c_{k,j} = c^{A_k}_{max}\}$ for $1 \le k \le l$. Note that $c_{k,a} = c_{k,b}$. Then, we can construct a new schedule (T, x) by setting

- $x_{k,a} = x^*_{k,a} - \epsilon$,
- $x_{k,b} = \epsilon$, and
- $x_{k,j} = x^*_{k,j}$ for $j \notin \{a, b\}$.

where $\epsilon = \min\{x^*_{k,a}, u_{k,b}\}$. Since jobs $(k, b), \ldots, (k, a-1)$ are completed earlier, we have $T \subseteq T^*$. Thus, since $c_{k,a} = c_{k,b}$,

$$z(T, x) = \sum_{(i,j)\in T} w_{i,j} + \sum_{i=1}^{l}\sum_{j=1}^{n_i} c_{i,j}x_{i,j} \le z(T^*, x^*) - (c_{k,a} - c_{k,b})\epsilon = z(T^*, x^*).$$

If $\epsilon = x^*_{k,a}$, jobs (k, a) and (k, b) belong to B_k and A_k, respectively, in (T, x). If $\epsilon = u_{k,b} < x^*_{k,a}$, both jobs, (k, a) and (k, b), belong to A_k in (T, x). We can apply this argument repeatedly until condition (2) is fulfilled. This completes the proof of (2).

(3) Consider an optimal schedule (T^*, x^*) with two partially compressed jobs (k, a) and (k, b), $a < b$, in chain k for $1 \le k \le l$. Let two new schedules (T, x) and (T', x') be constructed by setting

- $x_{k,a} = x^*_{k,a} + \epsilon$
- $x_{k,b} = x^*_{k,b} - \epsilon$,
- $x'_{k,a} = x^*_{k,a} - \epsilon$
- $x'_{k,b} = x^*_{k,b} + \epsilon$ and
- $x_{k,j} = x'_{k,j} = x^*_{k,j}$ for $\forall j \notin \{a, b\}$

where $\epsilon > 0$ is sufficiently small. Note that the set of tardy jobs does not change, i.e., $T = T' = T^*$. Then,

$$z(T, x) = z(T^*, x^*) + (c_a - c_b)\epsilon \text{ and } z(T', x') = z(T^*, x^*) - (c_a - c_b)\epsilon.$$

If $c_a - c_b \neq 0$, then $\min\{z(T, x), z(T', x')\} < z(T^*, x^*)$ which is a contradiction. Hence, $c_a - c_b = 0$. Then, we can construct a new schedule (T'', x'') by setting

- $x''_{k,a} = x^*_{k,a} + \epsilon$
- $x''_{k,b} = x^*_{k,b} - \epsilon$,
- $x''_{k,j} = x^*_{k,j}$ for $\forall j \notin \{a, b\}$,

where $\epsilon = \min\{u_{k,a} - x^*_{k,a}, x^*_{k,b}\}$. Then, in (T'', x''), either $x_{k,a} = u_{k,a}$ or $x_{k,b} = 0$ or both. Note that jobs $(k, a), \ldots, (k, b-1)$ are completed earlier in (T'', x'') than in (T^*, x^*) and, therefore, we have $T'' \subseteq T^*$. Since $c_{k,a} = c_{k,b}$, $\sum_{i=1}^{l} \sum_{j=1}^{n_i} c_{i,j} x''_{i,j} = \sum_{i=1}^{l} \sum_{j=1}^{n_i} c_{i,j} x^*_{i,j}$. Thus, we have

$$z(T'', x'') \leq z(T^*, x^*).$$

We can apply this procedure repeatedly until the number of partially compressed jobs in chain k becomes less than or equal to one. This completes the proof. ■

4.2 Reduction to Convex Programming Problem

Based on Theorem 1, we will reduce SubOutTree to the convex programming problem. For this reduction, the following notations will be used.

- Let $\sigma_i = (\sigma_i(1), \sigma_i(2), ..., \sigma_i(n_i))$ represent the sequence of jobs in chain i in order of non-decreasing compression cost, that is, $1 \leq \sigma_i(j) \leq n_i$ and $c_{i,\sigma_i(j)} \leq c_{i,\sigma_i(j+1)}$ for $1 \leq j \leq n_i - 1$, $i = 1, 2, ..., l$. Let ties in compression cost be broken by lower job index.
- Let δ_i be the total amount of compression in chain i, $i = 1, 2, ..., l$.
- Let H_i be the set of chains in the path from root job to leaf job (i, n_i) for $i \in L$.

Due to Theorem 1 we have

- $x_{i,\sigma_i(j)} = u_{i,\sigma_i(j)}$ for $0 \leq j \leq |A_i| - 1$,
- $0 < x_{i,\sigma_i(|A_i|)} \leq u_{i,\sigma_i(|A_i|)}$ for $|A_i| > 0$,
- $x_{i,\sigma_i(j)} = 0$ for $|A_i| + 1 \leq j \leq n_i$, and
- $\delta_i = \sum_{j=1}^{|A_i|} x_{i,\sigma_i(j)} = \sum_{j=1}^{|A_i|-1} u_{i,\sigma_i(j)} + x_{i,\sigma_i(|A_i|)}$

for each $1 \leq i \leq l$. Then, the total compression cost of chain i becomes $\int_0^{\delta_i} f_i(x)dx$, where

$$f_i(x) = \begin{cases} 0 & \text{for } x = 0 \\ c_{i,\sigma_i(j)} & \text{for } \sum_{j'=1}^{j-1} u_{i,\sigma_i(j')} < x \leq \sum_{j'=1}^{j} u_{i,\sigma_i(j')}, 1 \leq j \leq n_i. \end{cases} \quad (1)$$

Then, the total compression cost $F(\delta_1, \ldots, \delta_l)$ can be calculated as

$$F(\delta_1, \ldots, \delta_l) = \sum_{i=1}^{l} \int_0^{\delta_i} f_i(x)dx. \quad (2)$$

Lemma 2. *$F(\delta_1, \ldots, \delta_l)$ is a convex function.*

Proof Clearly, $\int_0^{\delta_i} f_i(x)dx$ is a convex function due to (1) and the definition of σ_i, $i = 1, ..., l$. Since $F(\delta_1, \ldots, \delta_l)$ is the sum of convex functions, it is a convex function. ■

Since all leaf jobs are JIT jobs due to Lemma 1,

$$\sum_{h \in H_i} \sum_{j=1}^{n_h} p_{h,j}^0 - \sum_{h \in H_i} \delta_h = \bar{d}_i \tag{3}$$

must hold for each $i \in L$. Hence, we can obtain the value of $F(\delta_1, \ldots, \delta_l)$ in an optimal schedule by solving the convex program (4).

$$\begin{array}{ll} \min & F(\delta_1, \delta_2, ..., \delta_l) \\ s.t. & \sum_{h \in H_i} \sum_{j=1}^{n_h} p_{h,j}^0 - \sum_{h \in H_i} \delta_h = \bar{d}_i \quad for \quad i \in L \\ & 0 \leq \delta_i \leq \sum_{j=1}^{n_i} u_{i,j}, \quad i = 1, 2, ..., l, \end{array} \tag{4}$$

However, there may exist multiple optimal solutions in the convex program (4). Let S^* be the set of optimal solutions to the convex program (4). Let $\{\rho_i(1), \rho_i(2), ..., \rho_i(\alpha_i)\}$ be the set of chains consisting of the path from the root chain to leaf chain i for $i \in L$ such that

- $\rho_i(1)$ and $\rho_i(\alpha_i)$ are the root chain and the leaf chain, respectively,
- Chain $\rho_i(k)$ is the successor of chain $\rho_i(k')$ if $k' < k$.

Lemma 3. *There exist a solution $(\delta_1^*, \delta_2^*, ..., \delta_l^*) \in S^*$ such that for an arbitrary solution $(\delta_1, \delta_2, ..., \delta_l) \in S^*$,*

$$\sum_{j=1}^{k} \delta_{\rho_i(j)}^* \geq \sum_{j=1}^{k} \delta_{\rho_i(j)}, \quad k = 1, 2, ..., \alpha_i \quad for \; i \in L.$$

Proof Let $\delta^1 = (\delta_1^1, \ldots, \delta_l^1) \in S^*$ and $\delta^2 = (\delta_1^2, \ldots, \delta_l^2) \in S^*$ be optimal solutions to (4) such that there is a path from the root chain to leaf chain i and a chain $\rho_i(h)$, $h = 1, \ldots, \alpha_i$, where

$$\sum_{j=1}^{k} \delta_{\rho_i(j)}^1 \geq \sum_{j=1}^{k} \delta_{\rho_i(j)}^2 \text{ for each } k = 1, 2, ..., h-1 \text{ and } \sum_{j=1}^{h} \delta_{\rho_i(j)}^1 < \sum_{j=1}^{h} \delta_{\rho_i(j)}^2.$$

Then, obviously, $\delta_{\rho_i(h)}^1 < \delta_{\rho_i(h)}^2$. Since total compression in each path from the root chain to a leaf chain is fixed, there exists a set of chains $\{h_1, \ldots, h_g\}$ such that

- for each path from $\rho_i(h)$ to a leaf chain which is a successor of $\rho_i(h)$ there is exactly one chain in $\{h_1, \ldots, h_g\}$ which is in this path and
- $\delta_{h_i}^1 > \delta_{h_i}^2$, $i = 1, \ldots, g$.

Since solutions δ^1 and δ^2 are optimal solutions to the convex program (4), there exist jobs $(\rho_i(h), \beta_0)$ in chain $\rho_i(h)$ and job $(h_{g'}, \beta_{g'})$ in chain $h_{g'}$, $g' = 1, \ldots, g$, such that

- $c_{\rho_i(h),\beta_0} = \sum_{g'=1}^{g} c_{h_{g'},\beta_{g'}}$,
- $x^1_{\rho_i(h),\beta_0} < u_{\rho_i(h),\beta_0}$ and $x^1_{h_{g'},\beta_{g'}} > 0$, $g' = 1, 2, ..., g$, where $x^1_{i,j}$ is the compression amount of job (i,j) in δ^1.

Now we can modify δ^1 by increasing $x^1_{\rho_i(h),\beta_0}$ by ϵ and decreasing $x^1_{h_{g'},\beta_{g'}}$, $g' = 1, \ldots, g$, by ϵ where

$$\epsilon = \min\{u_{\rho_i(h),\beta_0} - x^1_{\rho_i(h),\beta_0}, \min\{x^1_{h_{g'},\beta_{g'}}, g' = 1, \ldots, g\}\}.$$

By repeatedly applying the procedure above we can construct $(\delta^*_1, \ldots, \delta^*_l)$. ■

It is easy to see that $(\delta^*_1, ..., \delta^*_l)$ minimizes total tardiness penalty cost among all solutions in S^*. We can obtain $(\delta^*_1, ..., \delta^*_l)$ by solving the convex program (4) such that costs are perturbed according to

$$\bar{c}_{i,j} = (1 + \epsilon^{l-q_i})c_{i,j}, \quad i = 1, 2, ..., l, \quad j = 1, 2, ..., n_i, \tag{5}$$

where $\bar{c}_{i,j}$ is the perturbed compression cost rate of job (i,j) and $\epsilon > 0$ is sufficiently small, l is the number of chains, and q_i is the level of chain i. Note that the number of the optimal solution satisfying Lemma 3 is unique.

Theorem 2. *A solution which minimizes total tardiness penalty among solutions in S^* can be found in polynomial time.*

Proof It follows immediately from Lemma 3 and the polynomiality of convex program (4), see Boyd [3]. ■

4.3 Algorithm for SubOutTree

Based on Theorems 1 and 2, the following algorithm is obtained.

Algorithm SubOutTree

1. After perturbing $c_{i,j}$ according to (5), obtain $(\delta^*_1, \delta^*_2, ..., \delta^*_l)$ by solving the convex program (4).
2. For each chain i, $i = 1, ..., l$, compress jobs according to the order σ_i such that the total compression amount of chain i equals δ^*_i. That is, set

$$x^*_{i,\sigma_i(j)} = \min\left\{u_{i,\sigma_i(j)}, \max\left\{0, \delta^*_i - \sum_{j'=1}^{j-1} u_{i,\sigma_i(j')}\right\}\right\}.$$

3. Find the tardy jobs in chain i, $1 \le i \le l$. Let T^*_i be the set of the tardy jobs in chain i.
4. The total objective value becomes $\sum_{(i,j)\in T^*} w_{i,j} + \sum_{i=1}^{l}\sum_{j=1}^{n_i} x^*_{i,j}$.

Step (1) can be done in polynomial time according to Theorem 2. Steps 2 and 3, clearly, can be done in polynomial time. Thus, Algorithm SubOutTree finishes in polynomial time.

Theorem 3. *SubOutTree is polynomially solvable.*

Proof The optimal schedule can be obtained by Algorithm SubOutTree, which runs in polynomial time. The proof is complete. ■

5 Polynomiality of SubChain

In this section we consider an instance of SubChain. Since the precedence constraints form a single chain, the notation can be simplified by dropping chain index i, that is, we write j, p_j^0, d_j, u_j, w_j, x_j, and c_j instead of (i,j), $p_{i,j}^0$, $d_{i,j}$, $u_{i,j}$, $w_{i,j}$, $x_{i,j}$, and $c_{i,j}$, respectively. Without loss of generality, we assume that the number of jobs in the instance of SubChain under consideration is n.

5.1 Optimality Conditions

Suppose that there exists an optimal schedule such that $\{k_1, ..., k_m\}$ is the set of JIT jobs where $k_1 < \cdots < k_m$. For simplicity, let $k_0 = 0$ be a dummy job with $p_0^0 = 0$, $w_0^0 = 0$ and $d_0^0 = r_1$ where r_1 is the release time of the first job. A feasible choice of $\{k_1, ..., k_m\}$ implies that

$$d_{k_{q-1}} + \sum_{j=k_{q-1}+1}^{k_q} p_j^0 \geq d_{k_q} \text{ and} \tag{6}$$

$$d_{k_{q-1}} + \sum_{j=k_{q-1}+1}^{k_q} (p_j^0 - u_j) \leq d_{k_q} \tag{7}$$

must hold for $q = 1, 2, ..., m$. For the clarity of explanation, we introduce the following notation. For $q = 1, ..., m$, let A_q and B_q be the sets of compressed jobs and uncompressed jobs in $\{k_{q-1}+1, ..., k_q\}$, and let, furthermore, $c_{max}^{A_q} = \max\{c_j | j \in A_q\}$ and $c_{min}^{B_q} = \min\{c_j | j \in B_q\}$.

Theorem 4. *There exists an optimal schedule (T^*, x^*) such that*

1. $c_{max}^{A_q} \leq c_{min}^{B_q}$, $q = 1, ..., m$,
2. *if* $c_{max}^{A_q} = c_{min}^{B_q}$, *then* $\max\{j | j \in A_q, c_j = c_{min}^{B_q}\} < \min\{j | j \in B_q, c_j = c_{max}^{A_q}\}$, $q = 1, ..., m$,
3. *the number of a partially compressed jobs in A_q is at most one, $q = 1, ..., m$,*
4. *the jobs in $\{k_m + 1, ..., n\}$ are uncompressed.*

Proof Properties (1), (2), and (3) can be proven as Theorem 1, and Property (4) can be proven as Lemma 1. ■

According to Theorem 4, there exists an optimal schedule such that the total cost of compressing the jobs between JIT jobs k_{q-1} and k_q, $q = 1, ..., m$, is minimum. Note that there is exactly one schedule satisfying (1) to (4) in Theorem 4 for a given choice of $\{k_1, ..., k_m\}$.

5.2 Reduction to Shortest Path Problem

In Section 5.1, we derived optimality conditions assuming that the set of JIT jobs is known in advance. It remains to find the set of JIT jobs in an optimal schedule. Based on Theorem 4, this problem can be resolved by reducing it to the shortest path problem.

Lemma 4. *SubChain can be reduced to the shortest path problem.*

Proof Given an instance of SubChain, we construct an instance of the shortest path problem as follows. For $0 \leq \alpha \leq n+1$, let $N(\alpha)$ denote a node. Let $N(0)$ and $N(n+1)$ be the source node and the sink node, respectively.

Arc $(N(\alpha), N(\beta))$, $0 \leq \alpha < \beta \leq n$, exists if jobs α and β can be JIT jobs such that no job between α and β is a JIT job according to (6) and (7). Additionally, for each α, $0 \leq \alpha \leq n$, arc $(N(\alpha), N(n+1))$ exists.

Length $c_{\alpha,\beta}$ of arc $(N(\alpha), N(\beta))$, $1 \leq \alpha < \beta \leq n$, is calculated by the following procedure. The total compression

$$\delta = d_\alpha + \sum_{j=\alpha+1}^{\beta} p_j^0 - d_\beta$$

is derived from jobs α and β being JIT jobs. Now, it is easy to derive compression x_j of job j, $\alpha+1 \leq j \leq \beta$ according to Theorem 4. Then, since the start time of job $\alpha+1$ is known, we can find the corresponding set $T_{\alpha,\beta}$ of tardy jobs easily. Finally, let

$$c_{\alpha,\beta} = \sum_{j \in T_{\alpha,\beta}} w_j + \sum_{j=\alpha+1}^{\beta} c_j x_j.$$

Note that by Theorem 4, length of edge $(N(\alpha), N(\beta))$ represents the total costs occurring for jobs $\{\alpha+1, \ldots, \beta\}$ if jobs α and β are consecutive JIT jobs in a schedule.

Length $c_{0,\alpha}$ of arc $(N(0), N(\alpha))$, $1 \leq \alpha \leq n$, is calculated by a similar procedure. Then, length $c_{0,\alpha}$ represents the total costs occurring for jobs $\{1, \ldots, \alpha\}$ if job α is the first JIT jobs in a schedule.

Length $c_{\alpha,n+1}$ of arc $(N(\alpha), N(n+1))$, $0 \leq \alpha \leq n$, is derived by scheduling jobs $\alpha+1, \ldots, n$ without any compression. The start time of job $\alpha+1$ is zero if $\alpha = 0$ and d_α otherwise. Note that by Theorem 4, length $c_{\alpha,n+1}$ represents the total costs occurring for jobs $\{\alpha+1, \ldots, n\}$ if job α is the last JIT jobs in a schedule.

Finding a shortest path corresponds to choosing JIT jobs by visiting nodes. Moreover, the length of the shortest path is corresponding to the sum of total compression costs and total tardiness penalties in an optimal schedule. This completes the proof. ■

Theorem 5. *SubChain can be solved in $O(n^3)$.*

Proof The number of arcs in graph in derived from an instance of SubChain is $O(n^2)$ according to Lemma 4. Hence, we can find the shortest path by the algorithm in Ahuja et al. [1] in $O(n^2)$ time. The construction of the graph can be done in $O(n^3)$ time since computation of cost of each arc takes $O(n)$. This completes the proof. ■

6 Conclusions and Future Works

We consider a project scheduling problem with controllable processing times such that the precedence constraints graph is an out-tree. The objective is to minimize the sum of total compression costs and total tardiness penalties. The problem can be decomposed into two of subproblems, SubOutTree and SubChain, based on the set of the first and last JIT jobs

in each chain. We prove polynomiality of both subproblems by reducing them to the convex programming and the shortest path problem, respectively. Based on these results, we prove polynomiality of the problem when the number of chains is fixed.

It is still open to resolve the computational complexity of the problem with an arbitrary number of chains. Also, it is interesting to explore whether the techniques suggested in this paper can be applied to problems differing from the one at hand by precedence constraints structures, e.g., the problem with in-tree precedence constraints graph.

References

[1] R.A. Ahuja, K. Mehlhorn, J.B. Orlin, and R.E. Tarjan. *Faster algorithms for the shortest path problem.* Journal of the Association for Computing Machinery 37:213–223, 1990.

[2] J. Blazewicz, J. K. Lenstra, and A. H. G. Rinnooy Kan. *Scheduling subject to resource constraints: Classification and complexity*. Discrete Applied Mathematics, 5:11-24, 1983.

[3] T. Boyd, and L. Vandenberghe, Convex Optimization, Cambridge University PRess, 2004.

[4] P. De, J. Dunne, J.B, Ghosh, and C.E. Wells. *Complexity of the discrete time-cost trade-off problem for project networks*. Operations Research, 45:302-306, 1997.

[5] R.F. Dercker, J.E. Hebert, W.A. Verdini, P.H. Grimsrud, and S. Venkateshwar. *Nonlinear time/cost trade-off models in project management*. Computers & Industrial Engineering, 28(2):219-229, 1995.

[6] S.E. Elaghraby. *Activity Networks: Project planning and control by network models*. Wiley, New York, 1977.

[7] C.Y. Lee, and L. Lei. *Multiple-project scheduling with controllable project duration and hard resource constraint: some solvable cases*. Annals of Operations Research, 102:287-307, 2001.

[8] D. Shabtay, and G. Steiner. *A survey of scheduling with controllable processing times*. Discrete Applied Mathematics, 155:1643-1666, 2007.